



HAL
open science

Apprentissage de la représentation du style écrit, application à la recommandation d'articles d'actualité

Julien Hay

► **To cite this version:**

Julien Hay. Apprentissage de la représentation du style écrit, application à la recommandation d'articles d'actualité. Apprentissage [cs.LG]. Université Paris-Saclay, 2021. Français. NNT : 2021UP-ASG010 . tel-03420487

HAL Id: tel-03420487

<https://theses.hal.science/tel-03420487>

Submitted on 9 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Apprentissage de la représentation du style écrit, application à la recommandation d'articles d'actualité

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, ED Sciences et technologies de
l'information et de la communication (STIC)
Spécialité de doctorat : Informatique
Unité de recherche : Université Paris-Saclay, CNRS, Laboratoire
Interdisciplinaire des Sciences du Numérique, 91405, Orsay, France
Réfèrent : CentraleSupélec

**Thèse présentée et soutenue à Paris-Saclay,
le 9 mars 2021, par**

Julien Hay

Composition du jury :

Anne Vilnat Professeure des Universités, Université Paris-Saclay	Présidente
Eric Gaussier Professeur des Universités, Université Grenoble Alpes	Rapporteur & Examinateur
Thierry Poibeau Directeur de recherche, CNRS & ENS/PSL	Rapporteur & Examinateur
Armelle Brun Maîtresse de conférences HDR, Université de Lorraine	Examinatrice
Benjamin Piwowarski Chargé de recherche, Sorbonne Université	Examinateur
Max Chevalier Professeur des Universités, Université Toulouse III - Paul Sabatier	Examinateur
Bich-Liên Doan Professeure, CentraleSupélec	Directrice de thèse
Fabrice Popineau Professeur, CentraleSupélec	Co-encadrant de thèse

Titre : Apprentissage de la représentation du style écrit, application à la recommandation d'articles d'actualité

Mots clés : Apprentissage automatique, Traitement automatique du langage naturel, Apprentissage profond, Style écrit, Système de recommandation, Recommandation d'articles, Diversité

Résumé : La modélisation des utilisateurs est une étape essentielle lorsqu'il s'agit de recommander des produits et proposer des services automatiquement. Les réseaux sociaux sont une ressource riche et abondante de données utilisateur (e.g. liens partagés, messages postés) permettant de modéliser leurs intérêts et préférences. Dans cette thèse, nous proposons d'exploiter les articles d'actualité partagés sur les réseaux sociaux afin d'enrichir les modèles existants avec une nouvelle caractéristique textuelle : le style écrit.

Cette thèse, à l'intersection des domaines du traitement automatique du langage naturel et des systèmes de recommandation, porte sur l'apprentissage de la représentation du style et de son application à la recommandation d'articles d'actualité. Dans un premier temps, nous proposons une nouvelle méthode d'apprentissage de la représentation du texte visant à projeter tout document dans un espace stylométrique de référence. L'hypothèse testée est qu'un tel espace peut être généralisé par un ensemble suffisamment large d'auteurs de référence, et que les projections vectorielles des écrits d'un auteur « nouveau » seront proches, d'un

point de vue stylistique, des écrits d'un sous-ensemble consistant de ces auteurs de référence.

Dans un second temps, nous proposons d'exploiter la représentation stylométrique du texte pour la recommandation d'articles d'actualité en la combinant à d'autres représentations (e.g. thématique, lexicale, sémantique). Nous cherchons à identifier les caractéristiques les plus complémentaires pouvant permettre une recommandation d'articles plus pertinente et de meilleure qualité. L'hypothèse ayant motivé ces travaux est que les choix de lecture des individus sont non seulement influencés par le fond (e.g. le thème des articles d'actualité, les entités mentionnées), mais aussi par la forme (i.e. le style pouvant, par exemple, être descriptif, satirique, composé d'anecdotes personnelles, d'interviews). Les expérimentations effectuées montrent que non seulement le style écrit joue un rôle dans les préférences de lecture des individus, mais aussi que, lorsqu'il est combiné à d'autres caractéristiques textuelles, permet d'augmenter la précision et la qualité des recommandations en termes de diversité, de nouveauté et de sérendipité.

Title: Representation learning of writing style, application to news recommendation

Keywords: Machine Learning, Natural Language Processing, Deep Learning, Writing style, Recommender system, News recommendation, Diversity

Abstract: User modeling is an essential step when it comes to recommending products and offering services automatically. Social networks are a rich and abundant resource of user data (e.g. shared links, posted messages) that allow to model their interests and preferences. In this thesis, we propose to exploit news articles shared on social networks in order to enrich existing models with a new textual feature : the writing style.

This thesis, at the intersection of the fields of natural language processing and recommender systems, focuses on the representation learning of writing style and its application to news recommendation. As a first step, we propose a new representation learning method that aims to project any document into a reference stylometric space. The hypothesis being tested is that such a space can be generalized by a sufficiently large set of reference authors, and that the vector projections of the writings of a "new" author will be stylistically close to the writings of a

consistent subset of these reference authors.

In a second step, we propose to exploit the stylometric representation for news recommendation by combining it with other representations (e.g. topical, lexical, semantic). We seek to identify the most relevant and complementary characteristics that can allow a more relevant and better quality recommendation of articles. The hypothesis that motivated this work is that the reading choices of individuals are not only influenced by the content (e.g. the theme of news articles, the entities mentioned), but also by the form (i.e. the style that can, for example, be descriptive, satirical, composed of personal anecdotes, interviews). The experiments conducted show that not only does writing style play a role in individuals' reading preferences, but also that, when combined with other textual features, it increases the accuracy and quality of recommendations in terms of diversity, novelty and serendipity.

Apprentissage de la représentation du style
écrit, application à la recommandation
d'articles d'actualité

Julien Hay

Remerciements

Tout d'abord, je souhaiterais remercier **Mahmoud Zakaria** et **Abdelkrim Talhaoui**, directeurs et cofondateurs d'Octopeek, qui m'ont fait confiance et l'honneur d'avoir été le premier doctorant de l'entreprise, et sans lesquels je n'aurais pu m'aventurer dans le monde de la recherche durant ces années de thèse.

J'adresse de vifs remerciements à mes collègues et encadrants en entreprise **Ouassim Ait Elhara** et **Mahdi Hannoun** qui ont su m'apporter leur appui à de nombreuses reprises et faire preuve d'un sens critique dans l'objectif de me diriger techniquement et scientifiquement. Tous deux m'ont aussi beaucoup aidé lorsqu'il s'agissait d'améliorer la présentation de mes résultats, et notamment dans les derniers jours lors de ma préparation à la soutenance. Je remercie également **Lyes Benamsili** qui m'a soutenu et conseillé en début de thèse.

Je remercie ensuite tout particulièrement **Bich-Liên Doan**, ma directrice de thèse ainsi que **Fabrice Popineau**, co-encadrant de thèse, qui ont tous deux énormément contribué au succès de cette thèse. Je remercie Bich-Liên Doan pour sa bienveillance, pour m'avoir apporté la motivation et toute l'aide dont j'avais besoin, tout en me laissant une grande liberté dans mon travail. Je remercie Fabrice Popineau pour sa patience et pour m'avoir aidé et dirigé sur de nombreux aspects scientifiques et techniques.

Je tiens à remercier sincèrement **Eric Gaussier**, professeur des Universités, ainsi que **Thierry Poibeau**, directeur de recherche, pour avoir accepté d'être rapporteur et membre de mon jury de thèse. Leurs relectures et précieux conseils m'ont permis d'apporter des corrections importantes à ce manuscrit. Je remercie également **Anne Vilnat**, professeure des Universités, **Armelle Brun**, maîtresse de conférences HDR, **Benjamin Piwowarski**, chargé de recherche, et **Max Chevalier**, professeur des Universités, de m'avoir fait l'honneur d'être membre de mon jury de thèse, pour l'intérêt prononcé qu'ils ont eu à l'appréciation de mon travail et pour les échanges constructifs que nous avons eu lors de ma soutenance.

Bien entendu, je n'oublie pas de remercier tous mes collègues chez Octopeek ainsi qu'au LISN, et notamment **Benoît Choffin** avec qui j'ai partagé ces formidables années de thèse.

Je tiens sincèrement à remercier tous ceux qui ont participé au projet *Renewal*. Depuis ses débuts, ce projet a pour ambition d'améliorer l'état de l'art des algorithmes de recommandation par l'organisation de compétitions autour de cette tâche. Aujourd'hui, par le dévouement de chacun, et notamment d'**Erik M. Bray** et d'**Anne-Catherine Letournel**, cette ambition n'a pas faibli. Je les remercie de leur contribution à ce projet qui a encore du chemin à parcourir. Je n'oublie pas les membres du **staff du LISN**, et notamment **Laurent Darré**, qui m'a toujours apporté son aide, que ce soit sur *Renewal* ou sur d'autres aspects techniques de ma thèse.

Enfin et surtout, je tiens à remercier infiniment ma famille et mes proches, notamment **Chantal Hay**, **Jean-Claude Hay**, **Célia Hay** ainsi que ma conjointe **Ornanong Narat** qui m'ont soutenu tout au long de ces années de thèse et ont aussi contribué à la relecture du présent manuscrit.

Table des matières

Remerciements	1
Introduction	7
Contexte	7
Accès à l'information pertinente	7
Enrichissement des profils sur les réseaux sociaux	9
L'entreprise Octopeek	10
Problématiques scientifiques	11
Contributions	13
Publications	14
Plan de thèse	15
I Apprentissage de la représentation du style	17
1 État de l'art des méthodes de représentation du texte et de la stylométrie	18
1.1 Introduction	18
1.2 Représentation du texte	18
1.2.1 Représentation vectorielle du texte	19
1.2.2 Approches basées sur les réseaux de neurones profonds	26
1.3 La stylométrie	30
1.3.1 Définition et motivations	30
1.3.2 Solutions proposées dans la littérature	31
1.3.3 Amélioration des performances par masquage	32
1.3.4 Apprentissage de la représentation du style écrit	33
1.3.5 Limites identifiées dans l'état de l'art	34
1.4 Conclusion	35
2 Méthode de représentation du style écrit et cadre d'évaluation	36
2.1 Introduction	36
2.2 Le style écrit	37
2.3 La méthode de ciblage des indices intra-auteurs consistants	41
2.3.1 L'ensemble de référence et les ensembles nouveaux	42
2.3.2 L' <i>hypothèse de généralisation du style</i>	43
2.3.3 Évaluation de la qualité des représentations vectorielles	45
2.4 La <i>non-spécificité sémantique</i>	50
2.5 L' <i>hypothèse de filtrage</i>	52
2.6 Conclusion	54

3	Construction d'un corpus de référence stylistique	55
3.1	Introduction	55
3.2	Collecte des données	55
3.3	Élimination du bruit	56
3.4	Prétraitement et attribution de labels	58
3.5	Filtrage des indices révélateurs	61
3.5.1	Méthodologie de filtrage	61
3.5.2	L'algorithme <i>DocDist</i>	63
3.5.3	Génération d'un groupe	65
3.5.4	Génération des <i>n</i> -grams noirs	67
3.5.5	Résultats du filtrage sur <i>NewsID</i>	69
3.5.6	Avantages de la méthode	69
3.5.7	Importance de la difficulté à identifier les auteurs	70
3.5.8	Différences avec les méthodes de masquage	71
3.6	Extraction du <i>R-set</i> et des <i>U-sets</i>	71
3.7	Conclusion	73
4	Expérimentations sur les tâches de la stylométrie	75
4.1	Introduction	75
4.2	Partitionnement par auteur	76
4.2.1	Le modèle <i>Stylometric Neural Attention</i>	76
4.2.2	Le modèle <i>DistilBert</i> affiné	78
4.2.3	Les modèles <i>baselines</i> pour la stylométrie	79
4.2.4	Performance des modèles dans le partitionnement par auteur	80
4.2.5	Performance des combinaisons de modèles dans le partitionnement par auteur	84
4.2.6	Visualisation de partitions	85
4.3	Identification d'auteur	86
4.3.1	Procédure d'évaluation	86
4.3.2	Performance des modèles dans l'identification d'auteurs	87
4.3.3	Performance des combinaisons de modèles dans l'identification d'auteurs	88
4.3.4	Apprentissage par transfert	89
4.3.5	Carte d'attention textuelle	91
4.3.6	La <i>non-spécificité sémantique</i>	91
4.3.7	Impact du nombre de documents par auteur	93
4.4	Validation de l' <i>hypothèse de filtrage</i>	95
4.4.1	Impact du filtrage sur le partitionnement par auteur	96
4.4.2	Impact du filtrage sur l'identification d'auteurs	97
4.4.3	Impact du filtrage sur l'adéquation des modèles avec la <i>non-spécificité sémantique</i>	98
4.5	Conclusion	99
II	Recommandation d'articles d'actualité	102
5	État de l'art des systèmes de recommandation	103
5.1	Introduction	103
5.2	Systèmes de recommandation	103

5.2.1	Généralités	103
5.2.2	Filtrage collaboratif	105
5.2.3	Filtrage par le contenu	107
5.2.4	Méthodes hybrides	108
5.3	Évaluation des systèmes de recommandation	110
5.3.1	Évaluation <i>offline</i>	110
5.3.2	Évaluation qualitative	114
5.3.3	Évaluation <i>online</i>	121
5.3.4	Classification des méthodes et métriques d'évaluation	123
5.4	Recommandation d'articles d'actualité	127
5.4.1	Différences entre domaines de recommandation	127
5.4.2	Particularités de la recommandation d'articles d'actualité	129
5.4.3	Jeux de données pour la recommandation d'articles d'actualité	132
5.4.4	Algorithmes de recommandation	133
5.4.5	Limites identifiées dans l'état de l'art	135
5.5	Conclusion	139
6	Recommandation d'articles d'actualité par l'exploitation de représentations du texte	141
6.1	Introduction	141
6.2	Évaluation <i>offline</i> par découpage temporel	142
6.3	Algorithme générique de recommandation exploitant la représentation vectorielle des items	145
6.4	Exploitation de la complémentarité des représentations par hybridation des ordonnancements	147
6.4.1	L'algorithme d'hybridation <i>Resuhy</i>	147
6.4.2	La dominance et l'explicabilité d'une hybridation	151
6.4.3	Le rééchelonnage des ordonnancements	154
6.5	Conclusion	155
7	Récolte d'un jeu de données large pour la recommandation d'articles d'actualité	157
7.1	Introduction	157
7.2	Le jeu de données <i>Twinews</i>	158
7.2.1	Collecte des articles d'actualité	158
7.2.2	Prise en compte des <i>urls</i> réduites	161
7.2.3	Collecte des profils Twitter	162
7.2.4	Filtrage des profils	163
7.2.5	Particularités de <i>Twinews</i>	164
7.3	Évaluation <i>offline</i> dans le cadre de <i>Twinews</i>	166
7.3.1	Biais inhérents à l'évaluation <i>offline</i> et aux indicateurs de pertinence	166
7.3.2	Découpage de <i>Twinews</i>	170
7.4	Conclusion	171
8	Expérimentation de la recommandation basée sur le style	173
8.1	Introduction	173
8.2	Exploitation de modèles de représentation pour la recommandation d'articles d'actualité	173

8.2.1	La pondération <i>BM25</i>	174
8.2.2	Les modèles <i>baselines</i>	174
8.2.3	Optimisation des hyperparamètres	175
8.2.4	Métriques de précision d'ordonnement	176
8.2.5	Résultats obtenus sur <i>TwineWS</i>	176
8.3	Validation de l' <i>hypothèse de proximité partielle</i>	177
8.4	Combinaison des ordonnancements et complémentarité des représen- tations	179
8.4.1	Motivations	179
8.4.2	Impact du rééchelonnage sur la dominance et les métriques d'ordonnement	181
8.4.3	Résultats obtenus par hybridation des ordonnancements sur <i>TwineWS</i>	182
8.5	Au-delà de la précision d'ordonnement	184
8.5.1	Intérêts des mesures qualitatives	184
8.5.2	Rôle de l'hybridation	186
8.5.3	Fonctions de similarité	188
8.5.4	La diversité	189
8.5.5	La nouveauté	190
8.5.6	La sérendipité	191
8.5.7	La nouveauté stricte	192
8.5.8	Résultats qualitatifs des ordonnancements	193
8.5.9	Performance multi-critères	196
8.6	Conclusion	199
9	La plateforme d'évaluation <i>Renewal</i>	201
9.1	Introduction	201
9.2	Nouveautés apportées par <i>Renewal</i>	202
9.2.1	Architecture indépendante	203
9.2.2	Sources variées d'articles d'actualité et indépendance du contexte de recommandation	204
9.2.3	Historique utilisateur long terme	205
9.2.4	Mesurer la réelle satisfaction de l'utilisateur	205
9.2.5	Données mises à disposition aux systèmes de recommandation concurrents	207
9.2.6	Allègement de contraintes	207
9.3	Évaluation des systèmes concurrents	208
9.4	Prédiction des conditions nécessaires par simulation de compétitions .	211
9.4.1	Motivation	211
9.4.2	Paramètres initiaux des simulations	212
9.4.3	Simulation de compétitions	214
9.4.4	Résultats des simulations	217
9.4.5	Nombre de jours nécessaires en fonction du ratio utilisateurs- systèmes	219
9.5	Conclusion	220
	Conclusion	221
	Bibliographie	225

Introduction

Contexte

Accès à l'information pertinente

De nos jours, l'accès à l'information s'est transformé. Grâce au web, chaque individu peut trouver l'information qu'il cherche sur la quasi-totalité des sujets. Plus récemment, la multiplication des applications et des réseaux sociaux a rendu l'information encore plus riche et abondante. De surcroît, l'omniprésence des appareils mobiles amplifie considérablement cette consommation. Mais l'accès illimité à la connaissance et à l'actualité engendre, pour tout individu, une surabondance difficile à assimiler. Et cette surabondance conduit à la nécessité de filtrer et de personnaliser automatiquement l'information, chaque individu ayant un besoin spécifique et un temps limité (e.g. dans sa lecture de l'actualité, dans sa recherche d'informations). En effet, les articles et autres contenus présents sur les réseaux sociaux sont devenus si abondants qu'il est aujourd'hui très difficile, pour un individu, d'en dégager une information précise répondant à ses besoins, ses intérêts personnels, exprimés explicitement ou non.

Le domaine de la recherche d'information apporte des solutions dans la personnalisation de l'information lorsque l'utilisateur exprime « explicitement » un besoin. Ce domaine se compose de méthodes consistant à filtrer un ensemble d'« items »¹ (e.g. documents, pages web) en confrontant ceux-ci à une « requête » de l'utilisateur (Manning et al., 2008). Ces requêtes peuvent par exemple prendre la forme d'un texte court ou d'une image. Elles peuvent être à l'origine « vocales » et traduites en texte. Mais au-delà du contenu seul, ce domaine s'intéresse aussi aux contextes d'interaction utilisateurs-système² et aux liens entre les données. Les liens d'amitié dans les réseaux sociaux (e.g. Twitter, Facebook) (Pla Karidi et al., 2017), l'enrichissement des données par des sources externes (e.g. Wikipédia, DBPedia) (Carpineto et Romano, 2012) ou encore le contexte d'interaction (disponible via les smartphones et autres objets connectés) (Baldauf et al., 2007) sont autant d'indices permettant d'améliorer la précision des systèmes de recherche d'information par une personnalisation toujours plus fine des résultats envoyés à l'utilisateur.

1. Un « item », dans le contexte des systèmes de recommandation (ou des systèmes de recherche d'information, aussi appelés moteurs de recherche), est un élément du système qui est candidat à une recommandation pour les utilisateurs. Généralement, les items sont de même type (e.g. films, documents, produits).

2. Une « interaction utilisateur-système » correspond à toute action de l'utilisateur (e.g. requête textuelle, clics) sur l'interface du système. Un « système », dans le cadre de cette thèse, désignera un outil informatique permettant à un utilisateur d'effectuer une recherche (i.e. moteur de recherche) ou de recevoir des recommandations (i.e. système de recommandation).

Plus récemment, les techniques issues du domaine des systèmes de recommandation tentent de répondre aux besoins d'information « implicite » des utilisateurs en proposant des items pertinents selon le profil de chaque utilisateur, celui-ci étant souvent construit grâce à son historique d'interactions avec les items présents dans le système (Resnick et Varian, 1997). Dans ce domaine, il est admis que non seulement les utilisateurs sont en surabondance d'informations, donc ont un accès difficile à l'information pertinente et personnalisée, mais aussi qu'il leur est difficile d'exprimer un besoin explicite d'information. En effet, ils n'ont généralement pas connaissance de tous les sujets d'intérêts qu'ils jugeraient pertinents (Özgöbek et al., 2014). Pour les systèmes de recommandation, il s'agit alors de substituer les requêtes (utilisées dans le domaine de la recherche d'information) à des informations liées à l'utilisateur (e.g. ses consommations³ passées, des informations démographiques) afin de trouver automatiquement des items pertinents sur de nouveaux sujets d'intérêts ou des sujets liés à ce qu'il a déjà consommé. Les modèles les plus utilisés pour la recommandation sont ceux qui représentent les utilisateurs par leurs interactions avec le système, i.e. les items cliqués, aimés, notés. Et par l'exploitation de ces interactions, il est possible de capturer des similarités entre utilisateurs ou items (Koren et al., 2009). L'hypothèse sous-jacente est que si une personne A a la même opinion qu'une personne B sur un premier sujet, elle aura plus probablement une opinion proche de celle de B sur un second sujet qu'une opinion proche de celle d'une personne C prise au hasard sur ce second sujet. Cette méthode de filtrage est appelée le « filtrage collaboratif » (Resnick et Varian, 1997).

Cependant, ce type de modèle est fortement dépendant du système de recommandation en lui-même puisque ce sont les interactions avec celui-ci qui permettent de proposer de nouveaux items à l'utilisateur (e.g. la recommandation de produit sur Amazon, de films sur Netflix). Les utilisateurs étant représentés par leurs interactions observées dans le système, il est difficile de généraliser leur « modèle de profil » à d'autres types de données (recommandation « cross-domaines »), sauf dans les cas où les éléments de chaque domaine peuvent être liés, ou qu'il existe une intersection d'utilisateurs entre domaines (Cremonesi et al., 2011). Le filtrage collaboratif déduit donc les préférences d'un individu à travers ses interactions, mais ne tire pas parti des caractéristiques des items qui peuvent être extraits de leur contenu (Lops et al., 2011). Un autre problème de ce type de méthode est la gestion de nouveaux utilisateurs qui n'ont pas encore interagi avec le système : c'est le problème du « démarrage à froid utilisateur » (ou *user cold-start*) (Son, 2016).

La méthode duale lorsqu'il s'agit de personnaliser les recommandations est celle se basant sur le contenu. Elle consiste à « modéliser » les items par leur contenu et les utilisateurs par leur historique. Il est alors possible de calculer des similarités entre items candidats à une recommandation et ceux présents dans l'historique de l'utilisateur dans l'objectif de décider s'ils doivent être recommandés (Lops et al., 2011). Il est également possible d'inclure des données externes (e.g. issues de réseaux sociaux) dans la représentation des utilisateurs, ou d'autres caractéristiques (e.g. démographiques). Cet enrichissement permet notamment une modélisation des nouveaux utilisateurs et donc de résoudre le problème du démarrage à froid utilisateur. La prise en compte du contenu des items, quant à elle, permet de directement les

3. Le terme « consommer » est générique et désigne, pour un utilisateur utilisant un système de recommandation, le visionnage d'un film, la lecture d'un livre ou encore l'écoute d'un morceau de musique.

inclure parmi les candidats à une recommandation lorsque le filtrage collaboratif nécessite au moins une interaction entre ceux-ci et les utilisateurs. Cette approche permet donc de pallier le problème de « démarrage à froid item » (ou *item cold-start*) (Schein et al., 2002). Le démarrage à froid item est courant lorsque les items les plus pertinents sont ceux les plus récents comme dans la recommandation d'articles d'actualité. En effet, les items récents, étant donné qu'ils n'ont pas encore été « cliqués » ou « partagés » par les utilisateurs (i.e. aucune interaction ne les a encore impliqués), ne peuvent pas être recommandés par les méthodes standards basées sur le filtrage collaboratif. Bien entendu, il existe des algorithmes hybrides qui tentent de combiner les deux approches (i.e. filtrage par le contenu et filtrage collaboratif) et qui obtiennent généralement de meilleurs résultats (Burke, 2002; Danilova et Ponomarev, 2017).

L'inconvénient des méthodes reposant sur le contenu est qu'il est difficile de diversifier les recommandations étant donné que l'on se base sur l'historique de l'utilisateur (Kaminskas et Bridge, 2016). De même, il est difficile de recommander du contenu « nouveau » pour l'utilisateur. La personnalisation par le contenu conduit souvent à une restriction auto-alimentée de la consommation des items par l'utilisateur : un phénomène appelé « bulle de filtres » introduit pour la première fois par Pariser (2011). Le risque est donc de contraindre l'utilisateur à n'interagir qu'avec des items redondants, là où les méthodes de filtrage collaboratif permettent une diversification en tirant parti des intersections de préférences des utilisateurs. La littérature fait aussi référence à la « sérendipité » qui, plus qu'une diversité ou nouveauté de recommandation, consiste à recommander des items inattendus et pertinents (Kotkov et al., 2016).

Enrichissement des profils sur les réseaux sociaux

Sur les réseaux sociaux, nous considérons qu'il existe trois types de données pouvant permettre d'extraire de l'information des profils utilisateurs :

1. les données « statiques » telles que la date de naissance, le genre et toute information démographique, ou encore le texte descriptif (ou *bio*), la localisation (lorsqu'elle n'est pas mise à jour en temps réel) et les préférences exprimées explicitement ;
2. les données « produites » par les utilisateurs comme les messages, les images, les vidéos, ou encore les avis (e.g. mentions « j'aime », notes d'une à cinq étoiles) ;
3. les données « externes » liées aux données produites telles que les articles partagés, les *retweets*, les amis, les profils d'autres utilisateurs suivis.

Les données statiques apportent des informations précises sur l'utilisateur, mais sont cependant limitées. Les deux autres types de données se renouvellent périodiquement (e.g. de nouveaux messages des utilisateurs sont postés chaque jour) et sont plus riches d'informations. Il est courant que les utilisateurs postent des messages courts comme sur le réseau social Twitter. Ces messages peuvent contenir des liens vers des articles d'actualité ou de blog. Les utilisateurs peuvent aussi partager le contenu d'une personne ou d'une organisation qu'ils suivent. Dans cette thèse, nous proposons d'exploiter ce type de données : les données externes, et notamment les articles d'actualité et de blog. Nous pensons que les données externes peuvent permettre

d'enrichir un profil utilisateur en couvrant un large panel d'intérêts lié aux activités personnelles de celui-ci. En effet, les articles d'actualité ou les profils suivis, peuvent être liés, par exemple, aux activités sportives, aux opinions politiques (Meguebli et al., 2017), aux passe-temps et au métier de l'utilisateur. Nous estimons que ce type de données disponibles sur les réseaux sociaux permet un meilleur enrichissement comparé aux données d'autres plateformes telles que les plateformes de vidéo à la demande, de musique, de littérature. En effet, pour ces quelques exemples, la consommation du contenu proposé ne sera pas nécessairement autant en adéquation avec les activités de vie quotidienne de l'utilisateur. Ce type de données peut donc permettre une meilleure généralisation « cross-domaines » des profils utilisateurs pour différents objectifs, autres que la recommandation d'articles d'actualité tels que la recommandation de services, la recommandation de voyages.

Cependant, la plupart des travaux qui cherchent à extraire les intérêts de l'utilisateur à partir des données externes ne se concentrent que sur le fond, c'est-à-dire sur le contenu de ces données tel que les entités mentionnées, les lieux, le thème et plus généralement le vocabulaire employé (Abel et al., 2011; Capelle et al., 2012; Borges et Lorena, 2010). Nous pensons que le fond n'est pas le seul axe jouant un rôle dans les préférences de l'utilisateur et qu'il existe bien plus d'indices subtils pouvant apporter des informations utiles. Le fond s'oppose à la « forme », et lorsque l'on considère le texte écrit (e.g. articles, littérature), la forme est souvent désignée par le « style écrit » (Stamatatos, 2018). Les styles des articles d'actualité et de blog sont divers : e.g. humour, satire, description de faits, interviews, anecdotes. Chaque auteur et chaque source développe son style propre. Un article de blog, lorsque comparé à un article de journal en ligne, par exemple, aura ses propres spécificités stylistiques, même si ceux-ci portent sur le même sujet. Ces spécificités seront par exemple les auto-références et la présence d'anecdotes personnelles de l'auteur. Toutes ces caractéristiques de forme sont autant d'indices susceptibles de jouer un rôle dans les intérêts et les préférences des lecteurs. Par exemple, nous pouvons considérer qu'un utilisateur lisant des articles d'actualité de style « satirique » ou « caricatural » sur des sujets politiques aura des préférences de lecture et des opinions très différentes d'un utilisateur lisant des articles politiques d'un style « factuel », même lorsque ces articles sont, en apparence, du même bord politique lorsque l'on ne considère que la dimension « thématique » par exemple. Ainsi, le style écrit semble être une information essentielle lorsqu'il s'agit de recommander du contenu ou de proposer des services en accord avec les opinions et les préférences de l'utilisateur.

L'entreprise Octopeek

Cette thèse CIFRE est une collaboration entre l'entreprise Octopeek et l'école CentraleSupélec. Octopeek est une entreprise française pionnière dans les domaines de la gestion de données massives (ou *Big Data*), de la science des données et de l'intelligence artificielle. Elle propose un large catalogue de formations dans ces domaines ainsi que des services aux entreprises cherchant à optimiser leur gestion de données. Plus précisément, Octopeek répond aux problématiques actuelles des entreprises en proposant son expertise dans la mise en place de plateformes de gestion de données massives, ainsi que dans l'intégration d'algorithmes d'apprentissage automatique.

À travers les travaux initiés dans cette thèse CIFRE, Octopeek cherche, dans

un premier temps, à systématiser l’enrichissement de données par l’exploitation de ressources externes disponibles sur le web et des données issues des réseaux sociaux (e.g. message posté, contenu partagé), et, dans un second temps, à modéliser les préférences et intérêts de chaque individu afin d’améliorer les services personnalisés fournis par les entreprises.

Problématiques scientifiques

Dans cette thèse, nous proposons d’exploiter les données externes des réseaux sociaux pour l’enrichissement des profils utilisateurs et une série d’expérimentations autour de la tâche de recommandation d’articles d’actualité. Celle-ci nous permet de tester différentes hypothèses que nous détaillons tout au long de ce manuscrit. La recommandation d’articles d’actualité est une tâche particulière dans la mesure où il est « nécessaire » et « bénéfique » de prendre en compte le contenu des items dans le processus de recommandation (Zagheli et al., 2017) lorsque d’autres tâches telles que la recommandation de films peuvent bénéficier du filtrage collaboratif (Subramaniaswamy et al., 2017). La nécessité de prendre en compte le contenu des items dans la recommandation d’articles d’actualité s’explique par le fait que la majorité des articles pertinents à une recommandation sont tous « nouveaux » (i.e. récemment ajoutés dans le système) : c’est le problème du démarrage à froid item (de Souza Pereira Moreira et al., 2018). Les articles d’actualité ont une durée de vie courte en termes de pertinence (Gulla et al., 2016). Il est donc fondamental de tirer parti du contenu de ces items dans le but de recommander ceux les plus pertinents aux utilisateurs. Le bénéfice à l’exploitation du contenu s’explique par le fait que les items sont riches en informations sur le fond (e.g. thème, lieux, entités mentionnées) comme sur la forme (i.e. style écrit). En recommandation, les seuls travaux qui utilisent le style des écrits sont du domaine de la recommandation de livres (Alharthi et al., 2018a). À notre connaissance, aucun travail de recherche n’a encore proposé d’exploiter le style écrit pour la recommandation d’articles d’actualité.

Dans la littérature en traitement automatique du langage naturel, la stylométrie est un ensemble de techniques visant à extraire des informations utiles sur l’auteur d’un texte en analysant son style écrit. De nombreux travaux ont montré qu’il est possible d’identifier l’origine d’un texte (i.e. son auteur ou sa source dans le cas des articles d’actualité) par le style (Neal et al., 2017a; Tausczik et Pennebaker, 2010). D’autres ont montré que le texte, par le langage utilisé et sa structure, est révélateur d’indices de plus haut niveau tels que la personnalité du rédacteur (Mairesse et al., 2007). Généralement, le choix des descripteurs⁴ est motivé par leur performance observée dans les tâches de la stylométrie. Parfois ces descripteurs appartiennent de façon intuitive au périmètre du style écrit comme les « mots fonctions » (Goldstein-Stewart et al., 2009; Menon et Choi, 2011), et parfois ne correspondent qu’à de simples descripteurs du domaine du traitement automatique du langage naturel tels que les *n-grams* de caractères (Escalante et al., 2011; Stamatatos, 2007) ou les représentations distributionnelles de textes (Qian et al., 2017; Gupta et al., 2019; Bagnall, 2015). La plupart de ces travaux reposent donc sur l’implémentation de descripteurs (ou *feature engineering*) par des humains se basant sur des heuristiques

4. Un « descripteur » (ou *feature*) est une caractéristique extraite d’une donnée qui est destinée à être exploitée par des méthodes d’apprentissage automatique.

à partir de leur connaissance, un processus laborieux.

Cependant, des études récentes montrent que l'apprentissage de la représentation des données brutes est bénéfique pour de nombreuses raisons comme le décrit [Bengio et al. \(2013\)](#), et que cette automatisation permet l'extraction de caractéristiques que les humains ne peuvent pas capturer par leurs heuristiques ([Arora et Risteski, 2017](#)). Peu de travaux proposent, par un apprentissage non-supervisé, de projeter des documents dans un espace stylométrique en dimensions réduites ([Ding et al., 2019](#); [Jasper et al., 2018](#); [Boumber et al., 2019](#)), et aucun ne propose de généraliser la représentation de style dans l'objectif de projeter tout « nouveau » document. Or, dans un premier temps, nous pensons que les articles d'actualité et de blog sont une ressource profitable pour deux raisons :

1. leur disponibilité abondante et continue sur le web ;
2. le fait que leur exploitation ne nécessite pas d'annotation manuelle puisque les auteurs, les sources (e.g. journaux en ligne, blogs) et les noms de domaines peuvent être extraits automatiquement.

Et dans un second temps, nous pensons que, via l'assimilation de ces données par les récentes techniques de traitement automatique du langage naturel ([Devlin et al., 2019](#); [Yang et al., 2019](#)), il est possible de construire un espace stylométrique sur la base d'auteurs et de sources de référence, sous la condition que le corpus de documents soit suffisamment large pour permettre une généralisation du style écrit.

Dans ce manuscrit, nous tenterons de répondre aux questions de recherche suivantes :

***QR1** Est-il possible d'utiliser le volume conséquent de données disponibles sur le web pour améliorer l'extraction de caractéristiques stylométriques en exploitant les annotations de type auteur et source (e.g. journal en ligne, blog) ?*

***QR2** Peut-on entraîner un modèle de représentation à projeter un article d'actualité ou de blog dans un espace stylométrique général ?*

Afin de répondre à ces deux questions, nous proposons tout d'abord de construire un large corpus de référence stylistique. Nous introduisons deux propriétés inspirées de [Karlsgren \(2004\)](#) et [Holmes \(1998\)](#) qui permettent d'identifier les indices stylométriques de manière distributionnelle : la *consistance intra-auteur* et la *non-spécificité sémantique*. Enfin, nous proposons une méthode d'apprentissage de la représentation exploitant des réseaux de neurones profonds et le corpus de référence stylistique.

Comme développé précédemment, le style écrit, dans les articles d'actualité et de blog, peut jouer un rôle dans les préférences des utilisateurs et dans la modélisation de leur « profil ». Dans les tâches telles que la recommandation d'articles d'actualité, il est difficile d'optimiser à la fois la pertinence et la qualité de ces recommandations en termes de diversité, de nouveauté et de sérendipité ([Ribeiro et al., 2012](#)). Or, dans cette thèse, nous supposons que le style écrit, apportant une dimension supplémentaire et « complémentaire » dans la représentation des lecteurs, permet d'obtenir un bon compromis entre ces deux axes. Ainsi, nous répondrons aux questions de recherche suivantes :

***QR3** Les utilisateurs ont-ils des préférences stylistiques dans leur lecture de l'actualité et donc est-il possible d'améliorer la recommandation d'articles d'actualité en exploitant la similarité stylométrique ?*

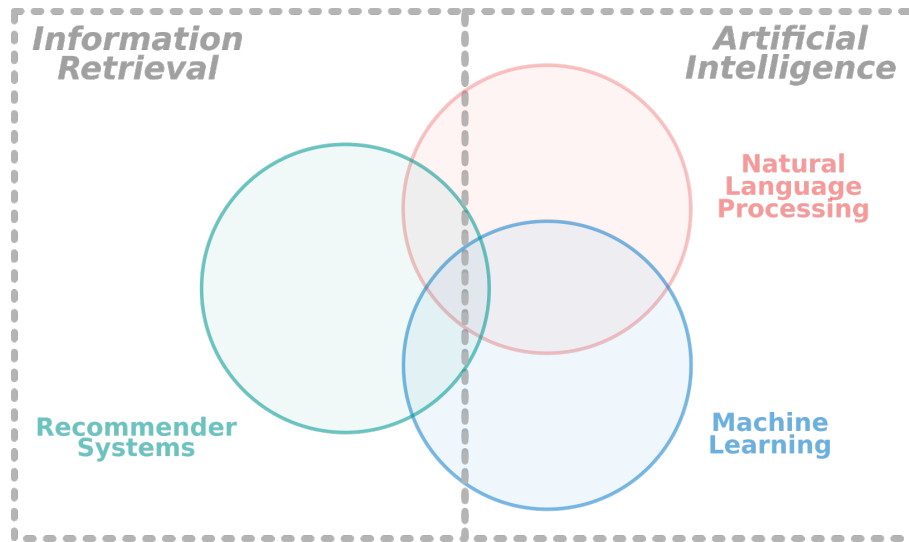


FIGURE 1 – Domaines de recherche de cette thèse

QR4 *Les caractéristiques de style sont-elles complémentaires avec les caractéristiques textuelles standards employées en recommandation d’articles d’actualité ?*

QR5 *Cette complémentarité permet-elle une amélioration qualitative des recommandations en termes de diversité, nouveauté et sérendipité ?*

Dans le but de répondre à ces questions, nous proposons le jeu de données « *TwineWS* » spécialement récolté pour les expérimentations de cette thèse. Ce jeu de données se compose d’approximativement 20 000 utilisateurs Twitter et de 300 000 articles d’actualité partagés dans les *tweets* des utilisateurs. Nous proposons un algorithme de recommandation basé sur la notion de *proximité partielle* à l’historique utilisateur et une méthode d’hybridation basée sur la pondération et le rééchelonnage des ordonnancements. Cette méthode d’hybridation nous permet de tester la qualité de la recommandation par le style lorsque cette représentation des items est combinée à des représentations standards.

En définitive, cette thèse propose d’exploiter les techniques de sémantique distributionnelle pour l’amélioration d’algorithmes de recommandation d’articles d’actualité. Elle se situe par conséquent à l’intersection des domaines du traitement automatique du langage naturel, de l’apprentissage automatique et des systèmes de recommandation comme l’illustre la figure 1.

Contributions

Dans cette thèse, nous proposons une méthode d’apprentissage de la représentation du style écrit se basant sur l’hypothèse qu’il est possible de généraliser le style à partir d’auteurs de références. Cette méthode permet de pallier les problèmes liés à l’implémentation de descripteurs (ou *feature engineering*) utilisée dans le domaine de la stylométrie et aux limites expérimentales (e.g. généralisation, quantité de données d’entraînement et de test) identifiées dans l’état de l’art de ce domaine. Pour cela, nous exploitons une ressource abondante : les articles d’actualité et de blog

disponibles sur le web. Cette contribution s’accompagne d’une nouvelle définition du style écrit introduisant des propriétés statistiques, mais aussi d’une nouvelle méthodologie d’évaluation incluant une métrique de partitionnement dite « interne » (ou *internal evaluation metric*) appelée *SimRank* et d’une nouvelle mesure appelée *TFIDF focus* permettant d’évaluer la capacité de nos modèles à capturer des indices textuels en adéquation avec l’une des propriétés du style écrit.

La deuxième contribution de cette thèse est la proposition d’une méthode de construction et de filtrage de corpus de référence. La méthode proposée se base sur la pondération TFIDF ainsi que de l’algorithme *DocDist* ayant de bonnes propriétés de scalabilité et permettant l’élimination exhaustive d’indices révélateurs des auteurs de référence dans le corpus. Cette méthode est motivée par l’*hypothèse de filtrage* stipulant que le filtrage du corpus de référence permet une amélioration des représentations stylométriques produites. Nous décrivons cette hypothèse au chapitre 2 et la méthode de filtrage au chapitre 3.

La troisième contribution est l’application de cette méthode au domaine de la recommandation d’articles d’actualité. Nous proposons un nouvel algorithme d’hybridation, appelé *Reswhy*, permettant de combiner des représentations vectorielles pour la recommandation basée sur le contenu. Ainsi, nous évaluons le potentiel des représentations stylométriques dans la recommandation d’articles d’actualité et montrons que le style écrit joue un rôle dans les préférences de lecture des utilisateurs. Dans cette thèse, nous développons différentes idées expliquant la cause des biais inhérents à l’évaluation des systèmes de recommandation en *offline*, mais également des biais d’évaluation *online* liés aux retours de pertinence tels que le taux de clics. Ces réflexions nous ont amenés à proposer une plateforme dédiée à l’organisation de compétitions consistant en l’évaluation *online* des systèmes de recommandation d’articles d’actualité. Cette plateforme, appelée *Renewal*, est la dernière contribution de cette thèse. Elle permet la résolution de biais mentionnés dans cette thèse et un allègement des contraintes imposées aux systèmes de recommandation dans les plateformes existantes.

Publications

Les publications résultant des travaux de recherche réalisés durant cette thèse sont les suivantes :

Hay et al. (2020c) *Representation learning of writing style*

Hay et al. (2020a) *Filtering a reference corpus to generalize stylometric representations*

Hay et al. (2020b) *Renewal: an online competition platform for news recommender systems*

Notre travail sur le projet *Renewal* a également été présenté (à travers un poster) en juin 2018 à l’école d’été DATAIA DS3, et en décembre 2018 à la conférence NeuIPS dans le *workshop* CiML (*Challenges in Machine Learning*).

Durant cette thèse, suite à un travail antérieur, nous avons également rédigé et publié deux articles portant sur la complémentarité de représentations sémantiques. Le premier article est un article de conférence intitulé « *Complémentarités de représentations lexicales pour la similarité sémantique* » (Hay et al., 2018) et le second un article de journal intitulé « *Automatically Selecting Complementary Vector*

Representations for Semantic Textual Similarity » (Hay et al., 2019). Cependant, la description de ce travail n’a pas été incluse dans ce manuscrit.

Enfin, nous projetons, par la suite, de publier le travail effectué dans les chapitres 6, 7 et 8 portant sur l’exploitation du style écrit dans la recommandation d’articles d’actualité.

Plan de thèse

Cette thèse est constituée de deux parties. La première partie rapporte le travail réalisé sur l’apprentissage de la représentation du style (domaine du traitement automatique du langage naturel) et la seconde rapporte le travail réalisé sur la recommandation d’articles d’actualité (domaine des systèmes de recommandation).

En première partie, le chapitre 1 est consacré à l’état de l’art du domaine du traitement automatique du langage naturel et de la stylométrie. Pour commencer, après un court aperçu du traitement automatique du langage naturel, nous détaillons quelques méthodes de ce domaine avec, notamment, les techniques de sémantique distributionnelle appliquées à la stylométrie. Dans les chapitres 2, 3 et 4, nous proposons une nouvelle méthode d’apprentissage de la représentation du style écrit visant à projeter des articles d’actualité et de blog dans un espace stylométrique de référence. Nous donnons une nouvelle définition du style au chapitre 2 sur la base de propriétés statistiques. Nous décrivons également la méthode d’apprentissage de la représentation proposée et le cadre d’évaluation utilisé. Nous décrivons la construction du jeu de données *NewsID* constituant un corpus de référence stylistique dans le chapitre 3. Enfin, dans le chapitre 4, nous présentons les résultats des expérimentations sur le partitionnement par auteur et l’identification d’auteurs validant l’intérêt de la méthode d’apprentissage de la représentation. Nous proposons également, dans ce chapitre, de tester l’impact du filtrage du corpus de référence sur la performance des modèles.

Dans la seconde partie, le chapitre 5 est consacré à l’état de l’art du domaine des systèmes de recommandation. Nous donnons un aperçu de ce domaine et détaillons les méthodes employées dans la recommandation d’articles d’actualité ainsi que les spécificités de cette tâche. Dans les chapitres 6, 7 et 8, nous étudions l’intérêt des représentations stylométriques sur la tâche de recommandation d’articles d’actualité. Pour cela, au chapitre 6, nous proposons deux algorithmes génériques de recommandation permettant de comparer et combiner des modèles de représentation d’items. Au chapitre 7, nous décrivons les particularités ainsi que la collecte du jeu de données *Twinews*. Nous discutons également des biais inhérents à l’évaluation *offline* sur ce type de jeu de données. Le chapitre 8 sera dédié à l’exploitation du style dans la recommandation d’articles d’actualité. Nous proposons pour cela la combinaison de différentes caractéristiques extraites du texte, dont les caractéristiques de style, afin de générer des recommandations d’articles. Nous testons la précision de ces recommandations ainsi que leur qualité grâce à différentes mesures telles que la diversité et la nouveauté.

Dans le chapitre 9, nous décrivons la plateforme *Renewal* dédiée à l’organisation de compétitions en conditions réelles pour la recommandation d’articles d’actualité. Le développement de ce projet a débuté durant cette thèse et a été motivé par les différentes problématiques que nous avons identifiées dans nos recherches, notamment concernant les biais inhérents aux évaluations « *offline* » des systèmes de recomman-

dation. Cette plateforme vise à évaluer dans un cadre « *online* » des algorithmes de recommandation par l'interaction temps réel d'utilisateurs sur une application mobile. Elle permettra, dans un premier temps, la confirmation de l'intérêt des algorithmes proposés dans cette thèse en termes de performances prédictives ainsi qu'en termes qualitatifs (e.g. diversité, nouveauté), mais aussi, dans un second temps, l'organisation de compétitions destinées à la communauté de recherche du domaine des systèmes de recommandation. Nous décrivons les contributions de cette nouvelle plateforme au regard des plateformes existantes ainsi que de son état d'avancement. Enfin, nous terminons ce manuscrit par une conclusion récapitulant les contributions de cette thèse et ses perspectives pour de futures recherches.

Première partie

Apprentissage de la représentation du style

Chapitre 1

État de l’art des méthodes de représentation du texte et de la stylométrie

1.1 Introduction

Dans ce chapitre, nous détaillons l’état de l’art des travaux réalisés durant cette thèse. La section 1.2 présente différentes méthodes de représentation vectorielle de documents issues du domaine du traitement automatique du langage naturel. Nous introduisons des modèles récents de la sémantique distributionnelle basés sur des réseaux de neurones profonds tels que les *transformers* (Vaswani et al., 2017). La section 1.3 est dédiée aux méthodes utilisées dans le domaine de la stylométrie, et notamment les méthodes présentées en section 1.2 appliquées à ce domaine. Nous détaillons les limites identifiées dans l’état de ce domaine, notamment concernant l’apprentissage de la représentation du style qui peut pallier les problèmes de l’implémentation de descripteurs (ou *feature engineering*). Nous reviendrons également, dans cette section, sur les questions de recherche présentées en introduction. Ces deux sections introduisent toutes les notions nécessaires à la lecture des chapitres 2, 3 et 4.

1.2 Représentation du texte

Dans cette section, nous décrivons les différentes étapes visant à construire la représentation d’un document. En informatique, un document peut-être défini comme une séquence finie (de taille supérieure à 0) de lettres (e.g. lettres alphanumériques, ponctuations). Un corpus est un ensemble de documents. La construction de la « représentation » d’un document consiste à le convertir d’une forme « compréhensible » par un humain en une forme « traitable » par un algorithme. Il s’agit de projeter les documents dans un espace vectoriel commun de sorte que l’algorithme puisse bénéficier des similarités entre documents. Deux documents seront alors représentés par des vecteurs de tailles égales et auront chacun leurs coordonnées dans cet espace vectoriel. Cette « projection » permet de retranscrire numériquement l’information que les humains assimilent naturellement à la lecture des documents.

Les algorithmes d'apprentissage automatique¹ peuvent alors, par l'utilisation de ces projections, résoudre des problèmes du traitement automatique du langage naturel (e.g. prédire la polarité en sentiment d'un document, prédire le thème d'un document). Généralement, plus les représentations retranscrivent les informations utiles du texte, plus les algorithmes d'apprentissage automatique sont performants dans la résolution de ces problèmes. Ces algorithmes, s'ils sont capables d'identifier des régularités dans la représentation des documents, peuvent alors, par apprentissage, associer des concepts à ces régularités. Ils peuvent, par exemple, apprendre que la présence du mot « *bon* » dans un document est associé à une polarité « positive », et ainsi généraliser en attribuant une polarité « positive » à tout nouveau document comprenant ce mot. Cependant, afin d'utiliser des algorithmes d'apprentissage automatique, il est nécessaire de pouvoir représenter un document de sorte à indiquer la présence des mots.

1.2.1 Représentation vectorielle du texte

Tokenisation et construction du vocabulaire

La première étape de la construction de la représentation d'un document est de le découper en plusieurs éléments. Une partie des éléments d'un document peut être commune à d'autres documents. Il s'agit alors d'exploiter ces ressemblances pour les projeter dans un espace vectoriel commun. Par défaut, un document est découpé en lettres. Cependant, afin que chaque élément qui compose un document corresponde à une unité sémantique de la langue, il est courant d'effectuer un découpage en mots. Comme nous le verrons dans les sections suivantes, il peut aussi être pertinent de découper les documents en *wordpieces* ou en *n-grams* de caractères. Les éléments considérés sont appelés des *tokens*, et le processus de découpage est appelé la *tokenisation*. Par exemple, dans la bibliothèque *NLTK*², les algorithmes de découpage des documents en mots s'appuient sur des expressions régulières³.

Notons $D = \{d_1, \dots, d_p\}$ un ensemble de documents (ou « corpus ») avec p le nombre de documents. Chaque document d_i est composé d'une suite ordonnée de *tokens* : $d_i = (w_1, \dots, w_{m_i})$ avec m_i la taille du document d_i . Chaque document a une taille finie supérieure à 0 : i.e. $\forall d \in D, |d| > 0$. L'étape suivante, après le découpage des documents en *tokens*, est la construction du vocabulaire à partir des documents d'un corpus découpés en *tokens*. Le vocabulaire $V = \{w_1, \dots, w_l\}$ du corpus D est l'ensemble des *tokens* des documents de D : $\forall d \in D, \forall w \in d, w \in V$ et $\forall w \in V, \exists d \in D$ s.t. $w \in d$.

Le vocabulaire ainsi construit est généralement large étant donné la présence de mots peu fréquents et de variations orthographiques (e.g. conjugaison des verbes, forme plurielle des noms). De même, certains mots ne sont parfois pas nécessaires à la représentation des documents pour une tâche particulière (e.g. ponctuations). Afin de réduire la taille du vocabulaire et de ne conserver que les éléments pertinents, il est possible de :

1. L'apprentissage automatique est un champ d'étude de l'intelligence artificielle donnant aux machines la capacité d'apprendre à partir de données par l'emploi de méthodes statistiques.

2. *NLTK (Natural Language Toolkit)* est une bibliothèque de traitement automatique du langage naturel disponible dans le langage *Python*.

3. Une expression régulière est une chaîne de caractères, qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles.

- Supprimer les éléments n'appartenant pas à une liste prédéfinie (liste de mots vides⁴, liste de ponctuations).
- Supprimer un certain nombre de mots parmi les plus fréquents⁵. Il peut aussi s'agir de supprimer une certaine proportion de ceux-ci.
- Supprimer un certain nombre de mots (ou une proportion) parmi les moins fréquents.
- Réduire les lettres majuscules en lettres minuscules. Ainsi, un mot, s'il est écrit en début de phrase ou non, correspondra au même élément dans le vocabulaire.
- Lemmatiser les mots, c'est-à-dire convertir différents lexèmes⁶ ayant une racine commune en leur forme canonique (e.g. « *chercha* » devient « *chercher* »). Il est aussi possible d'effectuer un *stemming*, c'est-à-dire un découpage des mots selon une heuristique définie à l'avance (e.g. « *chercha* » devient « *cherch* »).

La représentation sac de mots

La représentation d'un document est dépendante des *tokens* qui le composent ainsi que du vocabulaire considéré. Une des méthodes de représentation les plus courantes est la méthode « sac de mots ». Il s'agit de construire un vecteur v de taille $|V|$, avec pour chaque dimension, la valeur 1 si $V_i \in d$ et la valeur 0 si $V_i \notin d$. Une variante de cette représentation est de considérer non pas la présence mais le compte du *token* dans le document, i.e. une valeur entière supérieure à 0. La variante la plus utilisée est la pondération TFIDF. Il s'agit de prendre en compte à la fois la fréquence du *token* dans le document ainsi que l'inverse du nombre de fois que le *token* apparaît dans les documents du corpus (i.e. la *document frequency*). La représentation TFIDF d'un document d correspond à un vecteur de taille $|V|$, chaque dimension i de ce vecteur prenant la valeur :

$$\text{TFIDF}(d, i) = \text{tf}(V_i, d) \cdot \left(\log\left(\frac{1 + |D|}{1 + \text{df}(V_i, D)}\right) + 1 \right) \quad (1.1)$$

avec tf la fonction renvoyant la fréquence d'un *token* dans un document et df la fonction renvoyant le nombre de fois que le *token* apparaît dans les documents du corpus. Il existe plusieurs variantes de TFIDF, cette variante est celle retenue dans la bibliothèque *Scikit-learn*⁷ (Buitinck et al., 2013). Cette pondération permet de donner de l'importance aux mots du document qui sont fréquents dans celui-ci, mais rares dans le corpus, c'est-à-dire ceux qui rendent le document distinguable.

La représentation « sac de mots » consiste donc à représenter un document en indiquant quels sont les mots qui le composent. Cependant, elle ne prend pas en compte l'ordre des mots. Ainsi, les deux phrases ci-dessous, après suppression des mots vides et lemmatisation, seront représentées par le même vecteur sac de mots :

4. Une liste de mots vides (ou *stop word list*) est une liste des mots considérés comme les plus communs dans un langage. Un mot vide (e.g. « *un* », « *le* », « *la* ») ne permet pas de distinguer les textes les uns par rapport aux autres, c'est-à-dire qu'ils ont une fréquence semblable dans chacun des textes de la collection. Il n'existe pas de liste universelle mais différentes listes de tailles plus ou moins importantes.

5. La fréquence d'un mot est généralement soit définie par le compte de celui-ci dans le corpus, soit définie par le nombre de fois qu'il apparaît dans un document (i.e. la *document frequency*).

6. Un lexème est une unité de sens lexical qui est sous-jacente à un ensemble de mots reliés par inflexion.

7. *Scikit-learn* est une bibliothèque libre *Python* destinée à l'apprentissage automatique.

*À Paris, le président accueille le journal local.
Le président du journal local est accueilli à Paris.*

Pour pallier ce problème, il est possible de considérer l'ordre de mots dans le vocabulaire en faisant une extraction des *n-grams* avec n indiquant la longueur des suites de mots. Dans le vocabulaire des *2-grams* construit à partir d'un corpus composé des deux exemples ci-dessus, nous trouverons, par exemple, les éléments « *président du* », « *président accueille* » et « *journal local* ». Ainsi, il est possible de représenter l'information spatiale indiquant que le verbe « *accueillir* » réfère au sujet « *président* ». Un vocabulaire de *n-grams* avec $n > 1$ permet la représentation d'informations spatiales, et donc de prendre en compte l'ordre des mots dans les phrases. À noter qu'il est aussi possible d'appliquer la pondération TFIDF à un vocabulaire de *n-grams* avec $n > 1$.

La représentation sémantique

L'inconvénient de la représentation sac de mots est qu'elle considère chaque élément du vocabulaire comme étant de distance égale. Il n'est pas possible d'exploiter les relations entre les mots (e.g. synonymie, hyponymie) dans la représentation des documents. Par exemple, la représentation sac de mots de ces deux documents auront une distance maximale (i.e. la plus grande possible) puisqu'ils n'ont aucun mot en commun :

*À Paris, le président accueille le journal local.
Macron est interrogé par la presse de la capitale.*

Ces deux phrases ont cependant un sens proche, et représenter ce sens peut permettre une meilleure généralisation des connaissances acquises par un algorithme d'apprentissage automatique.

Imaginons un algorithme entraîné à prédire la polarité de documents. Cet algorithme a appris, par expérience, que le terme « *bon* » est souvent associé à une polarité positive. Imaginons que cet algorithme doit prédire la polarité d'un document comportant le mot « *fabuleux* ». Ce mot n'est présent que dans très peu de documents du corpus, l'algorithme n'a donc associé aucune polarité à ce mot. La représentation sac de mots n'indiquera pas que les mots « *bon* » et « *fabuleux* » sont sémantiquement proches. L'algorithme ne généralisera donc pas en associant également une polarité positive à l'emploi du terme « *fabuleux* ».

La sémantique distributionnelle est un domaine de recherche visant à développer des méthodes permettant de pallier ce problème. Les méthodes de représentation de la sémantique distributionnelle se fondent sur l'hypothèse avancée par Harris (1954), que les mots qui apparaissent souvent dans les mêmes contextes⁸ ont tendance à avoir un sens proche. Par exemple, dans un corpus donné, le mot « *voiture* » est souvent dans les mêmes phrases que le mot « *route* ». Le mot « *vélo* » est aussi souvent dans les mêmes phrases que le mot « *route* ». Selon la sémantique distributionnelle, cette observation indique que les mots *voiture* et *vélo* sont sémantiquement proches.

Les modèles thématiques (ou *topic models*), tels que l'analyse sémantique latente (ou *Latent Semantic Analysis*, abrégé LSA) et l'allocation de Dirichlet latente (ou

8. Le « contexte », dans le cadre de la sémantique distributionnelle, désigne généralement les termes voisins dans une phrase ou présents dans le même document.

Latent Dirichlet Allocation, abrégé LDA), utilisent ces informations dites de « co-occurrence » des mots à l'échelle du document. La méthode LSA consiste tout d'abord à construire une matrice de co-occurrences entre mots et documents (i.e. chaque valeur de la matrice correspond au compte d'un mot dans un document). Les mots correspondent aux lignes et les documents aux colonnes. Ensuite, une méthode de réduction de dimensions, la décomposition en valeurs singulières (ou *Singular Value Decomposition*, abrégé SVD), est utilisée afin de réduire le nombre de lignes. Cette méthode permet ainsi de représenter les documents en faible dimension par rapport au vocabulaire utilisé dans la représentation sac de mots. La méthode LDA, quant à elle, associe un ensemble de thèmes à chaque document selon différentes probabilités. Les thèmes sont attribués aléatoirement à chaque mot dans les documents selon une distribution de Dirichlet, puis mis à jour par un processus itératif. Tout comme LSA, cette méthode permet de représenter les documents selon une dimension « thématique ». Dans cette section, nous détaillons une méthode plus récente de la sémantique distributionnelle, Doc2Vec (Le et Mikolov, 2014), qui permet une meilleure représentation « sémantique » des documents comme nous le verrons dans les différents chapitres de ce manuscrit et comme démontré par Le et Mikolov (2014). Cette méthode repose sur une architecture de réseau de neurones et exploite les co-occurrences de mots à l'échelle de la phrase. L'avantage de Doc2Vec est qu'il permet l'entraînement d'un modèle de représentation des documents sur de grands corpus, ce qui permet une meilleure généralisation des représentations apprises. De plus, il prend en compte l'ordre des mots grâce à l'utilisation d'un contexte représenté par une fenêtre glissante sur les documents. Ainsi, deux documents comportant les mêmes mots mais dans un ordre différent (donc ayant un « sens » différent), seront représentés par deux vecteurs différents. Doc2Vec est une extension de Word2Vec, nous commençons donc par introduire ce dernier.

L'architecture du réseau de neurones Word2Vec, dans sa version CBOW (ou *continuous bag-of-words*), est donnée par la figure 1.1 et correspond à un réseau de neurones ayant trois couches. L'objectif de Word2Vec est d'obtenir une représentation vectorielle en basses dimensions des mots appartenant au vocabulaire d'un corpus donné. Ce réseau de neurones est entraîné à prédire un mot cible en fonction de plusieurs mots contexte donnés en entrée. Les mots contexte sont déterminés par une fenêtre glissante dans chaque phrase de chaque document d'un corpus. Le centre de cette fenêtre est le mot cible. Par exemple, une fenêtre de taille 3 contiendra 2 mots contexte et un mot cible. Une fenêtre de taille 5 contiendra 4 mots contexte et un mot cible. Chaque vecteur en entrée et sortie du réseau de neurones est un vecteur dit *one-hot*⁹ et correspond aux mots présents dans une fenêtre de contexte ainsi que le mot cible.

Les paramètres de ce réseau de neurones sont les matrices W pour la couche cachée et W' pour la couche de sortie. L'entrée du réseau de neurones correspond à la moyenne des vecteurs *one-hot* de tous les mots contexte. La sortie correspond au vecteur *one-hot* du mot cible. La fonction d'activation de la dernière couche est la fonction *softmax*. Les valeurs en sortie sont données par les équations suivantes :

$$u = W'^T W^T \bar{x} \quad (1.2)$$

9. Un vecteur *one-hot* est un vecteur composé de zéros, à l'exception d'une dimension ayant la valeur 1. La position de la valeur 1 indique une « catégorie » parmi un nombre de catégories égal à la taille du vecteur. Par exemple, dans le cas de Word2Vec, les catégories sont les mots appartenant au vocabulaire.

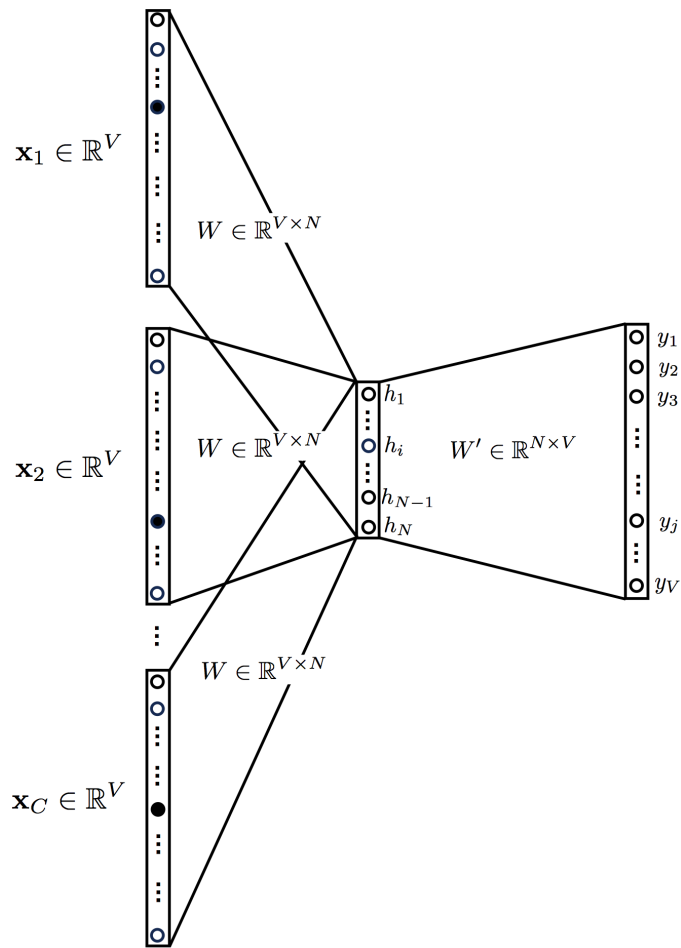


FIGURE 1.1 – Architecture de la version CBOW du réseau de neurones Word2Vec (Mikolov et al., 2013)

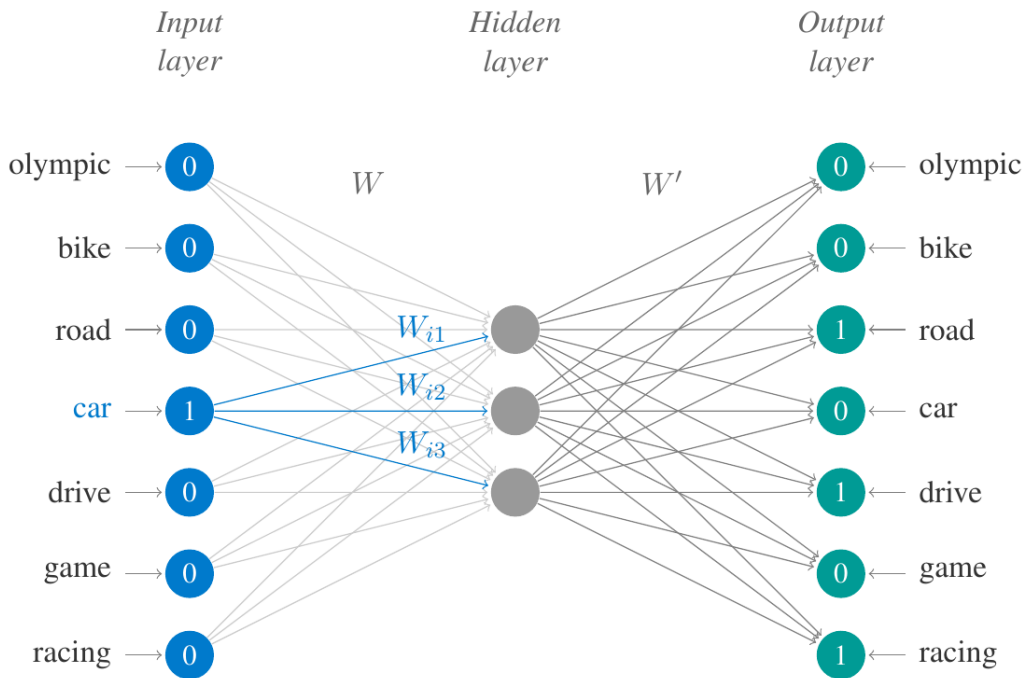


FIGURE 1.2 – Illustration de la variante Skip-gram de Word2Vec avec le mot cible « car », donné en entrée, ayant les mots contexte « road », « drive » et « racing ».

$$y = \text{softmax}(u) \quad (1.3)$$

Avec u les valeurs avant l'application de la fonction *softmax*. La variable \bar{x} est la moyenne des vecteurs *one-hot* des mots contexte. La fonction de perte s'appliquant à la dimension j^* du mot cible est la suivante :

$$\mathcal{L} = -u_{j^*} + \log \sum_i \exp(u_i) \quad (1.4)$$

Lors de la phase d'apprentissage, plusieurs exemples (mots contexte et mot cible) sont générés en *batches*¹⁰ et une rétropropagation du gradient permet de régler les paramètres du réseau de neurones. Afin d'éviter une mise à jour coûteuse du réseau de neurones à chaque itération due à l'utilisation de la fonction *softmax*, Mikolov et al. (2013) proposent d'utiliser la méthode du *negative sampling*. Cette méthode consiste à choisir aléatoirement un nombre limité de mots qui ne correspondent pas au mot cible, dits « négatifs », et de mettre à jour uniquement les poids correspondants dans le réseau de neurones par rétropropagation du gradient.

Afin d'obtenir la représentation vectorielle en basses dimensions d'un mot, il suffit, après entraînement du réseau de neurones sur un corpus, de donner en entrée le vecteur *one-hot* du mot et de prendre les valeurs en sortie de la couche cachée qui est en basse dimension. Lors de l'entraînement du réseau de neurones, les hyperparamètres les plus importants à choisir sont la taille de la couche cachée (généralement entre

10. En apprentissage automatique, et notamment lors de l'entraînement de réseaux de neurones, un *batch* est un ensemble d'exemples. L'entraînement par *batches* permet de prendre en compte plusieurs exemples lors de l'application de la rétropropagation du gradient afin d'obtenir une convergence plus rapide du réseau de neurones.

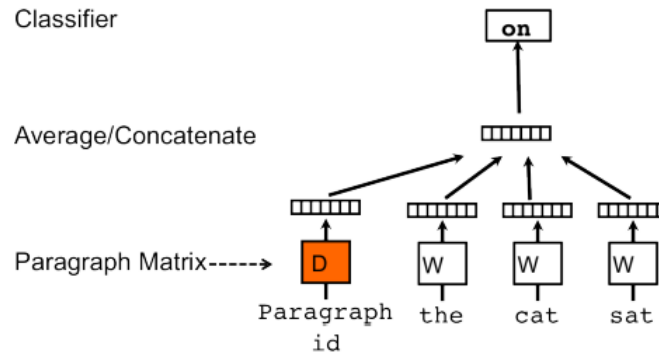


FIGURE 1.3 – Architecture de la version PV-DM du réseau de neurones Doc2Vec (Le et Mikolov, 2014)

100 et 1000) qui déterminera la taille des représentations vectorielles des mots, ainsi que la taille de la fenêtre glissante. Afin de régler la taille de la fenêtre glissante, notamment dans l’implémentation proposée dans *Gensim*¹¹ (Řehůřek et Sojka, 2010), il est nécessaire d’affecter une valeur au paramètre w (généralement entre 1 et 8). La taille réelle de la fenêtre glissante est obtenue en calculant $2w + 1$. Par exemple, $w = 2$ correspond à une fenêtre glissante de 5. La variante « Skip-gram » de Word2Vec consiste à entraîner le réseau de neurones à prédire les mots contexte à partir du mot cible donné en entrée. La figure 1.2 illustre le fonctionnement de la variante Skip-gram. La représentation du mot ayant l’indice i , i.e. « *car* », consiste à prendre les poids W_i du réseau de neurones.

Le modèle Doc2Vec, dans sa version la plus utilisée, *Distributed Memory Model of Paragraph Vectors* (abrégée PV-DM), consiste à ajouter des vecteurs *one-hot* uniques pour chaque document du corpus en entrée du réseau de neurones lors de l’entraînement du modèle Word2Vec dans sa version CBOW. La figure 1.3 illustre l’architecture Doc2Vec dans sa version PV-DM. Lors de l’entraînement du modèle, le vecteur *one-hot* du document courant est ajouté en entrée en plus des mots contexte. Pour un document, le réseau de neurones apprend à prédire chaque mot cible considéré dans le document ainsi qu’un vecteur de document unique. Tout comme pour le modèle Word2Vec CBOW, la moyenne des vecteurs *one-hot* est calculée pour l’entrée du réseau de neurones. Ainsi, à la fin du processus d’entraînement, chaque document et chaque mot du vocabulaire considéré est représenté sous la forme d’un vecteur sémantique. Pour obtenir la représentation d’un document inconnu, i.e. qui n’était pas présent dans le corpus d’entraînement, il est possible d’ajouter une colonne dans la matrice W , d’effectuer un entraînement du réseau de neurones sur le document nouveau sans modifier les paramètres correspondant aux mots et documents connus, et enfin, de prendre les poids inférés pour le document nouveau.

À noter que la similarité cosinus est la plus utilisée lorsqu’il s’agit de calculer la similarité sémantique des représentations vectorielles de deux mots (ou de deux

11. *Gensim*, disponible à l’adresse <https://radimrehurek.com/gensim>, est une bibliothèque Python implémentant des méthodes de la sémantique distributionnelle.

documents) A et B :

$$\text{cosine-sim}(A, B) = \frac{\sum_{i=1}^{|A|} A_i B_i}{\sqrt{\sum_{i=1}^{|A|} A_i^2} \sqrt{\sum_{i=1}^{|B|} B_i^2}} \quad (1.5)$$

1.2.2 Approches basées sur les réseaux de neurones profonds

La performance des algorithmes d'apprentissage automatique est dépendante des représentations vectorielles des documents. Plus celles-ci permettent de retranscrire les informations utiles à une tâche (e.g. le thème pour une classification de documents), plus les algorithmes seront performants dans cette tâche. Dans le traitement automatique du langage naturel, l'implémentation de descripteurs (ou *feature engineering*) consiste à convertir les données textuelles en une forme exploitable par les algorithmes d'apprentissage automatique. Le processus est souvent laborieux car demande aux humains de transférer leurs connaissances dans la construction de ressources permettant de générer la représentation des documents. Ces ressources peuvent, par exemple, correspondre à des lexiques tels que LIWC¹² (Pennebaker et al., 2001), ou encore la base de données WordNet¹³. Un autre exemple de ressource est la liste de mots vides, souvent créée manuellement et utilisée notamment dans la représentation sac de mots. L'implémentation de descripteurs (ou *feature engineering*) peut aussi consister en l'implémentation d'algorithmes tels que l'aligneur de texte proposé par Sultan et al. (2015) pour le calcul de similarités sémantiques. L'implémentation de ce type d'algorithme nécessite une connaissance a priori sur le langage.

Au lieu de cela, de récentes études ont montré qu'il est possible d'apprendre automatiquement à représenter des données brutes telles que des images et du texte (Liu et al., 2020). Comme décrit par Liu et al. (2020), ces représentations apprises sont utiles dans beaucoup de tâches de classification et de partitionnement (ou *clustering*). En effet, celles-ci permettent d'automatiquement sélectionner les informations discriminantes dans les données brutes et de les projeter dans un espace à basses dimensions (Arora et Risteski, 2017). De plus, ces informations discriminantes ne sont pas nécessairement capturées par les humains par l'implémentation de descripteurs.

Doc2Vec, que nous avons introduit en section 1.2.1, est un exemple de modèle d'apprentissage de la représentation. Ce réseau de neurones apprend à représenter les documents d'un corpus en réglant ses paramètres internes. Il permet également d'extraire la représentation de documents inconnus par inférence. Les réseaux de neurones profonds de type récurrent (abrégiés RNN), tels que les *Long Short-Term Memory* (abrégiés LSTM) (Hochreiter et Schmidhuber, 1997) et les *Gated Recurrent*

12. LIWC (*Linguistic Inquiry and Word Count*) est une ressource lexicale rassemblant différents termes dans des catégories linguistiques, psychologiques et thématiques.

13. La base de données WordNet a pour but de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise.

Units (abrégés GRU) (Cho et al., 2014), sont aussi capables d'apprendre la représentation de documents par apprentissage supervisé sur des tâches spécifiques telles que la traduction automatique. Il s'agit généralement d'exploiter des représentations de mots apprises de manière non supervisée (e.g. via Word2Vec) grâce à l'apprentissage par transfert¹⁴.

Ces réseaux de neurones profonds prennent en entrée des séquences de *tokens* et conservent une « mémoire » interne de contexte représentant les *tokens* déjà vus. Extraire la « mémoire » du réseau de neurones profond après avoir donné en entrée tous les *tokens* d'un document permet d'obtenir la représentation de celui-ci. Kiros et al. (2015) ont par exemple proposé d'apprendre la représentation de phrases grâce à des architectures de type GRU. À noter que la représentation des phrases est tout aussi essentielle en traitement automatique du langage naturel puisque la plupart des tâches consistent à comprendre le sens des phrases : e.g. traduction automatique, génération automatique de résumé, analyse de sentiment.

Suite aux travaux portant sur les réseaux de neurones récurrents, des réseaux de neurones profonds basés sur l'architecture « *transformer* » représentant le texte à l'échelle de la phrase ont permis d'atteindre des performances inégalées sur différentes tâches du traitement automatique du langage naturel (e.g. question réponse, traduction automatique, identification d'entités nommées). Comme nous le verrons dans les prochains chapitres, ce type de modèle peut être étendu à l'échelle du document. L'avantage des *transformers* est qu'ils ne perdent pas d'information de dépendance dans le cas de longues séquences de *tokens* étant donné que tous les mots sont traités simultanément par le réseau de neurones profond. Cette simultanéité permet aussi de paralléliser le processus de propagation des données en entrée et de rétropropagation du gradient. L'entraînement du réseau de neurones est, en conséquence, plus rapide. Les couches d'« attention » ainsi que les informations de positions ajoutées aux représentations des *tokens* permettent de modéliser les relations entre *tokens* dans une phrase. Ces informations sont, dans le cas des réseaux de neurones récurrents, encodés dans leur « mémoire ».

Les *transformers* sont des réseaux de neurones profonds de type *encoder-decoder* qui consistent en une série de blocs basés sur l'attention (Vaswani et al., 2017). Le mécanisme d'attention, à l'origine proposé par Bahdanau et al. (2015), permet la sélection automatique des parties d'un texte les plus pertinentes pour l'apprentissage d'un réseau de neurones. La figure 1.4 donne l'architecture de ce type de réseau de neurones profond. La partie *encoder* se situe à gauche sur la figure, et la partie *decoder* à droite. L'entrée du réseau de neurones profond correspond à une fenêtre de *tokens* observée dans le document, typiquement une phrase. Afin de prendre en compte l'ordre des *tokens* dans la fenêtre courante, une information de position est ajoutée aux représentations de ces *tokens*. Cette opération est représentée par le *positional encoding*. Les couches d'attention dans l'*encoder* et le *decoder* permettent de capturer des corrélations entre les différents *tokens* de la fenêtre courante. Plusieurs vecteurs d'attention sont utilisés afin de pouvoir prendre en compte tous les *tokens*. Ce type de réseau de neurones profond est utilisé pour convertir une séquence en une autre séquence, e.g. pour effectuer une traduction automatique. Ainsi, la sortie du *transformer* est un mot qui est prédit en fonction de la phrase donnée en entrée

14. L'apprentissage par transfert (ou *transfer learning*) est l'un des champs de recherche de l'apprentissage automatique qui vise à transférer des connaissances d'une ou plusieurs tâches sources vers une ou plusieurs tâches cibles.

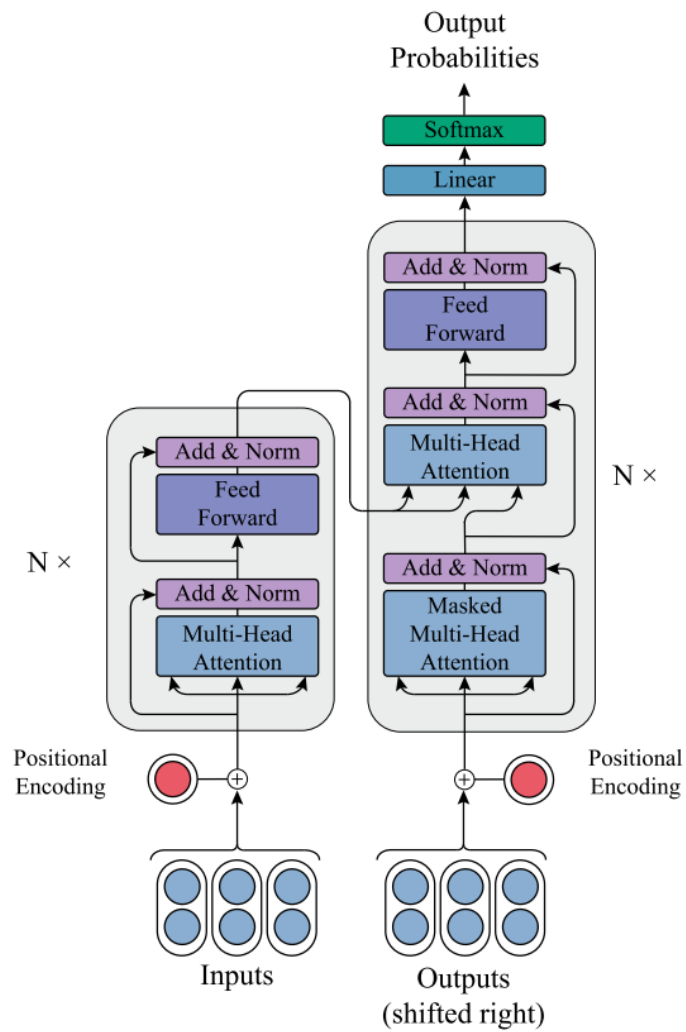


FIGURE 1.4 – Illustration de l'architecture des réseaux de neurones profonds de type *transformer* (Vaswani et al., 2017)

(langage source) ainsi que le mot précédent qui a été prédit par le réseau de neurones profond (langage cible).

Le modèle BERT (Devlin et al., 2019) se base sur la partie *encoder* de ce réseau de neurones profond. Son architecture consiste en 12 blocs d'*encoders* et est destinée à apprendre la représentation de phrases. Il utilise différents objectifs non supervisés : prédire des mots masqués dans la fenêtre courante et prédire la phrase suivante. Deux phrases sont données en entrée du réseau de neurones profond. Dans ces phrases, des mots sont masqués aléatoirement. Le modèle BERT doit prédire les mots masqués et prédire si la seconde phrase est effectivement la phrase suivante observée dans un document. BERT peut être entraîné sur un large corpus de façon non supervisée, puis affiné (ou *fine-tuned*) sur différentes tâches du traitement automatique du langage naturel. Par exemple, pour la classification de document, il est possible de remplacer la dernière couche de ce réseau de neurones par une couche de classification avec activation *softmax*, et affiner BERT en donnant une phrase (qui peut être étendue à 512 *tokens* pour représenter un document). La sortie du réseau de neurones est alors la classe du document et permet une correction d'erreur par rétropropagation du gradient. Si un document est plus long que 512 *tokens*, il est possible de le découper en plusieurs parties, ce qui permet d'augmenter le nombre d'exemples. Sun et al. (2019) décrivent différentes manières d'effectuer un affinage de BERT pour la classification. Les auteurs montrent que la dernière couche de BERT est la plus performante pour la classification, et donc qu'il est préférable de conserver toutes les couches du réseau de neurones profond lors de l'ajout d'une couche de classification. Les auteurs montrent aussi qu'il est possible d'effectuer de multiples affinages sur différentes tâches et un entraînement non supervisé sur les données de la tâche cible.

Plus récemment, Sanh et al. (2019) ont proposé une version du modèle BERT, appelée DistilBERT, moins coûteuse en mémoire et en temps d'entraînement, tout en gardant 97% des performances du modèle BERT original. La méthode utilisée est la distillation de connaissance qui consiste à transférer la connaissance d'un modèle large, en l'occurrence le modèle BERT original, dans un modèle plus « compact » tout en conservant les performances du modèle large. Le modèle DistilBERT est une version moitié moins « profonde » de BERT, c'est-à-dire que le nombre de couches du réseau de neurones profond est divisé par deux. Les auteurs ont aussi mis à disposition un modèle pré-entraîné sur un large corpus composé de livres en langue anglaise ainsi que d'articles Wikipédia en langue anglaise. Dans cette thèse, nous exploitons le modèle DistilBERT afin d'effectuer un affinage.

BERT et DistilBERT utilisent une *tokenisation* en *wordpieces*, c'est-à-dire en séquences de lettres fréquentes sélectionnées dans un corpus par l'algorithme *Byte Pair Encoding* (abrégé BPE) proposé initialement par Gage (1994). Le vocabulaire de *wordpieces* utilisé pour BERT est de taille 30 000. Cette *tokenisation* permet de limiter les inconvénients de la présence de mots rares dans un corpus comme le décrivent Sennrich et al. (2016). Par exemple, si le mot « *walking* » est rare dans un corpus, alors sa représentation pourra être composée des *wordpieces* « *walk* » et « *ing* ». La composition de ces deux *wordpieces* permet de bénéficier de la représentation du suffixe « *ing* » présent dans d'autres mots de la langue anglaise.

1.3 La stylométrie

1.3.1 Définition et motivations

En informatique, la stylométrie (ou *authorship analysis*) est un ensemble de méthodes visant à analyser le style des écrits inhérents à leur auteur. Les objectifs sous-jacents sont l'identification d'auteurs (ou *authorship attribution*, aussi appelé *authorship identification*) (Stamatatos, 2017), la vérification de paternité (ou *authorship verification*) (Boumber et al., 2019) et la caractérisation d'auteur (ou *author characterization / profiling*). Les applications de ce domaine sont nombreuses : la détection de fausses nouvelles (Rashkin et al., 2017), la détection d'agent communiquant (Ali, 2014), la recommandation de livres (Vaz et al., 2012b, 2013b), l'*hyperpartisanship prediction*¹⁵ (Potthast et al., 2018), la détection de plagiat, la cybersécurité ou encore l'analyse des messages malveillants sur les réseaux sociaux (Rocha et al., 2017).

La caractérisation d'auteur consiste à prédire des attributs démographiques relatifs à un auteur (e.g. âge, genre, éducation) par le texte écrit. L'identification d'auteurs et la vérification de paternité sont deux tâches plus répandues dans le domaine de la stylométrie. Ces tâches sont des tâches de classification. L'identification d'auteurs consiste à prédire l'auteur d'un document parmi un ensemble d'auteurs connus. À partir de documents dans un ensemble de test dont les auteurs sont inconnus, la tâche consiste à trouver quels sont les auteurs de ces documents à partir d'un ensemble d'entraînement contenant des documents de ces auteurs. Une variante dite *open-set* (opposée à *closed-set*) consiste à prédire l'auteur d'un document sachant que l'auteur peut être inconnu. Il s'agit donc de considérer une classe supplémentaire indiquant que l'auteur n'est pas connu. La difficulté de cette variante est de pouvoir identifier les documents dont l'auteur est inconnu à partir d'un ensemble d'entraînement ne contenant pas nécessairement ce type de documents. La vérification de paternité est une tâche de classification binaire. Il s'agit de prédire si deux documents sont du même auteur (Jasper et al., 2018).

Neal et al. (2017b) et Stamatatos (2009) donnent un aperçu des caractéristiques textuelles utilisées dans la stylométrie. Ces caractéristiques sont classées dans les catégories suivantes :

Lexicale Longueur des phrases, richesse de vocabulaire, compte de mots appartenant à un lexique (e.g. LIWC, mots vides), lisibilité, représentation sac de mots, représentation en *n-grams* de caractères.

Syntaxique Étiquettes morpho-syntaxiques (ou *Part-of-Speech tags*), ponctuations.

Sémantique Synonymes, représentations sémantiques, thématiques.

Structurelle Longueur des paragraphes, présence de citations.

Spécifique à un domaine Compte de mots dans un lexique spécialisé.

La plupart des expérimentations dans le domaine consistent en la représentation des documents par ces caractéristiques et en l'entraînement d'un modèle de classification (Houvardas et Stamatatos, 2006; Yang et al., 2018; Tausczik et Pennebaker, 2010; Goldstein-Stewart et al., 2009).

15. La tâche d'*hyperpartisanship prediction* est une tâche consistant à prédire si un article d'actualité est orienté politiquement.

1.3.2 Solutions proposées dans la littérature

En stylométrie, la représentation des documents la plus utilisée est la représentation en n -grams de caractères. Il s’agit de construire une représentation similaire à la représentation sac de mots mais en considérant le vocabulaire des n -grams de caractères. Les représentations par 3-grams sont les plus utilisées : le vocabulaire se constitue alors de toutes les suites de trois caractères dans le corpus. Selon [Neal et al. \(2017b\)](#), ce type de représentation a l’avantage, comparé à la représentation sac de mots, d’être indépendant du langage et être moins sensible au bruit (e.g. changement d’orthographe, erreurs grammaticales).

La représentation sac de mots appartenant à une liste de mots « fonctions » (e.g. « être », « avoir », « pour »), ainsi que des mesures de richesse de vocabulaire et de complexité de langage sont aussi employées. La mesure la plus utilisée est appelée le delta de Burrows ([Burrows, 2002](#)) qui consiste à comparer chaque document par l’usage des mots les plus fréquents observés dans le corpus. Il existe différentes variantes de cette mesure décrites par [Jannidis et al. \(2015\)](#) et [Argamon \(2008\)](#).

Les compétitions PAN ([Kestemont et al., 2019](#)) sont organisées tous les ans et portent sur les tâches de la stylométrie. Dans ces compétitions, la plupart des participants utilisent des algorithmes d’apprentissage automatique : des arbres de décision, des SVM ([Houvardas et Stamatatos, 2006](#)) ou encore des forêts d’arbres décisionnels¹⁶. [Zheng et al. \(2006\)](#) ont par exemple utilisé un SVM pour identifier les auteurs de messages. Les auteurs ont utilisé différentes caractéristiques textuelles telles que des caractéristiques lexicales, syntaxiques et spécifiques au domaine. [Marukat et al. \(2014\)](#) ont aussi utilisé un SVM pour l’identification d’auteur de messages sur un forum. Les auteurs ont utilisé au total 53 caractéristiques différentes et ont montré que le SVM était plus performant. De nombreuses études ont montré que les SVM sont plus performants que les autres algorithmes d’apprentissage automatique. Selon [Neal et al. \(2017b\)](#), cette performance résulte de la capacité des SVM à gérer les données dites « creuses ».

D’autres études ont employé des algorithmes basés sur la méthode des k plus proches voisins avec des fonctions de similarité telles que le delta de Burrows utilisé par [Sterling et Argamon \(2006\)](#) ou encore la divergence de Kullback-Leibler utilisée par [Arun et al. \(2009\)](#). [Stamatatos \(2007\)](#) propose aussi de considérer une fonction de similarité basée sur le n -grams de caractères. Des méthodes bayésiennes ont aussi été utilisées telles que LDA pour l’identification d’auteurs dans des e-mails et articles de blog par [Seroussi et al. \(2014\)](#) ainsi que pour des articles de recherche par [Rosen-Zvi et al. \(2004\)](#).

De nombreuses méthodes, représentations et fonctions de similarité ont été proposées dans différentes études sur la stylométrie. Cependant, d’après l’aperçu de la compétition PAN de l’année 2019 par [Kestemont et al. \(2019\)](#), les modèles les plus performants dans la tâche d’identification d’auteurs sont les SVM combinés à des représentations « sac de mots » ainsi que les représentations basées sur des n -grams de caractères comme l’ont proposé récemment [Amann \(2019\)](#) et [Bacciu et al. \(2019\)](#).

Les différentes méthodes et représentations employées dans la stylométrie ont aussi permis une amélioration de performances dans des tâches externes liées aux

16. L’algorithme des forêts d’arbres décisionnels (*random forest*) est une technique d’apprentissage automatique combinant les concepts de sous-espaces aléatoires et de *bagging*. L’algorithme des forêts d’arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents.

articles d’actualité telles que la tâche de vérification de faits politiques (ou *political fact-checking*) et de détection de fausses nouvelles. Par exemple, [Rashkin et al. \(2017\)](#) ont montré que les descripteurs LIWC aident à la prédiction de fausses nouvelles. Cependant, les auteurs ont également montré qu’un simple réseau de neurones récurrents de type LSTM permet un gain plus important en précision. Plus récemment, [Potthast et al. \(2018\)](#) ont montré obtenir une meilleure précision sur la tâche d’*hyperpartisanship prediction* et à distinguer les fausses nouvelles des nouvelles satiriques en utilisant des descripteurs stylométriques et un classifieur de type forêts d’arbres décisionnels.

1.3.3 Amélioration des performances par masquage

Comme nous l’avons vu, les tâches du traitement automatique du langage naturel comportent des étapes de prétraitement : filtrage des mots vides, lemmatisation, etc. L’objectif est de réduire la complexité du langage et la taille du vocabulaire pour que les algorithmes d’apprentissage automatique puissent trouver des régularités dans le texte ([Lourdusamy et Abraham, 2018](#)). Il s’agit de mettre en avant les caractéristiques du texte utiles à une tâche particulière. Ces méthodes de prétraitement sont généralement utilisées conjointement à la représentation sac de mots et de classifieurs (e.g. SVM).

Dans le domaine de la stylométrie, [Stamatatos \(2017\)](#) a proposé une méthode similaire à ce type de prétraitement. Cette méthode est appelée la distorsion de texte (ou *text distortion*) et vise à remplacer les mots sémantiquement forts, i.e. les mots peu fréquents et qui permettent de distinguer un document dans un corpus, par des *tokens* spéciaux. Il s’agit, dans la variante la plus performante de la méthode *text distortion*, de remplacer les caractères de ces mots par le caractère « # ». À l’origine, cette méthode fut utilisée par [Granados et al. \(2011\)](#) pour masquer les mots fréquents et améliorer la performance de différents modèles dans la classification de documents. Mais pour la stylométrie, l’objectif est de masquer les mots liés à des thèmes ou au genre des documents. Ces caractéristiques sont, selon [Stamatatos \(2017\)](#), des caractéristiques qui ne dépendent pas du style des auteurs. L’avantage du « masquage » est que l’on conserve la structure des phrases, contrairement à d’autres méthodes de prétraitement telles que la suppression des mots vides. Cette méthode de prétraitement a permis l’amélioration de l’identification d’auteurs, et notamment d’un modèle SVM utilisant la représentation *3-grams* de caractères. Elle a permis une amélioration dans des conditions défavorables : lorsque le thème ou le genre des documents d’un même auteur change entre l’ensemble d’entraînement et l’ensemble de test ([Stamatatos, 2017, 2018](#)).

Suite à ces travaux, [Halvani et al. \(2020\)](#) ont proposé un prétraitement proche consistant à masquer les mots thématiques par leur étiquette morpho-syntaxique. Leur méthode est appelée POSNoise. La méthode POSNoise permet une amélioration comparée à la méthode proposée par [Stamatatos \(2018\)](#) dans la vérification de paternité sur différents jeux de données.

L’objectif de ces méthodes est d’effectuer un prétraitement des corpus afin de rendre la représentation des documents robuste à un changement de thème ou de genre. Ainsi, les modèles peuvent généraliser le style des auteurs des jeux de données utilisés et être plus performants en conditions réelles (i.e. lorsque les auteurs écrivent sur différentes thématiques). Dans cette thèse, nous proposons en section 3.5 une

méthode consistant à filtrer un corpus pour l'identification d'auteurs. Cependant, les objectifs de cette méthode sont différents comparée aux méthodes de masquages proposées par [Stamatatos \(2018\)](#) et [Halvani et al. \(2020\)](#). Nous détaillons toutes les différences en section 3.5.8.

1.3.4 Apprentissage de la représentation du style écrit

Plus récemment, les réseaux de neurones profonds ont aussi été utilisés pour la stylométrie. [Mohsen et al. \(2016\)](#) proposent un *Stacked Denoising AutoEncoder* pour l'identification d'auteurs sur le jeu de données *Reuter C50*¹⁷. Tout comme le propose [Brocardo et al. \(2017\)](#) et [Surendran et al. \(2017\)](#), les auteurs de cette étude utilisent des caractéristiques dites *handcrafted* (e.g. *n-grams*, *n-grams* de caractères, mots fréquents) mais n'utilisent pas le texte brut.

Dans l'objectif de résoudre le problème de l'implémentation de descripteurs (ou *feature engineering*) pour la stylométrie, d'autres études ont, par la suite, utilisé des réseaux de neurones profonds pour traiter le texte brut. Par exemple, [Qian et al. \(2017\)](#) proposent d'utiliser un GRU pour l'identification d'auteurs. Les auteurs utilisent un réseau de neurones profond de type siamois composé de deux GRUs pour la vérification de paternité. Ce réseau de neurones profond donne un score à partir de deux documents indiquant si les deux documents sont du même auteur. Leur modèle montre être plus performant qu'un classifieur de type *Gradient Boosting* utilisant différentes caractéristiques textuelles. D'autres travaux ont aussi montré l'intérêt des réseaux de neurones récurrents (i.e. les GRUs et LSTMs) dans ces tâches ([Gupta et al., 2019](#); [Bagnall, 2015](#); [Zhou et Wang, 2016](#)). Plus récemment, [Boumber et al. \(2018\)](#) ont proposé un réseau de neurones profond de type CNN pour effectuer une identification d'auteurs sur des articles de recherche. La difficulté étant de pouvoir identifier plusieurs auteurs pour chaque article.

Au-delà de l'utilisation des réseaux de neurones profonds pour la classification, peu d'études consistent en l'apprentissage de la représentation du style par un processus de généralisation. Dans cette section, nous décrivons les différents travaux portant sur ce sujet, ou s'en approchant. Nous décrivons ensuite, dans la prochaine section, les limites de ces différents travaux. [Qian et al. \(2015\)](#) sont les premiers à se reposer sur un jeu de données « externe » afin de pré-entraîner un modèle pour la stylométrie. Les auteurs entraînent un classifieur de type SVM à identifier des auteurs et utilisent ce modèle pour l'identification d'auteur dans un nouveau jeu de données. Cette étude est la première étude visant à généraliser des caractéristiques de style. Cependant, ces travaux reposent sur des caractéristiques textuelles dites *handcrafted* et ne consistent pas en l'apprentissage de la représentation du style.

[Boumber et al. \(2019\)](#) proposent un réseau de neurones récurrents pour la vérification de paternité. Le modèle est entraîné à représenter des documents dans la tâche de vérification de paternité. Leur jeu de données est découpé en un jeu d'entraînement et de test ne comprenant pas les mêmes auteurs. Le modèle est donc pré-entraîné sur le jeu d'entraînement, puis utilisé sur le jeu de données de test par apprentissage par transfert. Les auteurs proposent d'effectuer cette expérimentation sur plusieurs jeux de données : ceux mis à disposition lors des compétitions PAN, le jeu de données *Amazon Reviews* composé de 300 auteurs ayant publié au moins

17. *Reuter C50* est un jeu de données composé de 5 000 articles d'actualité disponibles à l'adresse https://archive.ics.uci.edu/ml/datasets/Reuter_50_50

40 messages ainsi qu’un jeu de données appelé MLPA composé de 20 auteurs ayant publié chacun 20 articles scientifiques sur le sujet de l’apprentissage automatique. Les auteurs de cette étude montrent que leur modèle arrive à atteindre des performances supérieures aux modèles vainqueurs des compétitions PAN 2013 et PAN 2015.

Ding et al. (2019) proposent un modèle consistant à apprendre de manière non-supervisée la représentation de documents selon différents axes (i.e. thématique, lexical). Les auteurs proposent de combiner les deux variantes de Doc2Vec, i.e. PV-DM et PV-DBOW, pour l’entraînement de leur réseau de neurones profond. Ils évaluent leur modèle sur l’identification d’auteurs sur le jeu de données *IMDb62*¹⁸ et sur la vérification de paternité sur le jeu de données *PAN14*¹⁹. Par exemple, pour la vérification d’auteur, Ding et al. (2019) utilisent leur modèle afin d’inférer la représentation de nouveaux documents et calculent une similarité cosinus afin de déduire si les documents sont du même auteur. Pour l’identification d’auteurs, ils infèrent la représentation des documents grâce à leur modèle et utilisent un algorithme de régression logistique afin d’effectuer l’identification des auteurs. Leur modèle obtient de meilleures performances comparé à différentes *baselines* standards.

Jasper et al. (2018) proposent un modèle permettant de représenter les documents dans un corpus de romans en langue anglaise. Le modèle est composé de représentations distributionnelles de mots fastText (Mikolov et al., 2018) (entraîné sur un corpus d’article Wikipédia en langue anglaise) ainsi qu’un réseau de neurones profond de type « *stacked LSTM* ». Ils utilisent une fonction de perte triplet proposée par Hoffer et Ailon (2015) afin d’apprendre la représentation des documents à partir de deux documents d’un même auteur et d’un troisième document d’un autre auteur. Leur modèle est performant sur un sous-ensemble du jeu de données *PAN14* pour la tâche de vérification de paternité. Leurs expérimentations sont construites de sorte à évaluer la capacité du modèle à identifier les auteurs de textes courts. Ainsi, leur modèle est adapté aux textes courts issus de réseaux sociaux.

1.3.5 Limites identifiées dans l’état de l’art

Comme nous l’avons vu, la plupart des travaux dans le domaine de la stylométrie reposent sur l’implémentation de descripteurs (ou *feature engineering*) avec des représentations sac de mots. Les quelques travaux que nous avons présentés dans la sous-section précédente visent à entraîner des réseaux de neurones profonds sur les tâches de la stylométrie afin de pallier les problèmes de l’implémentation de descripteurs (ou *feature engineering*). Cependant, aucun de ces travaux ne propose d’entraîner un modèle sur un grand nombre de documents dans le but de généraliser les représentations apprises à différents jeux de données, dont le « domaine » change (e.g. thématique, article d’actualité, de blog). Par exemple (Boumber et al., 2019) utilisent un jeu de données composé d’un nombre restreint d’auteurs. Ces différents travaux exploitent divers jeux de données, mais chaque expérimentation est conduite sur un jeu de données unique dédié aux tâches de la stylométrie.

Dans cette thèse, nous proposons d’explorer une approche différente. Étant donné la large disponibilité d’articles d’actualité et de blog sur le web, nous proposons d’ex-

18. Le jeu de données *IMDb62* est disponible à l’adresse https://umlt.infotech.monash.edu/?page_id=266/.

19. Le jeu de données *PAN14* est disponible à l’adresse <https://pan.webis.de/clef14/pan14-web/>.

exploiter ceux-ci pour l'entraînement d'un modèle visant à généraliser la représentation du style. Contrairement à ce que proposent [Ding et al. \(2019\)](#) qui effectuent un apprentissage non supervisé, nous voulons prendre en compte les labels attachés à ce type de données : les auteurs que l'on peut extraire du contenu HTML et le nom des journaux en ligne que l'on peut extraire des noms de domaines. Ce type de données a pour avantage de permettre un entraînement supervisé sans nécessité d'effectuer de labélisation manuelle sur un grand nombre d'exemples. Ainsi, nous pouvons exploiter un contenu textuel (i.e. les articles d'actualité abondants sur le web) aussi large que le contenu des jeux de données utilisés dans l'entraînement non supervisé des récents modèles de la sémantique distributionnelle tels que BERT. Comme nous le décrirons au chapitre 2, nous effectuons donc un apprentissage supervisé afin d'entraîner un réseau de neurones profond sur la base d'un large nombre d'auteurs de référence. Ainsi, nous répondrons à la question de recherche *QR1* portant sur l'exploitation des données de type articles d'actualité et de blog dans l'entraînement d'un réseau de neurones profond pour la stylométrie.

Nous proposons également d'évaluer directement la qualité des représentations générées par notre modèle sur la tâche de partitionnement par auteur. Cette tâche nous permettra d'évaluer si notre modèle produit des représentations stylométriques de qualité sans effectuer d'affinage sur les documents de nouveaux auteurs présents dans les jeux de test, et donc évaluer la généralisation de notre modèle. Ensuite, nous évaluerons celui-ci sur l'identification d'auteurs. Ainsi, nous répondrons à la question de recherche *QR2* portant sur la représentation de documents dans un espace stylométrique général.

1.4 Conclusion

Cet état de l'art a permis de donner un aperçu du domaine du traitement automatique du langage naturel pour la représentation du texte grâce à des méthodes statistiques. Nous avons également donné un aperçu du domaine de la stylométrie, et notamment des différentes notions dont la compréhension est nécessaire à la lecture des chapitres 2, 3 et 4. Ceux-ci porteront sur les questions de recherche *QR1* et *QR2* et permettront d'apporter des solutions aux limites de l'état de l'art exposées en section 1.3.5.

Chapitre 2

Méthode de représentation du style écrit et cadre d'évaluation

2.1 Introduction

Dans ce chapitre, nous présentons une nouvelle méthode d'apprentissage de la représentation visant à projeter des documents dans un espace stylométrique de référence. Les récentes études du domaine de la stylométrie se concentrent sur l'implémentation de descripteurs (ou *feature engineering*) afin de représenter le style des documents et d'améliorer la performance de modèles prédictifs sur différentes tâches (e.g. l'identification d'auteurs, la vérification d'auteurs). Au lieu de cela, nous proposons de directement projeter les documents dans un espace stylométrique en nous appuyant sur un ensemble de documents et d'auteurs de référence. Nous voulons capturer les caractéristiques discriminatives des différents styles propres aux auteurs (e.g. de livres, d'articles) afin de les exploiter dans différentes tâches telles que les tâches de la stylométrie ou des tâches extrinsèques (e.g. la recommandation d'articles d'actualité).

Les écrits d'un auteur sont généralement consistants dans leur style (Karlgrén, 2004) même lorsque l'auteur couvre un grand nombre de thèmes différents (Patchala et Bhatnagar, 2018). Notre méthode d'apprentissage de la représentation exploite cette consistance que nous appelons « *consistance intra-auteur* ». Celle-ci est l'une des propriétés que compose la définition du style écrit proposée en section 2.2 de ce chapitre.

En section 2.3, nous introduisons l'hypothèse principale de notre étude sur le style écrit : l'*hypothèse de généralisation du style*. Dans cette même section, nous détaillons la méthode de ciblage des indices intra-auteurs consistants, qui nous permettra, dans les futures chapitres, de valider cette hypothèse. Nous introduisons *SimRank*, une nouvelle métrique de partitionnement permettant d'évaluer la qualité de représentations vectorielles. Nous détaillons l'intérêt de cette nouvelle métrique.

La seconde propriété du style, la *non-spécificité sémantique*, suggère que les indices linguistiques relevant du style tendent à ne porter que peu d'informations sur le fond, le thème, les entités, etc. En section 2.4, nous proposons une mesure, le *TFIDF focus*, permettant de vérifier si les modèles entraînés par la méthode de ciblage des indices intra-auteurs consistants sont en adéquation avec la *non-spécificité sémantique*.

Enfin, en section 2.5, nous introduisons la seconde hypothèse : l'*hypothèse de*

Hommage à Notre-Dame de Paris

Blog A l'alarme, citoyens!

Comment dire l'indicible ? Comment exprimer l'inexprimable ? Comment raconter l'impensable ? L'effroi est tel que les mots, pour ce genre d'événement, historique à plus d'un titre, n'existent pas !

[...] Lundi 15 avril 2019. Il est 22 heures. À l'heure où j'écris ces lignes, la cathédrale Notre-Dame de Paris, joyau architectural, artistique et spirituel, vieux de plus de huit siècles, est en train de brûler, ravagé par un énorme incendie, sous mes yeux effarés, incrédules. [...] Je me souviens. Longtemps, dans les années 90, j'ai moi-même habité à Paris

Anaphores

Autoréférences

FIGURE 2.1 – Exemple de style spécifique du blog *A l'alarme, citoyens!*

Notre Dame de Paris : un symbole brûlé

France Culture

La cathédrale parisienne a été ravagée par un incendie lundi soir. La piste accidentelle est envisagée. Pendant que l'enquête se poursuit, de nombreuses personnes viennent voir ce qu'il reste du monument. Une cathédrale connue dans le monde entier. [...] Des milliers de personnes sont venues constater les dégâts.

[...] « On voit une cathédrale détruite, une cathédrale comme on ne l'a jamais vue, se désole Margot. C'est un paysage urbain qui se transforme par cette destruction partielle. »

Journalisme factuel

Citations et interviews

FIGURE 2.2 – Exemple de style spécifique au journal *France Culture*

filtrage. Celle-ci stipule que la suppression des phrases les plus « révélatrices » des auteurs, dans le cadre de la méthode de ciblage des indices intra-auteurs consistants, permet une meilleure généralisation du style écrit.

2.2 Le style écrit

Les auteurs d'articles d'actualité et d'articles de blog ont chacun leur style propre. La figure 2.1 donne un exemple d'article de style blog. Les articles de blog, comme nous pouvons le voir sur cet exemple, ont la particularité de comporter des références à l'auteur lui-même, avec, par exemple, des anecdotes personnelles de l'auteur concernant un évènement en particulier.

Les deux autres articles (figures 2.2 et 2.3) sont des articles sur le même sujet mais d'un style différent. Le premier correspond à un style que l'on pourrait qualifier de « journalisme factuel », i.e. exposant des faits et n'exprimant aucune opinion. Le second est plus « littéraire ». À noter que nous considérons certains éléments discursifs particuliers, tels que les citations sur le deuxième article, comme relevant

Notre-Dame de Paris: la «cathédrale» est éternelle

Jean Pruvost dans *Le Figaro*

Impossible de ne pas avoir la gorge nouée devant le spectacle effroyable de la Cathédrale Notre-Dame de Paris en flammes, immense symbole de pierre et de bois, s'effaçant, en cendres rougeoyantes, d'un paysage parisien presque millénaire. [...] L'histoire d'une cathédrale est plus forte que le feu, on ne peut l'anéantir par le feu. [...] Eh bien à cette indestructible littérature fera écho la reconstructible cathédrale.

Style littéraire

[...] Le Moyen Âge gothique primitif et flamboyant - quelle sinistre résonance... - poussait un cri déchirant dans le brasier.

Références historiques

FIGURE 2.3 – Exemple de style spécifique à un rédacteur du journal *Le Figaro*

du style. La typographie est importante dans la caractérisation du style puisque permet, par exemple, d'identifier les citations grâce aux guillemets. Sur le deuxième article, la citation relève du style puisque montre la volonté de l'auteur d'exprimer une idée par l'intermédiaire de personnes interrogées, cette façon de présenter les faits étant différente d'une simple description de ceux-ci.

Un auteur peut adopter plusieurs styles, l'un d'eux pouvant être similaire au style d'un autre auteur. [Karlgrén \(2004\)](#) définit le style comme une tendance consistante et distinguable à faire certains choix linguistiques (e.g. syntaxe, vocabulaire, développements, descriptions, répétitions).

A consistent and distinguishable tendency to make [some of these] linguistic choices.

[Karlgrén \(2004\)](#)

De plus, [Karlgrén \(2004\)](#) explique que les écrits sont bien plus que ce dont ils parlent.

Texts are much more than what they are about.

[Karlgrén \(2004\)](#)

Toute différence textuelle qui n'est pas sémantique ou thématique appartient aux choix stylistiques de l'auteur. Différentes expressions peuvent avoir un sens commun, faire référence aux mêmes objets, aux mêmes événements, mais être pourtant constituées de différents mots, différentes syntaxes, volonté de l'auteur de laisser transparaître un contexte, une orientation, parfois une émotion ([Argamon et al., 2005](#)).

Les articles d'actualité ont montré être écrits avec un style spécifique par l'utilisation, entre autres, d'adverbes pour les dates (e.g. « *The governor Thursday*

announced... ») ou encore d'anthropomorphisations¹ (e.g. « *The 1990s saw an increase in crime... »*). Ce style particulier est appelé le *journalese* (Dickson et Skole, 2012). Les titres de journaux ont aussi un style spécifique, appelé *headline*, avec, par exemple, la suppression d'articles (e.g. « *(A) Disaster as (a) hurricane strikes (the) south coast of (the) United States* », les parenthèses indiquent la suppression des articles) (Weir, 2009). Une autre démonstration de la particularité stylistique des articles d'actualité est la tendance récente à la mise en avant d'articles par des titres « piège à clics » (ou *clickbait*) sur internet (Chakraborty et al., 2016). Le principe du piège à clics est d'écrire des titres attrayants ne reflétant pas ou amplifiant le contenu de l'article. L'objectif est d'inciter les utilisateurs à cliquer pour générer du revenu.

Enfin, mentionnons les « guides de style » ou « code typographique » des journaux qui influencent non seulement la typographie (e.g. la structure des paragraphes, les citations, l'utilisation de l'italique) mais aussi l'usage de la langue elle-même (Cameron, 1996).

In contrast to other textual online contents (professional or user generated) authors of news are usually people with journalistic background, who are following certain journalistic standards, style, and language.

Sertkan et al. (2019)

Par exemple, un guide de style peut plus ou moins encourager l'utilisation de la voix active contre la voix passive. Certains journaux en ligne peuvent avoir leurs propres guides de style que leurs rédacteurs suivent, rendant le texte « cohérent » ou « consistant » pour le lecteur. Nous noterons également que la plupart des journaux ont une « ligne éditoriale »², parfois une orientation politique, influençant les sujets publiés et la façon de les présenter (Sertkan et al., 2019). Comme le souligne Hicks (2002), garder une consistance dans le style permet au lecteur de se concentrer sur le sens, de ne pas détourner son attention sur des détails de forme.

Variation that has no purpose is distracting. By keeping a consistent style in matters of detail a publication encourages readers to concentrate on what its writers are saying.

Hicks (2002)

Étant donné que les journaux en ligne suivent ces pratiques de manière plus ou moins constante dans le temps, il est possible d'utiliser ces caractéristiques comme base pour les distinguer.

Les articles de blog ont aussi leurs particularités stylistiques comme les autoréférences, les anecdotes personnelles ou encore les anaphores comme dans cet exemple tiré d'un article de *The Tim Ferris Show* :

*Might that be an overreaction? Might I be misinformed?
Totally.*

Tim Ferriss sur *The Tim Ferris Show*

1. L'anthropomorphisation est l'action de donner à un animal ou une chose, une représentation humaine, un aspect ou un comportement humain.

2. La ligne éditoriale d'un média représente l'ensemble de leurs choix influençant leur communication tels que les choix moraux, éthiques, thématiques, etc.

Au contraire, les articles d'actualité peuvent parfois être plus littéraires, ou prendre la forme d'analyses factuelles d'évènements, d'interviews ou de citations comme dans cet exemple tiré de *The Washington Post* :

Director General [...] said Saturday that "it's impossible to predict which direction this epidemic will take".

Simon Denyer et collègues sur *The Washington Post*

Le style est plus ou moins prononcé selon les passages dans les écrits d'un auteur, il est difficile de le définir de façon précise, et, sachant un document, de trouver une suite de mots qui appartienne strictement au style de l'auteur. Chaque mot ou suite de mots est un mélange de fond et de forme, c'est-à-dire de contenu sémantique et de style. Ils ne peuvent généralement pas être associés ni au style, ni au contenu, mais à une combinaison des deux. C'est pourquoi il est difficile d'extraire des caractéristiques du texte.

It is difficult to distinguish whether a lexical token occurring in a sentence is mainly due to the topics of the document or the author's lexical preference.

Ding et al. (2019)

À partir des documents du corpus de référence, nous cherchons à extraire des structures latentes qui appartiennent au périmètre du style écrit. Nous soutenons que ces structures latentes peuvent être capturées par des réseaux de neurones profonds, et plus précisément des modèles de type « récurrent » avec une couche d'attention capable de se concentrer sur des termes liés au style. D'un point de vue linguistique, ces structures latentes correspondent à des fragments lexicaux, syntaxiques ou structurels de phrases ou de paragraphes.

Pour extraire les caractéristiques de style de manière distributionnelle, nous cherchons à capturer des structures latentes (lexicales, syntaxiques, structurelles), que nous appellerons par la suite des « indices linguistiques », satisfaisant ces deux propriétés :

La *consistance intra-auteur* la propriété d'être consistant dans les documents d'un même auteur ;

La *non-spécificité sémantique* la propriété de ne porter que peu d'informations sur ce qui distingue sémantiquement le document (e.g. thèmes, entités mentionnées) dans un corpus.

Cette définition³ est inspirée de [Karlgrén \(2004\)](#) et [Holmes \(1998\)](#) et consiste à dire que le style des documents est représenté par des structures linguistiques qui sont consistantes pour chaque auteur (permettant leur identification) mais sémantiquement pauvres vis-à-vis du contenu du document (e.g. thème, entités). En effet, ce sur quoi porte le document est une contrainte qui impose à l'auteur d'employer un vocabulaire spécifique. Les termes qui appartiennent à ce vocabulaire spécifique portent une forte

3. Dans ce chapitre, nous ferons référence aux propriétés de *consistance intra-auteur* et de *non-spécificité sémantique* comme définissant le style, cependant, celles-ci permettent plus exactement de « caractériser » les indices relevant du style écrit.

valeur sémantique vis-à-vis du thème du document, et au contraire, véhiculent moins probablement le style de l’auteur.

L’extraction d’indices linguistiques appartenant au style écrit dans sa définition est difficile. De même, il est difficile d’extraire exhaustivement tout type de caractéristiques pouvant aider dans les tâches du domaine de la stylométrie (Bischoff et al., 2020). Les descripteurs employés auront des performances variables selon la tâche. Il s’agit généralement non pas de trouver les caractéristiques textuelles représentant au mieux le style de son auteur, mais plutôt de choisir les descripteurs qui auront les meilleures performances pour une tâche donnée. Ces descripteurs appartiennent soit au périmètre du style écrit dans sa définition courante comme les « mots fonctions » (ou *function words*) (Goldstein-Stewart et al., 2009; Menon et Choi, 2011), soit ne correspondent qu’à de simples descripteurs du domaine du traitement automatique du langage naturel tels que les *n-grams* de caractères (Escalante et al., 2011; Stamatatos, 2007) ou les représentations distributionnelles de textes (Qian et al., 2017; Gupta et al., 2019; Bagnall, 2015). De plus, il est difficile de savoir à l’avance si une caractéristique spécifique sera pertinente dans une tâche spécifique.

Étant donné que l’implémentation de descripteurs (ou *feature engineering*) souffre de ces quelques problèmes, nous tentons de définir et de capturer le style en exploitant ses propriétés distributionnelles. En effet, la première propriété, la *consistance intra-auteur*, vise à éviter l’implémentation de descripteurs en s’appuyant sur la distribution des indices qui sont consistants pour chaque auteur de référence. La seconde propriété, la *non-spécificité sémantique*, suit la constatation de Karlgren (2004), que les écrits sont bien plus que ce dont ils parlent. Ainsi, les indices linguistiques relatifs au thème, au sujet, à l’entité, auront tendance à être moins représentatifs du style de l’auteur. C’est une propriété mesurable par les couches d’attention que l’on peut ajouter aux réseaux de neurones profonds. Ainsi, dans les prochains chapitres, la *consistance intra-auteur* guidera notre méthodologie tandis que la *non-spécificité sémantique* sera une propriété à mesurer afin de déterminer si les caractéristiques extraites par le réseau de neurones profond sont en adéquation avec celle-ci.

2.3 La méthode de ciblage des indices intra-auteurs consistants

Le partitionnement d’auteur (ou *authorship clustering*) est une tâche nous permettant d’établir la qualité de représentations vectorielles de document produites par un modèle. Il s’agit de produire des représentations vectorielles reflétant au mieux l’appartenance d’un document à son auteur. Comme nous l’avons développé en section 2.2, le style, par la *consistance intra-auteur*, est un indice permettant cette mise en relation. A contrario, le thème (ou *topic*), pouvant changer d’un document à l’autre pour un même auteur dans les articles d’actualité, ne reflétera pas nécessairement l’appartenance d’un document à son auteur. Si les représentations d’un modèle *A* permettent de mieux regrouper les documents par auteur que les représentations d’un modèle *B*, alors nous considérerons les représentations du modèle *A* de meilleure qualité. Plus formellement, cette qualité est déterminée par les métriques de partitionnement dites « internes » (*internal evaluation metrics*). Dans cette section, nous introduisons la métrique *SimRank*, particulièrement adaptée à l’évaluation du partitionnement de documents.

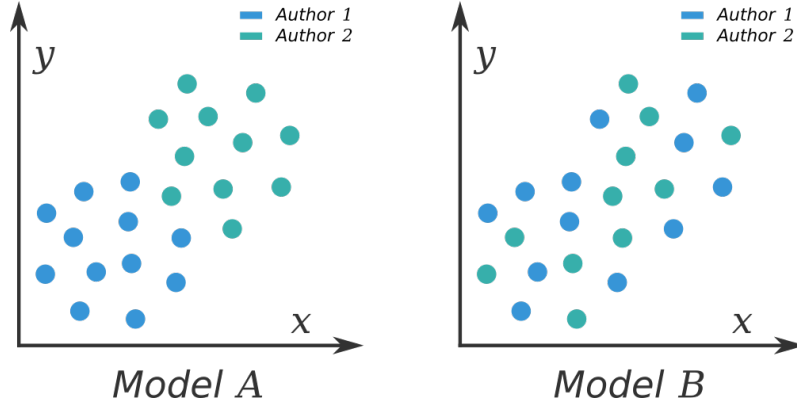


FIGURE 2.4 – Partitionnements d’auteurs issus d’un modèle A et d’un modèle B

Sur la figure 2.4, le système A , à gauche, parvient à regrouper les vecteurs de documents (ayant pour valeurs les coordonnées x et y) par auteur contrairement au système B . Nous considérons alors le système A comme supérieur dans sa capacité à produire des représentations vectorielles de documents reflétant son appartenance à un auteur.

2.3.1 L’ensemble de référence et les ensembles nouveaux

Notons $D = \{d_1, \dots, d_n\}$ un ensemble de documents. Les documents prennent la forme d’une séquence de vecteurs sémantiques de mots⁴ donc $\forall i \in \{1, \dots, n\}, d_i \in \mathbb{R}^l \times \mathbb{R}^w$ avec w la dimension des vecteurs de mots. Les documents sont de taille égale l , i.e. ils ont autant de vecteurs de mots. Les documents de taille inférieure à l sont complétés par des vecteurs remplis de zéros jusqu’à obtenir la taille l . Les documents trop longs seront tronqués. Notons $A = \{a_1, \dots, a_m\}$ un ensemble d’auteurs. Chaque document a été écrit par un et un seul auteur dans A et chaque auteur a écrit au moins un des documents de D .

Un ensemble de référence R -set (ou *reference set*) est un sous-ensemble strict de A et de D . Il est unique. On notera R -set = (D^r, A^r) avec $A^r \subsetneq A$ impliquant $D^r \subsetneq D$. Chaque document de D^r a été écrit par un auteur dans A^r . Notons $|D^r| = n^r, |A^r| = m^r$ tel que $n^r > m^r$.

Un ensemble nouveau U -set (ou *unseen set*) est un sous-ensemble strict de A et de D . Un ensemble nouveau a les mêmes propriétés qu’un ensemble de référence :

$$U\text{-set} = (D^u, A^u)$$

$$A^u \subsetneq A$$

$$D^u \subsetneq D$$

$$|D^u| = n^u$$

$$|A^u| = m^u$$

$$n^u > m^u$$

4. Une séquence de vecteurs sémantiques de mots est une liste de représentations vectorielles de mots appartenant à un lexique connu. Ces vecteurs sont généralement appris par des méthodes non-supervisées telles que *Word2Vec* (Mikolov et al., 2013).

Cependant, aucun auteur de A^u n'appartient à A^r . Plus formellement, $A^r \cap A^u = \emptyset$ impliquant $D^r \cap D^u = \emptyset$. Pour la validation de l'*hypothèse de généralisation du style*, nous construisons un unique R -set et de multiples U -sets. À noter que l'ensemble de référence doit être plus grand que tout ensemble nouveau : $n^r > n^u, m^r > m^u$.

Notons $V^r \in [0, 1]^{m^r}$ un ensemble de vecteurs tel que $\forall v \in V^r, \sum_{i=1}^{m^r} v_i = 1$, typiquement un vecteur de distribution de probabilités (e.g. un vecteur *softmax*) sur A^r . Dans cet ensemble de vecteurs, nous définissons un sous-ensemble de vecteurs « vérité terrain » (ou *ground truth*) $G^r = \{g_1^r, \dots, g_{m^r}^r\}$. Chaque document dans D^r est associé à un et uniquement un vecteur de G^r . Ces vecteurs indiquent l'appartenance d'un document à un auteur dans A^r . Ils sont dits *onehot*, c'est-à-dire qu'ils sont remplis de zéros à l'exception d'un 1 à la position (ou index) de l'auteur auquel elle correspond dans A^r .

2.3.2 L'hypothèse de généralisation du style

Suivant la définition du style donnée en section 2.2, nous proposons de valider l'*hypothèse de généralisation du style* et d'introduire une nouvelle méthode ayant pour objectif d'apprendre une représentation vectorielle du style. Nous voulons construire d'un modèle pré-entraîné capable de projeter tout nouveau document dans un espace stylométrique général. Nous appelons cette méthode la « méthode de ciblage des indices intra-auteurs consistants ». Elle consiste à exploiter un large ensemble de référence, le R -set afin d'entraîner un modèle à reconnaître les éléments linguistiques consistants pour un même auteur. Plus nous disposons de documents par auteur, plus nombreux seront les indices à forte valeur sémantique (e.g. le thème, les entités nommées) pouvant permettre la reconnaissance d'un auteur. Or, plus les aspects sont nombreux, plus le risque d'intersection de ces indices entre auteurs augmente, rendant plus difficile la distinction des auteurs et laissant alors apparaître la consistance de style écrit des auteurs. Cette méthode se fonde sur l'*hypothèse de généralisation du style* que nous détaillons.

Notons $f^r : \mathbb{R}^l \times \mathbb{R}^w \rightarrow V^r$ une fonction projetant les documents donnés en entrée, représentés par des séquences de vecteurs sémantiques de mots, de D^r vers V^r . L'objectif de f^r est de projeter chaque document $d_i^r, i \in \{1, \dots, n^r\}$ de telle sorte que la projection (ou représentation vectorielle) soit la plus proche possible de son vecteur *ground truth* correspondant dans G^r que de tout autre vecteur *ground truth*.

L'hypothèse principale de ce chapitre, l'*hypothèse de généralisation du style*, stipule que deux représentations de deux documents nouveaux définies par f^r seront plus probablement similaires si les deux documents ont été écrits par le même auteur que si ce n'était pas le cas. La figure 2.5 illustre l'*hypothèse de généralisation du style*. Les ensembles bleus dans V^r correspondent aux représentations des documents de D^r . La fonction f^r permet de projeter les documents de D^r dans V^r de telle sorte que la représentation d'un document est proche de son vecteur *ground truth* correspondant en rouge. Les ensembles verts correspondent aux représentations des documents nouveaux dans V^r . L'*hypothèse de généralisation du style* stipule que des représentations proches, dans les ensembles verts sur la figure, correspondront plus probablement à des documents écrits par le même auteur.

Plus formellement, notons un ensemble nouveau U -set respectant ces propriétés :

$$U\text{-set} = (D^u, A^u)$$

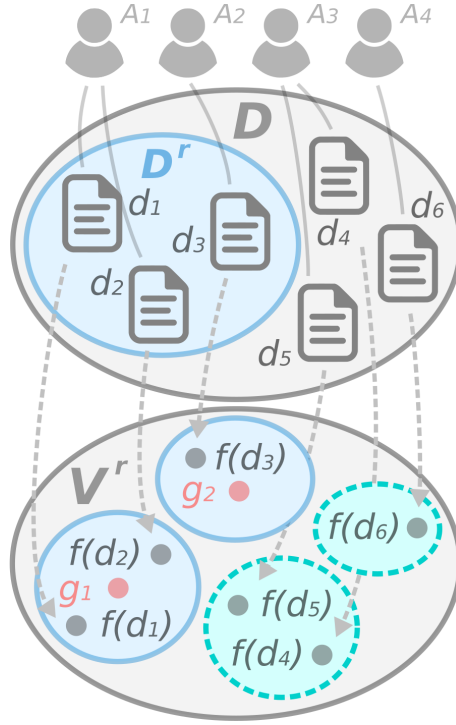


FIGURE 2.5 – Illustration de l’hypothèse de généralisation du style

$$D^u \subsetneq D$$

$$A^u \subsetneq A$$

$$|D^u| = n^u$$

$$|A^u| = m^u$$

Nous rappelons que chaque document dans D^u a été écrit par un auteur dans A^u et $n^u > m^u$. Il n’existe pas d’intersection de document entre le R -set et le U -set : $A^r \cap A^u = \emptyset$ impliquant $D^r \cap D^u = \emptyset$. L’hypothèse de généralisation du style stipule que la projection de documents de D^u (le U -set) dans V^r en utilisant f^r permet de calculer des similarités de telle sorte que les documents similaires de D_u sont plus probablement écrits par les mêmes auteurs dans A_u . Par conséquent, la fonction f^r permet le partitionnement par auteur dans un espace stylométrique général qu’elle définit.

Intuitivement, nous émettons l’hypothèse que tout nouveau document écrit par un auteur inconnu est similaire, en termes de style, à des documents écrits par un sous-ensemble d’auteurs connus, et qu’un autre document du même auteur sera probablement similaire, en termes de style, aux documents du même sous-ensemble d’auteurs connus. Les auteurs de référence ne permettront pas d’exposer exhaustivement tous les styles écrits existants et possibles. Cependant, un nombre suffisant d’auteurs de référence permettra de construire une approximation du style suffisante pour positionner chaque nouvel auteur dans l’espace de référence. Un auteur nouveau pourra écrire dans un style nouveau, mais celui-ci pourra être représenté par une combinaison de styles de référence. Il pourra avoir un style partiellement similaire aux styles d’un sous-ensemble d’auteurs, mais la consistance

de ces similarités « partielles » permettra de bien séparer cet auteur des autres dans l'espace de référence.

Comme expliqué précédemment, plus nous disposons de documents par auteur dans le R -set, plus la *consistance intra-auteur* se révélera. Or, dans le chapitre suivant, nous verrons que lorsque seules les personnes physiques (e.g. journaliste, blogueur) sont considérées en tant que classes, il est très difficile de simultanément disposer d'assez de documents pour construire un jeu de données large et de restreindre raisonnablement le nombre d'auteurs pour que l'entraînement d'un modèle soit possible. Une hypothèse sous-jacente à l'*hypothèse de généralisation du style* est alors que se reposer sur des documents regroupés par journal / blog (représenté par leur nom de domaine) est suffisant pour obtenir de bonnes performances dans un partitionnement d'auteur sur les U -sets. Plus précisément, nous émettons l'hypothèse que généraliser une consistance intra-journal / intra-blog permette un bon partitionnement des U -sets (même pour un U -set uniquement constitué d'auteurs physiques), que ce soit par l'existence de consignes stylistiques (e.g. guide stylistique, ligne éditoriale, code typographique) ou par la prédominance d'un auteur physique pour un journal ou un blog donné. La construction d'un R -set suffisamment large (avec un grand nombre d'auteurs et un grand nombre de documents par auteur) permet au modèle entraîné sur ces données de généraliser suffisamment le style en s'appuyant sur le plus d'auteurs possible. Dans le chapitre 3, nous décrirons la construction d'un tel jeu de données.

L'objectif final est de disposer d'un modèle approximant f^r capable de calculer des similarités stylométriques sans connaissance de l'auteur du document, i.e. sans que le modèle ait eu besoin d'être entraîné sur de nouveaux auteurs, donc sans affinage (ou *fine-tuning*). Pour résumer, le modèle entraîné permettra de représenter les documents d'un auteur inconnu de telle sorte que les représentations vectorielles refléteront au mieux l'appartenance des documents à cet auteur. Ceci est rendu possible par la projection des documents dans un espace stylométrique général ayant été construit sur la base d'un ensemble d'auteurs et de documents de référence par la méthode de ciblage des indices intra-auteurs consistants. La projection d'un document nouveau dans cet espace rapproche les auteurs nouveaux des auteurs de référence et permet ainsi le partitionnement par auteur.

2.3.3 Évaluation de la qualité des représentations vectorielles

Les métriques standards pour le partitionnement

Dans l'objectif d'évaluer nos modèles ainsi que les *baselines*, nous commençons par générer les représentations vectorielles $V = \{v_1, \dots, v_p\}$ en utilisant tous les $d_p^u \in D^u$ (le U -set) et un modèle donné de telle sorte que $\forall v \in V, |v| > 0$. Ensuite, nous évaluons la capacité des représentations vectorielles à bien partitionner les documents en fonction d'une mesure de similarité et des labels vérité $L = \{L_1, \dots, L_p\}$ de telle sorte que $L_i = L_j$ si et seulement si d_i^u a le même auteur que d_j^u . Nous nous basons sur ces deux métriques standards mesurant la qualité d'un partitionnement :

L'indice de *Calinski-Harabasz* (Caliński et Harabasz, 1974) Défini par le rapport de la somme de la dispersion entre partitions et de la dispersion dans les partitions (la dispersion est définie comme la somme des distances au

carré). Il sera abrégé *CalHar*. L'indice de *Calinski-Harabasz*, pour un ensemble de données E , est donné par :

$$CalHar = \frac{\text{tr}(B_k)}{\text{tr}(W_k)} \times \frac{|E| - k}{k - 1} \quad (2.1)$$

Avec k le nombre de partitions de l'ensemble E . B_k est la matrice de variance inter-groupes et W_k la matrice de variance intra-groupes :

$$W_k = \sum_{q=1}^k \sum_{x \in C_q} (x - c_q)(x - c_q)^T \quad (2.2)$$

$$B_k = \sum_{q=1}^k |q|(c_q - c_E)(c_q - c_E)^T \quad (2.3)$$

Avec C_q l'ensemble des points de la partition q , c_q le centre de la partition q et c_E le centre de E .

L'indice de *Davies-Bouldin* (Davies et Bouldin, 1979) Défini par la similarité moyenne entre les partitions. La similarité est une mesure qui compare la distance entre les partitions avec la taille des partitions. Un indice proche de 0 signifie que les partitions sont de bonne qualité. Il sera abrégé *DavB*. L'indice de *Davies-Bouldin*, pour k partitions, est donné par :

$$DavB = \frac{1}{k} \sum_{i=1}^k \max_{i \neq j} \frac{s_i + s_j}{d_{ij}} \quad (2.4)$$

Avec s_i le diamètre de la partition et d_{ij} la distance entre les centroïdes des partitions i et j .

Ces métriques utilisent la similarité euclidienne. Elles prennent en entrée V et L et retournent un score reflétant la qualité du partitionnement.

La métrique *SimRank*

Dans le domaine de la recherche d'information, il est important de pouvoir bien ordonner des documents en fonction de leur représentation et de la représentation d'une requête. La requête peut prendre la forme d'un fragment de texte (e.g. dans les moteurs de recherche) ou de l'historique d'un utilisateur (e.g. dans les systèmes de recommandation). La méthode la plus simple permettant de générer un ordonnancement⁵ (ou *ranking*) d'un ensemble de documents est de considérer la similarité entre les représentations des documents et la représentation de la requête. La métrique la plus utilisée lorsqu'il s'agit d'évaluer la qualité d'un ordonnancement est la métrique de *Normalized Discounted Cumulative Gain* (abrégée *nDCG*) (Järvelin et Kekäläinen, 2002). Dans notre expérimentation, les métriques standards (e.g. l'indice de *Calinski-Harabasz*), mesurant la qualité d'un partitionnement, n'indiquent pas, en moyenne, à quel point les documents sont bien ordonnés les uns par rapport aux autres sachant leur représentation et une mesure de similarité. Nous introduisons donc une nouvelle

5. La génération d'un ordonnancement (ou *ranking*) correspond à la mise en ordre d'éléments dans un vecteur.

métrique évaluant la qualité d'un partitionnement, appelée *SimRank*, basée sur la métrique *nDCG* issue du domaine de la recherche d'information.

Pour introduire *nDCG*, notons l'ensemble de documents $D = \{d_1, \dots, d_n\}$ et R un ordonnancement de ces documents tel que $|R| = k$, $R_i \in D$, $i \in \{1, \dots, k\}$ et

$$\forall i \in \{1, \dots, k\}, \forall j \in \{1, \dots, k\}$$

$$i \neq j \Rightarrow R_i \neq R_j$$

À noter que $k \leq n$, i.e. l'ordonnancement R ne contient pas nécessairement tous les documents. Nous considérons l'ordonnancement R donné, mais en pratique, il est généré par un système sachant une requête donnée et un ensemble de documents D .

Notons un vecteur de pertinence graduée rel tel que $|rel| = k$ et $rel_i \in \{0, 1\}$, $i \in \{1, \dots, k\}$. rel_i indique si le document correspondant dans R est pertinent ($rel_i = 1$) ou non ($rel_i = 0$).

La fonction *nDCG* donne un score qui prend un vecteur de pertinence graduée rel :

$$\text{nDCG}(rel) = \frac{\text{DCG}(rel)}{\text{iDCG}(rel)} \quad (2.5)$$

Le *nDCG* inclue le score *DCG* (Järvelin et Kekäläinen, 2002) :

$$\text{DCG}(rel) = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i + 1)} \quad (2.6)$$

et est normalisé par le score *DCG* idéal :

$$\text{iDCG}(rel) = \sum_{i=1}^{\text{sum}(rel)} \frac{1}{\log_2(i + 1)} \quad (2.7)$$

Dans notre expérimentation, nous pouvons construire un ordonnancement R_p de tous les documents du *U-set* pour un document donné $d_p^u \in D^u$. Ce dernier correspondrait donc à la requête dans une tâche du domaine de la recherche d'information. Nous rappelons que D^u correspond à tous les documents du *U-set*. Cet ordonnancement sera un vecteur ordonné de tous les documents de D^u suivant la similarité cosinus de leur représentation vectorielle.

Notons rel^p le vecteur de pertinence graduée tel que $rel_i^p = 1$ si le document correspondant dans R^p a été écrit par l'auteur de d_p^u et $rel_i^p = 0$ si le document correspondant a été écrit par un autre auteur. À noter que $|rel^p| = |D^u|$. Nous pouvons maintenant appliquer la métrique *nDCG* sur rel^p qui contiendra toujours au moins un document pertinent, impliquant $\text{nDCG}(rel^p) > 0$. Afin de normaliser le score *nDCG* entre 0 et 1, nous donnons l'équation (2.8) introduisant *wDCG*, le pire *DCG* :

$$\text{nDCG}'(rel) = \frac{\text{DCG}(rel) - \text{wDCG}(rel)}{\text{iDCG}(rel) - \text{wDCG}(rel)} \quad (2.8)$$

L'équation (2.9) donne le *wDCG* :

$$\text{wDCG}(rel) = \sum_{i=\text{count}_0(rel)}^{|rel|} \frac{1}{\log_2(i + 1)} \quad (2.9)$$

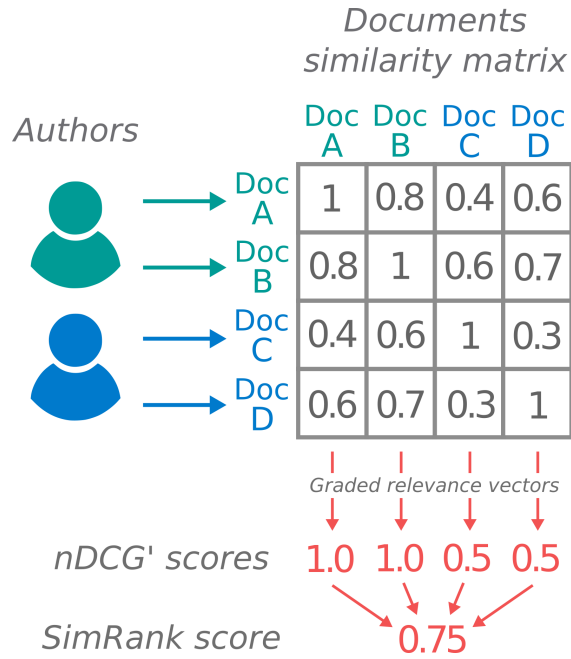


FIGURE 2.6 – Illustration du calcul du score *SimRank*

Avec $\text{count}_0(\text{rel}) = \sum_{u=1}^{|\text{rel}|} -(\text{rel}_u - 1)$, i.e. le compte de zéros dans rel . Nous normalisons $nDCG$ en utilisant le pire ordonnancement car, dans notre cas, nous trions toujours sans exception l'ensemble des documents en fonction d'un document « requête », alors que le score $nDCG$ est généralement calculé sur un sous-ensemble de documents qui sont renvoyés par un moteur de recherche. Or, la valeur minimale si nous avons utilisé le $nDCG$ original ne serait pas 0.

Finalement, le score *SimRank* correspond à la moyenne des $nDCG'$ appliqués à $REL^u = \{\text{rel}^1, \dots, \text{rel}^{|D^u|}\}$ l'ensemble des vecteurs de pertinence graduée pour chaque $d_p^u \in D^u$ avec $|REL^u| = |D^u|$:

$$\text{SimRank}(REL) = \frac{\sum_{p=1}^{|REL|} nDCG'(\text{rel}^p)}{|REL|} \quad (2.10)$$

La figure 2.6 illustre le calcul du score *SimRank*. L'exemple comprend deux auteurs et quatre documents. La matrice de similarité des documents permet de produire les vecteurs de pertinence graduée. Puis les scores $nDCG'$ sont calculés sur chaque vecteur. Pour terminer, le score *SimRank* est calculé à partir de l'ensemble des scores $nDCG'$.

Démonstration empirique de l'intérêt de *SimRank*

Comme nous l'avons détaillé en section 2.2, un auteur peut adopter plusieurs styles. Par exemple, un auteur écrivant sur des sujets politiques peut parfois écrire des articles de style « factuel » et « descriptif », et occasionnellement écrire dans un tout autre registre : e.g. « humouristique », « satirique ». Ainsi, les articles de cet auteur formeront deux partitions distinctes dans un espace stylométrique, comme s'ils étaient écrits par deux auteurs ayant des styles différents. La prise en compte de l'ordre des exemples et non des distances, ainsi que le fait que les poids attribués aux exemples

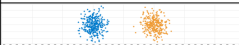










Id	Clusters	600 samples			Random #samples in [400, 2400]		
		SimRank ($\pm 2\sigma$)	CalHar ($\pm 2\sigma$)	DavB ($\pm 2\sigma$)	SimRank ($\pm 2\sigma$)	CalHar ($\pm 2\sigma$)	DavB ($\pm 2\sigma$)
1		1.0 (± 0)	7503 (± 660)	0.25 (± 0.01)	1.0 (± 0)	17K ($\pm 14K$)	0.25 (± 0)
2		0.9 (± 0.01)	2322 (± 111)	0.42 (± 0.01)	0.9 (± 0.01)	5383 ($\pm 4K$)	0.42 (± 0)
3		1.0 (± 0)	2218 (± 75)	0.5 (± 0)	1.0 (± 0)	5142 ($\pm 4K$)	0.51 (± 0)
4		0.69 (± 0.01)	0.03 (± 0.08)	170 (± 393)	0.68 (± 0.01)	0.03 (± 0.08)	255 (± 668)
5		0.59 (± 0.01)	0.01 (± 0.02)	304 (± 747)	0.58 (± 0.01)	0.01 (± 0.03)	442 (± 890)
6		0.59 (± 0.01)	146 (± 5)	2 (± 0.03)	0.58 (± 0.01)	349 (± 280)	2.01 (± 0.02)
7		0.71 (± 0.02)	554 (± 24)	1.02 (± 0.02)	0.71 (± 0.02)	1310 ($\pm 1K$)	1.02 (± 0.01)
8		0.62 (± 0.01)	57 (± 3)	3.03 (± 0.1)	0.61 (± 0.01)	134 (± 111)	3.03 (± 0.07)
9		0.66 (± 0)	258 (± 20)	1.27 (± 0.05)	0.65 (± 0.01)	613 (± 484)	1.27 (± 0.03)
10		0.52 (± 0)	0.07 (± 0.14)	158 (± 456)	0.52 (± 0.01)	0.07 (± 0.15)	234 (± 695)
11		0.53 (± 0)	1 (± 2)	38 (± 109)	0.52 (± 0.01)	1.04 (± 2.13)	54 (± 166)

FIGURE 2.7 – Moyenne des scores *SimRank*, *CalHar* et *DavB* des 1000 partitionnements générés aléatoirement de 11 configurations différentes

décroissent par l'utilisation de nDCG, nous permet de minimiser les effets de ce multi-partitionnement. En effet, dans le calcul du score nDCG sur une colonne (voir l'exemple de la figure 2.6), plus les documents sont différents du document référence (lié à la colonne considérée), moins une erreur de classement sera pénalisante.

Les métriques de partitionnement standards telles que *DavB* et *CalHar*, quant à elles, considèrent la forme des partitionnements créés. Elles sont plus sensibles aux distances entre les exemples. Ainsi, si les exemples d'une classe forment deux partitions, ces métriques indiqueront une moins bonne qualité de partitionnement, même lorsque les exemples appartenant à ces partitions sont distants des exemples des autres classes. L'intérêt de la métrique *SimRank*, en plus de permettre une meilleure prédiction des performances des modèles dans la classification comme nous le verrons au chapitre 4, est qu'elle ne pénalise pas les modèles qui répartissent les exemples dans plusieurs partitions distantes des partitions correspondantes aux autres classes.

Pour démontrer cet effet, nous avons généré aléatoirement le partitionnement d'exemples appartenant à deux classes. La figure 2.7 montre l'ensemble des configurations que nous avons générées. Ces 11 configurations correspondent à tous les arrangements possibles de quatre partitions : deux partitions comprenant les exemples d'une classe en bleu et deux autres comprenant les exemples de l'autre classe en orange. La couleur d'un exemple indique sa classe. Pour chaque ligne du tableau, nous avons tout d'abord généré 1000 partitionnements aléatoires du même nombre d'exemples. Nous avons choisi un nombre de 600 exemples. Les scores correspondants sont ceux de la partie gauche du tableau. Chaque graphique dans la colonne *Clusters* correspond à la répartition des exemples pour un unique partitionnement parmi les 1000 générés. Ensuite, les scores de la partie droite du tableau correspondent aussi, pour chaque ligne, à la génération aléatoire de 1000 partitionnements, la différence étant que nous avons fait varier le nombre d'exemples total de 400 à 2400.

Ainsi, les scores *SimRank*, *CalHar* et *DavB* obtenus correspondent à la moyenne

pour les 1 000 partitionnements de la ligne correspondante. Les premières configurations correspondent à des partitionnements que nous considérons de qualité, les exemples de chaque classe étant bien séparés. Les dernières configurations correspondent à des partitions qui mélangent des exemples de deux classes. Nous considérons que celles-ci sont de moins bonne qualité.

Comme le montre ce tableau, les scores *SimRank* semblent effectivement moins pénalisés par le multi-partitionnement des classes. Par exemple, selon les métriques *CalHar* et *DavB*, les configurations 4 et 5 donnent des partitionnements de moins bonne qualité que toutes les autres configurations, et notamment les deux dernières (i.e. 10 et 11). Lorsque l'on compare les configurations 4 et 9 (la configuration 9 ayant une partition composée d'exemples de différentes classes), les métriques *CalHar* et *DavB* indiquent que la configuration 4 donne des partitionnements de moins bonne qualité tandis que *SimRank* indique qu'elles ont une qualité proche. De même, les pires partitionnements possibles donnés par la configuration 11 sont considérés de meilleure qualité que les partitionnements de la configuration 4 par les métriques *CalHar* et *DavB*. Nous rappelons qu'un score *DavB* proche de 0 indique que le partitionnement est de bonne qualité selon cette métrique.

Une autre particularité de *SimRank* est sa stabilité sur une même configuration. En effet, les mesures 2σ (indiquant l'intervalle pour 95% des valeurs) de toutes les configurations sont proches de 0 pour *SimRank*. En revanche, pour certaines configurations, notamment les dernières, les mesures 2σ des métriques *CalHar* et *DavB* ont une grande variance. La partie droite du tableau, montre aussi que la métrique *SimRank* reste stable même lorsque le nombre d'exemples varie. Cette faible variation des scores *SimRank* permet de comparer des modèles même lorsque les données sur lesquelles ils sont évalués sont différentes.

À noter également que les scores *SimRank* sont normalisés entre 0 et 1, il est ainsi possible d'avoir une intuition de la performance d'un modèle lorsque le meilleur partitionnement possible aura un score de 1. Un score de 0 correspond au score *SimRank* le plus bas que l'on puisse obtenir. Cependant, cette valeur est théorique. En pratique, il est plus pertinent de comparer les scores *SimRank* (des modèles de représentation que l'on évalue) à un score obtenu à partir d'un modèle générant des représentations aléatoires. Dans nos exemples, ce score serait proche de 0.5 comme le montre les configurations 10 et 11, les exemples pour ces configurations étant tous positionnés aléatoirement dans les mêmes partitions. Nous utiliserons cette méthode de comparaison dans les expérimentations du chapitre 4.

En définitive, la métrique *SimRank* présente ces quelques avantages comparée aux métriques standards :

1. elle est peu sensible au multi-partitionnement d'une classe ;
2. les scores sont normalisés entre 0 et 1 ;
3. la variance pour une même configuration est faible ;
4. la variance pour une même configuration est faible même lorsque le nombre d'exemples varie.

2.4 La *non-spécificité sémantique*

La *non-spécificité sémantique* est la seconde propriété de la définition du style donnée dans ce chapitre. Cette propriété suggère que les indices linguistiques relevant

du style tendent à ne porter que peu d’informations sur le fond, le thème, les entités, etc. Les indices à forte valeur sémantique qui identifieront, par exemple, un thème, sont les indices permettant la différenciation du document dans un corpus. Une méthode très utilisée lorsqu’il s’agit de pondérer les termes d’un document par leur spécificité vis-à-vis du document est la pondération TFIDF.

Nous cherchons à vérifier si les modèles entraînés par la méthode de ciblage des indices intra-auteurs consistentants respectent la propriété de *non-spécificité sémantique*. Dans cet objectif, nous évaluons à quel point la méthode de ciblage des indices intra-auteurs consistentants permet d’effectivement capturer les indices non spécifiques aux documents. Nous proposons d’utiliser l’attention des réseaux de neurones profonds appris par cette méthode ainsi que les poids TFIDF des termes que composent les documents. Nous cherchons à calculer un score indiquant à quel point les modèles, entraînés sur le *R-set*, portent leur attention sur des termes à poids TFIDF faibles. Dans le chapitre 4, afin de valider l’adéquation des modèles avec la *non-spécificité sémantique*, nous comparerons ceux-ci à des modèles n’ayant pas été entraînés par la méthode de ciblage des indices intra-auteurs consistentants.

La mesure proposée, le *TFIDF focus*, consiste à calculer la moyenne des poids TFIDF multipliés par les poids d’attention des modèles. Plus formellement, notons A la matrice d’attention du modèle pour un U -set donné de taille $w \times d$. Chaque ligne de la matrice correspond à l’attention d’un document du U -set choisi avec w le nombre de mots que composent les documents et d le nombre de documents que compose le U -set. Une ligne de la matrice A correspond aux poids d’attention en sortie de la couche d’attention du réseau de neurones profond lorsque l’on donne un document du U -set en entrée à celui-ci.

Notons T la matrice TFIDF de même taille $w \times d$. La matrice T est calculée à partir de tous les documents du U -set sans contrainte sur la taille du vocabulaire. À noter que T n’est pas la matrice *sparse* TFIDF habituellement utilisée lorsqu’il s’agit d’obtenir des descripteurs TFIDF mais une matrice, qui, pour chaque document associe le poids TFIDF des mots le composant. Si un mot se répète dans les mots que compose un document, il y aura alors plusieurs fois le même poids dans la ligne correspondante. Nous calculons A' et T' les matrices normalisées par lignes de respectivement A et T . Les valeurs de chaque ligne (correspondant aux documents) de ces deux matrices sont normalisées entre 0 et 1 :

$$\forall i \in \{1, \dots, d\}, A'_i = \text{stdnorm}(A_i)$$

$$\forall i \in \{1, \dots, d\}, T'_i = \text{stdnorm}(T_i)$$

de telle sorte que la somme des valeurs d’une ligne est égale à 1, impliquant que la somme de tous les termes de la matrice est égale à d :

$$\forall M' \in \{A', T'\}, \sum_{i=1}^d \sum_{j=1}^w M'_{ij} = d$$

Nous utilisons la fonction de normalisation *Min-Max* qui prend X un vecteur et renvoie le vecteur normalisé :

$$\text{stdnorm}(X) = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Le score *TFIDF focus*, pour une matrice d'attention A' donnée et une matrice TFIDF T' donnée correspondra donc à :

$$\text{TFIDFFocus}(A', T') = \frac{\sum_{i=1}^d \sum_{j=1}^w A'_{ij} \cdot T'_{ij}}{d} \quad (2.11)$$

En calcul matriciel, l'équation 2.11 est strictement équivalente à la *trace* de la multiplication de la matrice A' avec la transposée de la matrice T' , le tout divisé par le nombre de documents que compose le *U-set* :

$$\text{TFIDFFocus}(A', T') = \frac{\text{Tr}(A' \cdot T'^T)}{d} \quad (2.12)$$

Ainsi, plus la mesure *TFIDF focus* est basse pour un modèle donné et un *U-set* donné, plus cela indiquera que le modèle se concentre sur des mots à faible valeur TFIDF dans le *U-set* concerné, et plus cela indiquera que le modèle se concentrera sur des indices linguistiques en adéquation avec la propriété de *non-spécificité sémantique*.

2.5 L'hypothèse de filtrage

Non seulement il est difficile d'identifier précisément quelles caractéristiques relèvent du style écrit (Bischoff et al., 2020), mais il est également difficile d'extraire des caractéristiques qui ne capturent pas d'informations thématiques en même temps (Stamatatos, 2018) puisque ce type d'informations est généralement utile à la reconnaissance d'un auteur (Seroussi et al., 2014).

Ideally, stylometric features should not be affected by shifts in topic or genre variations and they should only depend on personal style of the authors. However, it is not yet clear how the topic/genre factor can be separated from the personal writing style.

Stamatatos (2017)

Cependant, en conditions particulières « *cross-domain* » (en thème et en genre « littéraire »), c'est-à-dire lorsque le thème ou le genre change pour un même auteur entre l'ensemble d'entraînement et de test, les caractéristiques thématiques n'aident pas à la reconnaissance d'un auteur. C'est pourquoi de récentes études proposent des méthodes de masquage de portions de texte consistant à cacher le thème et le genre d'un document afin d'améliorer la reconnaissance des auteurs (Stamatatos, 2018, 2017; Halvani et al., 2020).

Dans le chapitre 3, nous proposons une méthode de filtrage du corpus de référence. La différence avec les méthodes de masquage est que nous ne ciblons pas les mots liés au thème ou au genre mais ceux spécifiques à leurs auteurs dans le corpus de référence. L'objectif final est différent : nous cherchons à rendre plus difficile l'identification d'un auteur afin que les modèles de réseaux de neurones profonds puissent capturer des indices subtils et consistants dans le texte sans se reposer sur des mots trop évidents sur l'appartenance du document à son auteur. Cette méthode n'est donc pas destinée à directement améliorer la performance d'un modèle dans l'identification d'auteurs sur un jeu de données spécifique, mais consiste à filtrer un corpus de

référence afin d'améliorer l'extraction de traits stylistiques de documents d'auteurs inconnus. L'avantage de notre méthode est qu'elle permet de conserver le texte brut et garantir une cohérence dans les données d'entrée dans les jeux d'entraînement (i.e. les *R-set*) et les jeux de test (i.e. les *U-set*).

L'*hypothèse de filtrage* stipule que, dans le cadre de la méthode de ciblage des indices intra-auteurs consistants, la suppression des phrases du *R-set* les plus « révélatrices » de l'auteur permet de produire de meilleures représentations du style écrit, en adéquation avec la *non-spécificité sémantique*. Plus précisément, l'*hypothèse de filtrage* stipule que le filtrage des phrases « révélatrices » d'un auteur dans le corpus de référence :

1. permet de projeter des documents nouveaux dans un espace stylométrique de façon à améliorer d'autant plus les performances dans l'identification d'auteurs et le partitionnement par auteur ;
2. permet au modèle de se concentrer d'autant plus sur des mots sémantiquement faibles comme les « mots fonctions » permettant ainsi une adéquation du modèle avec la propriété de *non-spécificité sémantique*.

Les phrases les plus « révélatrices » sont celles contenant des séquences de mots propres à l'auteur, i.e. les séquences de mots très utilisées par un auteur et très peu par le reste des auteurs dans le corpus. Le processus de filtrage décrit au chapitre 3 nous permet de filtrer un jeu de données en supprimant les phrases « révélatrices » des auteurs. Nous utiliserons la pondération TFIDF sur des *n-grams* avec différents *n*. Les *n-grams* ciblés sont ceux qui ont une fréquence élevée dans les documents d'un même auteur et une faible fréquence inverse dans les documents du corpus, i.e. qui sont rares dans le corpus. Le *R-set* filtré ainsi obtenu nous permet de valider l'*hypothèse de filtrage*.

Intuitivement, moins il y a d'indices révélateurs dans le jeu d'entraînement, plus le modèle doit réussir à identifier des indices subtils et dispersés afin d'obtenir de bonnes performances dans l'identification d'auteurs. Il atteindra de bonnes performances lorsque les représentations vectorielles des documents du corpus de référence généraliseront suffisamment chaque auteur de référence en identifiant des séquences subtiles et non des éléments lexicaux propres aux auteurs. Ainsi, le filtrage permet de concentrer l'attention du modèle sur des séquences consistantes pour un auteur et moins spécifiques aux documents. À noter que, grâce à la sémantique distributionnelle et l'utilisation de vecteurs sémantiques de mots ou de *wordpieces*, ces séquences « consistantes » ne devront pas être nécessairement composées des mêmes mots, d'où le fait qu'elles soient « subtiles ». Elles peuvent être composées de différents mots mais malgré tout être « similaires » lorsque l'on considère les vecteurs sémantiques que composent ces séquences (e.g. en termes de similarité cosinus).

L'intérêt d'un réseau de neurones récurrents est dans sa capacité à identifier ce type de séquences, contrairement aux méthodes de représentation classiques telles que les descripteurs stylométriques standards ou encore les représentations sac de mots. Une seconde intuition ayant motivé l'*hypothèse de filtrage* est que le filtrage peut jouer un rôle important dans l'adéquation du modèle avec la *non-spécificité sémantique*. En effet, les mots les plus sémantiquement forts liés au thème d'un document et aux thèmes spécifiques à un auteur seront moins nombreux. Et moins il y aura de mots spécifiques aux auteurs, plus les mots sur lesquels le réseau de neurones profond se concentrera seront proches des mots employés par un nouvel auteur (dans les *U-sets*), permettant ainsi une meilleure généralisation des représentations stylométriques.

Prenons un exemple : si les documents du journal *CNN* sont tous composés de la séquence "*Written by CNN*" alors le modèle focalisera son attention sur cette séquence afin d'identifier *CNN*. Il se focalisera sur peu d'autres séquences potentiellement plus subtiles. Les séquences plus subtiles n'aideront pas à l'identification de l'auteur qui est déjà maximale grâce à cette séquence. En effet, les séquences révélatrices telles que "*Written by CNN*" provoqueront l'arrêt de l'entraînement du modèle lorsque la courbe d'apprentissage sur l'ensemble de validation commencera à être constante ou à décroître. Dans notre exemple, les séquences de mots trahissent de façon évidente l'auteur des documents. Cependant, lors du filtrage effectué grâce à la méthode proposée au chapitre 3, l'exploitation de *n-grams* de différentes tailles nous permettra de filtrer aussi des phrases moins évidentes mais tout aussi révélatrices.

2.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle définition du style écrit sur la base de propriétés distributionnelles : la *consistance intra-auteur* et la *non-spécificité sémantique*. Nous avons proposé une nouvelle méthode d'apprentissage de la représentation (ou *representation learning*) du style : la méthode de ciblage des indices intra-auteurs consistants. Celle-ci nous permettra, au chapitre 4, de valider l'*hypothèse de généralisation du style*. La méthode proposée implique la construction d'un corpus de référence stylistique. Dans le chapitre 3, nous détaillons la construction d'un tel corpus et proposons une méthode de filtrage. Ce filtrage nous permettra de valider l'*hypothèse de filtrage* au chapitre 4.

Chapitre 3

Construction d'un corpus de référence stylistique

3.1 Introduction

Dans l'objectif de tester l'*hypothèse de généralisation du style* et l'*hypothèse de filtrage* que nous avons introduites au chapitre 2, nous proposons de créer un corpus de référence stylistique, que nous appelons *NewsID*¹, dont nous allons extraire deux types d'ensembles : un *R-set* pour l'entraînement de nos modèles sur un grand nombre de documents ainsi que différents *U-sets* nous permettant de tester notre modèle et valider nos hypothèses.

Cette thèse se concentre sur des documents de type articles d'actualité et de blog pour la large disponibilité de ce type de contenu sur internet. L'application de la méthode de ciblage des indices intra-auteurs consistants sur un jeu de données large est ainsi rendue possible grâce à ce type de données. Dans ce chapitre, nous commençons par décrire, en section 3.2, les données assemblées qui constitueront le jeu de données *NewsID*. Nous proposons ensuite, aux sections 3.3 et 3.4 de nettoyer ce corpus et d'attribuer des labels spécifiques à chaque document. En section 3.3, nous proposons une méthode de filtrage. Le corpus filtré par cette méthode nous permettra, au chapitre 4, de valider l'*hypothèse de filtrage*. Enfin, nous terminons en section 3.6 par décrire le *R-set* et les *U-sets* extraits de *NewsID*.

3.2 Collecte des données

La première étape de la construction du jeu de données *NewsID* et de l'extraction du *R-set* ainsi que des *U-sets* est de disposer de D , un grand ensemble de documents. Les jeux de données existants dans la littérature ne permettaient pas l'entraînement d'un modèle sur suffisamment de données. Le jeu de données *BlogCorpus* (Schler et al., 2006) ne comprend que des articles de type blog. Le jeu de données *RCV1* destiné à la classification d'articles par catégories (Lewis et al., 2004) est seulement composé d'articles du journal Reuters. D'autres jeux de données, principalement destinés à l'évaluation de modèles sur la tâche d'attribution d'auteur, ne sont composés que d'un faible nombre d'articles (Stamatatos et al., 2018). De plus, chaque jeu de données de

1. Le jeu de données *NewsID* est mis à disposition à l'adresse <https://github.com/hayj/DeepStyle>.

la littérature, pris individuellement, ne permettait pas de disposer de suffisamment d'articles par auteur. Nous avons donc choisi d'assembler un grand nombre d'articles de différents jeux de données de la littérature étant chacun suffisamment grand individuellement afin de disposer d'une base d'entraînement et d'évaluation solide pour nos modèles.

AllTheNews, MediaHouses et NewsAndBlog jeux de données du site web Kaggle rassemblant des articles d'actualité et de blog en langue anglaise ;

BlogCorpus (The Blog Authorship Corpus) jeu de données rassemblant des articles de blog en langue anglaise (Schler et al., 2006) ;

ICWSM2009 et ICWSM2011 jeux de données de l'entreprise Spinn3r pour les *workshops* ICWSM 2009 et ICWSM 2011 rassemblant une très grande quantité de pages web, dont des articles d'actualité et de blog (Burton et al., 2009, 2011) ;

Taonews jeu de données comprenant de nombreux articles d'actualité de différents journaux (~ 1 million) collectés durant cette thèse grâce à une bibliothèque implémentée en *Python*². Cette bibliothèque est dépendante de différentes bibliothèques de récolte (*Selenium*³, *Requests*⁴) ainsi que d'extraction de contenu (*newspaper3k*⁵). Les liens des articles issus de ce jeu de données ont été extraits de *tweets*, puis nous avons collecté les données en téléchargeant le contenu des liens sélectionnés. Les *tweets* de *Taonews* ont été téléchargés sur archive.org.

Ces jeux de données permettent de rassembler 63 millions de documents sous la même structure avec les champs *url*, *text*, *authors*, *domain* et *title*. Nous ajoutons un champ *author* correspondant au premier auteur du champ *authors* s'il n'est pas vide. Par la suite, nous n'utiliserons que le champ *author*. Seuls les documents de langue anglaise⁶ ayant plus de 100 caractères et des lignes d'en moyenne plus de 8 caractères ont été conservés.

3.3 Élimination du bruit

L'assemblage de nombreux articles hétérogènes introduit du bruit, i.e. des articles ne présentant aucun intérêt à une étude sur le style ou sur l'attribution d'auteur. Ce fut par exemple le cas d'articles tels que les pages web ne comprenant que des menus, les produits d'occasion en vente, les articles ne comprenant que des publicités et très peu de texte, etc. Afin de pallier ce problème, nous procédons ainsi :

2. La bibliothèque de récolte développée durant cette thèse est disponible à l'adresse <https://github.com/hayj/WebCrawler>.

3. *Selenium*, disponible à l'adresse <https://github.com/SeleniumHQ/selenium>, est un *framework* de test informatique développé en *Java*. Il permet d'interagir avec différents navigateurs web.

4. *Requests*, disponible à l'adresse <https://github.com/psf/requests>, est une bibliothèque permettant l'utilisation du protocole HTTP en *Python*.

5. *Newspaper3k*, disponible à l'adresse <https://github.com/codelucas/newspaper>, est une bibliothèque *Python* pour l'extraction de données d'articles d'actualité.

6. La bibliothèque *langdetect*, disponible à l'adresse <https://pypi.org/project/langdetect>, a été utilisée pour le filtrage des documents par langue.

1. Prétraitement simple du jeu de données. Tout d'abord, nous réduisons chaque espace blanc⁷ à un unique espace (ou à un unique retour à la ligne si l'espace blanc en contient au moins un). Nous éliminons ensuite les lignes commençant par des indicateurs identifiés manuellement tels que "*Share this article*", "*All rights reserved*" ou encore "*Advertisement*". Plus de 300 indicateurs ont été collectés manuellement avec la lecture de quelques exemples du jeu de données.
2. Classification de 300 articles⁸ en équilibrant le nombre d'articles par nom de domaine. Pour chaque article, nous avons donné un nombre entre 0 et 1 reflétant la qualité de l'article, i.e. contenu assez long, peu de publicités, peu de menus, un encodage correct, suffisamment de phrases bien formées, etc.
3. Implémentation de l'extraction de différents descripteurs.
4. Choix d'un seuil déterminant de façon binaire si un article est de qualité ou non.
5. Entraînement d'un modèle⁹ sur les données annotées.
6. Élimination des articles du jeu de données dont la qualité est prédite comme inférieure au seuil par le modèle.

Suite au prétraitement simple du jeu de données et à la labélisation manuelle de 300 articles, nous implémentons l'extraction des descripteurs suivants :

tooLongDocument Indique si le document fait plus de 60000 caractères.

length Nombre de caractères.

linesCount Nombre de lignes.

tokensCount Nombre de mots (un élément de ponctuation est compté comme un mot).

longLinesCount Nombre de longues lignes (supérieur à 140 caractères).

shortLinesCount Nombre de lignes courtes (inférieur à 20 caractères).

meanLinesLength Nombre de caractères par ligne en moyenne.

longLinesRatio et **shortLinesRatio** Proportion de nombre de longues lignes et proportion de nombre de lignes courtes.

stopwordsPunctRatio Nombre de mots vides et de ponctuations sur nombre de mots au total. La liste de mots vides utilisée est la liste des 500 mots les plus courants de la langue anglaise.

nonSWPMeanOverlap Nombre moyen de doublons pour les mots non vides et n'étant pas des éléments de ponctuation.

upperWordCount et **upperWordRatio** Nombre et proportion de mots constitués uniquement de majuscules.

nonWordCount et **nonWordRatio** Nombre et proportion de *token*¹⁰ ne portant aucune lettre.

7. Un espace blanc est une suite de caractères blancs tels que les espaces, les retours à la ligne ou encore les tabulations.

8. L'outil *Annotator*, développé durant cette thèse, a permis la classification de ces articles. Cet outil est disponible à l'adresse <https://github.com/hayj/Annotator>.

9. Le modèle pré-entraîné ainsi que l'ensemble de l'implémentation sont disponibles à l'adresse <https://github.com/hayj/NewsTools>.

10. Un *token* est une unité lexicale extraite d'un texte. Le découpage d'un texte en plusieurs *tokens* permet le traitement de celui-ci en séquence.

htmlCharCount et **htmlCharRatio** Nombre et proportion de caractères appartenant à des balises *html*.

hasUpperRatio Proportion de mots comportant au moins une majuscule.

lineStartWithNonWordRatio Nombre de lignes commençant par un *token* n'ayant aucune lettre.

encodingProbCount Nombre de caractères faisant partie d'une liste de caractères significatifs d'un problème d'encodage.

L'ensemble des descripteurs est extrait de l'ensemble du jeu de données annoté. Nous choisissons le seuil de 0.75 afin de classer chaque exemple comme étant de qualité ou non. Plusieurs modèles standards de classification sont évalués sur une validation croisée de 10 *folds*. Le modèle *Linear Support Vector Classification*¹¹ (Buitinck et al., 2013) est retenu avec une précision de 90%. Après analyse qualitative, il a été observé que les exemples mal classés sont ceux proches du seuil. Cette précision était donc suffisante dans l'objectif d'éliminer tout document mal formé. Une fois le modèle sélectionné, nous entraînons un modèle neuf sur l'ensemble des données afin d'effectuer de futures prédictions sur de nouvelles données. Nous disposons désormais d'un modèle capable de prédire la qualité d'un document. Nous l'exploitons dans les étapes suivantes.

3.4 Prétraitement et attribution de labels

Cette étape a pour objectif le prétraitement complet du jeu de données ainsi que l'attribution de labels à chaque document. À chaque document, nous indiquons, dans un champ, s'il est prédit comme étant un article de qualité par le modèle entraîné à l'étape précédente.

L'objectif du jeu de données *NewsID* est de permettre d'évaluer le modèle de classification sur la tâche d'attribution d'auteur. Il est donc nécessaire de disposer d'informations sur l'origine d'un article. Nous avons à notre disposition deux types d'informations : la source, généralement le journal (représenté par le nom de domaine lié à l'article) et les auteurs. Afin d'inclure un maximum de documents dans notre jeu de données, nous exploitons ces deux informations. Les sources présentent, comme les auteurs, une consistance de style dans leur écrit comme expliqué au chapitre 2. Ainsi, il est pertinent de considérer également ces labels dans notre jeu de données. Nous créons deux labels à partir de ces informations :

domain Le nom de domaine de l'article extrait de l'*url* (ou directement disponible pour *BlogCorpus*). Le nom de domaine fait généralement référence à un journal en ligne ou au nom d'un blog.

authorial_domain Le premier auteur concaténé au nom de domaine du document. Le caractère "●" permet la séparation. Si aucun auteur n'a été extrait du document, ce label est alors non défini. Cette concaténation nous permet d'éviter les auteurs homonymes entre domaines.

Par la suite, pour chaque document, seul l'un de ces deux labels constituera l'unique label attribué au document. Un document aura donc pour label son auteur

11. Implémentation d'un *SVM* (Cortes et Vapnik, 1995) basée sur la bibliothèque *libsvm* avec un noyau linéaire. La documentation de ce modèle est disponible à l'adresse <https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>.

ou son nom de domaine (correspondant généralement à un journal en ligne). Nous détaillons le processus qui a permis ce choix.

Tout d’abord, nous créons deux listes. Une liste *ads* contenant tous les *authorial_domain* que nous voulons conserver, une seconde liste *domains* contenant tous les *domain* que nous voulons conserver. Ces listes nous permettront, par la suite, d’éliminer les documents qui n’ont pas au moins leur *authorial_domain* dans *ads* ou leur *domain* dans *domains*.

Dans la liste *ads*, nous ajoutons les *authorial_domain* respectant ces conditions (pour les documents ayant comme label l’*authorial_domain* en question) :

1. Il existe au moins 50 documents de qualité.
2. Il existe au moins 60% de documents de qualité. Nous avons bien entendu conservé les articles de mauvaise qualité afin de pouvoir calculer cette proportion. Ainsi, il nous est possible d’éliminer l’ensemble des documents d’un *authorial_domain* donné si la proportion d’articles de qualité montre la présence d’une forte majorité d’articles de mauvaise qualité. Nous pouvons ainsi considérer les articles de qualité de ce même *authorial_domain* comme de mauvaise qualité. Cette considération permet d’éviter les faux positifs, dans les cas où des articles sont jugés de bonne qualité mais ne présentant aucun intérêt pour cette étude.
3. Il existe au moins 8 *authorial_domain* sous-jacents au domaine lié. C’est-à-dire qu’il existe au moins huit auteurs sous le même nom de domaine. En effet, nous ne conservons un auteur que si suffisamment d’autres auteurs du même domaine existent. Ce seuil nous permet d’éviter de retenir des auteurs extraits automatiquement qui ne correspondent en réalité qu’à une catégorie spécifique d’un site d’actualité ou d’un blog (annonce des propriétaires du *site web*, éditorial) ou dans le cas où l’auteur extrait est commun à tous les articles et dont l’orthographe change au cours du temps.

Dans la liste *domains*, nous ajoutons tous les *domain* liés aux *authorial_domain* de la liste *ads*. Puis nous ajoutons les *domain* respectant ces conditions (pour les documents ayant comme label le *domain* en question) :

1. Il existe au moins 200 documents de qualité.
2. Il existe au moins 60% de documents de qualité.

Chaque document de qualité ayant son *authorial_domain* dans la liste *ads* ou son domaine dans la liste *domains* est alors conservé et prétraité comme suit :

- Les balises *html* sont filtrées.
- Les caractères ou suite de caractères spécifiques au *html* sont remplacés par leur équivalent *utf-8*.
- Les suites de plus de trois mêmes caractères sont réduites à une suite de trois.
- Les émoticônes composées de plusieurs caractères *ASCII* sont remplacées par leur équivalent *utf-8*.
- Les différentes apostrophes et *quotes* sont normalisées par leur plus proche équivalent *ASCII*.
- Les accents sont supprimés.
- Les *urls* sont supprimées.
- Les caractères spéciaux ou mots contenant des caractères spéciaux sont supprimés.

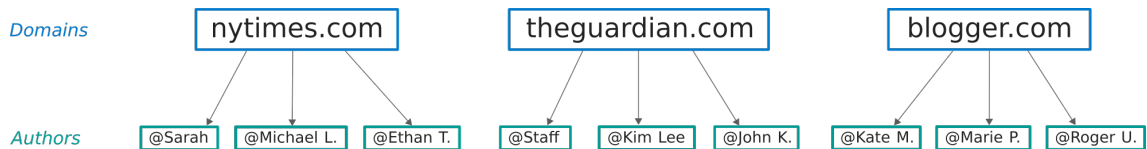


FIGURE 3.1 – Arbre de sous-jacence des labels de *NewsID*

Puis chaque document est tokenisé. La *tokenization* en phrases est effectuée par la bibliothèque *NLTK*¹² en *Python*. La *tokenization* des phrases en mots est effectuée par la bibliothèque *spaCy*¹³ en *Python*. Les documents tokenisés sont filtrés comme suit :

- Les suites de *tokens* ne comprenant pas de lettre sont réduites au premier *token*.
- Les prix, les nombres, les emails et les localisations internet (ou *net locations*) sont remplacés par des *tokens* spécifiques.
- Les suites de plus de deux phrases courtes (moins de deux *tokens*) sont éliminées.

Seuls les documents de plus de 25 *tokens* et de plus de 3 phrases sont conservés. Les documents doublons sont supprimés. De nouveau, afin de prendre en compte le fait que des documents ont été supprimés, nous éliminons les *authorial_domain* de la liste *ads* qui ont moins de 50 documents ainsi que les *domain* de la liste *domains* qui ont moins de 1 000 documents. Nous éliminons également les *authorial_domain* de la liste *ads* pour lesquels il existe moins de 8 *authorial_domain* sous-jacents au domaine lié. Nous donnons un identifiant unique à chaque document.

Le label définitif de chaque document est déterminé comme suit :

1. Si son *authorial_domain* est présent dans la liste *ads*, le label est alors son *authorial_domain*.
2. Sinon, si son *domain* est présent dans la liste *domains*, son label est alors son *domain*.
3. Sinon le document est éliminé du jeu de données.

Chaque document est alors assigné soit à un *authorial_domain*, soit à un *domain*. Si nous considérons l'arbre de sous-jacence entre domaines et auteurs illustré sur la figure 3.1, alors chaque document est situé sur un nœud du niveau *Domains* ou du niveau *Authors*. Enfin, dans l'objectif de ne pas obtenir de déséquilibre trop grand en termes de nombre de documents par label, nous limitons le nombre de documents par label à 30 000.

Nous appelons « labels » les labels associés aux documents dans tous les jeux de données utilisés. Ces labels sont utiles lorsqu'il s'agit d'apprendre un modèle ou de vérifier la qualité d'un partitionnement à partir des données. Les labels sont les données attachées aux « exemples » (i.e. les documents) dans le jeu de données et réfèrent à la classe des exemples (i.e. leur auteur). Ces labels peuvent être de type *domain* ou de type *authorial_domain*, et correspondront donc soit à des journaux en

12. *Natural Language Toolkit (NLTK)* est une bibliothèque en *Python* implémentant différents outils de traitement automatique des langues. Disponible à l'adresse <https://www.nltk.org>.

13. *spaCy* est une bibliothèque *Python open source* pour le traitement avancé du langage naturel. Disponible à l'adresse <https://spacy.io>.

ligne, soit à des blogs, soit à des rédacteurs (i.e. personnes physiques) associés à un journal ou à un blog.

3.5 Filtrage des indices révélateurs

3.5.1 Méthodologie de filtrage

Afin de tester l'*hypothèse de filtrage*, nous proposons de filtrer les indices trop évidents¹⁴ des documents de *NewsID* pour la classification par auteur. Nous voulons éliminer les phrases contenant des *n-grams* pour $n \in \{1, 2, 3\}$ qui ont une haute valeur TFIDF. Ainsi, les phrases portant de trop forts indices sur la classe du document (son auteur) seront éliminées, rendant le label du document plus difficile à prédire. Nous choisissons d'éliminer les phrases entières et non les mots afin de conserver la structure des phrases. Nous aurions pu conserver ces phrases et ne masquer que les mots trop évidents avec des *tokens* de masquage comme le propose [Stamatatos \(2018\)](#). Mais ce choix imposerait de :

1. soit effectuer le même type de prétraitement pour les documents nouveaux dont on veut extraire des caractéristiques de style, ce qui aurait pour effet de dénaturer les documents et d'éliminer des indices potentiellement utiles ;
2. soit de conserver la forme originelle des documents nouveaux, sans la présence du *token* de masquage, ce qui aurait pour effet de créer une légère incohérence entre données d'entraînement et de test.

De plus, la suppression des phrases permet d'éliminer des phrases récurrentes pour certains auteurs telles que les mentions légales ou les invitations à commenter l'article. Nous considérons que ces phrases ne sont pas pertinentes pour la représentation du style.

L'objectif de l'algorithme présenté à cette étape est donc de disposer d'une liste de *n-grams* noirs pour chaque classe du jeu de données de référence (*R-set*). Ces *n-grams* noirs seront extraits des vecteurs TFIDF selon leur valeur et un seuil de filtrage donné. Ils permettront ensuite d'éliminer des phrases du jeu de données. Nous choisissons de considérer non seulement les *1-grams* mais également les *n-grams* avec $n > 1$ afin de pouvoir capturer des suites de mots significatives. En effet, par la suite, si l'algorithme de prédiction se base sur un réseau de neurones récurrents, il sera alors possible de représenter des informations en séquence. Prenons par exemple ces deux documents :

President cooks pasta. My friend talks about politics.
My friend cooks pasta. President talks about politics.

La suite de mots "*President cooks pasta*" portera une information sémantique particulièrement unique et inhabituelle, ce que les réseaux de neurones récurrents (ou réseaux de neurones profonds de type *transformers*) sont capables de représenter. Ces trois mots pris séparément, comme dans le deuxième document, porteront une information sémantique tout à fait différente. De plus, nous pouvons imaginer que cette suite de mots puisse être le gimmick d'un auteur, ce qui l'identifiera de façon

14. L'implémentation de cette méthode de filtrage est disponible sous la forme d'une bibliothèque à l'adresse <https://github.com/hayj/AuthFilt>

probable. Or, les vecteurs TFIDF de ces deux documents seront exactement les mêmes avec la seule utilisation du vocabulaire des *1-grams*.

À cette étape, afin de calculer les valeurs TFIDF et ainsi extraire des *n-grams* noirs de chaque classe, nous devons considérer que chaque document du corpus est la concaténation de l'ensemble des documents d'une même classe dans *NewsID*. Nous appellerons ces documents les *class-documents*. Le calcul des vecteurs TFIDF de ces *class-documents* est coûteux étant donné le nombre de documents constituant le jeu de données *NewsID* et la taille des *class-documents* construits. Nous rappelons que disposer d'un corpus de référence est nécessaire afin que le réseau de neurones profond, entraîné dessus, puisse capturer les structures stylométriques latentes des auteurs de référence. Or, il est difficile de distribuer la construction du vocabulaire et le calcul TFIDF sur un corpus aussi large. C'est d'autant plus le cas lorsque l'on prend en compte tous les éléments du vocabulaire des *2-grams* et *3-grams*. Enfin, le dernier inconvénient lié au calcul des valeurs TFIDF sur l'ensemble du jeu de données est que les documents ne sont pas homogènes en nombre de mots. Il existe un fort déséquilibre en termes de quantité de documents entre les classes, même si ce déséquilibre a été réduit par le processus de la section précédente. Nous avons observé qu'un tel déséquilibre ne permettait pas de choisir un seuil de filtrage unique pour l'extraction des *n-grams* noirs parmi l'ensemble des classes, c'est-à-dire un seuil de filtrage qui permettrait de choisir des *n-grams* noirs qui sont tous autant « révélateurs » d'une classe. En effet, le choix d'un seuil TFIDF lorsque les *class-documents* sont très déséquilibrés conduit à une différence dans la sélection des *n-grams* noirs entre classes, que ce soit en nombre de *n-grams* noirs que dans leur réelle spécificité vis-à-vis de la classe.

Une solution simple à ces différents problèmes est de créer des groupes de documents équilibrés. Chaque groupe sera constitué d'un sous-ensemble des classes et d'un sous-ensemble des documents d'une classe. La création de groupes permet à la fois de répartir la charge de calcul mais également d'équilibrer la taille des *class-documents*. Ils nous permettent d'obtenir des valeurs TFIDF comparables et ainsi choisir un seuil de filtrage pour chacun d'entre eux. Ils permettent aussi d'éviter d'avoir recours à des méthodes telles que le *feature hashing* (Weinberger et al., 2009) qui a pour objectif de distribuer le calcul, mais ne permet pas une prise en compte du vocabulaire entier.

Le filtrage du *R-set* se compose de trois étapes :

1. La génération de groupes avec un nombre limité de classes, un nombre limité de documents et un nombre équilibré de *tokens* par classe.
2. Pour chacun de ces groupes, le calcul des poids TFIDF du vocabulaire des *1-grams*, des *2-grams* et des *3-grams* sur les *class-documents*¹⁵ de chaque classe dans le groupe.
3. Pour chacun de ces groupes, l'extraction des *n-grams* qui sont les plus révélateurs de leur classe, i.e. ayant un poids TFIDF élevé. Nous sélectionnons ces *n-grams*, que nous appelons « *n-grams* noirs », en utilisant un seuil sur les poids TFIDF. Nous choisissons ce seuil de sorte que lorsque l'on supprime les phrases contenant au moins un des *n-grams* noirs sélectionnés, une certaine proportion des phrases dans le groupe est supprimée. Cette proportion (ou

15. Le *class-document* d'une classe est la concaténation de tous les documents appartenant à cette classe.

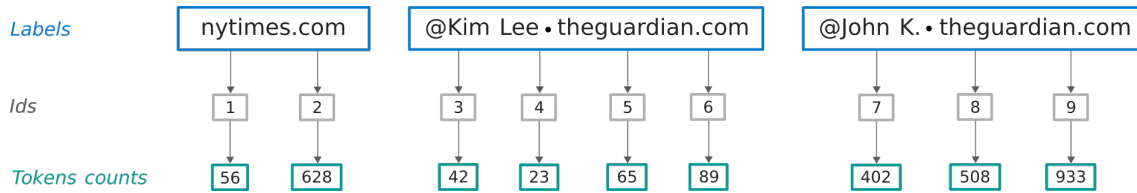


FIGURE 3.2 – Structure de données de type *TokensCount-structure* et exemple sur trois labels et neuf identifiants

« ratio ») est un paramètre que nous choisissons à l’avance.

Idéalement, un algorithme générant cet ensemble de groupes devrait respecter ces quelques propriétés :

Représentativité Créer des groupes de taille suffisante pour que le vocabulaire soit correctement représenté dans un groupe, c’est-à-dire qu’il n’y ait pas un nombre important de mots absents du groupe mais présent dans la plupart des autres groupes. Pour cela, 100 000 documents semble être le nombre minimal permettant de respecter cette propriété.

Restriction Créer des groupes de taille suffisamment réduite afin d’éviter une surcharge de calcul sur un groupe pour le calcul des vecteurs TFIDF.

Équilibre Chaque *class-document* de chaque groupe doit compter le même nombre de *tokens* à $\pm 5\%$.

Complétude Chaque document doit être sélectionné dans au moins un des groupes.

Convergence L’algorithme doit se terminer en un temps fini.

3.5.2 L’algorithme *DocDist*

Nous proposons l’algorithme *DocDist* (*document distribution*). Cet algorithme utilise des structures de type *TokensCount-structure*. Une structure *TokensCount-structure* est un dictionnaire clef-valeur dont les clefs sont les labels et les valeurs sont des dictionnaires clef-valeur. Chacun de ces dictionnaires est composé des identifiants des documents appartenant au label. À chaque identifiant est associé un unique nombre qui est le nombre de *tokens* que compose le document. La figure 3.2 illustre la structure de données *TokensCount-structure*. Cette structure est calculée à partir du jeu de données *NewsID*. La norme d’une structure *TokensCount-structure* a , notée $|a|$, sera le nombre de *tokens* de tous les documents composant la structure. Nous noterons $a[l]$ le dictionnaire des documents correspondant à la classe l dans une structure *TokensCount-structure* a . De la même manière, la norme d’un dictionnaire de documents, notée $|a[l]|$, sera le nombre total de *tokens* des documents de la classe l .

L’algorithme 1 correspond au pseudo-code de *DocDist*. Cet algorithme utilise deux structures *TC-struct* :

remaining correspondant aux documents n’ayant pas encore été sélectionnés dans un groupe.

selected correspondant aux documents déjà sélectionnés dans au moins un des groupes.

Algorithm 1 Algorithme de distribution optimale des documents

```
1: procedure DocDist(remaining : TC, maxTokensPerBucket : integer, variance-  
   Ratio : float)  
2:   selected  $\leftarrow$  new empty TC-struct  
3:   buckets  $\leftarrow$   $\emptyset$   
4:   while remaining is not empty do  
5:     bucket  $\leftarrow$  makeBucket(remaining, selected, maxTokensPerBucket)  
6:     ok  $\leftarrow$  isValidBucket(bucket, varianceRatio)  
7:     changed = false  
8:     if ok then  
9:       newRemaining, newSelected  $\leftarrow$  copy of remaining, selected  
10:      Adding bucket's ids in newSelected  
11:      Removing bucket's ids from newRemaining  
12:      changed  $\leftarrow$   $|newRemaining| - |remaining| \neq 0$   
13:      if changed then  
14:        buckets  $\leftarrow$  buckets  $\cup$  {bucket}  
15:        remaining, selected  $\leftarrow$  newRemaining, newSelected  
16:      if  $\neg(ok \wedge changed)$  then  
17:        remaining  $\leftarrow$  prune(remaining, bucket)  
18:   return buckets  
19: end procedure
```

La création des différents groupes consiste à exécuter l'algorithme 1 en donnant en paramètre la structure *remaining* initiale constituée de l'ensemble des labels et documents du corpus ainsi qu'un nombre déterminé à l'avance *maxTokensPerBucket* désignant le nombre maximum de *tokens* que doit contenir un groupe. L'algorithme crée itérativement des groupes de documents, jusqu'à épuisement de la structure *remaining*, et les ajoute dans la liste *buckets*. La structure *selected* permet de retenir les documents déjà assignés. Un groupe est créé par la fonction *makeBucket* qui prend les deux structures citées précédemment ainsi que la variable *maxTokensPerBucket*. Nous détaillerons la fonction *makeBucket* par la suite. Chaque groupe est ajouté ou passé selon deux critères : s'il a permis de retirer des documents de la structure *remaining* et si le groupe est considéré valide. Pour terminer, l'algorithme renvoie l'ensemble des groupes générés.

La fonction *isValidBucket* utilisée à la ligne 6 dans l'algorithme 1 permet de connaître la validité d'un groupe selon l'écart à la moyenne du nombre de *tokens* de chaque label sélectionné. Si le compte de *tokens* d'un des labels dépasse d'un certain ratio défini par le paramètre *varianceRatio* en dessous ou au-dessus de la moyenne, alors le groupe sera considéré comme non valide. Cette vérification permet le respect de la propriété d'équilibre. En effet, les *class-documents* auront un nombre de *tokens* à peu près équivalent dans chaque groupe.

À la ligne 17, la fonction *prune* prend en paramètre la structure *TC-struct remaining* ainsi que le groupe *bucket* courant et renvoie une nouvelle structure *TC-struct remaining* élaguée. Cet élagage permet de garantir la propriété de convergence. En effet, il est possible qu'aucun document de *remaining* n'ait été sélectionné par *makeBucket*, il est donc nécessaire de supprimer des documents afin de ne pas boucler sur des sélections similaires. La fonction *prune* supprime le plus long document et le

document le plus court de la classe qui a la plus grande divergence à la moyenne dans le groupe *bucket* courant en termes de nombre de *tokens* par labels. Malheureusement, cette fonction empêche l'algorithme de garantir la propriété de complétude, cependant, sur le jeu de données *NewsID*, cette fonction n'a finalement supprimé qu'un nombre négligeable de documents.

3.5.3 Génération d'un groupe

La fonction *makeBucket*, dont le pseudo-code est donné dans l'algorithme 2, permet la génération d'un groupe selon les deux structures *remaining* et *selected*. À la ligne 2, la variable *maxLabelsPerBucket* garantit la propriété de restriction puisque seulement un sous-ensemble de tous les labels disponibles sera retenu dans le groupe. À partir de la ligne 6, nous sélectionnons les labels qui constitueront notre groupe. À la ligne 6, nous sélectionnons des labels dans *remaining* en donnant la priorité aux labels qui ont le moins de *tokens*. Cela permet d'éliminer en priorité les labels ayant peu de documents restants.

À la ligne 10, nous sélectionnons des labels dans *selected* afin de compléter la sélection précédente et ainsi d'obtenir suffisamment de labels afin de garantir la représentativité. À cette ligne, la priorisation des labels ayant le plus de *tokens* dans *selected* permet de disposer de labels ayant une grande combinaison de documents possible et ainsi garantir un équilibrage facile avec les labels déjà sélectionnés. À la ligne 14, nous calculons le nombre minimum de *tokens* que nous allons sélectionner par label. Enfin, à la ligne 23, pour chaque label choisi, nous sélectionnons les documents qui constitueront un *class-document*, en optimisant le nombre de *tokens*. Nous commençons par sélectionner les documents dans *remaining*, puis nous complétons avec ceux dans *selected*. La fonction *makeBucket* renvoie les documents sélectionnés dans la structure *TC-struct bucket*.

À noter qu'un groupe aura toujours au moins un label de *remaining* mais pourra n'être composé que de documents déjà sélectionnés issus de *selected* s'il est impossible de sélectionner des documents de *remaining* en garantissant la propriété d'équilibre. En effet, la taille des documents, très hétérogène, ne permettra pas toujours de consommer *remaining*, une combinaison de documents équilibrée pouvant être impossible à trouver, même en complétant avec *selected*.

La fonction *optIdsSelection* (pour "*optimal ids selection*") utilisée aux lignes 25 et 27 est un algorithme qui cherche à sélectionner un ensemble de documents représentés par leur identifiant en étant le plus proche d'un nombre de *tokens* cible. Elle prend donc en entrée ce nombre de *tokens* cible et un dictionnaire de documents. Pour rappel, les clés sont les identifiants des documents candidats et chaque valeur le nombre de *tokens* que compose le document. Cette fonction prend un troisième paramètre optionnel indiquant quels identifiants ont déjà été sélectionnés. En sortie, nous obtenons un dictionnaire de documents correspondant à un sous-ensemble des documents candidats ainsi que de ceux déjà sélectionnés. L'algorithme étant implémentable de plusieurs manières, nous n'en donnons pas le pseudo-code. Pour *NewsID*, nous avons implémenté une version simple qui consomme aléatoirement des documents et s'arrête lorsque la somme des *tokens* dépasse le nombre de *tokens* cible. Cette version s'est trouvé être suffisante en pratique puisqu'elle permettait une convergence de *DocDist* suffisamment rapide sur *NewsID*. Plusieurs itérations de cette sélection sont lancées. La meilleure sélection est renvoyée en sortie.

Algorithm 2 Génération d'un groupe

```
1: procedure MAKEBUCKET(remaining : TC, selected : TC, maxTokensPerBucket : integer)
2:    $maxLabelsPerBucket \leftarrow \frac{\text{Count of labels in } remaining \text{ and } selected}{4}$ 
3:   if  $maxLabelsPerBucket < 4$  then
4:      $maxLabelsPerBucket \leftarrow 4$ 
5:    $chosenLabels \leftarrow \emptyset$ 
6:    $labels \leftarrow$  labels of remaining in ascending order of tokens count
7:   while  $|chosenLabels| < maxLabelsPerBucket$  do
8:      $chosenLabels \leftarrow chosenLabels \cup labels_1$ 
9:     Removing  $labels_1$ 
10:   $labels \leftarrow$  labels of selected in descending order of tokens count
11:  while  $|chosenLabels| < maxLabelsPerBucket$  do
12:     $chosenLabels \leftarrow chosenLabels \cup labels_1$ 
13:    Removing  $labels_1$ 
14:   $minTokens \leftarrow -1$ 
15:  for label in chosenLabels do
16:     $c \leftarrow |remaining[label]| + |selected[label]|$ 
17:    if  $minTokens = -1$  or  $c < minTokens$  then
18:       $minTokens \leftarrow c$ 
19:   $totalTokens \leftarrow minTokens * |chosenLabels|$ 
20:  if  $totalTokens > maxTokensPerBucket$  then
21:     $minTokens \leftarrow \lfloor \frac{maxTokensPerBucket}{|chosenLabels|} \rfloor$ 
22:  bucket  $\leftarrow$  new empty TC-struct
23:  for label in chosenLabels do
24:    if  $label \in remaining$  and  $|remaining[label]| > 0$  then
25:       $bucket[label] \leftarrow optIdsSelection(remaining[label], minTokens)$ 
26:    if  $label \in selected$  and  $|selected[label]| > 0$  then
27:       $bucket[label] \leftarrow optIdsSelection(selected[label],$   

 $minTokens, bucket[label])$ 
28:  return bucket
29: end procedure
```

Sur *NewsID*, étant donné le grand déséquilibre existant entre les classes issues des *authorial_domain* (entre 50 et 2000 documents par *authorial_domain*) et ceux issus des *domain* (entre 1000 et 30000 documents par *domain*), nous avons commencé par créer deux premières structures *TokensCount-structure*. La première correspond à une structure ayant pour clefs l'ensemble des *domain* présents dans les labels de *NewsID* et la seconde correspond à une structure ayant pour clefs tous les *authorial_domain* présents dans les labels de *NewsID*. Une troisième et dernière structure est initialisée, ses clefs sont l'ensemble des *domain* liés aux *authorial_domain* présents dans les labels de *NewsID*. En effet, il est pertinent de pouvoir identifier les *n-grams* noirs des journaux communs à plusieurs *authorial_domain* en plus de ceux spécifiques à un auteur.

3.5.4 Génération des n -grams noirs

À partir de ces trois listes de groupes, nous devons générer un dictionnaire clef-valeur contenant tous les n -grams noirs. Ce dictionnaire de n -grams noirs aura pour clef les labels présents dans *NewsID* et pour valeur une liste de n -grams. Ce dictionnaire nous permettra par la suite de filtrer le jeu de données. Un n -gram noir est un n -gram souvent employé par un auteur et très peu par l'ensemble des autres, qui permet donc d'identifier facilement celui-ci dans une tâche de classification. Un n -gram avec $n > 1$ sera la concaténation sous forme de chaîne de caractères des *tokens* composant le n -gram avec un espace permettant la séparation. Sa reconstitution peut se faire par le découpage de la chaîne de caractères par le caractère espace. Nous avons choisi de représenter les n -grams ainsi afin de facilement les indexer dans des structures de type table de hachage telles que les dictionnaires en *Python*.

Algorithm 3 Génération des n -grams noirs

```
1: procedure GENERATEBLACKNGRAMS(classDocuments, minNgrams, maxNgrams, delRatio)
2:   tfidf  $\leftarrow$  new dictionary
3:   cumDists  $\leftarrow$  new dictionary
4:   for ngrams  $\leftarrow$  minNgrams to maxNgrams do
5:     ngDocs  $\leftarrow$  generate  $n$ -grams of classDocuments
6:     tfidf[ngrams]  $\leftarrow$  compute TFIDF vectors of ngDocs
7:     maxVals  $\leftarrow$  find max TFIDF value for each sentence in ngDocs
8:     cumDists[ngrams]  $\leftarrow$  compute the TFIDFCumDist-struct for ngrams
9:     blackNgrams  $\leftarrow$  search black  $n$ -grams using delRatio, cumDists, tfidf
10:  return blackNgrams
11: end procedure
```

L'algorithme 3 *generateBlackNgrams*¹⁶ prend en entrée la liste des *class-documents* d'un groupe. À noter que la position d'un *class-document* dans la liste *documents* correspondra donc à une classe. Chaque *class-document* est une liste de phrases. Une phrase est composée de *tokens*. L'algorithme prend également un nombre minimum et maximum de n -grams à considérer pour le vocabulaire ainsi qu'un ratio de suppression. Le ratio de suppression permet d'indiquer quelle proportion de phrases sera filtrée par les n -grams noirs sélectionnés lorsque ceux-ci seront utilisés pour filtrer les phrases. Par exemple, pour un ratio de 0.3 (qui a été choisi pour *NewsID*), l'algorithme *generateBlackNgrams* cherchera les seuils de valeur TFIDF permettant la sélection des n -grams noirs éliminant 30% des phrases au total. Nous avons choisi de supprimer 30% des phrases du corpus de référence car il correspond, selon nous, à un bon compromis. Ainsi, le filtrage du corpus peut avoir un impact significatif lors de la phrase d'entraînement, mais évite l'élimination d'un trop grand nombre de phrases qui peuvent véhiculer le style des auteurs. L'algorithme 3 renvoie une liste de listes de n -grams correspondants aux n -grams noirs de chaque *class-document*. Nous exécutons l'algorithme *generateBlackNgrams* pour chaque groupe en parallèle.

À la ligne 6, le calcul des vecteurs TFIDF est fait sur les documents aplatis, c'est-à-dire les phrases concaténées. Nous ne donnons aucune contrainte sur la taille du

16. Le code *Python* de l'algorithme *generateBlackNgrams* est disponible à l'adresse <https://github.com/hayj/NLPTools>

vocabulaire. Le vocabulaire correspondra à l'ensemble des n -grams pour un n donné. Pour *NewsID*, nous avons choisi d'extraire les n -grams noirs de 1 à 3. Les variables d'entrée $minNgrams$ et $maxNgrams$ sont donc initialisées à respectivement 1 et 3.

Nous calculons $\forall s \in S, TFIDF_{max}(s)$ à la ligne 7, avec S l'ensemble de phrases du corpus de documents et $TFIDF_{max}$ la fonction donnant la valeur TFIDF maximale des *tokens* composant la phrase donnée en paramètre. À la ligne 8, nous utilisons ensuite une structure de données que nous appelons *TFIDFCumDist-struct* (*TFIDF Cumulative Distribution*). Cette structure de données est un dictionnaire clef-valeur qui associe à chaque valeur TFIDF x le nombre de phrases qui ont pour valeur TFIDF maximale (parmi les valeurs TFIDF des *tokens* que compose la phrase) une valeur supérieure ou égale à la valeur x . Les valeurs, en clefs dans ce dictionnaire, correspondront à un intervalle de valeurs TFIDF. Plus formellement, cette structure de données représentera une approximation discrétisée de la fonction de répartition (ou *cumulative distribution*) :

$$f : \mathbb{R} \rightarrow \mathbb{N}$$

$$x \mapsto |\{s : s \in S, TFIDF_{max}(s) \geq x\}|$$

L'objectif est de chercher un y tel que :

$$y = \underset{x}{\operatorname{argmin}}(\operatorname{abs}(f(x) - r \cdot |S|))$$

avec abs la fonction valeur absolue et r le ratio choisi (0.3 pour *NewsID*). La résolution de cette équation est rendue constante (un nombre d'itérations correspondant au nombre d'intervalles choisis¹⁷) grâce à la construction de la structure de données *TFIDFCumDist-struct*. En effet, il suffit de parcourir itérativement la structure dans l'ordre jusqu'à un compte de phrases correspondant à 30% des phrases du corpus et de retenir la valeur TFIDF correspondante.

À la ligne 9, nous générons des structures de n -grams noirs¹⁸ en utilisant *tfidf* et *cumDists*. Prenons par exemple un ratio de suppression de 0.3, la difficulté est de trouver un nouveau ratio de suppression de telle sorte que les n -grams noirs générés puissent supprimer effectivement 30% des phrases. Ce n'est évidemment pas le cas si nous faisons une extraction indépendante à partir du vocabulaire des 1-grams, puis des 2-grams et enfin des 3-grams. En effet, avec un ratio de suppression de 0.3, nous commençons par générer les 1-grams noirs permettant de supprimer 30% des phrases du corpus, puis nous générons les 2-grams noirs supprimant 30% des phrases et enfin nous générons les 3-grams noirs supprimant 30% des phrases. L'ensemble de ces n -grams (de 1 à 3 pour *NewsID*), assemblés par *class-document* en une seule et unique structure de n -grams noirs, supprimera finalement plus de 30% des phrases s'il existe des phrases ayant des n -grams en collision¹⁹ avec une structure de n -grams parmi celles générées mais pas toutes. En reprenant l'exemple donné précédemment, si l'on trouve une phrase ayant le 3-gram "*President cooks pasta*", celui-ci pourra

17. Le nombre d'intervalles choisis pour *NewsID* est de 1000

18. Une structure de n -grams noirs est une liste de même taille que le nombre de *class-documents*. Un élément de cette liste est constitué des n -grams noirs relatifs au *class-document* dont la position dans la liste correspond. Elle est à différencier de l'objectif final qui est d'obtenir un dictionnaire clef-valeur mettant en relation des labels à des n -grams noirs.

19. Une phrase, appartenant à un *class-document* donné, est dite en collision avec une structure de n -grams noirs si l'un de ses n -grams appartient à la liste des n -grams noirs liés au *class-document* dans la structure.

appartenir à la structure de 3-grams noirs sans qu'aucun des mots, individuellement, soit présent dans la structure des 1-grams noirs. Cette phrase sera donc une phrase de plus à supprimer lorsque seront générés les 3-grams noirs, rendant le ratio de suppression initial caduc.

Afin de trouver ce nouveau ratio de suppression, nous faisons donc une recherche dichotomique entre 0 et *delRatio* jusqu'à trouver les *n-grams* noirs supprimant une proportion de phrases correspondant au ratio de suppression initial. Indépendamment, pour chaque *n* donné, les *n-grams* noirs sont générés selon la procédure décrite précédemment. Les trois structures de *n-grams* noirs générés avec le nouveau ratio de suppression sont fusionnées (i.e. les listes de chaque *class-document* sont concaténées) et renvoyées par l'algorithme *generateBlackNgrams*.

Finalement, pour chaque liste de groupes, nous collectons autant de structures de *n-grams* noirs que la liste comprend de groupes en utilisant *generateBlackNgrams*, et nous fusionnons celles-ci dans un dictionnaire mettant en relation les labels à leurs *n-grams* noirs. Nous procédons ensuite au filtrage du jeu de données *NewsID* en supprimant de chaque document les phrases dont un de ses *n-grams* appartient à la liste des *n-grams* concernée par le label du document. Bien entendu, une condition de suppression particulière est appliquée à la liste de groupes correspondant à l'ensemble des *domain* liés aux *authorial_domain* présents dans les labels de *NewsID* : nous filtrons les phrases des *authorial_domain* en prenant les *n-grams* noirs du *domain* associé. Le jeu de données *NewsID* comprend à la fois, pour chaque document, un champ correspondant aux phrases non filtrées et un champ correspondant aux phrases filtrées.

3.5.5 Résultats du filtrage sur *NewsID*

La figure 3.3 montre les phrases conservées et supprimées de trois documents du *R-set*. Les phrases supprimées sont les phrases en rouge et les phrases conservées sont celles en vert. Les mots soulignés sont ceux appartenant à un des *n-grams* noirs de la classe du document. Comme nous pouvons le voir, certains mots faisant référence au journal en ligne comme "*Washington Post*" et "*The Denver Post*" sont supprimés. Les phrases apparaissant dans beaucoup de documents d'une même classe sont automatiquement supprimées lors du filtrage comme la phrase « *Comments are moderated and may not appear immediately* », réduisant ainsi le nombre de phrases non pertinentes pour une représentation du style. Certains *n-grams* spécifiques à des auteurs comme "*Lauren and Steph*", ou des mots ayant une orthographe particulière comme "*tomorrow*" sont aussi détectés et supprimés par le processus de filtrage.

3.5.6 Avantages de la méthode

La méthode de filtrage que nous proposons est parallélisable (chaque tâche étant totalement indépendante) et offre donc les propriétés de scalabilité requises lorsqu'il s'agit de traiter de larges corpus tels que celui que nous avons collecté. Ce corpus de référence stylistique nécessite d'être grand étant donné que nous cherchons à généraliser des caractéristiques de style à partir d'auteurs de référence. La méthode de filtrage proposée permet de traiter des corpus plus grands avec un impact linéaire sur le temps de calcul. Elle permet de trouver des *n-grams* « révélateurs » d'auteurs (selon un ratio de suppression paramétrable) dans différents sous-ensembles de documents,

Sarah Kaplan@washingtonpost.com

March for Science participants walk along Constitution Avenue to the Capitol on April 22. (Astrid Riecken for The Washington Post). [...] Conceived in the wake of President Trump's inauguration, and galvanized by his efforts to slash environmental protections and cut the federal budget [...] So we want to know: What has changed for you since the March for Science? Have you altered anything about your life or work? Have your colleagues? Do you feel part of a "global movement"? Do you think the March for Science achieved its goals? Please let us know what you think using the form below.

Adrian Dater@denverpost.com

[...] He has been at The Denver Post hockey columnist and sports feature writer. [...] I would caution against everyone getting their hopes up , though. This is Peter Forsberg after all. But he looked MUCH better at practice today than two days ago. I know he worked with doctors on adjustments to the brace on his right foot in the last two days. [...] Comments are moderated and may not appear immediately.

3030784@blogger.com

Hey I am Josh Debner and I am the kid who sits home and does nothing on fri. nights [...] with my family FUN FUN FUN :-/ two weeks in a row now. [...] I had ALL my academy classes... It is block scheduling so having 4 45min classes is much better than 2 1.5 hour ones... I found since i was absent I don't have to make up a lab YAHOO. Hmmm Bus ride home ate lunch. [...] Lauren and Steph went to movies with Mike and Joe.. yea they had fun. Busy day tomorrow. [...]

FIGURE 3.3 – Filtrage de phrases sur trois exemples de documents

ceux-ci étant à la fois suffisamment représentatifs et traitables en un temps court. Les bibliothèques de calcul distribué telles que *Spark* (Zaharia et al., 2016), qui n'utilise pas la notion de *buckets*, ne permettent pas d'effectuer ce type de calcul étant donné qu'elles partagent un vocabulaire entre nœuds. Or, le vocabulaire que nous considérons (jusqu'aux 3-grams) est grand.

Une possibilité alternative à cette méthode de filtrage aurait été d'utiliser la technique du *feature hashing* proposée par Weinberger et al. (2009) et implémentée dans *Spark* (Zaharia et al., 2016), mais celle-ci ne permettait pas une prise en compte du vocabulaire dans son entièreté. Sans cette prise en compte exhaustive, il aurait été nécessaire de supprimer les phrases contenant des mots hors du vocabulaire (ou ceux dont le *hash* est en collision) pour que le corpus de référence soit filtré de façon précise, évitant tout biais dans les expérimentations futures.

Le traitement du corpus de référence (extraction des *n-grams* noirs et suppression des phrases) a été effectué en parallèle sur 30 serveurs de calcul ayant chacun 64 Go de mémoire vive, des processeurs *Intel Xeon Silver 4112* et une partition disque réseau commune (permettant l'accès au jeu de données *NewsID*). L'extraction des *n-grams* noirs dura une journée et la suppression des phrases approximativement une heure. La génération des *buckets* s'est faite sur l'une de ces machines et pris approximativement une heure.

3.5.7 Importance de la difficulté à identifier les auteurs

Pour les algorithmes de classification, l'utilisation des documents filtrés dans les *U-sets* permet d'évaluer les modèles sur des jeux de données plus difficiles. En effet, pour les expérimentations du chapitre 4, nous voulons disposer de jeux de

données suffisamment difficiles dans la tâche d’identification d’auteurs. Comme le montre [Stamatatos \(2018\)](#), la performance des méthodes état de l’art en identification d’auteurs est très fortement affectée par la diversité de thèmes présents dans les écrits d’un auteur. [Stamatatos \(2018\)](#) explique qu’il est difficile de représenter le style d’un auteur sans que celui-ci soit lié aux thèmes utilisés dans ses écrits. Il propose une méthode de masquage des indices de thème tout en préservant les éléments linguistiques pouvant relever du style de l’auteur. Cette méthode permet l’amélioration des méthodes état de l’art en identification d’auteurs.

Dans le même objectif, nous avons filtré *NewsID* afin de disposer de jeux de tests difficiles et ainsi avoir une marge de progression plus grande en termes de performances dans l’identification d’auteurs. En effet, il est courant que, pour obtenir des précisions proches de 100%, les modèles les plus performants soient ceux ne se concentrant que sur des éléments de vocabulaire spécifiques à certains auteurs étant donné qu’ils écrivent généralement sur des thèmes proches, impliquant l’emploi d’un vocabulaire thématique consistant. [Sapkota et Solorio \(2012\)](#) ont par exemple obtenu une précision proche de 100% sur le jeu de données *PAN-2012* ([Inches et Crestani, 2012](#)) dédié à l’identification d’auteurs en utilisant des descripteurs standards tels que des *n-grams*, le compte de phrases, de mots, des étiquettes morpho-syntaxiques, etc. Il est alors très difficile de pouvoir capturer des indices relevant du style lorsque les modèles atteignent une grande précision sur la base d’un ensemble restreint d’indices lexicaux.

3.5.8 Différences avec les méthodes de masquage

La méthode de masquage proposée par [Stamatatos \(2018\)](#) est décrite en section 1.3.3. Contrairement à cette méthode, nous ne masquons pas les mots liés aux thèmes, mais nous supprimons les phrases comprenant des séquences propres aux auteurs en nous basant sur la pondération TFIDF. Disposer de tels jeux de données permet à la fois d’évaluer le pouvoir de généralisation des modèles dans l’identification d’auteurs mais également de garantir leur robustesse lorsqu’ils sont utilisés dans des conditions réelles avec une identification des auteurs difficile, i.e. lorsque les auteurs n’emploient que très peu d’éléments lexicaux propres. Le filtrage des *U-sets* effectué sur *NewsID* nous permet de disposer de ce type de jeu de test.

En définitive, l’objectif final de notre méthode est légèrement différent du masquage proposé par [Stamatatos \(2018\)](#) : nous cherchons à rendre l’identification d’un auteur plus difficile afin d’entraîner un réseau de neurones profond capable de représenter des structures subtiles et consistantes pour un auteur, de sorte qu’il ne se repose pas sur des séquences de mots trop évidentes, ce qui aurait pour conséquence de limiter la représentation du style écrit. Cette méthode n’est pas destinée à améliorer directement les performances d’un modèle dans l’identification d’auteurs connus sur un ensemble de données spécifique, mais à filtrer un corpus de référence afin de mieux capturer les caractéristiques de style d’auteurs inconnus.

3.6 Extraction du *R-set* et des *U-sets*

À partir de *NewsID*, nous extrayons un *R-set* en prenant les classes ayant le plus de documents. Nous réservons cependant quelques domaines pour les *U-sets* en les choisissant manuellement. Dans le *R-set*, nous incluons également des classes

représentées par des *authorial_domain* afin de le rendre suffisamment hétérogène. Nous sélectionnons ceux ayant le plus de documents afin de disposer d'un *R-set* large. Le *R-set* est composé de 1 200 classes et de 3.3 millions de documents avec un maximum de 30 000 documents par classe et une moyenne de 3 000 documents par classe.

Nous extrayons plusieurs *U-sets* en prenant des classes au hasard (non présentes dans le *R-set*) et des documents au hasard appartenant à ces classes. Nous faisons varier le nombre de documents par classe (entre 50 et 200) et le nombre de classes (entre 50 et 200). Ceci nous permet de disposer de *U-sets* pour lesquels la classification est difficile et d'autres pour lesquels la classification est facile²⁰. Certains documents filtrés lors de l'étape précédente ne possèdent plus assez de phrases, ils sont alors supprimés. Enfin, les classes qui ont moins de 50 documents sont éliminées.

Nous obtenons 454 *U-sets* différents dont une majorité est composée à la fois de labels de type *domain* et de labels de type *authorial_domain*. Un nombre plus réduit de *U-sets* dédiés aux auteurs de domaines spécifiques sont également générés : *blogger.com*, *livejournal.com*, *washingtonpost.com*, *breitbart.com*, *businessinsider.com*, *cnn.com*, *guardian.co.uk*, *theguardian.com* et *nytimes.com*. Cet ensemble de *U-sets* permet de mettre à disposition un grand nombre de jeux de données pour la tâche d'identification d'auteurs.

Les *U-sets* utilisés dans le chapitre 4 sont les suivants :

NewsID-50-50 5 *U-sets* rassemblant des labels faisant référence à des journaux (i.e. labels de type *domain*) et des labels faisant référence à des auteurs d'articles d'actualité et de blog (i.e. labels de type *authorial_domain* d'auteurs physiques).

BlogCorpus-50-50 5 *U-sets* du jeu de données *BlogCorpus* (Schler et al., 2006) composé uniquement de labels faisant référence à des auteurs d'articles de blog (i.e. *authorial_domain*).

Lors de nos expérimentations, nous utilisons aussi d'autres *U-sets* spécifiques, qui, comme pour *BlogCorpus-50-50*, ne comprennent que des auteurs physiques (rédacteur dans un journal ou blogueur) d'un même domaine. Ces *U-sets* ne seront donc composés que de labels de type *authorial_domain* avec un auteur en partie gauche et un domaine en partie droite (correspondant à un journal ou un blog) qui sera unique pour un même *U-set*.

LiveJournal-50-50 5 *U-sets* du site web *livejournal.com*, réseau social permettant aux utilisateurs de tenir un blog.

TheGuardian-50-50* et *GuardianUK-50-50 2 *U-sets* des sites web *theguardian.com* et *guardian.co.uk* appartenant à *The Guardian*, journal d'information britannique.

WashingtonPost-50-50 *U-set* du site web *washingtonpost.com*, appartenant au journal américain *The Washington Post*.

Breitbart-50-50 *U-set* du site web *breitbart.com* appartenant à *Breitbart News*, média politique conservateur américain.

20. La tâche de classification (tâche d'identification d'auteurs) sur un *U-set* composé de 50 classes et 200 documents par classe aura tendance à être plus simple que sur un *U-set* composé de 200 classes et 50 documents par classe.

BusinessInsider-50-50 *U-set* du site web d'information américain *businessinsider.com* situé à New York.

CNN-50-50 *U-set* du site web *cnn.com*, journal en ligne du groupe *Cable News Network*.

NYTimes-50-50 *U-set* du site web *nytimes.com* appartenant au journal d'information new-yorkais *The New York Times*.

Dans le nom de chaque *U-set* nous indiquons deux nombres :

1. le nombre de labels différents dans le *U-set* ;
2. le nombre de documents par classe dans le *U-set*.

Certains types de *U-sets* sont multiples, i.e. nous construisons plusieurs *U-sets* de mêmes caractéristiques (ayant les mêmes domaines avec le même nombre d'exemples) mais en répartissant les auteurs dans différents *U-sets*. Ceci nous permet de disposer de plus de *U-sets* pour nos tests sans intersection d'auteurs. C'est le cas des *U-sets* suivants : *NewsID-50-50*, *BlogCorpus-50-50* et *LiveJournal-50-50*.

Dans le jeu de données *NewsID*, nous avons concaténé les auteurs des articles avec le nom de domaine des journaux en ligne et des blogs. Après avoir étudié les labels de notre jeu de données, nous avons remarqué qu'une grande majorité (approximativement 99%) des auteurs communs à plusieurs journaux sont en réalité des homonymes. Ces auteurs utilisent des pseudonymes courts (*Alex*, *Angie*, *Carl*, *Erik*, *Lucy*, etc.). Ainsi, nous n'avons pas différencié les rares cas où un auteur semble avoir écrit pour plusieurs journaux (e.g. *Andrew Restuccia* pour *politico.com* et *thehill.com*) par un étiquetage manuel. Les auteurs sont consistants dans leurs écrits, et comme nous l'avons vu, les journaux en ligne aussi, notamment parce que leurs auteurs doivent suivre un guide de style et une ligne éditoriale. Mais même dans les cas où il est difficile de différencier les articles d'un auteur ayant écrit pour deux journaux différents, nous soutenons que l'existence de deux labels pour ce même auteur n'aura que peu d'impact sur l'apprentissage de nos modèles et la représentation stylométrique des documents. En effet, si le modèle ne parvient pas à établir les différences entre les documents de cet auteur, la conséquence sera que les représentations internes du réseau de neurones profond seront proches pour les documents de l'un ou l'autre des deux labels qu'il doit prédire. La proximité des représentations internes du réseau de neurones profond pour ces deux auteurs de référence n'impliquera pas nécessairement l'ajout de bruit dans la projection des documents « nouveaux » dans les *U-sets*.

3.7 Conclusion

Dans ce chapitre, nous avons présenté le jeu de données *NewsID* construit durant cette thèse ainsi que son prétraitement. Ce jeu de données possède plusieurs caractéristiques importantes pour notre étude sur le style. Tout d'abord, celui-ci est composé principalement d'articles d'actualité et de blogs, ce qui permet de maintenir une cohérence dans le domaine des données utilisées tout au long des différentes expérimentations de cette thèse, et notamment lorsque nous testerons les modèles entraînés par la méthode de ciblage des indices intra-auteurs consistants au chapitre 8. Ensuite, la classification des articles selon leur pertinence nous a permis de retirer le « bruit » (items non pertinents à l'étude du style écrit) du jeu de données. La taille

conséquence de ce jeu de données et le découpage qui a été effectué nous permettent de valider l'*hypothèse de généralisation du style* dans le prochain chapitre. De même, nous validerons l'*hypothèse de filtrage* grâce au filtrage des indices trop évidents à la reconnaissance d'un auteur. Le processus de filtrage proposé est efficient et offre les propriétés de scalabilité requises grâce à la distribution du calcul en *buckets*.

Chapitre 4

Expérimentations sur les tâches de la stylométrie

4.1 Introduction

Dans ce chapitre, nous cherchons à valider l'*hypothèse de généralisation du style* stipulant que les documents d'auteurs inconnus tendent à être similaires, en termes de style, aux documents écrits par les mêmes auteurs de référence. Plus précisément, l'hypothèse stipule que, d'une part, il est possible de représenter de nouveaux documents d'auteurs inconnus en généralisant des traits stylistiques d'un ensemble de documents et d'auteurs connus suivant la propriété de *consistance intra-auteur*, et que, d'autre part, les documents d'un même auteur, même inconnu, tendront à avoir des représentations stylométriques similaires dans cet espace générique.

La méthode d'apprentissage de la représentation proposée (i.e. la méthode de ciblage des indices intra-auteurs consistants) permet de représenter les documents dans un tel espace. Elle permet le partitionnement de documents par auteurs en s'appuyant sur des traits stylistiques. Pour cela, nous entraînons un réseau de neurones profond sur un large jeu de données de référence constitué d'articles d'actualité et de blogs. Ce partitionnement est possible même lorsque le modèle n'est pas entraîné sur les nouveaux auteurs que l'on considère. Comme nous le verrons, cette méthode permet au modèle de généraliser et d'être robuste au changement de domaine (i.e. changement de thème, de genre). Afin de valider empiriquement cette méthode, nous évaluons le modèle proposé sur sa capacité à partitionner les documents d'auteurs inconnus en utilisant des métriques standards de partitionnement ainsi que la nouvelle métrique *SimRank*. En complément, nous testons cette méthode dans le contexte d'apprentissage par transfert pour la tâche d'identification d'auteurs.

Dans ce chapitre, afin de valider l'*hypothèse de généralisation du style*, nous implémentons la méthode de ciblage des indices intra-auteurs consistants sur le partitionnement par auteur en proposant le modèle *SNA* et *DBert-ft* en section 4.2. Dans la section 4.3, nous testons nos modèles sur l'identification d'auteurs et montrons à quel point ces représentations sont bénéfiques dans le cadre de l'apprentissage par transfert. Pour terminer, en section 4.4, nous proposons une expérimentation permettant la validation de l'*hypothèse de filtrage*.

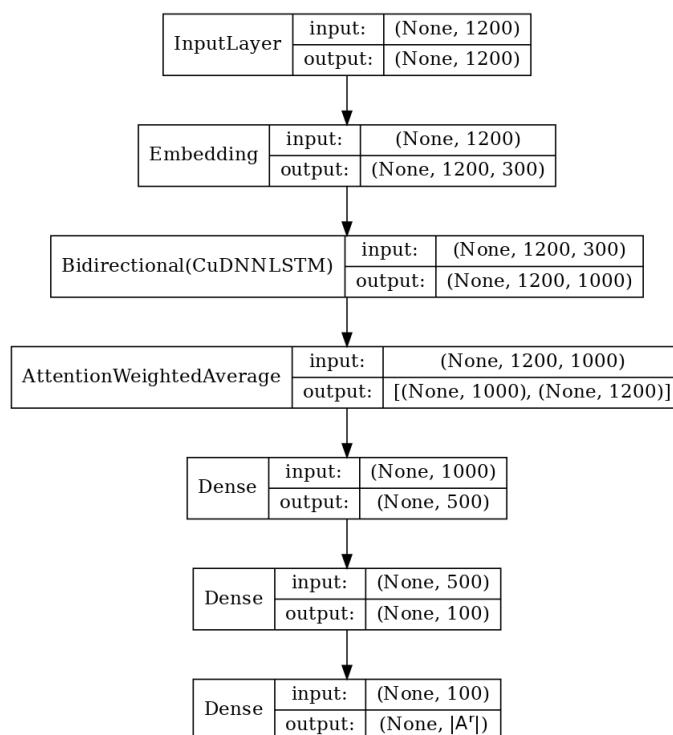


FIGURE 4.1 – *Flow graph* du modèle *SNA* (*Stylometric Neural Attention*)

4.2 Partitionnement par auteur

Dans cette section, nous implémentons la méthode d’apprentissage de la représentation proposée au chapitre 2, la méthode de ciblage des indices intra-auteurs consistants, dans l’objectif de valider l’*hypothèse de généralisation du style*. Nous détaillons les modèles testés et les *baselines* utilisées pour le partitionnement par auteur. Les métriques utilisées pour cette tâche sont celles présentées en section 2.3. Enfin, nous terminons avec le détail des résultats du partitionnement par auteur sur différents *U-sets*.

4.2.1 Le modèle *Stylometric Neural Attention*

Pour tester l’*hypothèse de généralisation du style*, nous devons apprendre la fonction f^r introduite en section 2.3. Dans cet objectif, nous proposons d’entraîner un réseau de neurones profond sur le *R-set* en suivant la méthode d’apprentissage de la représentation. La tâche est donc l’identification d’auteurs. Les données sont les documents représentés par des suites de vecteurs sémantiques et les prédictions en sortie sont les auteurs du *R-set*. L’entraînement du modèle se repose sur la première propriété de notre définition du style : la *consistance intra-auteur*. Le ciblage sur les indices propres aux auteurs est rendu possible par la taille de l’ensemble *R-set* et notamment le grand nombre de documents par auteur. Ainsi, nous dirigeons l’entraînement du modèle de telle sorte qu’il puisse identifier les indices linguistiques consistants de chaque auteur.

Le modèle *SNA* (*Stylometric Neural Attention*) est un modèle basé sur un réseau de neurones récurrents de type LSTM (Hochreiter et Schmidhuber, 1997). Le modèle

est bidirectionnel et est principalement basé sur l'architecture proposée par Zhou et al. (2016). Les entrées du modèle sont les vecteurs sémantiques GloVe 840B (Pennington et al., 2014) de taille 300. Les documents sont réduits à 1 200 mots. La première couche de *SNA* est un LSTM bidirectionnel possédant 500 unités. Étant donné que les indices de style seront localisés à des séquences particulières dans les documents, i.e. l'ensemble du document ne laissera pas nécessairement transparaître le style de l'auteur, nous introduisons une couche d'attention permettant au modèle de focaliser son attention sur des sous-séquences du document. Nous ajoutons deux couches denses possédant 500 unités. La dernière couche est une couche dite *softmax*, chaque dimension de cette couche de sortie correspondra à un auteur dans A^r . La fonction de perte est la *multi-class log loss*.

La figure 4.1 donne plus de précisions sur l'architecture du modèle à travers un *flow graph*¹, notamment les dimensions des entrées et sorties de chaque couche. Dans ce *flow graph*, nous introduisons une couche *Embedding* qui permet de projeter les mots dans un espace vectoriel. Dans le cas où cette couche est présente, les entrées du réseau de neurones sont alors les indices des mots dans le vocabulaire. Cette couche est facultative. Son ajout a pour conséquence de rendre le réseau de neurones plus grand et donc plus difficile à charger en mémoire GPU. Si nous choisissons de retirer cette couche, les entrées du réseau de neurones seront alors les projections des mots (*word embeddings*) directement. La conséquence de l'absence de cette couche est que les projections de mots ne pourront pas être affinées sur le jeu de données utilisé.

Des *dropouts* de probabilité 0.2 sont ajoutés aux sorties des couches à partir de la couche d'attention jusqu'à la dernière. Le modèle est implémenté grâce à la bibliothèque *Keras*² (Chollet et al., 2015) utilisant *TensorFlow*³ (Abadi et al., 2015). À noter que nous utilisons l'implémentation CUDA⁴ de l'architecture LSTM appelée CuDNNLSTM afin d'accélérer l'entraînement de notre modèle sur les serveurs de calcul disposant d'une carte graphique Nvidia⁵.

Finalement, afin de projeter les documents d'un *U-set*, nous nous servons de la représentation interne du modèle afin d'obtenir les représentations vectorielles des documents. Pour des raisons de généralisation, nous ne prenons pas la couche *softmax* comme représentation vectorielle de chaque document du *U-set* mais la sortie d'une couche intermédiaire du réseau de neurones profond, en l'occurrence de la sortie de la couche d'attention.

Nous introduisons un second modèle, une simplification de *SNA* sans couche

1. Un *flow graph* (ou *computation graph*) est la représentation d'un calcul mathématique arbitraire sous forme de graphe. C'est un graphe acyclique dirigé (DAG) dans lequel les nœuds correspondent à des opérations mathématiques ou à des variables et les arêtes correspondent au flux de valeurs intermédiaires entre les nœuds. Cette abstraction permet de représenter les réseaux de neurones profonds.

2. La bibliothèque *Keras* (Chollet et al., 2015) permet d'interagir avec les algorithmes de réseaux de neurones profonds et de machine learning, notamment les algorithmes implémentés dans *TensorFlow*.

3. La bibliothèque *TensorFlow* (Abadi et al., 2015), dotée d'une interface pour *Python* et *Julia*, est un outil *open source* d'apprentissage automatique et permet notamment l'implémentation de réseaux de neurones profonds.

4. CUDA (*Compute Unified Device Architecture*) est une technologie de GPGPU (*General-Purpose Computing on Graphics Processing Units*), i.e. utilisant un processeur graphique (GPU) pour exécuter des calculs généraux à la place du processeur (CPU).

5. Nvidia Corporation est une entreprise spécialisée dans la conception de processeurs graphiques, de cartes graphiques et de puces graphiques.

d'attention et non bidirectionnel. Cette version simple correspond à un LSTM avec deux couches denses et une couche de classification *softmax*. Elle nous permet d'évaluer l'impact de l'ajout de la couche d'attention et de la bidirectionnalité sur les résultats. Afin d'obtenir les représentations vectorielles issues de ce modèle, nous prenons les sorties du LSTM.

4.2.2 Le modèle *DistilBert* affiné

De récentes architectures de réseaux de neurones profonds, les *transformers*, ont montré permettre une meilleure représentation de phrases sur différentes tâches du traitement automatique du langage naturel. Différentes architectures basées sur les *transformers* ont été proposées ces dernières années et des modèles pré-entraînés ont été mis à disposition (Devlin et al., 2019; Yang et al., 2019; Sanh et al., 2019). Ces modèles, par l'apprentissage par transfert, permettent de s'affranchir d'un entraînement sur de larges corpus et ainsi d'exploiter les représentations internes du langage apprises lors de l'entraînement non-supervisé de ceux-ci. Pour cette expérimentation, nous proposons d'utiliser l'un de ces modèles pré-entraînés, d'effectuer un affinage (ou *fine-tuning*) selon la méthode décrite au chapitre 2, et d'évaluer l'efficacité du modèle dans la tâche de partitionnement par auteur.

Nous choisissons d'utiliser comme base le modèle *DistilBert* (Sanh et al., 2019). Nous lui ajoutons une couche *softmax* de classification comme décrit pour *SNA*. Pour cela, nous utilisons la bibliothèque *transformers*⁶ (Wolf et al., 2019) en *Python*. Sanh et al. (2019) décrivent ce modèle comme une version 40% réduite du modèle BERT original, tout en conservant 99% de ses capacités et en étant 60% plus rapide.

Nous appelons ce modèle « *DBert-ft* »⁷. Pour l'entraînement de celui-ci sur le *R-set*, nous laissons l'ensemble des couches entraînaibles pour qu'il puisse ajuster son attention sur les séquences consistantes de chaque auteur. Nous utilisons les mêmes documents que pour l'entraînement de *SNA*. Les documents du *R-set* sont filtrés par la méthode décrite en section 3.5. Le texte est ensuite découpé en sous-mots appelés *wordpieces* (Wu et al., 2016). Le vocabulaire *wordpieces* est de taille 30 000. La suite de *wordpieces* correspond donc à l'entrée de *DBert-ft* et est directement convertie en *wordpieces embeddings* par une couche de type *embeddings* en entrée du réseau de neurones.

Chaque document est réduit à 512 *wordpieces*. Cependant, afin de disposer de plus d'exemples d'entraînement, nous décomposons un exemple d'entraînement du *R-set* en plusieurs tranches de 512 *wordpieces*. Si un document est composé de plusieurs tranches (i.e. le document est composé de plus de 512 *wordpieces*) et que la dernière tranche est de taille inférieure à 50, cette dernière n'est pas retenue. Chaque tranche de document nous donne donc un nouvel exemple d'entraînement avec le même label que celui du document original.

Pour les *U-sets*, nous prenons les représentations internes de la dernière couche du *transformer* en donnant en entrée les *wordpieces* de chaque document. Dans les cas où les documents sont plus grands que 512 *wordpieces*, nous moyennons les

6. La bibliothèque *transformers* est une bibliothèque *Python* rassemblant des réseaux de neurones profonds pré-entraînés pour le traitement automatique du langage naturel. Ces modèles sont utilisables sous *TensorFlow 2.0* et *PyTorch*.

7. Le modèle *SNA* pré-entraîné ainsi que le code ayant permis son entraînement sont mis à disposition à l'adresse <https://github.com/hayj/DeepStyle>.

représentations vectorielles extraites afin de garantir une meilleure représentation des documents en tenant compte de toutes les tranches, chacune pouvant contenir des informations relatives à l’auteur. Cette méthode a été testée et validée expérimentalement sur un petit échantillon de données lorsque que comparée à une simple troncature des documents à 512 *wordpieces*. Le modèle *DBert-ft* ainsi que *SNA* et sa version simplifiée *LSTM* sont entraînés sur un serveur de calcul disposant d’une carte graphique NVIDIA TITAN V.

Dans ce chapitre, nous proposons d’évaluer une version non affinée du modèle *DistilBert* afin d’évaluer la pertinence de la méthode proposée en section 2.3 et ainsi valider l’hypothèse de généralisation du style. Nous extrayons les représentations vectorielles du modèle *DistilBert* n’ayant pas été affiné en prenant les représentations internes de la dernière couche du *transformer* et en moyennant les représentations des tranches de documents comme décrit pour *DBert-ft*. La différence de performance entre *DBert-ft* et *DBert* nous permettra d’évaluer l’efficacité de la méthode de ciblage des indices intra-auteurs consistants.

4.2.3 Les modèles *baselines* pour la stylométrie

Dans l’objectif d’obtenir une base comparative pour les modèles *SNA* et *DBert-ft*, nous introduisons différents modèles *baselines* dans notre expérimentation :

Random Vecteur aléatoire pour chaque document. Les scores reportés dans le tableau de résultats seront la moyenne de vecteurs générés aléatoirement (différentes dimensions et variances).

TFIDF Vecteurs *sparses* TFIDF des documents. Les vecteurs sont de la taille du vocabulaire du jeu de données *U-set* testé. Nous ne limitons pas la taille du vocabulaire. Ces vecteurs étant *sparses*, nous ne pouvons pas donner les scores CalHar et DavB. En effet, ces deux métriques ne peuvent être utilisées qu’avec des vecteurs de taille réduite.

TFIDF+SVD et TFIDF+NMF Vecteurs TFIDF des documents réduits en dimensions par la méthode SVD (Halko et al., 2009) et par factorisation par matrices non négatives (Cichocki et Phan, 2009). Nous avons cherché les meilleurs paramètres (nombre de dimensions, suppression des mots vides, taille maximum du vocabulaire, etc.) de ces deux algorithmes implémentés dans *Scikit-learn* (Buitinck et al., 2013) par la méthode *grid search* sur un ensemble de validation et avons retenu uniquement les modèles obtenant de meilleurs résultats. Les meilleures réductions en dimensions sont la réduction à 50 dimensions pour *TFIDF+SVD* et la réduction à 100 dimensions pour *TFIDF+NMF*.

Stylo Descripteurs stylométriques (dits *handcrafted*) communément utilisés en stylométrie tels que des scores de *readability*, des mesures de richesse de vocabulaire, le compte de phrases, la longueur moyenne des phrases, etc.

LDA Vecteurs de thèmes (ou *topics*) LDA (Blei et al., 2003) (*Latent Dirichlet allocation*) de dimension 100 de chaque document. Nous avons effectué une recherche des meilleurs paramètres afin de ne retenir que le meilleur modèle.

Sent2Vec Représentation des documents par projection en utilisant un modèle *Sent2Vec* pré-entraîné sur des données Wikipédia de langue anglaise

(Pagliardini et al., 2018). Nous reportons à la fois la moyenne des représentations des phrases que composent les documents (abrégé *Sent2Vec-m*) et les représentations des documents entiers (abrégé *Sent2Vec*).

Doc2Vec Représentation des documents par projection en utilisant un modèle *Doc2Vec* (Le et Mikolov, 2014) pré-entraîné sur le *R-set* de manière non-supervisée. Nous avons entraîné 22 modèles avec différents paramètres et n’avons retenu que celui obtenant les meilleurs scores. Nous utilisons l’implémentation de *Gensim* (Řehůřek et Sojka, 2010).

USent Représentation des documents par projection en utilisant le modèle *Universal Sentence Encoder* pré-entraîné sur des données Wikipédia de langue anglaise et des articles d’actualité de langue anglaise (Cer et al., 2018). Comme pour le modèle *Sent2Vec*, nous donnons la moyenne des projections des phrases (abrégé *USent-m*) ainsi que la projection des documents entiers (abrégé *USent*).

InferSent Représentation des documents par projection en utilisant le modèle *InferSent* pré-entraîné sur un jeu de données d’inférence dans le langage naturel (Conneau et al., 2017). Comme pour le modèle *Sent2Vec*, nous donnons la moyenne des projections des phrases (abrégé *InferSent-m*) ainsi que la projection des documents entiers (abrégé *InferSent*).

BERT Moyenne des projections des phrases que composent les documents en utilisant le modèle *BERT* pré-entraîné sur un large jeu de données de livres en langue anglaise ainsi que d’articles Wikipédia en langue anglaise (Devlin et al., 2019). Nous utilisons *bert-as-service* (Xiao, 2018), une bibliothèque permettant la projection des documents par le modèle *BERT* pré-entraîné.

Nous noterons que pour tous les modèles *baselines* dont nous avons calculé la moyenne des projections des phrases, nous avons également calculé la somme des projections. Cependant, les représentations vectorielles issues de ces sommes obtiennent, de manière générale, des performances plus basses dans le partitionnement par auteur, nous n’avons donc pas retenu cette opération pour les différentes *baselines* concernées.

Contrairement à ce que proposent Stamatatos (2013) et Sapkota et al. (2015), la pondération TFIDF se trouve être une meilleure représentation des documents que la représentation par *3-grams* de caractères pour la tâche de partitionnement par auteur et la tâche d’identification d’auteur sur *NewsID*. La *baseline* TFIDF utilise un découpage des documents par mots et non par *3-grams* de caractères. De plus, cette représentation bénéficie d’une pondération qui prend en compte la fréquence des termes dans le corpus. Dans nos expérimentations, nous avons utilisé la représentation par *3-grams* de caractères en considérant différents paramètres (e.g. fréquence de *n-grams* contre présence de *n-grams*) mais avons observé des performances moindres comparées à celles de la *baseline* TFIDF. Nous n’avons donc pas ajouté cette représentation dans les tableaux de résultats.

4.2.4 Performance des modèles dans le partitionnement par auteur

Afin de tester l’hypothèse de généralisation du style, nous suivons ces différentes étapes :

Model	<i>NewsID-50-50</i> (5)			<i>BlogCorpus-50-50</i> (5)		
	SimRk	CalHar	DavB	SimRk	CalHar	DavB
<i>Random</i>	0.185	1.001	14.771	0.185	1.001	14.775
<i>TFIDF</i>	0.418	N/A	N/A	0.353	N/A	N/A
<i>TFIDF+SVD</i>	0.496	20.793	4.208	0.415	13.358	4.335
<i>TFIDF+NMF</i>	0.478	11.233	4.443	0.410	9.988	4.884
<i>LDA</i>	0.327	59.108	7.445	0.286	44.05	9.498
<i>Stylo</i>	0.302	20.031	49.431	0.287	16.945	56.976
<i>Sent2Vec</i>	0.242	2.046	8.408	0.214	1.62	8.967
<i>Sent2Vec-m</i>	0.317	5.946	6.559	0.280	4.999	6.885
<i>Doc2Vec</i>	0.486	7.216	5.796	0.374	5.234	5.893
<i>USent</i>	0.447	12.93	5.344	0.317	7.166	6.011
<i>USent-m</i>	0.469	18.856	4.791	0.349	10.888	5.199
<i>InferSent</i>	0.341	15.190	6.129	0.286	8.051	6.656
<i>InferSent-m</i>	0.402	21.696	5.082	0.343	13.768	5.241
<i>BERT</i>	0.414	18.587	4.939	0.340	11.885	5.093
<i>LSTM</i>	0.586	45.255	3.581	0.381	20.493	4.414
<i>SNA</i>	0.613	22.342	3.385	0.427	11.54	4.137
<i>DBert</i>	0.369	26.741	6.247	0.277	13.299	7.851
<i>DBert-ft</i>	0.630	31.185	3.548	0.404	15.424	4.456

TABLE 4.1 – Résultats du partitionnement par auteur sur les *U-sets* *NewsID-50-50* et *BlogCorpus-50-50*

1. Entraînement des modèles *SNA*, *LSTM* et *DBert-ft* sur le *R-set* sur la tâche d’identification d’auteurs ;
2. Projection des documents des *U-sets* en utilisant les modèles pré-entraînés ;
3. Évaluation des représentations vectorielles en utilisant les métriques de partitionnement et les labels des *U-sets*.

Le tableau 4.1 montre la performance des modèles *SNA* et *DBert-ft* comparés aux différentes *baselines* sur les *U-sets* *NewsID-50-50* et *BlogCorpus-50-50*. Nous indiquons le nombre de *U-sets* utilisés entre parenthèses dans tous les tableaux de résultats. Les scores correspondent ainsi aux moyennes obtenues sur les ensembles de *U-sets* de même type. Dans ce tableau, la *baseline* *Random* permet de rendre compte du score minimum qu’un modèle doit obtenir. Comme le montre ce tableau, les scores de *SNA* et *DBert-ft* sont supérieurs sur la métrique *SimRank*.

LSTM n’est pas aussi performant que *SNA* mais est tout de même supérieur à la plupart des *baselines*. Cela montre l’intérêt de la méthode proposée mais également que l’introduction d’une couche d’attention et de la bidirectionnalité de *SNA* semblent jouer un rôle dans la performance du modèle. D’après le tableau, le modèle non affiné *DBert* obtient des performances très inférieures à *DBert-ft*, ce qui, de nouveau, montre la pertinence de la méthode proposée.

Nous noterons les bonnes performances des *baselines* *Doc2Vec*, *TFIDF+SVD* et *TFIDF+NMF*. Cela s’explique par l’entraînement non-supervisé du modèle *Doc2Vec* sur le *R-set*. Les *baselines* *TFIDF+SVD* et *TFIDF+NMF* sont également performantes, et d’autant plus sur le *U-set* *BlogCorpus-50-50*. La représentation fine du vocabulaire par la pondération TFIDF est donc une *baseline* pertinente dans l’objectif

de partitionner par auteur. Cependant, les modèles *SNA* et *DBert-ft* semblent apporter une dimension supplémentaire par la représentation sémantique des documents.

Les modèles de représentation textuelle état de l’art proposés en *baselines* dans cette expérimentation (*Sent2Vec*, *USent*, *InferSent* et *BERT*) n’obtiennent pas de scores comparables sur cette tâche. Cela peut s’expliquer du fait qu’ils n’aient pas bénéficié de la méthode de ciblage des indices intra-auteurs consistants. À noter que la moyenne des projections des phrases permet d’obtenir de meilleurs résultats pour ces différents modèles.

Pour la métrique DavB, le modèle *SNA* obtient de meilleures performances sur ces deux types de *U-sets*. Cependant, les scores de *DBert-ft* et *LSTM*, ayant été entraînés par la méthode de ciblage des indices intra-auteurs consistants, obtiennent des résultats proches. Pour rappel, plus l’indice DavB est bas, plus le partitionnement est considéré de qualité. À noter que l’indice DavB de la *baseline Stylo* est plus grand que les vecteurs générés aléatoirement (*baseline Random*). Cela s’explique du fait que *Stylo* a tendance à générer des partitions ayant des formes et positions similaires. Les formes et positions des partitions étant similaires, l’indice DavB de *Stylo* en est pénalisé. En effet, l’indice DavB mesure la différence de taille des partitions et leur distance les unes par rapport aux autres, alors que les autres mesures n’en tiennent pas compte.

Cependant, il est à noter que les partitions *Stylo* obtiennent des scores plus bas pour les autres métriques également. Il semblerait que cela soit dû au fait que les représentations *Stylo* ne contiennent que peu de caractéristiques permettant la distinction des documents. En effet, les descripteurs *Stylo* ne représentent un document que superficiellement en donnant, par exemple, des scores de lisibilité ou de richesse de vocabulaire. Cela confirme l’importance de la prise en compte des éléments de vocabulaire dans les documents.

Concernant la métrique CalHar, la *baseline LDA* obtient des performances largement supérieures aux autres modèles malgré des performances inférieures sur les autres métriques. Ces scores semblent indiquer que les représentations *LDA* permettent d’obtenir des partitions ayant de bonnes propriétés de dispersion mais que d’après *SimRank*, ces représentations ne permettent pas un bon ordonnancement des documents par auteurs. Cependant, comme le montrent [Rendón et al. \(2011\)](#), cet indice est moins informatif que l’indice DavB lorsqu’il s’agit de prédire les performances d’un modèle sur des tâches extrinsèques. Cette observation se vérifiera d’ailleurs par la suite lorsque nous testerons les modèles sur l’identification d’auteurs. Nous verrons que le modèle *LDA* obtient des scores inférieurs malgré sa performance au regard de l’indice CalHar. À noter que le modèle *LSTM* obtient également de bons scores CalHar mais que ses scores sur les autres métriques sont plus bas que ceux de sa version augmentée *SNA*.

Le tableau 4.2 donne les scores *SimRank* des modèles sur d’autres *U-sets* spécifiques à des journaux présentés au chapitre précédent. Les résultats obtenus sur ces *U-sets* montrent que les *baselines* basées sur la pondération TFIDF avec réduction de dimension *SVD* et *NMF* sont performantes. Ce qui confirme l’intérêt d’une prise en compte fine du vocabulaire pour cette tâche. Les modèles *USent-m* et *Doc2Vec* obtiennent également de bons scores *SimRank* en moyenne.

Le modèle *DBert-ft* est cependant supérieur à ces modèles sur une partie des *U-sets*. À noter que *SNA* obtient également de bonnes performances mais inférieures à *DBert-ft*. Tout comme pour le tableau de résultat 4.1, *LSTM* est performant

Model	<i>LiveJournal-50-50 (5)</i>	<i>WashingtonPost-50-50</i>	<i>Breitbart-50-50</i>	<i>BusinessInsider-50-50</i>	<i>CNN-50-50</i>	<i>GuardianUK-50-50</i>	<i>TheGuardian-50-50</i>	<i>NYTimes-50-50</i>
<i>Random</i>	0.185	0.185	0.185	0.185	0.185	0.185	0.185	0.185
<i>TFIDF</i>	0.370	0.396	0.346	0.405	0.38	0.485	0.388	0.456
<i>TFIDF+SVD</i>	0.440	0.439	0.387	0.456	0.436	0.509	0.418	0.505
<i>TFIDF+NMF</i>	0.435	0.443	0.362	0.432	0.403	0.504	0.414	0.480
<i>LDA</i>	0.304	0.327	0.257	0.316	0.300	0.404	0.269	0.345
<i>Stylo</i>	0.270	0.274	0.247	0.234	0.245	0.303	0.223	0.266
<i>Sent2Vec</i>	0.221	0.245	0.227	0.245	0.229	0.332	0.247	0.246
<i>Sent2Vec-m</i>	0.289	0.311	0.271	0.313	0.296	0.392	0.294	0.348
<i>Doc2Vec</i>	0.417	0.416	0.375	0.441	0.399	0.538	0.405	0.472
<i>USent</i>	0.368	0.365	0.343	0.425	0.381	0.529	0.393	0.429
<i>USent-m</i>	0.403	0.424	0.354	0.439	0.404	0.547	0.406	0.479
<i>InferSent</i>	0.296	0.337	0.272	0.33	0.316	0.412	0.305	0.359
<i>InferSent-m</i>	0.364	0.381	0.31	0.382	0.361	0.468	0.345	0.423
<i>BERT</i>	0.370	0.382	0.319	0.379	0.353	0.471	0.352	0.432
<i>LSTM</i>	0.403	0.334	0.376	0.393	0.383	0.451	0.347	0.385
<i>SNA</i>	0.438	0.358	0.405	0.406	0.407	0.465	0.345	0.406
<i>DBert</i>	0.313	0.369	0.317	0.32	0.346	0.414	0.339	0.391
<i>DBert-ft</i>	0.441	0.436	0.416	0.439	0.444	0.499	0.375	0.453

TABLE 4.2 – Scores *SimRank* du partitionnement par auteur sur les *U-sets* spécifiques

Model	<i>TFIDF+SVD</i>	<i>TFIDF+NMF</i>	<i>LDA</i>	<i>Stylo</i>	<i>Sent2Vec-m</i>	<i>Doc2Vec</i>	<i>USent-m</i>	<i>InferSent-m</i>	<i>BERT</i>	<i>SNA</i>	<i>DBert</i>	<i>DBert-ft</i>
<i>TFIDF+SVD</i>	.455	.461	.410	.276	.318	.431	.453	.394	.380	.466	.334	.474
<i>TFIDF+NMF</i>		.439	.378	.276	.311	.430	.443	.383	.379	.464	.333	.474
<i>LDA</i>			.309	.277	.320	.431	.409	.384	.381	.466	.336	.474
<i>Stylo</i>				.276	.277	.285	.277	.277	.279	.279	.278	.316
<i>Sent2Vec-m</i>					.302	.434	.323	.322	.364	.458	.335	.475
<i>Doc2Vec</i>						.430	.431	.422	.340	.457	.386	.492
<i>USent-m</i>							.416	.385	.379	.468	.337	.474
<i>InferSent-m</i>								.374	.381	.456	.347	.475
<i>BERT</i>									.378	.438	.372	.477
<i>SNA</i>										.463	.381	.473
<i>DBert</i>											.339	.472
<i>DBert-ft</i>												.474

TABLE 4.3 – Scores *SimRank* moyens des combinaisons de représentations vectorielles pour le partitionnement par auteur

mais obtient des scores plus bas que *SNA*. De même, *DBert* obtient des scores inférieurs à *DBert-ft*, montrant ainsi la pertinence de la méthode de ciblage des indices intra-auteurs consistants.

Les modèles *DBert-ft*, *SNA* et *LSTM* ont été entraînés sur le *R-set* majoritairement composé de labels de type *domain*. Malgré cela, et d’après le tableau 4.2, il semblerait que ces modèles parviennent à généraliser suffisamment leurs représentations pour permettre un bon partitionnement par auteur, même dans le cas de *U-sets* uniquement composés de labels correspondant à des rédacteurs de journaux en ligne et de blogs (i.e. des labels de type *authorial_domain*).

4.2.5 Performance des combinaisons de modèles dans le partitionnement par auteur

Dans l’objectif de montrer la complémentarité des représentations de ces différents modèles, nous proposons de les combiner et de les évaluer par la métrique *SimRank*.

Le tableau 4.3 donne les scores *SimRank*, sous forme de matrice, des combinaisons par paires de modèles. Chaque cellule de la matrice correspond à la moyenne des *SimRank* obtenue sur les 22 *U-sets* utilisés précédemment. Pour rappel, les scores *SimRank* sont normalisés entre 0 et 1. Dans le tableau, pour des raisons de mise en forme, nous affichons ces nombres réels sans le premier chiffre qui est toujours 0. La diagonale ne donne pas de combinaison de modèles mais la moyenne des scores pour le modèle concerné.

Ce tableau montre que la moyenne des scores de *DBert-ft* seul est supérieure aux autres modèles. La meilleure combinaison de modèles est *DBert-ft* et *Doc2Vec*. D’après ce tableau, le modèle *DBert-ft* semble être complémentaire avec la plupart

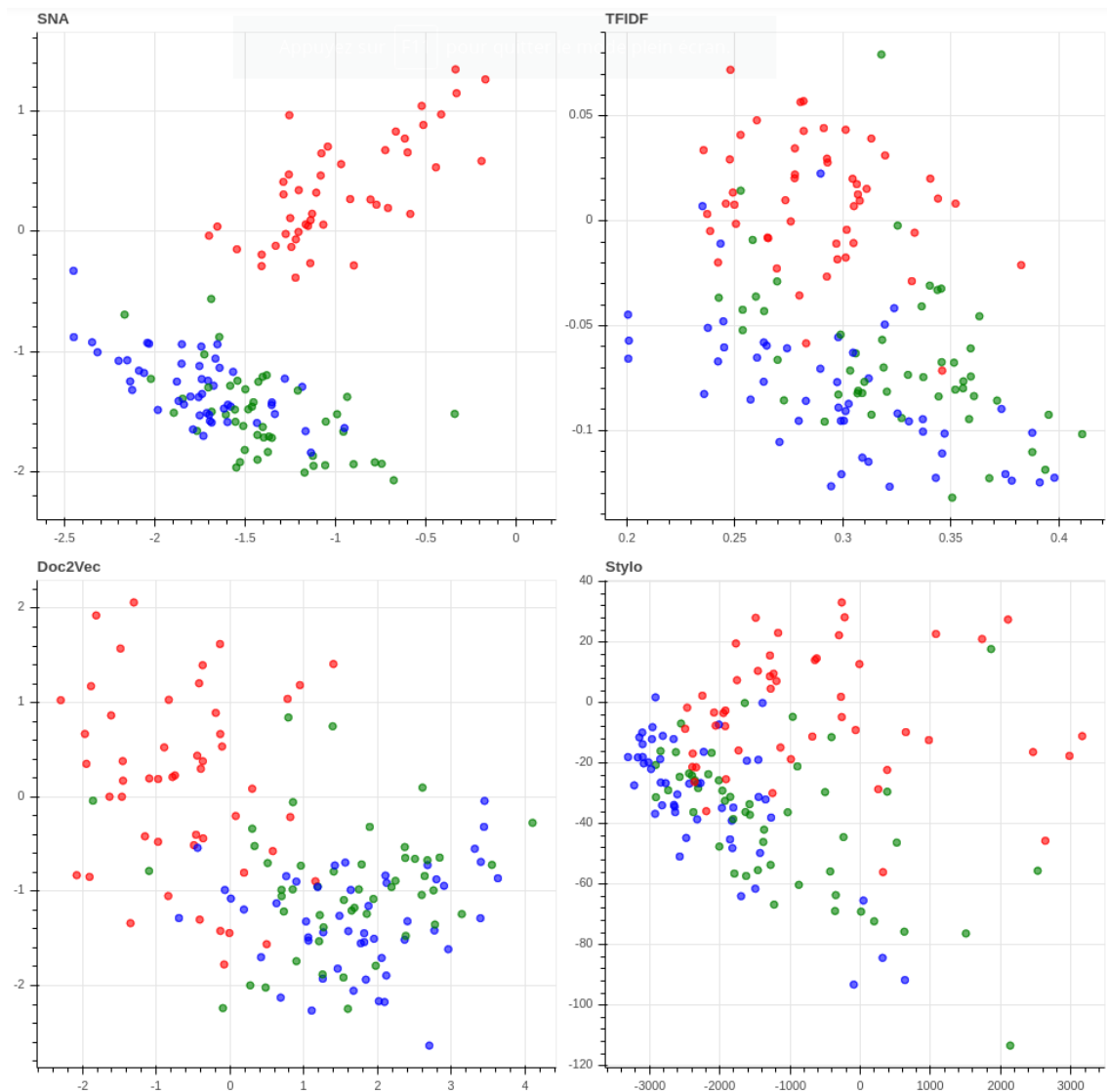


FIGURE 4.2 – Regroupement de trois auteurs par le modèle *SNA* ainsi que trois *baselines*

des autres modèles. À noter que la combinaison *DBert-ft* et *Doc2Vec* est supérieure à *DBert-ft* seul, la combinaison de représentations vectorielles complémentaires est donc pertinente.

Au regard de l'ensemble de ces résultats, nous avons montré que la méthode de ciblage des indices intra-auteurs consistants permet un gain significatif dans la tâche de partitionnement par auteur, validant ainsi l'*hypothèse de généralisation du style*.

4.2.6 Visualisation de partitions

Sur la figure 4.2, nous assemblons le partitionnement de trois auteurs pour le modèle *SNA* en haut à gauche ainsi que les *baselines* *TFIDF*, *Doc2Vec* et *Stylo*. Ce partitionnement correspond au partitionnement des documents de trois auteurs pris au hasard dans un *U-set* de type *NewsID-50-50*. Nous effectuons une réduction à deux dimensions afin de placer chaque document sur un plan à deux dimensions.

Les couleurs correspondent aux différents auteurs. Nous remarquons que le modèle *SNA* permet un partitionnement plus net des vecteurs par auteur, contrairement au partitionnement des autres modèles. En effet, l’auteur en rouge est nettement séparé des auteurs en vert et bleu. Cependant, les auteurs vert et bleu se superposent sur les quatre graphes, indiquant la difficulté des modèles à discriminer les documents de ces deux auteurs.

4.3 Identification d’auteur

Dans la section 4.2, nous avons montré l’intérêt de la méthode de ciblage des indices intra-auteurs consistants et validé l’*hypothèse de généralisation du style*. Cependant, les modèles n’ont été évalués que sur la tâche de partitionnement par auteur. Nous proposons donc d’évaluer les modèles et leur combinaison sur l’identification d’auteurs (tâche de classification). Cette expérimentation nous permettra de confirmer l’intérêt de la méthode proposée et d’observer la cohérence de performance des modèles *SNA* et *DBert-ft* sur deux tâches de la stylométrie. Elle nous permettra également d’analyser la pertinence des métriques de partitionnement utilisées lorsqu’il s’agit d’avoir un aperçu de la qualité des représentations vectorielles produites par les différents modèles dans le cadre d’autres tâches. Il est notamment intéressant d’analyser la pertinence de la métrique *SimRank* et ses différences dans l’évaluation des modèles avec les métriques *CalHar* et *DavB*.

Dans cette section, nous testons les modèles présentés en section précédente sur les mêmes jeux de test *U-sets*. Nous observons ensuite les performances du modèle *SNA* dans le cadre de l’apprentissage par transfert, puis nous donnons un exemple de carte d’attention du modèle *SNA*. Nous testons l’une des propriétés du style définie dans le chapitre 2 : la *non-spécificité sémantique*. Pour cela nous proposons la métrique *TFIDF focus*. Enfin, nous étudions l’impact du nombre d’auteurs et du nombre de documents par auteurs des *U-sets* dans la performance du modèle.

4.3.1 Procédure d’évaluation

La procédure nous permettant de comparer les modèles sur la tâche d’identification d’auteurs est la suivante :

1. Pour tous les modèles, nous générons les vecteurs représentatifs des documents de chaque *U-set*.
2. Nous choisissons un modèle de classification standard : *SGDClassifier* implémenté dans *Scikit-learn*. Ce modèle utilise l’algorithme du gradient stochastique pour la classification.
3. Nous entraînons ce modèle sur le *U-set* avec pour entrée les vecteurs représentatifs d’un modèle. La sortie du modèle étant une distribution de probabilité sur les classes du *U-sets*. Nous choisissons la fonction de perte *hinge* correspondant à un *SVM* linéaire. Nous réservons 80% des documents pour l’ensemble d’entraînement et 20% pour l’ensemble de validation.
4. Nous arrêtons l’entraînement du modèle *SGDClassifier* lorsque l’exactitude⁸ (ou *accuracy*) stagne ou décroît.

8. L’exactitude (ou *accuracy*) est une mesure de performance utilisée notamment dans des tâches de classification, elle donne le nombre de vrais positifs sur le nombre total d’échantillons.

Model	<i>NewsID-50-50 (5)</i>	<i>BlogCorpus-50-50 (5)</i>
<i>MostFreq</i>	0.020	0.020
<i>TFIDF+SVD</i>	0.554	0.550
<i>TFIDF+NMF</i>	0.489	0.499
<i>LDA</i>	0.193	0.135
<i>Stylo</i>	0.111	0.117
<i>Sent2Vec-m</i>	0.418	0.377
<i>Doc2Vec</i>	0.513	0.459
<i>USent-m</i>	0.557	0.477
<i>InferSent-m</i>	0.628	0.647
<i>BERT</i>	0.582	0.558
<i>SNA</i>	0.671	0.580
<i>Dbert</i>	0.582	0.489
<i>Dbert-ft</i>	0.745	0.564

TABLE 4.4 – Scores d’exactitude de l’identification d’auteurs sur les *U-sets NewsID-50-50* et *BlogCorpus-50-50*

5. Nous retenons la meilleure exactitude obtenue sur l’ensemble de validation lors de l’entraînement du modèle.

Nous choisissons *SGDClassifier* suite à son évaluation et comparaison sur un *U-set* ne faisant pas partie de ceux testés dans cette section. *SGDClassifier* permet d’obtenir des résultats optimaux comparés à d’autres modèles standards de classification. De plus, celui-ci présente l’avantage de n’être que peu sensible à des modifications d’hyperparamètres.

Dans cette procédure, nous n’utilisons pas d’ensembles de test car, mis à part le nombre d’itérations effectuées, aucun hyperparamètre n’est optimisé. Retenir la meilleure exactitude sur l’ensemble de validation nous permet d’obtenir une bonne approximation de la performance du modèle *SGDClassifier* ainsi que des représentations vectorielles qu’il utilise.

4.3.2 Performance des modèles dans l’identification d’auteurs

Le tableau 4.4 donne les résultats obtenus sur les *U-sets NewsID-50-50* et *BlogCorpus-50-50*. Dans ce tableau, nous ne reportons pas la *baseline TFIDF* sans réduction en dimensions qui obtient des scores plus bas que lorsque l’on effectue une réduction en dimensions. De même, nous reportons uniquement les moyennes des projections de phrases pour *Sent2Vec*, *USent*, *InferSent* et *BERT*. Nous donnons la *baseline MostFreq* qui correspond à un classifieur prédisant uniquement la classe la plus représentée dans le jeu de données, i.e. le score de *MostFreq* correspond au nombre d’exemples de la classe la plus représentée divisé par le nombre d’exemples total. Les *U-sets* étant équilibrés, le score de cette *baseline* est de 0.02. Cette *baseline* naïve indique le score minimum qu’un modèle doit obtenir.

Dans ce premier tableau, nous remarquons que les modèles les plus performants sont *SNA* et *DBert-ft* sur *NewsID-50-50* et *InferSent-m* sur *BlogCorpus-50-50*. Les scores des autres modèles sont équilibrés. Les *baselines Stylo* et *LDA* sont cependant

Model	<i>LiveJournal-50-50 (5)</i>	<i>WashingtonPost-50-50</i>	<i>Breitbart-50-50</i>	<i>BusinessInsider-50-50</i>	<i>CNN-50-50</i>	<i>GuardianUK-50-50</i>	<i>TheGuardian-50-50</i>	<i>NYTimes-50-50</i>
<i>MostFreq</i>	0.020	0.020	0.020	0.020	0.020	0.020	0.020	0.020
<i>TFIDF+SVD</i>	0.487	0.448	0.412	0.490	0.433	0.578	0.458	0.548
<i>TFIDF+NMF</i>	0.458	0.442	0.337	0.389	0.356	0.501	0.394	0.474
<i>LDA</i>	0.165	0.182	0.096	0.162	0.149	0.193	0.128	0.208
<i>Stylo</i>	0.100	0.080	0.080	0.074	0.081	0.080	0.052	0.080
<i>Sent2Vec-m</i>	0.357	0.450	0.300	0.410	0.321	0.780	0.372	0.502
<i>Doc2Vec</i>	0.408	0.478	0.388	0.489	0.381	0.814	0.416	0.525
<i>USent-m</i>	0.465	0.476	0.382	0.480	0.397	0.736	0.46	0.565
<i>InferSent-m</i>	0.535	0.528	0.48	0.553	0.449	0.850	0.534	0.614
<i>BERT</i>	0.482	0.468	0.408	0.516	0.410	0.82	0.476	0.577
<i>SNA</i>	0.477	0.429	0.484	0.452	0.422	0.734	0.428	0.556
<i>Dbert</i>	0.478	0.528	0.442	0.464	0.422	0.789	0.492	0.594
<i>Dbert-ft</i>	0.526	0.537	0.537	0.541	0.454	0.796	0.488	0.617

TABLE 4.5 – Scores d’exactitude de l’identification d’auteurs sur les *U-sets* spécifiques

inférieures à l’ensemble des autres modèles.

Le tableau 4.5 donne les résultats obtenus sur les autres *U-sets* spécifiques à un journal en ligne. Nous remarquons que, comme pour les *U-sets* précédents, les modèles les plus performants sont les modèles *DBert-ft* et *InferSent-m*.

4.3.3 Performance des combinaisons de modèles dans l’identification d’auteurs

Le tableau 4.6 donne les scores moyens, sous forme de matrice, des modèles seuls ainsi que les scores des combinaisons de modèles par paires. Le modèle *DBert-ft* obtient la meilleure moyenne d’exactitude. Une nouvelle fois, comme pour la tâche de partitionnement par auteur, la combinaison *DBert-ft* et *Doc2Vec* permet d’obtenir de meilleures performances que les autres combinaisons. Le score de cette combinaison est supérieur au score de *DBert-ft* seul, ce qui montre que la combinaison de modèles est pertinente et que ces deux modèles sont complémentaires. Ils permettent de capturer différentes caractéristiques utiles pour l’identification d’auteurs.

À noter que le modèle *InferSent* est performant sur cette tâche malgré ses scores plus faibles lorsque testé sur la tâche de partitionnement par auteur. Cela peut s’expliquer par le fait que la plupart des dimensions des vecteurs *InferSent* ne jouent pas de rôle majeur dans le partitionnement et que le *SVM* utilisé parvient à isoler des dimensions très informatives lorsqu’il s’agit d’identifier un auteur.

De nouveau, les performances de *DBert-ft* et l’amélioration par rapport à sa

Model	<i>TFIDF+SVD</i>	<i>TFIDF+NMF</i>	<i>LDA</i>	<i>Stylo</i>	<i>Sent2Vec-m</i>	<i>Doc2Vec</i>	<i>USent-m</i>	<i>InferSent-m</i>	<i>BERT</i>	<i>SNA</i>	<i>DBert</i>	<i>DBert-ft</i>
<i>TFIDF+SVD</i>	.514	.552	.525	.096	.464	.475	.599	.629	.553	.581	.547	.598
<i>TFIDF+NMF</i>		.460	.472	.099	.444	.476	.576	.618	.543	.573	.550	.598
<i>LDA</i>			.163	.098	.420	.477	.518	.590	.541	.555	.541	.600
<i>Stylo</i>				.098	.098	.108	.103	.097	.102	.101	.097	.191
<i>Sent2Vec-m</i>					.404	.494	.466	.504	.544	.576	.550	.609
<i>Doc2Vec</i>						.472	.474	.491	.526	.543	.519	.641
<i>USent-m</i>							.499	.612	.538	.579	.550	.598
<i>InferSent-m</i>								.594	.560	.598	.578	.604
<i>BERT</i>									.536	.621	.571	.616
<i>SNA</i>										.552	.598	.614
<i>DBert</i>											.522	.610
<i>DBert-ft</i>												.597

TABLE 4.6 – Scores d’exactitude moyens des combinaisons de représentations vectorielles pour l’identification d’auteurs

version non affinée *DBert* confirment l’intérêt de la méthode proposée en section 2.3. À noter que *LDA* obtient un score *CalHar* haut, mais se trouve être moins performant sur la tâche de classification. Cette métrique ne semble pas permettre une bonne approximation des performances dans la tâche de classification.

4.3.4 Apprentissage par transfert

Afin d’évaluer le bénéfice qu’apporte l’utilisation du modèle *SNA* dans le cadre de l’apprentissage par transfert, nous proposons d’entraîner un modèle *double-branches* basé sur le modèle *SNA* sur la tâche d’identification d’auteurs. Nous appelons ce modèle *double-branches* « *Dual-SNA* ». Dans cette section, nous ne cherchons pas à obtenir les meilleures performances possibles (e.g. par l’utilisation de *DBert-ft*) mais à étudier les couches d’attention. Pour des raisons que nous détaillons en section 4.3.6, nous avons choisi le modèle *SNA* pour cette expérimentation.

Nous dupliquons toutes les couches de la première couche jusqu’à la couche d’attention. Puis, nous concaténons la sortie des deux branches. Nous ajoutons ensuite deux couches denses avec les mêmes paramètres que pour le modèle *SNA*. La dernière couche est une couche *softmax* avec un nombre d’unités qui dépend du nombre d’auteurs dans le jeu de données. Nous entraînons deux versions de ce modèle *double-branches* sur deux *U-sets* : *NewsID-50-50* et *BlogCorpus-50-50*.

La branche de gauche de la première version du modèle est initialisée avec les poids du modèle *SNA* entraîné sur le *R-set*. Le modèle, dans sa première version, a donc bénéficié d’un transfert de connaissances. En interne, dans le réseau de neurones, il représentera les documents des *U-sets* de la même manière qu’il les représentait lors de l’entraînement sur le *R-set*. Cette branche de gauche est configurée comme

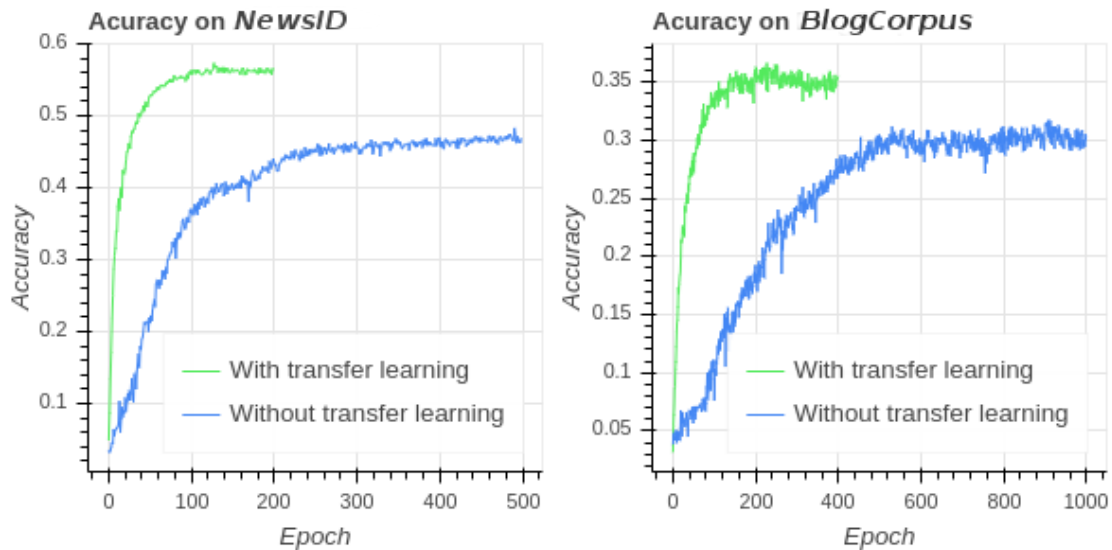


FIGURE 4.3 – Exactitude du modèle *Dual-SNA* sur *NewsID-50-50* et *BlogCorpus-50-50*

immuable, i.e. l’entraînement du réseau de neurones profond, avec le processus de rétropropagation du gradient, n’aura aucun impact sur les poids de cette branche. La branche de droite est initialisée avec des valeurs aléatoires. La seconde version du modèle *Dual-SNA* ne bénéficie pas de transfert de connaissances. Chacune des branches de cette version est donc initialisée avec des valeurs aléatoires.

Le modèle *Dual-SNA* nous permet de comparer l’exactitude obtenue pour les deux versions et ainsi voir l’impact du transfert de connaissances sur l’identification d’auteurs. De plus, cela nous permet d’afficher, pour la première version du modèle *Dual-SNA*, la superposition des deux cartes d’attention textuelle⁹ des deux branches et ainsi observer sur quel mot se concentre telle ou telle branche.

Sur la figure 4.3, nous assemblons les courbes d’apprentissage « étape/exactitude » (ou *epoch/accuracy*) des deux versions du modèle *Dual-SNA* : la version ayant bénéficié du transfert de connaissances en vert et l’autre en bleu. Le graphe de gauche correspond à la courbe d’apprentissage des deux versions du modèle sur *NewsID-50-50*. Le graphe de droite correspond à la courbe d’apprentissage des deux versions du modèle sur *BlogCorpus-50-50*.

Comme nous pouvons le voir sur les deux graphes, le transfert de connaissances en vert permet un gain d’approximativement 10% sur les deux *U-sets*. Il permet également une convergence plus rapide de l’apprentissage. Sur *BlogCorpus-50-50*, la convergence est visible entre 150 et 200 *epochs*. Sur *NewsID-50-50*, la convergence est visible à approximativement 100 *epochs*. Cette expérimentation montre que l’entraînement de *SNA* sur le *R-set* permet effectivement un transfert de connaissances et que les représentations internes apprises aident à l’identification des auteurs pour d’autres jeux de données.

9. Une carte d’attention textuelle, issue d’un réseau de neurones profond, correspond à la superposition de texte et d’une bande de couleur plus ou moins transparente. L’intensité de couleur de la bande indique le niveau d’attention que porte le réseau de neurones profond sur le texte se trouvant en dessous.

Welcome to day two of cold , nasty , wet weather
 . Ick . Rain is so bad by itself ... But when you
 mix it with a hella cold temperature and nasty
 wind ... Not so much fun anymore . I hate days
 like these ... Two cold and icky to do anything
 so you stuck in the house ALL day long ... All
 trapped and junk ... I hate feeling trapped ...
 But in a way I guess there is a bright side ... I
 finally have no reason out of folding up all the
 clothes and getting my closet organized ... Gee ,
 I much rather have rain than all the snow we
 got !

FIGURE 4.4 – Carte d’attention textuelle du modèle *Dual-SNA*

4.3.5 Carte d’attention textuelle

Afin d’observer sur quel mot chaque branche du modèle *Dual-SNA* porte son attention, nous générons la carte d’attention textuelle en superposant les deux attentions de la première version du modèle. À noter qu’un niveau d’attention élevé sur un mot correspondra en réalité à l’attention d’une séquence plus ou moins longue dont le dernier mot est celui surligné. En effet, le modèle *SNA* est basé sur une architecture de type réseau de neurones récurrents et représente donc les séquences de mots. La figure 4.4 montre la carte d’attention textuelle sur un document exemple. L’attention en rouge (bande haute) correspond à l’attention de la branche de gauche ayant bénéficié du transfert de connaissances. L’attention en vert (bande basse) correspond à l’attention de la branche de droite.

Cette figure montre que l’attention de chaque branche est concentrée sur des mots différents. Ainsi, chaque branche représente un document d’une manière différente. La branche de gauche se concentre sur les mots qui ont été jugés pertinents pour l’identification des auteurs dans le jeu de données de référence *R-set*. La branche de droite, en revanche, se concentre uniquement sur les mots pertinents pour l’identification des auteurs du *U-set*. Cette expérimentation d’apprentissage par transfert et la carte d’attention semblent montrer que les représentations de la branche pré-entraînée permettent une augmentation des performances sur la tâche d’identification d’auteurs sur ces deux jeux de données. Cela peut aussi être dû à la taille du *U-set* qui ne permet pas au modèle sans apprentissage par transfert de disposer de suffisamment d’indices linguistiques pertinents pour effectuer une identification d’auteurs fiable.

Nous ne pouvons tirer aucune conclusion de cette carte d’attention puisqu’elle ne correspond qu’à un seul exemple et qu’il est difficile de réellement donner une interprétation aux mots ciblés par chaque branche. Cependant, grâce à la mesure *TFIDF focus* introduite en section 4.3.6, il est possible de calculer si, en moyenne, les mots sur lesquels la branche pré-entraînée focalise son attention respectent la propriété de *non-spécificité sémantique*.

4.3.6 La *non-spécificité sémantique*

Dans cette section, nous cherchons à vérifier si les modèles entraînés par la méthode de ciblage des indices intra-auteurs consistentants portent leur attention sur des indices linguistiques en adéquation avec la *non-spécificité sémantique*. Pour cela, nous utilisons la mesure *TFIDF focus* introduite en section 2.4. Pour cette expérimentation, nous utilisons le modèle *SNA* et non le modèle *DBert-ft* pour trois

<i>SNA trained on</i>	<i>U-set 1</i>	<i>U-set 2</i>	<i>U-set 3</i>	<i>U-set 4</i>	<i>U-set 5</i>	<i>Mean</i>
<i>Target U-sets</i>	0.642	0.650	0.606	0.601	0.661	0.632
<i>Other U-set</i>	0.611	0.591	0.576	0.559	0.623	0.592
<i>R-set</i>	0.497	0.479	0.477	0.459	0.507	0.483

TABLE 4.7 – *TFIDF focus* de différents modèles *SNA* sur 5 *U-sets NewsID-50-200*

<i>SNA trained on</i>	<i>U-set 1</i>	<i>U-set 2</i>	<i>U-set 3</i>	<i>U-set 4</i>	<i>U-set 5</i>	<i>Mean</i>
<i>Target U-sets</i>	0.668	0.722	0.734	0.645	0.702	0.694
<i>Other U-set</i>	0.637	0.670	0.670	0.606	0.648	0.646
<i>R-set</i>	0.547	0.579	0.575	0.526	0.560	0.557

TABLE 4.8 – *TFIDF focus* de différents modèles *SNA* sur 5 *U-sets BlogCorpus-50-160*

raisons :

1. Nous ne cherchons pas à utiliser le modèle plus performant mais à évaluer si la méthode de ciblage des indices intra-auteurs consistants permet aux couches d’attention de se concentrer sur des indices linguistiques en adéquation avec la *non-spécificité sémantique* ;
2. L’entraînement de *SNA* demande moins de ressources, réduisant ainsi le temps d’exécution ;
3. Le modèle *DBert-ft* utilise une *tokenisation* par *wordpieces* (Wu et al., 2016) et il est plus compliqué d’utiliser la mesure *TFIDF focus* sans introduire de biais étant donné que les poids *TFIDF* sont calculés sur une *tokenisation* par mots.

Ainsi, nous comparons le modèle *SNA* entraîné par la méthode de ciblage des indices intra-auteurs consistants à d’autres modèles *SNA* (ayant donc la même structure) entraînés sur des *U-sets*. Les différences de *TFIDF focus* observées montreront à quel point les représentations internes des documents du modèle *SNA* principal satisfont la propriété de *non-spécificité sémantique*. Dans cette expérimentation, comme défini dans la section 4.2, les documents sont de taille $w = 1\,200$. Ainsi, les lignes des matrices A' et T' utilisées pour calcul du *TFIDF focus* sont de taille 1 200.

Lors du calcul d’un *TFIDF focus*, nous choisissons un unique *U-set*. Cependant, nous répétons le calcul sur plusieurs *U-sets* (10 au total) afin d’observer si la différence est significative. Les tableaux 4.7 et 4.8 montrent les scores obtenus pour différents modèles. Pour chaque cellule du tableau, nous générons une matrice A' et une matrice T' différentes, liées au *U-set* de la colonne correspondante. Dans le tableau 4.7, nous choisissons 5 *U-sets* de type *NewsID-50-200* ayant 50 classes différentes et 200 documents par classe. Dans le tableau 4.8, nous choisissons 5 *U-sets* de type *BlogCorpus-50-160* ayant 50 classes différentes et 160 documents par classe.

Les lignes de résultats de ces tableaux sont au nombre de trois :

1. La première ligne de résultats du tableau correspond aux *TFIDF focus* de matrices A' et T' générées à partir d’un modèle *SNA* entraîné sur le *U-set* utilisé pour la colonne correspondante. Sur cette ligne, il existe autant de modèles *SNA* que de *U-sets* cibles. Nous choisissons de donner ces scores *TFIDF focus* afin de comparer l’attention pour un modèle qui a été entraîné sur les documents pour lesquels nous voulons extraire l’attention.

2. La deuxième ligne du tableau correspond aux *TFIDF focus* d'un modèle *SNA* entraîné sur un autre *U-set* composé de 50 classes et de 150 documents par classe. Celui-ci n'est pas présent dans les *U-sets* pour lesquels nous calculons les attentions. Il peut donc être considéré comme un *R-set* composé de beaucoup moins de documents que le *R-set* original. Nous choisissons de donner ces scores *TFIDF focus* afin d'observer si l'entraînement du modèle sur un *U-set* externe est suffisant pour que le modèle concentre son attention sur des termes à poids TFIDF faibles. Ce modèle est évalué dans les mêmes conditions que le modèle *SNA* entraîné sur le *R-set*, il n'a cependant pas bénéficié d'un jeu de données large (le *R-set*) avec plus de classes et un plus grand nombre de documents par classe.
3. À la troisième et dernière ligne, nous reportons les *TFIDF focus* du modèle *SNA* entraîné sur le *R-set*.

Dans ces tableaux, pour chaque type de *U-set*, nous donnons, en dernière colonne, la moyenne de tous les *TFIDF focus*. Ces tableaux montrent que le modèle *SNA* entraîné sur le *R-set* porte son attention sur des termes à poids TFIDF faibles de façon plus importante que ceux entraînés sur les *U-set* cibles (première ligne), indiquant ainsi que la méthode de ciblage des indices intra-auteurs consistants permet, en effet, de concentrer l'attention du modèle sur des termes non spécifiques par rapport à un modèle ayant été entraîné sur les données cibles.

Le modèle entraîné sur un *U-set* externe (deuxième ligne) obtient un *TFIDF focus* plus faible que les modèles entraînés sur les *U-sets* cibles (première ligne). Ce résultat est cohérent du fait de la différence de vocabulaire entre les *U-sets* cibles et le *U-set* externe. Les modèles entraînés sur les *U-sets* cibles sont en mesure de détecter les termes clés permettant l'identification d'une classe, ce qui, inévitablement, concentrera l'attention du modèle sur ces termes. Plus le modèle sera entraîné sur les données du *U-set*, plus il sera précis dans l'identification des classes en détectant les termes clés, impliquant que le modèle focalisera son attention sur ces termes. Or, comme mentionné précédemment, ces termes clés sont généralement ceux permettant de distinguer les documents dans le corpus et ont donc plus probablement des poids TFIDF importants.

Pour finir, les tableaux montrent que le modèle *SNA* entraîné sur le *R-set* obtient des *TFIDF focus* encore plus faibles que le modèle *SNA* entraîné sur un *U-set* externe. La différence est plus importante qu'entre les deux premières lignes avec approximativement un *TFIDF focus* 10% moins grand. Ce dernier résultat indique qu'un entraînement sur un large jeu de données est un autre facteur clé dans la capacité du modèle à concentrer son attention sur des termes non spécifiques.

4.3.7 Impact du nombre de documents par auteur

[Boumber et al. \(2018\)](#) ont étudié l'impact du nombre de documents par auteur sur la performance de leur modèle basé sur une architecture *CNN*. Ils ont montré que plus le réseau de neurones dispose de documents par auteur, plus il sera performant dans la tâche d'identification d'auteurs par rapport à leur *baseline MLP* basée sur *Doc2Vec* ([Le et Mikolov, 2014](#)). Plus leur modèle dispose de documents par auteur, plus il pourra généraliser l'identification des auteurs du jeu de données, évitant ainsi le surapprentissage. Ils ont également montré que la performance de leur modèle

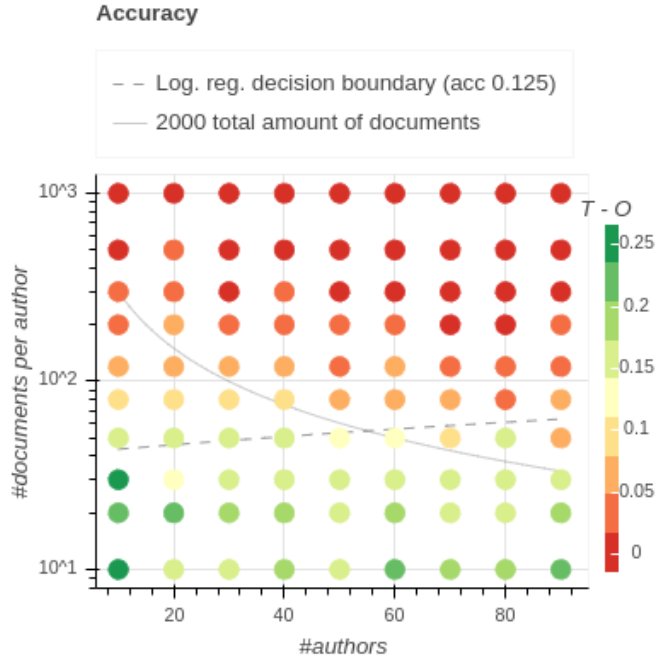


FIGURE 4.5 – Différence d’exactitude des deux versions du modèle *Dual-SNA*

CNN et de la *baseline* diminuent de la même manière lorsqu’ils augmentent le nombre d’auteurs.

Au lieu de cela, nous proposons d’étudier la corrélation entre le bénéfice à utiliser l’apprentissage par transfert et deux caractéristiques propres au jeu de données : le nombre d’auteurs et le nombre de documents par auteur. À cette fin, nous entraînons et évaluons les deux versions du modèle *Dual-SNA* sur 90 *U-sets* différents. Pour chaque combinaison de contraintes sur le nombre d’auteurs et le nombre de documents par auteur, les *U-sets* sont générés. Les classes ainsi que les documents sont choisis aléatoirement. 20% des documents sont réservés au test afin de déterminer le score final. Le « bénéfice » à utiliser l’apprentissage par transfert correspondra à la différence d’exactitude entre les deux versions du modèle *Dual-SNA*. Cette expérimentation permet de montrer dans quelles conditions il est pertinent de mettre en place l’apprentissage par transfert dans la tâche d’identification d’auteurs pour les articles d’actualité et de blog.

La figure 4.5 montre la différence d’exactitude dans la tâche d’identification d’auteurs entre les deux versions du modèle *Dual-SNA*. Les 90 points représentent chacun un *U-set* différent. Nous faisons varier le nombre d’auteurs de 10 à 90 et le nombre de documents par auteur de 10 à 1000. Les couleurs de chaque point indiquent la différence d’exactitude. Nous soustrayons l’exactitude de la seconde version du modèle *Dual-SNA* sans apprentissage par transfert à la première version. Une couleur rouge signifie que la différence d’exactitude, pour le *U-set* correspondant, est proche de 0. Une couleur verte, au contraire, signifie que la différence d’exactitude est grande. Les différences d’exactitude se situent entre 0 et 0.25. À noter que l’axe correspondant au nombre de documents par auteur est sur une échelle logarithmique.

Nous ajoutons deux courbes à la figure 4.5, la courbe discontinue correspond à une courbe de type *LSDB* (*logistic regression decision boundary*) indiquant la limite

entre deux catégories de *U-sets* : les *U-sets* pour lesquels l'apprentissage par transfert est très bénéfique avec une différence d'exactitude au-dessus de 0.125 et les autres *U-sets* avec une différence en dessous de 0.125. La courbe continue est une *LSDB* imaginaire et indique les coordonnées pour lesquelles les *U-sets* ont 2 000 documents au total. Cette courbe permet de positionner l'angle que prendrait une courbe *LSDB* si le bénéfice à utiliser de l'apprentissage par transfert était corrélé uniquement au nombre de documents. À noter que sans l'échelle logarithmique, cette courbe serait linéaire.

Comme nous le voyons par les couleurs ainsi que la courbe *LSDB* discontinue, le nombre d'auteurs (axe des abscisses) n'est pas corrélé au bénéfice. De plus, la courbe *LSDB* discontinue ne semble pas suivre la courbe *LSDB* imaginaire, indiquant une faible corrélation entre le bénéfice et le nombre de documents. Cependant, la courbe *LSDB* discontinue semble indiquer que le bénéfice est corrélé linéairement au nombre de documents par auteur puisqu'elle correspond à une fonction relativement constante sur ce graphe. Finalement, il semble que l'utilisation de l'apprentissage par transfert soit beaucoup plus pertinent lorsque le jeu de données comporte des classes sous-représentées, i.e. pas assez d'échantillons par auteur pour que le modèle puisse bien discriminer les auteurs. D'après cette figure, 50 documents par classe permettent d'obtenir un bénéfice important à utiliser l'apprentissage par transfert important avec une différence d'exactitude d'en moyenne 0.125.

4.4 Validation de l'hypothèse de filtrage

Comme expliqué dans la section 3.5, les documents du *R-set* ainsi que ceux des *U-sets* possèdent deux champs donnant accès au contenu du document filtré et non filtré. Ces champs sont les suivants :

sentences Correspond au texte prétraité et *tokenisé*.

filtered_sentences Correspond au champ *sentences* avec certaines phrases supprimées par la méthode de filtrage décrite en section 3.5. Une proportion de 30% des phrases a été supprimée du jeu de données *NewsID*. Ces phrases sont celles permettant une identification simple de l'auteur du document.

Chacun de ces champs correspond à une liste de listes. Le premier niveau rassemble les phrases que compose le document et le deuxième niveau rassemble les *tokens* de chaque phrase.

Afin de valider l'hypothèse de filtrage, nous entraînons deux modèles *SNA*. Le premier modèle, *SNA-nf*, est entraîné sur les champs *sentences* du *R-set*. Le second modèle, *SNA-f*, est entraîné sur les champs *filtered_sentences* du *R-set*. Nous reproduisons les expérimentations de ce chapitre afin d'évaluer la différence de performance des deux modèles. Nous utilisons les mêmes *U-sets* que dans la section 4.2 sauf pour les *U-sets* de type *NewsID-50-50* qui sont maintenant au nombre de 100 (au lieu de 5) dans les expérimentations de cette section. Tout d'abord, nous testons les modèles sur le partitionnement par auteur. Puis nous testons les modèles sur l'identification d'auteurs. Enfin, nous analysons l'impact du filtrage sur la mesure *TFIDF focus*.

4.4.1 Impact du filtrage sur le partitionnement par auteur

De la même manière qu’en section 4.2, nous testons les modèles sur le partitionnement par auteur. Nous utilisons les mêmes métriques. Le tableau 4.9 reporte les scores obtenus sur chacun des *U-sets*.

Chacun des modèles est testé à la fois sur les *U-sets* filtrés (reportés avec le label *Filt. U-set*) par la méthode décrite en section 3.5 mais également sur les *U-sets* non filtrés (reportés avec le label *Base U-set*). Lors de l’analyse de ces résultats, il faut donc faire la différence entre deux filtrages :

1. Le filtrage du *R-set* pour l’entraînement des réseaux de neurones profonds.
2. Le filtrage des *U-sets* pour l’évaluation des réseaux de neurones profonds.

Nous évaluons toutes les combinaisons de filtrage entraînement / évaluation. Ceci nous permet d’établir si la bonne performance de *SNA-f* est due à son évaluation sur les *U-sets* filtrés. Autrement dit, nous pourrions observer s’il existe des biais dans les scores obtenus par *SNA-f* lorsqu’il est entraîné sur un *R-set* filtré.

Dans la partie de gauche du tableau 4.9, la première colonne indique si le *U-set* est filtré ou non, la deuxième colonne donne le modèle évalué (donc le filtrage de *R-set*) et enfin la troisième indique quelle métrique est utilisée pour la ligne correspondante. Nous comparons donc *SNA-nf* et *SNA-f* sur les *U-sets* non filtrés puis *SNA-nf* et *SNA-f* sur les *U-sets* filtrés.

Sur les *U-sets* non filtrés, nous remarquons que *SNA-f* est plus performant sur les métriques *SimRank* et *DavB* (indice de *Davies-Bouldin*) sauf pour les *U-sets NewsID-50-50*. Cependant, la différence de performance est relativement basse. Les résultats pour la métrique *CalHar* (indice de *Calinski-Harabasz*) sont globalement à l’avantage de *SNA-nf* puisque plus performants sur une majorité des *U-sets*. Nous faisons les mêmes observations sur les résultats de la deuxième partie du tableau pour les *U-sets* filtrés : *SNA-f* est plus performant sur les métriques *SimRank* et *DavB*.

Il semblerait donc que pour cette tâche, la performance des deux modèles soit assez partagée en fonction des *U-sets*. Les résultats de *SNA-f* aux différentes métriques montrent :

1. D’après les scores *SimRank*, les similarités *cosinus* des projections de documents de *SNA-f* permettent de meilleurs ordonnancements des documents.
2. D’après les scores *DavB*, les partitions (ou *clusters*) de *SNA-f* semblent être plus différentes.
3. Cependant, d’après les scores *CalHar*, les partitions de *SNA-nf* semblent plus dispersées.

La différence basse des résultats entre les *U-sets* filtrés (ligne *Filtered* de la première colonne) et les *U-sets* non filtrés (ligne \neg *Filtered* de la première colonne) semble indiquer que *SNA-f* généralise suffisamment pour obtenir des performances proches même dans le cas de *U-sets* non filtrés, i.e. des *U-sets* plus faciles dans l’identification des auteurs. Il n’existe donc pas de biais lié au filtrage du *R-set*. La difficulté des *U-sets* dans l’identification des auteurs implique une plus grande difficulté dans le partitionnement par auteur comme le montre les scores dans le tableau. En effet, les scores de la deuxième partie du tableau sont en moyenne plus bas que les scores de la première partie du tableau.

<i>U</i> -set filtering													
	<i>SNA R</i> -set filtering	Metric	<i>NewsID-50-50 (100)</i>	<i>BlogCorpus-50-50 (5)</i>	<i>LiveJournal-50-50 (5)</i>	<i>WashingtonPost-50-50</i>	<i>Breitbart-50-50</i>	<i>BusinessInsider-50-50</i>	<i>CNN-50-50</i>	<i>GuardianUK-50-50</i>	<i>TheGuardian-50-50</i>	<i>NYTimes-50-50</i>	
<i>Filtered</i>	<i>Filtered</i>	<i>Filt.</i>	<i>SimRk</i>	0.73	0.45	0.46	0.32	0.56	0.35	0.41	0.25	0.34	0.426
		<i>CalHar</i>	107	28.8	30.1	15.2	64.6	35.9	50.4	6.5	9.2	35.8	
		<i>DavB</i>	2.61	3.84	5.28	8.29	4.29	7.37	7.57	11.1	8.29	5.71	
<i>Filtered</i>	<i>Filtered</i>	<i>Filt.</i>	<i>SimRk</i>	0.72	0.48	0.48	0.37	0.53	0.43	0.45	0.27	0.44	0.46
		<i>CalHar</i>	38.1	13.1	23.1	27.5	16.2	18.8	15.9	8.9	13.2	36.4	
		<i>DavB</i>	3.01	3.77	5.02	6.82	4.02	6.13	6.64	10.8	6.02	6.27	
<i>Filtered</i>	<i>Filtered</i>	<i>Filt.</i>	<i>SimRk</i>	0.55	0.39	0.40	0.33	0.37	0.36	0.36	0.30	0.43	0.38
		<i>CalHar</i>	37.0	22.0	21.4	11.0	16.9	20.8	15.3	8.1	18.8	13.7	
		<i>DavB</i>	3.55	4.29	5.58	7.09	5.6	6.06	6.16	7.79	4.96	5.91	
<i>Filtered</i>	<i>Filtered</i>	<i>Filt.</i>	<i>SimRk</i>	0.63	0.42	0.43	0.35	0.40	0.40	0.40	0.34	0.46	0.40
		<i>CalHar</i>	23.1	11.5	21.5	16.5	11.8	14.4	15.3	7.0	11.9	12.2	
		<i>DavB</i>	3.35	4.13	5.23	6.88	5.07	5.86	5.93	7.15	5.08	5.5	

TABLE 4.9 – Impact du filtrage sur le partitionnement par auteur

4.4.2 Impact du filtrage sur l’identification d’auteurs

Les résultats de l’expérimentation précédente semblent montrer que *SNA-f* est supérieur. Cependant, la métrique de partitionnement *CalHar* est à l’avantage de *SNA-nf*. Comme nous l’avons vu dans la section 4.3, cette métrique ne permet pas d’obtenir une bonne approximation de la performance des modèles sur une tâche de classification. Nous proposons donc de vérifier la différence de performance de ces deux modèles sur la tâche d’identification d’auteurs.

L’expérimentation est identique à celle décrite en section 4.3 : pour tous les *U*-sets, nous ne générons que les vecteurs stylométriques issus de *SNA-nf* et *SNA-f*. Nous concaténons à ces représentations les vecteurs *sparses* TFIDF pour chaque document. La combinaison des modèles *SNA* et TFIDF nous permet d’obtenir de meilleurs scores de classification afin d’observer une différence entre *SNA-f* et *SNA-nf* au plus proche des performances maximales attendues sur chaque *U*-set. Nous pouvons ainsi observer, dans le cas d’une amélioration par le filtrage du *R*-set, si l’introduction de la représentation TFIDF efface cette amélioration.

Nous reportons dans le tableau de résultats 4.10 les scores du modèle *SGDClassifier* entraîné sur les *U*-sets spécifiés par la colonne. La ligne indique les descripteurs utilisés pour l’entraînement du modèle parmi les combinaisons suivantes : *SNA-nf*, *SNA-f*, *SNA-nf + TFIDF*, *SNA-f + TFIDF*. Comme pour les autres expérimentations, les trois premiers *U*-sets correspondent à la moyenne des scores de plusieurs *U*-sets de même type. Le nombre de *U*-sets utilisés est donné entre parenthèses.

Ce tableau présente quatre paires de lignes comparatives. La première compare

<i>U-set filtering</i>	Model	<i>NewsID-50-50 (100)</i>	<i>BlogCorpus-50-50 (5)</i>	<i>LiveJournal-50-50 (5)</i>	<i>WashingtonPost-50-50</i>	<i>Breitbart-50-50</i>	<i>BusinessInsider-50-50</i>	<i>CNN-50-50</i>	<i>GuardianUK-50-50</i>	<i>TheGuardian-50-50</i>	<i>NYTimes-50-50</i>
<i>Filt.</i>	<i>SNA-nf</i>	0.80	0.61	0.51	0.41	0.70	0.44	0.41	0.59	0.28	0.58
	<i>SNA-f</i>	0.78	0.66	0.53	0.48	0.68	0.54	0.51	0.76	0.33	0.61
	<i>TFIDF+SNA-nf</i>	0.81	0.62	0.51	0.42	0.70	0.45	0.43	0.58	0.28	0.58
	<i>TFIDF+SNA-f</i>	0.79	0.73	0.56	0.51	0.71	0.55	0.53	0.77	0.35	0.63
<i>Filtered</i>	<i>SNA-nf</i>	0.64	0.50	0.43	0.39	0.40	0.40	0.38	0.76	0.36	0.48
	<i>SNA-f</i>	0.69	0.57	0.47	0.42	0.48	0.45	0.42	0.73	0.43	0.55
	<i>TFIDF+SNA-nf</i>	0.64	0.51	0.43	0.39	0.40	0.39	0.39	0.75	0.36	0.49
	<i>TFIDF+SNA-f</i>	0.70	0.65	0.51	0.46	0.50	0.48	0.46	0.77	0.46	0.60

TABLE 4.10 – Impact du filtrage sur l’identification d’auteurs (scores d’exactitude)

SNA-nf et *SNA-f* sur les *U-sets* non filtrés. La deuxième paire de lignes reprend les mêmes modèles mais les vecteurs *sparses* TFIDF sont ajoutés en tant que descripteurs pour chaque document par une concaténation. Les troisième et quatrième paires de lignes correspondent aux mêmes types de descripteurs mais évalués sur les *U-sets* filtrés.

Comme le montre ce tableau, les vecteurs de *SNA-f* obtiennent, pour la majorité des paires de lignes, une meilleure exactitude. À noter que pour *NewsID-50-50* sur les *U-sets* non filtrés, *SNA-nf* obtient une meilleure exactitude. Les vecteurs TFIDF sont suffisamment complémentaires pour permettre une amélioration des performances sur tous les *U-sets*. Ces vecteurs n’ont pas d’impact sur les conclusions de cette expérimentation, ils ne font que compléter les descripteurs *SNA*. L’amélioration par le filtrage est donc significative même lorsque les modèles exploitent déjà des vecteurs représentatifs « *baseline* » performants : les vecteurs TFIDF.

Finalement, ce tableau indique que les vecteurs *SNA-f* permettent une identification des auteurs plus performante, ce qui valide l’*hypothèse de filtrage*. Les vecteurs *SNA-f* obtiennent de meilleures performances également sur les *U-sets* non filtrés, montrant la capacité du modèle à généraliser ses représentations stylométriques. Nous noterons la différence d’exactitude entre les scores sur les *U-sets* filtrés et les *U-sets* non filtrés, indiquant que la difficulté des *U-sets* filtrés est effectivement plus grande pour cette tâche.

4.4.3 Impact du filtrage sur l’adéquation des modèles avec la *non-spécificité sémantique*

Nous voulons évaluer l’impact du filtrage sur la mesure *TFIDF focus*. Dans ce but, nous calculons le *TFIDF focus* de *SNA-nf* et *SNA-f* sur l’ensemble des *U-sets*. Nous reportons les scores dans le tableau 4.11. Comme précédemment, nous testons

<i>U</i> -set filtering	<i>SNA R</i> -set filtering	<i>NewsID-50-50 (100)</i>	<i>BlogCorpus-50-50 (5)</i>	<i>LiveJournal-50-50 (5)</i>	<i>WashingtonPost-50-50</i>	<i>Breitbart-50-50</i>	<i>BusinessInsider-50-50</i>	<i>CNN-50-50</i>	<i>GuardianUK-50-50</i>	<i>TheGuardian-50-50</i>	<i>NYTimes-50-50</i>
\neg <i>Filt.</i>	\neg <i>Filt.</i>	0.385	0.54	0.42	0.14	0.23	0.25	0.16	0.11	0.11	0.17
	<i>Filt.</i>	0.003	0.08	0.06	0.12	0.19	0.23	0.17	0.13	0.10	0.14
<i>Filt.</i>	\neg <i>Filt.</i>	0.006	0.14	0.12	0.26	0.50	0.57	0.44	0.55	0.37	0.38
	<i>Filt.</i>	0.005	0.11	0.09	0.22	0.39	0.46	0.35	0.44	0.30	0.32

TABLE 4.11 – Impact du filtrage sur le *TFIDF focus*

sur les deux versions de *U*-sets : \neg *Filtered* et *Filtered*.

Ce tableau montre que le modèle *SNA-f* concentre son attention sur des termes à poids TFIDF faibles, le score *TFIDF focus* étant plus faible pour tous les *U*-sets sauf pour *CNN-50-50* et *GuardianUK-50-50* en version \neg *Filtered*. Le filtrage du jeu de données de référence *R*-set permet donc d’accentuer le ciblage de l’attention du modèle *SNA* sur des termes faiblement sémantiques tels que les mots fonctions. En définitive, le filtrage du *R*-set permet non seulement d’améliorer les performances des réseaux de neurones profonds dans le partitionnement par auteur et l’identification d’auteurs mais permet aussi aux modèles de satisfaire d’autant plus la seconde propriété de notre définition du style : la *non-spécificité sémantique*.

4.5 Conclusion

Dans ce chapitre, nous avons validé expérimentalement l’*hypothèse de généralisation du style* grâce à l’application de la méthode de ciblage des indices intra-auteurs consistants sur la tâche de partitionnement par auteur sur 22 ensembles de test. Cette hypothèse stipule que, d’une part, nous pouvons représenter des documents nouveaux d’auteurs inconnus en généralisant les caractéristiques de style d’auteurs de références, et que, d’autre part, si deux représentations de deux documents nouveaux sont proches, cela indique que ces documents ont plus probablement été écrits par le même auteur.

Les modèles *SNA* et *DBert-ft* ont prouvé l’intérêt de la méthode de ciblage des indices intra-auteurs consistants grâce aux résultats obtenus sur la tâche de partitionnement par auteur comparés aux différentes *baselines* implémentées. Nous avons montré que l’ajout d’une couche d’attention et de la bidirectionnalité au modèle *SNA* permettaient d’obtenir de meilleurs résultats. De plus, nous avons montré que la méthode de ciblage des indices intra-auteurs consistants permettait d’améliorer les résultats de *DBert-ft* par rapport à son modèle non affiné *DBert*, validant ainsi l’*hypothèse de généralisation du style*.

Dans une seconde expérimentation, nous avons testé les différents modèles proposés sur la tâche d’identification d’auteurs. Les modèles les plus performants pour

cette tâche sont ceux ayant le meilleur score de partitionnement. Nous avons notamment constaté que les représentations vectorielles du modèle *DBert-ft* obtiennent les meilleurs résultats sur les deux tâches et que la combinaison *DBert-ft* et *Doc2Vec* est la meilleure combinaison de modèles sur les deux tâches. Ces deux modèles semblent être complémentaires dans les tâches de la stylométrie.

La métrique *SimRank*, proposée dans ce chapitre, permet de mesurer la qualité d'un partitionnement. Elle se base sur nDCG et évalue la qualité de l'ordonnement des documents les uns par rapport aux autres, suivant une fonction de similarité. Celle-ci permet une bonne approximation des performances de la tâche source, le partitionnement par auteur, vers la tâche cible, l'identification d'auteurs. Grâce à la métrique *TFIDF focus* proposée au chapitre 2, nous avons montré que les modèles entraînés par la méthode de ciblage des indices intra-auteurs consistants satisfaisaient la seconde propriété de notre définition du style : la « *non-spécificité sémantique* ».

Enfin, nous avons validé expérimentalement une seconde hypothèse, l'*hypothèse de filtrage*, en comparant différents modèles entraînés, d'une part, sur un ensemble de référence, et, d'autre part, sur ce même ensemble dont 30% des phrases « révélatrices » ont été filtrées. Cette expérimentation nous a permis de conclure que le filtrage de l'ensemble de référence nous permettait d'augmenter les performances de nos modèles dans le cadre de méthode de ciblage des indices intra-auteurs consistants et que ce filtrage satisfaisait d'autant plus la *non-spécificité sémantique*. Le processus de filtrage proposé permet un gain moyen de $\sim 5\%$ sur la tâche d'identification d'auteurs et de $\sim 3.5\%$ sur le partitionnement par auteur selon la métrique *SimRank*.

À travers l'exploitation du corpus de référence, i.e. le *R-set* du jeu de données *NewsID*, nous avons démontré que se baser sur un ensemble d'auteurs et de sources de référence suffisamment large permet de généraliser les représentations stylométriques de « nouveaux » documents. Ainsi, nous avons répondu à la question de recherche *QR1* formulée en introduction de ce manuscrit. Puis, grâce aux résultats obtenus par les modèles *SNA* et *DBert-ft* sur les deux tâches de la stylométrie proposées, nous avons validé la faisabilité de notre méthode d'apprentissage de la représentation du style, celle-ci permettant de s'affranchir de l'implémentation de descripteurs. Ainsi, nous avons répondu à la question de recherche *QR2*.

En perspective de ces travaux, nous pourrions tester l'entraînement des modèles « état de l'art » les plus performants tels qu'*InferSent* (Conneau et al., 2017), *BERT* (Devlin et al., 2019) et *XLNet* (Yang et al., 2019) par la méthode de ciblage des indices intra-auteurs consistants. Nous pourrions également tester notre méthode sur d'autres domaines, notamment le domaine des « *fanfictions*¹⁰ » avec les challenges d'identification d'auteurs *PAN19* (Kestemont et al., 2019). Comme nous l'avons montré, nos modèles pré-entraînés peuvent aider le domaine de la stylométrie avec la tâche d'identification d'auteurs (via l'apprentissage par transfert). Nous pourrions aussi tester les modèles sur une seconde tâche : la vérification d'auteur (ou *authorship verification*).

Les modèles proposés dans ce chapitre permettent de projeter tout document dans un espace stylométrique d'auteurs de référence. Les caractéristiques de style capturées par ces modèles peuvent donc également permettre l'amélioration de modèles dans des tâches externes telles que la détection de fausses nouvelles. Dans ce domaine de recherche, les modèles entraînés par la méthode de ciblage des indices intra-auteurs

10. Une fanfiction est un récit que certains fans écrivent pour augmenter ou transformer l'univers d'un produit médiatique (e.g. roman, manga, série télévisée, film).

consistants peuvent permettre l'identification de sources non fiables. Il est possible d'apprendre conjointement à identifier une source dans un ensemble de référence, mais également sa disposition à écrire de fausses nouvelles (via une classification « multi-labels ») grâce à des données annotées tel que proposé par [Baly et al. \(2018\)](#). Entraîner un modèle à discriminer les journaux ou blogs en plus des auteurs physiques, comme nous le proposons, pourrait permettre d'aider à détecter les fausses nouvelles puisqu'elles sont généralement écrites par des journaux spécifiques considérés comme non fiables.

Dans les chapitres suivants, après un état de l'art sur les systèmes de recommandation, nous détaillons dans quelle mesure le style écrit peut être exploité dans le cadre de la recommandation d'articles d'actualité. Par les expérimentations proposées, nous tentons de répondre aux questions de recherche *QR3*, *QR4* et *QR5*.

Deuxième partie

Recommandation d'articles
d'actualité

Chapitre 5

État de l’art des systèmes de recommandation

5.1 Introduction

Dans ce chapitre, la section 5.2 donne un aperçu du domaine des systèmes de recommandation et la section 5.3 un aperçu des méthodes et métriques d’évaluation du domaine. Dans cette dernière, nous proposons une taxonomie de ces méthodes et métriques. Nous proposons également une discussion sur les biais d’évaluation *offline* faisant écho au travail réalisé aux chapitres 7 et 8 sur la nécessité d’effectuer l’évaluation qualitative des algorithmes de recommandation et au chapitre 9 sur la nécessité d’avoir aussi recours à l’évaluation *online*. Dans la section 5.4, nous donnons un aperçu du domaine de la recommandation d’articles d’actualité. Nous analysons les limites de ce domaine dans l’exploitation du style écrit pour la recommandation d’articles d’actualité et revenons sur les questions de recherche *QR3*, *QR4* et *QR5*. Ces trois sections introduisent toutes les notions nécessaires à la lecture des chapitres 6, 7, 8 et 9.

5.2 Systèmes de recommandation

5.2.1 Généralités

Un système de recommandation est une application permettant à des utilisateurs de « consommer » des items lui étant recommandés selon différents critères. La consommation des items peut se traduire par la lecture d’articles dans des systèmes de recommandation d’articles d’actualité (e.g. Google Actualités, Yahoo Actualités), le visionnage de films dans des systèmes de recommandation de films (e.g. Netflix, Disney+, Prime Video) ou encore l’achat de produits sur des sites e-commerces (e.g. Amazon, eBay, Rakuten). Ces plateformes sont des plateformes commerciales et utilisent des algorithmes de recommandation privés. L’objectif de ces algorithmes est d’améliorer l’expérience des utilisateurs en leur proposant des items susceptibles de les intéresser, et ainsi de générer plus de revenus par une consommation intensifiée. Les modèles économiques de ces plateformes commerciales sont divers : contenus sponsorisés, publicités, achats intégrés, abonnements. Cependant, malgré cet objectif intéressé, la « recommandation » consiste, en définitive, à satisfaire l’utilisateur afin de le fidéliser, celui-ci pouvant se tourner vers la concurrence. Ces outils sont

donc bénéfiques lorsque le contenu est abondant. En effet, les utilisateurs n’ont pas la capacité, notamment en temps, de sélectionner par leurs propres moyens le contenu pouvant satisfaire leur besoin. D’un point de vue de l’utilisateur, cette personnalisation peut néanmoins avoir des conséquences non souhaitables :

Atteinte à la vie privée Comme montré par [Madejski et al. \(2011\)](#), pour une majorité d’utilisateurs, les « intentions de partage » en termes de confidentialité des données privées ne sont pas en adéquation avec les paramètres réglés dans les applications.

Addictions Les réseaux sociaux sont susceptibles de provoquer des addictions ([Andreassen, 2015](#)), surtout lorsque l’accès aux plateformes est facilité par les smartphones.

Surconsommation La personnalisation peut encourager une surconsommation néfaste pour les utilisateurs, notamment sur les plateformes e-commerce.

Bulles de filtres Un phénomène qui tendrait à restreindre la diversité de consommation (e.g. articles d’actualité, messages d’autres utilisateurs) de chaque individu et ainsi provoquer un « isolement intellectuel » de ceux-ci comme l’explique [Pariser \(2011\)](#). Nous détaillerons les bulles de filtres (ou *filter bubble*) en section 5.3.2.

La plupart des algorithmes de recommandation sont issus de la recherche dans le domaine des systèmes de recommandation. Pour cela, la communauté peut s’appuyer sur différents jeux de données servant à évaluer leurs algorithmes. Par exemple, *MovieLens* ([Harper et Konstan, 2015](#)) est un jeu de données permettant d’expérimenter des algorithmes de recommandation de films. De même, *LastFM* ([Bertin-Mahieux et al., 2011](#)) est dédié à la recommandation de musique, *CiteULike* ([Wang et al., 2013](#)) à la recommandation de ressources bibliographiques et *Delicious* ([Rossi et Ahmed, 2015](#)) à la recommandation de pages web. L’évaluation exploitant les jeux de données statiques est dite « *offline* ». Ce type de jeux de données permet de prédire, avec plus ou moins de biais, les performances d’un algorithme de recommandation s’il était placé en conditions réelles. Il existe aussi des plateformes mises à disposition de la communauté de recherche permettant d’effectuer l’évaluation « *online* » d’algorithmes, c’est-à-dire une évaluation impliquant des utilisateurs en temps réel. *NewsREEL* ([Hopfgartner et al., 2016](#)) est un exemple de plateforme de recommandation *online* qui propose aux équipes de recherche d’évaluer leurs algorithmes de recommandation d’articles d’actualité par l’intermédiaire de différents journaux en ligne.

Les algorithmes de recommandation exploitent les données d’« interactions » entre utilisateurs et items qui sont récoltées sur une période donnée. Ces données peuvent prendre la forme de retours de pertinence « implicites » tels que les clics, les visionnages et les achats, ou « explicites » tels que les notes, les mentions « j’aime » et les préférences données en réponse à un questionnaire. Ces retours de pertinence sont dits « implicites » lorsqu’ils sont nécessaires à l’utilisation de la plateforme. Les clics, par exemple, sont des actions nécessaires lorsqu’il s’agit, pour l’utilisateur, de lire le contenu d’un article d’actualité. Un retour de pertinence est dit « explicite » lorsque l’action vise à donner une information sur les intérêts de l’utilisateur. Il est généralement relatif à un item ou à une catégorie d’items. La note qu’attribue un utilisateur à un film, par exemple, peut être considérée comme un retour de pertinence explicite. En effet, elle apporte des informations sur l’intérêt de l’utilisateur vis-à-vis

du film. Cette action n'est, néanmoins, pas nécessaire au visionnage des films sur la plateforme. L'avantage des retours de pertinence explicites est qu'ils permettent de disposer, en amont, d'informations sans que l'utilisateur ait encore interagi avec le système. La plateforme Netflix, par exemple, demande aux nouveaux inscrits une sélection d'items que l'utilisateur a déjà consommé et aimé. Les retours de pertinence explicites peuvent être utiles lorsqu'il est difficile d'enregistrer des retours implicites ou lorsque ceux-ci ne sont pas suffisamment fiables. Cependant, contrairement aux retours de pertinence implicites, ce type de retour pertinence est généralement très peu exhaustif et nécessite à l'utilisateur de consacrer du temps à effectuer les actions requises (e.g. annotations, sélection), ce qui peut altérer son « expérience ».

Les algorithmes exploitant les données d'interactions passées des utilisateurs appartiennent à la méthode de « filtrage collaboratif ». Il s'agit de prédire les futures interactions d'un utilisateur pour déterminer quels items doivent être recommandés, ou prédire les notes que l'utilisateur attribuerait aux items. Les algorithmes de recommandation peuvent également exploiter le contenu des items et les informations relatives aux utilisateurs, mais aussi des informations externes comme celles disponibles sur les profils issus des réseaux sociaux des utilisateurs, ou encore les informations disponibles sur Wikipédia concernant un item. Les algorithmes exploitant le contenu des items appartiennent à la méthode du « filtrage par le contenu ». L'hybridation du filtrage collaboratif et du filtrage par le contenu constitue une autre catégorie de méthode. Dans cette section, nous détaillons ces trois méthodes de recommandation.

5.2.2 Filtrage collaboratif

Le filtrage collaboratif est une méthode consistant à prédire les préférences d'un utilisateur en exploitant ses préférences exprimées implicitement ou explicitement dans le passé et celles d'autres utilisateurs du système. Pour un utilisateur en particulier, il s'agit de prédire la pertinence d'un item candidat à une recommandation en cherchant la pertinence observée de cet item chez des utilisateurs similaires, i.e. ayant les mêmes préférences. L'hypothèse sous-jacente est que si une personne A a la même opinion qu'une personne B sur un premier sujet, elle aura plus probablement une opinion proche de celle de B sur un second sujet qu'une opinion proche de celle d'une personne C prise au hasard sur ce second sujet.

Le filtrage collaboratif consiste dans un premier temps à construire la matrice d'interactions utilisateurs-items. Celle-ci se compose généralement :

1. soit de retours de pertinence explicites qui peuvent, par exemple, correspondre à des notes attribuées aux items entre 1 et 5 ;
2. soit de retours de pertinence implicites qui peuvent être binaires : 1 signifiant que l'item est pertinent et 0 que l'utilisateur n'a pas encore interagi avec l'item, e.g. qu'il ne l'a jamais cliqué ou visionné.

Avec U l'ensemble des utilisateurs du système et I l'ensemble des items du système, la matrice d'interaction P est alors de taille $|U| \cdot |I|$ et l'élément $P_{i,j}$ correspondra à la valeur du retour de pertinence entre l'utilisateur U_i et l'item I_j . Les items candidats à une recommandation doivent pouvoir être identifiés dans la matrice. Par exemple, dans le cas d'une matrice d'interactions composée de retours de pertinence implicites, une valeur de 0 indiquera que l'item est « candidat » à une recommandation.

Pour un utilisateur U_i représenté par la ligne P_i de la matrice d'interaction, le filtrage collaboratif consiste ensuite à trouver les utilisateurs les plus similaires et à effectuer des recommandations sur la base de ceux-ci. Les utilisateurs les plus similaires sont obtenus en effectuant la similarité entre P_i et toutes les autres lignes de la matrice P correspondant aux autres utilisateurs du système. Les fonctions de similarité utilisées sont diverses : la similarité euclidienne, la similarité cosinus, la corrélation de Pearson. Le choix de cette fonction dépendra du type de données que contient la matrice d'interactions. Enfin, pour prédire la pertinence d'un item candidat (i.e. dont la valeur est 0 dans le vecteur P_i), il est possible d'effectuer une moyenne pondérée de la pertinence de cet item dans le vecteur des utilisateurs les plus similaires. Cette méthode est appelée le « filtrage collaboratif utilisateur » (ou *user-based collaborative filtering*). De même, il est possible de « compléter » la matrice d'interactions en calculant des similarités entre colonnes, c'est-à-dire entre items : c'est le « filtrage collaboratif item » (ou *item-based collaborative filtering*).

Par exemple, pour la recommandation de films, Sarwar et al. (2001) utilisent le filtrage collaboratif item sur le jeu de données *MovieLens* (Harper et Konstan, 2015). Jin et al. (2004) proposent d'employer le filtrage collaboratif utilisateur sur les jeux de données *MovieRating* et *EachMovie*. Enfin, Wang et al. (2006) fusionnent les deux approches pour améliorer la recommandation de films sur les jeux de données *MovieLens* (Harper et Konstan, 2015) et *EachMovie*.

Le filtrage collaboratif utilisateur et le filtrage collaboratif item sont dits « basés sur la mémoire ». Ces méthodes sont coûteuses en temps de calcul puisqu'il est nécessaire de prendre en compte la matrice entière pour effectuer une unique recommandation (Xue et al., 2005). À l'opposé, le filtrage collaboratif « basé sur des modèles » permet un temps de calcul réduit lors de la génération de recommandations. Il s'agit généralement d'effectuer une réduction en dimension de la matrice d'interactions par des méthodes telles que la décomposition en valeurs singulières (ou *singular value decomposition*, abrégée SVD) (Halko et al., 2009) afin de rendre le calcul des similarités moins coûteux. D'autres méthodes telles que les réseaux de neurones de type « machine de Boltzmann restreinte » (ou *Restricted Boltzmann machine*, abrégé RBM) permettent de pallier le fait que la matrice d'interactions est creuse (ou *sparse*), i.e. lorsqu'elle contient beaucoup de valeurs à 0, rendant les recommandations impossibles à effectuer. Salakhutdinov et al. (2007) ont par exemple montré que les RBM permettent d'obtenir de meilleures recommandations sur les données Netflix (Bennett et Lanning, 2007) pour la recommandation de films. Plus récemment, Sedhain et al. (2015) ont montré que des réseaux de neurones de type *auto-encoder* peuvent obtenir de meilleures performances comparés aux RBM sur les jeux de données *MovieLens* (Harper et Konstan, 2015) et Netflix (Bennett et Lanning, 2007). La procédure consiste à utiliser un *auto-encoder* pour réduire en dimension des vecteurs items « partiels », i.e. des vecteurs dont certaines interactions utilisateurs-système sont manquantes, puis à reconstruire ceux-ci dans le but d'effectuer des recommandations en prédisant les interactions manquantes.

Malgré les performances des systèmes de recommandation basés sur le filtrage collaboratif, ceux-ci présentent différents désavantages :

Le démarrage à froid item (ou *item cold start*) Lorsqu'un nouvel item est introduit dans le système, il est difficile de le recommander étant donné qu'aucun utilisateur n'a encore interagi avec.

Le démarrage à froid utilisateur (ou *user cold start*) Il est difficile de re-

commander des items à un nouvel *utilisateur* étant donné qu'il n'a encore rien consommé.

La scalabilité Correspond à la capacité d'un système à supporter un changement d'échelle (e.g. en termes de données, d'activité, de trafic, etc.) sans altération de ses fonctionnalités. Dans le cadre des systèmes de recommandation, le nombre d'utilisateurs et d'items présents sur certaines plateformes nécessite un temps de calcul considérable.

La sparsity Certaines plateformes ont beaucoup d'items, ce qui a pour conséquence que la plupart des utilisateurs n'ont interagi qu'avec un sous-ensemble restreint des items disponibles. En conséquence, le calcul des similarités entre utilisateurs ne permet pas de générer des recommandations puisque les utilisateurs ont très peu d'interactions communes.

La couverture (ou *coverage*) et les items *long-tail* Les items n'ayant été impliqués dans aucune interaction utilisateur-item ne peuvent pas être recommandés. Autrement dit, le système, par les recommandations qu'il génère, ne couvre pas l'ensemble des items. Ces items sont dits « *long-tail* ». Le système a alors une faible « couverture » (ou *coverage*), parfois aussi appelée « diversité agrégée » (ou *aggregate diversity*).

5.2.3 Filtrage par le contenu

Le filtrage par le contenu est une méthode visant à exploiter les données attachées aux items et aux utilisateurs dans l'objectif d'effectuer des recommandations. Ces données peuvent être disponibles directement dans le système de recommandation ou correspondre à un enrichissement par des ressources externes au système. L'idée est de recommander à l'utilisateur les items similaires à ceux qu'il a aimés dans le passé. À la différence du filtrage collaboratif, la décision de recommandation d'un item ne se fonde pas sur les interactions d'autres utilisateurs mais sur le contenu de celui-ci. Ainsi, si une plateforme de recommandation n'est composée que d'un seul utilisateur, il est possible d'effectuer une recommandation. Autrement dit, le nombre d'utilisateurs inscrits dans la plateforme n'a aucun impact sur les recommandations générées.

Il existe de nombreuses procédures permettant d'effectuer des recommandations par filtrage par le contenu. La plus courante consiste à :

1. Construire un « profil utilisateur » à partir de l'historique d'interactions (e.g. lectures, visionnages) de celui-ci.
2. Calculer les similarités entre ce profil et des items candidats.
3. Recommander les items candidats les plus similaires.

Par exemple, pour la recommandation de pages web, [Van Meteren et Van Someren \(2000\)](#) proposent de construire les vecteurs TFIDF de toutes les pages, puis de construire les profils utilisateurs en faisant la moyenne des vecteurs TFIDF de leur historique, et enfin de recommander les articles candidats les plus similaires suivant une similarité cosinus. [Abel et al. \(2011\)](#) proposent d'utiliser à la fois les *tweets* (via les entités nommées, les *hashtags*) d'utilisateurs Twitter ainsi que le contenu d'articles pour effectuer une recommandation d'articles d'actualité basée sur le contenu.

Le filtrage par le contenu est généralement moins performant que le filtrage collaboratif car il ne bénéficie pas des relations de préférences entre items qui peuvent

être déduites de l'ensemble des interactions utilisateurs-items. Ces « relations » peuvent être considérées comme des caractéristiques latentes qui ne peuvent pas être capturées par des méthodes d'extraction de connaissances se basant sur le contenu des items. Cependant, le filtrage par le contenu est utile lorsqu'il n'existe pas (ou très peu) d'interactions communes entre utilisateurs. Il permet de pallier les différents problèmes mentionnés pour le filtrage collaboratif, et notamment la recommandation de nouveaux items (i.e. problème du démarrage à froid item). Il présente néanmoins un désavantage majeur :

La sur-spécialisation (ou *overspecialization*) Correspond à l'incapacité de l'algorithme de recommandation à proposer du contenu divers, inattendu et nouveau par rapport à l'historique de l'utilisateur.

Cependant, la majorité de ces algorithmes ne sont pas radicalement basés sur le contenu. Il s'agit souvent de modèles entraînés sur l'historique de tous les utilisateurs. Les recommandations d'un utilisateur sont donc influencées par l'apprentissage du modèle sur les données d'autres utilisateurs (Wang et al., 2018; Volkovs et al., 2017). Ces algorithmes utilisent donc, comme ceux basés sur le filtrage collaboratif, des données de type « collaboratif ». En effet, les décisions prises par l'algorithme, pour un utilisateur en particulier, dépendent aussi de la consommation des autres utilisateurs.

5.2.4 Méthodes hybrides

Dans la littérature, il est courant de considérer un algorithme comme étant « hybride » lorsqu'il consiste en l'hybridation d'un ou plusieurs algorithmes basés sur le filtrage collaboratif et d'un ou plusieurs algorithmes basés sur le filtrage par le contenu. L'hybridation est effectuée grâce à ces quelques méthodes décrites par Burke (2002) :

Mixage (ou *Mixed*) Cette méthode consiste simplement à présenter toutes les recommandations issues des différents algorithmes.

Échange (ou *Switching*) Consiste à choisir un algorithme parmi plusieurs selon le contexte courant. Cette méthode nécessite des paramètres supplémentaires qui peuvent être choisis à l'avance ou optimisés.

Pondération (ou *Weighted*) Les scores de pertinence attribués aux items par les différents algorithmes sont combinés numériquement.

Combinaison de descripteurs (ou *Feature combination*) Utilise conjointement les représentations des différents algorithmes en entrée à un algorithme général.

Cascade Consiste à utiliser un premier algorithme afin d'effectuer une première sélection des items pertinents, puis à raffiner cette sélection par un second algorithme. Cette méthode est utilisée notamment lorsque l'on cherche à réduire le temps de calcul en filtrant l'ensemble des items candidats par une méthode peu coûteuse, puis à raffiner par une seconde méthode plus coûteuse mais aussi plus précise. Bien entendu, il est possible d'effectuer un filtrage en cascade par l'utilisation de plus de deux algorithmes.

Augmentation de descripteurs (ou *Feature augmentation*) Les prédictions d'un premier algorithme (e.g. scores de pertinence, représentations vectorielles) sont données en entrée à un second algorithme.

Meta-niveau (ou *Meta-level*) Un modèle est appris par un premier algorithme et donné en entrée à un second.

Ces méthodes permettent de bénéficier des avantages du filtrage collaboratif (e.g. précision, qualité, diversité, nouveauté) et du filtrage par le contenu (e.g. couverture, démarrage à froid, scalabilité). Elles permettent donc l'utilisation conjointe d'algorithmes « complémentaires » (Ben Ticha, 2015). Par exemple, Aslanian et al. (2016) proposent une méthode hybride améliorant à la fois la nouveauté et la pertinence dans la recommandation de films avec le jeu de données *MovieLens* (Harper et Konstan, 2015). Ils parviennent à réduire le problème de démarrage à froid lié au filtrage collaboratif. Ces algorithmes d'hybridation emploient à la fois le filtrage collaboratif et le filtrage par le contenu et sont, par conséquent, considérés comme appartenant à une catégorie à part. Cependant, il est à noter que ces méthodes d'hybridation peuvent aussi être utilisées pour combiner plusieurs algorithmes de recommandation qui peuvent appartenir à une même catégorie (Burke, 2002). Comme nous le verrons au chapitre 8, au-delà de la combinaison d'algorithmes, l'hybridation permet aussi l'utilisation de représentations complémentaires d'items lorsque celles-ci capturent différentes caractéristiques liées au contenu.

La méthode par pondération a pour avantage de ne nécessiter que les sorties d'algorithmes déjà établis, c'est-à-dire les scores de pertinence pour chaque item candidat. Selon Danilova et Ponomarev (2017), cette méthode est la plus utilisée (69%) dans la littérature du domaine. Elle ne nécessite pas de modifier un algorithme existant et permet le réglage du poids accordé à chaque algorithme. Ainsi, si les algorithmes d'une hybridation optimisent différents critères (e.g. précision, diversité), alors il est possible de donner l'avantage à certains de ces critères en réglant le poids des algorithmes. Par exemple, Ribeiro et al. (2012) proposent l'utilisation de l'hybridation par pondération de huit algorithmes afin de régler la balance entre diversité, nouveauté et pertinence, les poids étant optimisés en fonction de chaque utilisateur par un algorithme génétique. Cai et al. (2020) emploient également un algorithme génétique afin d'optimiser la diversité et la pertinence des recommandations sur le jeu de données *MovieLens* (Harper et Konstan, 2015). Suriati et al. (2017) proposent une hybridation du filtrage par le contenu et du filtrage collaboratif sur les données *MovieLens* (Harper et Konstan, 2015) et montrent qu'elle permet d'obtenir une meilleure « couverture ». Javari et al. (2016) proposent le même type d'expérimentation et montrent que l'hybridation permet de régler un compromis entre nouveauté, diversité et pertinence des recommandations. Zhou et al. (2010) améliorent à la fois la diversité et la pertinence sur les jeux de données Netflix (Bennett et Lanning, 2007) et *Delicious* (Rossi et Ahmed, 2015).

La méthode de pondération consiste à calculer une combinaison linéaire des scores de chaque item. Après normalisation entre 0 et 1 des scores en sortie de chaque algorithme, le calcul du score hybride de l'item i revient à calculer :

$$S_i(P, W) = \frac{\sum_{u=1}^{|W|} W_u P_{u,i}}{|W|} \quad (5.1)$$

Avec W le vecteur des poids attribués à chaque algorithme et P la matrice donnant les scores de chaque item pour chaque algorithme de l'hybridation. Le vecteur W est de taille n le nombre d'algorithmes pris en compte dans l'hybridation et la matrice P est de taille $n \cdot m$ avec m le nombre d'items candidats. Ainsi, W_u correspondant au poids attribué à l'algorithme u et $P_{u,i}$ correspondant au score de pertinence de l'item

i par l’algorithme u . Bien entendu, lorsque les scores de pertinence de tous les items ne sont pas retournés par un algorithme, alors l’implémentation de l’hybridation par pondération nécessite de faire un choix pour les items manquants (e.g. valeur par défaut, exclusion). Pour terminer, il s’agit ensuite de réordonner les items selon leur score hybride puis d’effectuer une recommandation en sélectionnant les items les plus pertinents. À noter qu’il est possible d’apprendre le vecteur W pour chaque utilisateur comme le propose [Ribeiro et al. \(2012\)](#) pour la recommandation de films et [Claypool et al. \(1999\)](#) pour la recommandation d’articles d’actualité.

5.3 Évaluation des systèmes de recommandation

Dans cette section, nous décrivons les méthodes d’évaluation *offline*, qualitative et *online* des systèmes de recommandation. Nous terminons par proposer une nouvelle classification des méthodes et métriques d’évaluation dédiées aux systèmes de recommandation.

5.3.1 Évaluation *offline*

L’évaluation *offline* consiste à exploiter un jeu de données d’interaction utilisateurs-items. Ce type de jeu de données est souvent l’« enregistrement » des activités des utilisateurs sur une plateforme leur proposant d’« interagir » (e.g. clics, lectures, visionnage) avec des items (e.g. films, articles, musiques, produits). Cette plateforme peut intégrer ou non un algorithme de recommandation. Comme expliqué en section 5.2.1, ces jeux de données se composent soit de retours de pertinence explicites correspondant par exemple à des notes de 1 à 5 comme dans le jeu de données *MovieLens* ([Harper et Konstan, 2015](#)), soit de retours de pertinence implicites correspondant par exemple à la « lecture » d’un titre comme dans le jeu de données *LastFM* ([Bertin-Mahieux et al., 2011](#)).

La plupart des protocoles d’évaluation sur ce type de jeu de données consistent à effectuer un découpage aléatoire des données d’interaction en un « ensemble d’entraînement » (ou *train set*) et un « ensemble de test » (ou *test set*). Par exemple, pour le jeu de données *MovieLens*, et comme le propose [Cremonesi et al. \(2010\)](#), le découpage consiste à sélectionner aléatoirement 1.4% des interactions utilisateurs-items pour constituer l’ensemble de test et sélectionner le reste pour constituer l’ensemble d’entraînement. Dans ce jeu de données, il existe un ensemble d’items $I = \{i_1, \dots, i_{|I|}\}$ et un ensemble d’utilisateurs $U = \{u_1, \dots, u_{|U|}\}$. Les interactions utilisateurs-items sont représentées par un ensemble de triplets $P = \{p_1, \dots, p_{|P|}\}$ et $\forall p \in P, p = (u \in U, i \in I, s)$ avec u un utilisateur, i un item et s une note entre 1 et 5. Le découpage consiste donc en la sélection aléatoire d’éléments de P . Comme précisé par [Cremonesi et al. \(2010\)](#), il est nécessaire de ne conserver que les triplets dont la note est égale à 5 dans l’ensemble de test. En effet, ces triplets sont ceux correspondant aux films que les utilisateurs ont appréciés. Ils permettent donc d’évaluer les algorithmes de recommandation, leur objectif étant de prédire que ces films sont les plus pertinents.

Après l’entraînement d’un modèle testé sur l’ensemble d’entraînement, le protocole d’évaluation de celui-ci consiste ensuite à, pour chaque triplet $p = (u, i, s)$ dans l’ensemble de test :

1. Créer un ensemble composé de i ainsi que de 1000 autres items pris au hasard parmi les items qui n'ont pas été notés par u .
2. Prédire les notes de ces 1001 items en utilisant le modèle pré-entraîné.
3. Évaluer l'« ordonnancement »¹ (ou *ranking*) ainsi créé avec les métriques décrites dans cette section.
4. Le score final du modèle correspond à la moyenne des scores de chaque ordonnancement.

Comme proposé par [Sedhain et al. \(2015\)](#), il est possible de découper l'ensemble d'entraînement en un ensemble d'entraînement et un ensemble de validation, ce dernier permettant d'effectuer une optimisation d'hyperparamètres. Ils proposent aussi d'effectuer plusieurs découpages aléatoires afin de rendre l'évaluation des modèles plus fiable et non dépendante de l'échantillonnage. Pour les jeux de données composés de retours de pertinence implicite tels que *LastFM*, [Vargas et Castells \(2011\)](#) proposent de déduire la note d'un item entre 1 et 5 (que l'utilisateur aurait donné) à partir de la fréquence d'interaction entre l'utilisateur et l'item. Nous donnons un autre exemple de protocole d'évaluation pour la recommandation d'articles d'actualité en section 7.3 basé sur celui proposé par [Abel et al. \(2011\)](#).

Les métriques dites *offline* appartiennent à trois catégories distinctes : les métriques d'erreur, les métriques de précision et les métriques d'ordonnancement.

Les métriques d'erreur

Les métriques d'erreur (ou *error metrics*) utilisent la prédiction des items sans considérer d'ordre. Lorsqu'il s'agit d'évaluer la prédiction de notes (retours de pertinence explicites), il est courant d'utiliser soit l'erreur absolue moyenne (ou *mean absolute error*, abrégée MAE), soit la racine de l'erreur quadratique moyenne (ou *root mean squared error*, abrégée RMSE).

Notons $M \subset P$ l'ensemble d'entraînement et $T \subset P$ l'ensemble de test de telle sorte que $M \cap T = \emptyset$. Notons r la fonction renvoyant la note prédite d'un item i pour un utilisateur u . La métrique MAE consiste à calculer :

$$\text{MAE}(T) = \frac{1}{|T|} \sum_{(u,i,s) \in T} |r(u,i) - s| \quad (5.2)$$

Et la métrique RMSE consiste à calculer :

$$\text{RMSE}(T) = \sqrt{\frac{1}{|T|} \sum_{(u,i,s) \in T} (r(u,i) - s)^2} \quad (5.3)$$

Les métriques de précision

Lorsque les retours de pertinence sont implicites et que les prédictions consistent à estimer si un item est pertinent (valeur booléenne), il est possible d'utiliser la précision et le rappel. Notons R l'ensemble des tuples (u, i) utilisateurs-items prédits comme pertinents, i.e. que l'item i est prédit pertinent pour l'utilisateur u . Notons T

1. Un « ordonnancement » (ou *ranking*) est une suite d'items ordonnés selon leur score de pertinence attribué par un algorithme de recommandation. L'item le plus haut est prédit comme étant le plus pertinent et l'item le plus bas est prédit comme étant le moins pertinent.

l'ensemble de test comprenant les tuples (u, i) utilisateurs-items réellement pertinents. La précision consiste à calculer :

$$\text{precision}(R, T) = \frac{|R \cap T|}{|R|} \quad (5.4)$$

Et le rappel consiste à calculer :

$$\text{recall}(R, T) = \frac{|R \cap T|}{|T|} \quad (5.5)$$

La F-mesure (ou F-score) est la moyenne harmonique de la précision et du rappel :

$$\text{F-score}(R, T) = 2 \cdot \frac{\text{precision}(R, T) \cdot \text{recall}(R, T)}{\text{precision}(R, T) + \text{recall}(R, T)} \quad (5.6)$$

Si l'algorithme de recommandation renvoie également un ensemble \bar{R} d'items non pertinents, il est alors possible de calculer l'« exactitude » (ou *accuracy*) :

$$\text{accuracy}(R, \bar{R}, T, \bar{T}) = \frac{|R \cap T| + |\bar{R} \cap \bar{T}|}{T + \bar{T}} \quad (5.7)$$

avec \bar{T} l'ensemble des items réellement non pertinents.

Les métriques d'ordonnement

Les métriques de précision ne prennent en considération que les items jugés pertinents par l'algorithme de recommandation. Il s'agit souvent d'appliquer un seuil aux scores retournés par l'algorithme. À l'opposé, les métriques d'ordonnement (ou *ranking metrics*) prennent en compte l'ordre des items calculé à partir des scores de pertinence. Au lieu d'utiliser une information binaire, elles exploitent donc une information continue, permettant ainsi de différencier plus précisément la performance d'un algorithme par rapport à un autre. C'est pourquoi ces métriques sont souvent considérées plus fiables que les métriques de précision puisque prennent en compte les variations de pertinence entre items (Kumar et al., 2017).

Lorsque l'on coupe un ordonnancement à un nombre k d'items (les plus pertinents), il est possible d'utiliser les métriques de précision et de rappel présentées précédemment. Elles sont alors appelées précision partielle et rappel partiel, et sont dénotées *precision@k* (abrégée *p@k*) et *recall@k* (abrégée *r@k*). Étant donné qu'elles prennent en considération l'ordre des items lors du découpage de l'ordonnement, elles peuvent être considérées comme des métriques d'ordonnement (Kouki et Saïd, 2018).

Notons R l'ensemble ordonné de tous les items « candidats », contenant en conséquence au moins un item réellement pertinent et des items non pertinents. Notons T l'ensemble de tous les items réellement pertinents dans R . La métrique *average precision* (abrégée AP) est la moyenne des précisions partielles pour chaque item réellement pertinent (le k correspondant, pour chaque item, au rang de cet item dans R) :

$$\text{AP}(R, T) = \frac{1}{|T|} \sum_{i \in T} p@{\text{rank}(i, R)}(R, T) \quad (5.8)$$

avec *rank* la fonction renvoyant le rang d'un item i dans R .

La métrique appelée *mean average precision* (abrégée MAP) est la moyenne des *average precision* pour tous les ordonnancements générés par l'algorithme. Notons R' l'ensemble des ordonnancements générés par un algorithme et T' l'ensemble des items pertinents dans ces ordonnancements. La *mean average precision* consiste à calculer :

$$\text{MAP}(R', T') = \frac{1}{|R'|} \sum_{i=1}^{|R'|} \text{AP}(R'_i, T'_i) \quad (5.9)$$

La métrique *mean reciprocal rank* (abrégée MRR) est la moyenne, pour chaque ordonnancement, de l'inverse du rang de l'item le plus pertinent. Elle donne donc un score indiquant à quel point les premiers items pertinents des ordonnancements sont proches de la première position. Elle consiste par conséquent à calculer :

$$\text{MRR}(R', T') = \sum_{i=1}^{|R'|} \frac{1}{\text{minrank}(R'_i, T'_i)} \quad (5.10)$$

avec *minrank* la fonction renvoyant la position du premier item pertinent d'un ordonnancement.

La métrique nDCG ([Järvelin et Kekäläinen, 2002](#)) est la plus utilisée des métriques d'ordonnancement. Cette métrique donne un score de précision d'ordonnancement avec une pondération décroissante en fonction de la position des items. Notons rel un vecteur de pertinence graduée correspondant à un vecteur de valeurs booléennes tel que $|rel| = k$ et $rel_i \in \{0, 1\}, i \in \{1, \dots, k\}$. La valeur rel_i indique si le document correspondant dans l'ordonnancement R est pertinent ($rel_i = 1$) ou non ($rel_i = 0$). La fonction nDCG donne un score qui prend un vecteur de pertinence graduée rel :

$$\text{nDCG}(rel) = \frac{\text{DCG}(rel)}{\text{iDCG}(rel)} \quad (5.11)$$

Le nDCG inclut le score DCG :

$$\text{DCG}(rel) = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i + 1)} \quad (5.12)$$

et est normalisé par le score DCG idéal, équivalent à un DCG calculé sur un ordonnancement idéal :

$$\text{iDCG}(rel) = \sum_{i=1}^{\text{sum}(rel)} \frac{1}{\log_2(i + 1)} \quad (5.13)$$

Le score nDCG final d'un algorithme est la moyenne des scores nDCG de tous les ordonnancements générés par l'algorithme, donc appliqué sur l'ensemble de tous les vecteurs de pertinence graduée issus de R' et T' .

Il existe d'autres métriques d'ordonnancement détaillées par [Kouki et Said \(2018\)](#) telles que HL et NDPM. Il existe aussi des métriques issues du domaine de l'apprentissage automatique telles que ROC et AUC ([Bamber, 1975](#)). Cependant, dans cette thèse, nous n'utiliserons que les métriques détaillées dans cette section.

5.3.2 Évaluation qualitative

Différents travaux ont montré que la précision des recommandations n'est pas toujours corrélée à la satisfaction des utilisateurs (Ziegler et al., 2005). Par exemple, Garcin et al. (2014) ont montré que leur meilleur modèle évalué en conditions *offline* consistant à recommander les items les plus populaires n'est pas performant lorsque testé en conditions *online*. De même, Said et al. (2013) ont montré que deux algorithmes de recommandation totalement différents en termes de performances *offline* et d'items recommandés peuvent être jugés comme d'utilité similaire en conditions *online*. Ainsi, il est parfois nécessaire de recourir à des techniques telles que la diversification afin d'améliorer la satisfaction des utilisateurs (McNee et al., 2006). En conditions *offline*, néanmoins, ces techniques peuvent conduire à une diminution de la précision des algorithmes de recommandation (Vargas et al., 2014) mais sont nécessaires lorsqu'il s'agit d'évaluer la qualité des recommandations et leur utilité pour les utilisateurs finaux. Les principaux axes d'amélioration étudiés dans la littérature du domaine sont la « diversité », la « nouveauté » et la « sérendipité ». Ces axes d'amélioration peuvent être mesurés par des métriques dédiées, notamment en conditions *offline* afin de ne pas se restreindre à mesurer la précision des recommandations. Ces métriques, que nous appellerons les « métriques qualitatives », sont opposées aux « métriques de pertinence » que nous avons détaillées en section 5.3.1. Elles sont appelées, dans la littérature du domaine, les *beyond-accuracy metrics* (Kaminskas et Bridge, 2016), *non-accuracy metrics* (Ziegler et al., 2005) ou encore *discovery utilities* (Zhao et Lee, 2016).

La diversité est la propriété d'une liste de recommandations à être composée d'items différents d'un point de vue de leur contenu. La nouveauté, quant à elle, est la propriété d'une liste à contenir des items que l'utilisateur percevra comme nouveaux, c'est-à-dire différents de ce qu'il a déjà consommé. La sérendipité est la propriété d'une liste à contenir des items à la fois pertinents, nouveaux, et surprenants (ou « inattendus »).

Différentes études ont montré l'intérêt des critères qualitatifs. Par exemple, via une « étude utilisateur », Pu et al. (2011) ont interrogé plusieurs volontaires sur l'utilité d'un système de recommandation et ont trouvé une corrélation avec la nouveauté. Ziegler et al. (2005) ont montré que la diversification de listes de recommandations de livres permet d'augmenter la satisfaction des utilisateurs. Candillier et al. (2011) parviennent aux mêmes conclusions sur la recommandation d'articles de blog. Maksai et al. (2015) montrent que la sérendipité permet une augmentation du nombre de clics dans la recommandation d'articles d'actualité.

Dans cette thèse, nous proposons l'idée que le manque de diversité et de nouveauté des algorithmes de recommandation, notamment ceux basés sur le contenu, est la conséquence des habitudes de consommation des utilisateurs. Les utilisateurs n'interagissent avec les plateformes (intégrant un algorithme de recommandation ou non) que dans un temps limité, i.e. ils ne basent par leur choix de consommation sur un parcours exhaustif du catalogue d'items proposé. Leur consommation tend à se restreindre à leurs intérêts déjà établis, lorsque la recherche de nouveaux items demanderait un effort et nécessiterait une « compréhension » (e.g. lecture d'un article sur un sujet inconnu) ou une « adaptation » (e.g. découverte d'un nouveau genre musical).

Nous pensons que les utilisateurs ne sont pas en mesure d'estimer, par eux-mêmes, les bénéfices que pourrait leur apporter la recherche de nouveaux items.

Leur consommation est donc guidée par la « loi du moindre effort » (ou « ignorance rationnelle »), un concept introduit par [Downs \(1957\)](#). L'« ignorance rationnelle » est un comportement humain consistant à s'abstenir d'acquérir des connaissances sur un sujet en particulier lorsque le coût de la « recherche » ou de l'« apprentissage » dépasse les bénéfices estimés que la connaissance apporterait. Les utilisateurs ne savent pas si une recherche aboutira effectivement à la découverte de nouveaux intérêts qui pourront les satisfaire au moins autant que leurs intérêts déjà établis. Cette ignorance les conduit à préférer se restreindre à ce qu'ils connaissent, plutôt que de fournir un effort de recherche potentiellement infructueux.

En conséquence, l'historique de consommation d'un utilisateur ne correspond qu'à un sous-ensemble de ses intérêts existants et potentiels :

Ses intérêts existants Tout d'abord, le fait que l'utilisateur ne dispose que d'un temps limité implique qu'il ne consommera pas nécessairement des items couvrant tous ses centres d'intérêts existants. Par exemple, prenons un utilisateur ayant comme préférences musicales le genre « pop » et le genre « rock ». Sur une plateforme de *streaming* de musique donnée, cet utilisateur n'écoute que des morceaux de musique de genre « pop ». Il n'écoute pas de morceaux de genre « rock » sur cette plateforme mais dans d'autres contextes (e.g. sur une plateforme concurrente, par l'intermédiaire de supports physiques). Alors que ses intérêts correspondent à deux genres musicaux différents, la plateforme de *streaming* de musiques observe la consommation d'uniquement l'un des deux.

Ses intérêts potentiels Ensuite, la loi du moindre effort implique que ses habitudes de consommation auront tendance à limiter la découverte d'intérêts nouveaux. Reprenons l'utilisateur de l'exemple précédent. Imaginons que, s'il s'était intéressé au genre musical « blues », il identifierait celui-ci comme faisant partie de ses préférences musicales. Cet utilisateur pourra estimer que l'écoute de morceaux nouveaux, appartenant à un genre musical nouveau, sera dans la plupart des cas non satisfaisante. Ainsi, il ne consacra pas de temps à écouter de genres musicaux nouveaux, ce qui exclura sa découverte du genre « blues ».

L'historique d'un utilisateur n'est donc jamais aussi diversifié qu'il pourrait l'être. Or, les systèmes de recommandation se basent sur les items historiques, et recommandent, en conséquence, uniquement des items similaires à ceux appartenant à ce sous-ensemble restreint d'intérêts.

Lorsque l'on s'intéresse à l'analyse qualitative des systèmes de recommandation, il est nécessaire de différencier deux processus distincts :

L'évaluation qualitative Processus consistant à évaluer le résultat de l'exécution d'un algorithme de recommandation, i.e. la liste de recommandations qu'il génère. Pour cela, il s'agit d'employer les différentes métriques que nous détaillons dans cette section. Il est plus courant d'évaluer un critère qualitatif dans le cadre d'une expérimentation *offline*. En effet, le critère optimisé en conditions *online* est la satisfaction de l'utilisateur.

L'amélioration qualitative Processus consistant à améliorer un algorithme ou une liste de recommandations. Il s'agit généralement d'optimiser un critère ayant un impact sur les métriques qualitatives du premier point. Ce processus peut conduire à une baisse de précision des recommandations dans le cadre

d'une évaluation *offline*, mais à une hausse de la satisfaction des utilisateurs lorsque les algorithmes sont évalués en conditions *online* (Ziegler et al., 2005; McNee et al., 2006). Cette hausse de la satisfaction n'est néanmoins pas garantie, et seule une « étude utilisateur » (ou *user study*) peut garantir qu'une amélioration qualitative *offline* ait un impact sur la satisfaction des utilisateurs en *online*.

Dans cette sous-section, nous commençons par expliquer le phénomène de bulle de filtres lié à la diversité, la nouveauté et la sérendipité. Puis, nous décrivons ces trois métriques qualitatives en donnant à la fois un aperçu bibliographique de leur « évaluation » ainsi que de leur « amélioration ».

Le phénomène de bulle de filtres

La diversité et la nouveauté de contenu sont associées au phénomène de bulle de filtres (ou *filter bubble*) décrit par Pariser (2011). Ce phénomène est l'isolement des utilisateurs lorsqu'ils consomment de l'information filtrée et personnalisée par des systèmes (e.g. moteurs de recherche, systèmes de recommandation, fil d'actualité sur les réseaux sociaux). De plus, comme dans une chambre d'écho², ce phénomène tendrait à s'auto-alimenter puisque, par exemple, plus un individu lira de recommandations sur un sujet en particulier, plus le système de recommandation considérera ce sujet comme pertinent pour les recommandations suivantes, restreignant de plus en plus les lectures de l'individu à ce seul sujet. Ce filtrage est la conséquence de l'objectif des systèmes qui consiste à proposer aux utilisateurs du contenu satisfaisant et facile à consommer au regard de leurs propres intérêts et opinions. Selon Pariser (2011), l'utilisateur est alors très rarement confronté à des opinions opposées, ce qui a pour conséquence de limiter son libre arbitre. Il consulte uniquement du contenu qui renforce son propre point de vue sans remise en cause par des opinions divergentes. Le phénomène de bulle de filtres est lié au biais cognitif de confirmation correspondant à une tendance des individus à ne chercher, retenir et diffuser que des arguments et opinions confirmant leurs croyances préconçues. Si une opinion est en accord avec les croyances d'un individu, alors celui-ci pourra plus facilement l'adopter tandis que la considération d'une opinion divergente nécessitera de résoudre des incohérences.

Selon Pariser (2011), les bulles de filtres ne sont pas des phénomènes souhaitables puisque nuisent à la démocratie (Helberger, 2019). Elles peuvent aussi engendrer une désinformation par la consultation de fausses nouvelles (Spohr, 2017). Dans la lecture de l'actualité, l'utilisation de médias personnalisés a montré être corrélée à des phénomènes tels la « ségrégation idéologique » (Flaxman et al., 2016). Cependant, l'impact de la personnalisation semblerait être limité (Bakshy et al., 2015), et l'« isolement » être la conséquence des choix de lectures propres des utilisateurs (Flaxman et al., 2016). Certaines études soulignent aussi que, aujourd'hui, les utilisateurs, par les réseaux sociaux, auraient une plus large diversité de lecture (Barberá, 2014), et donc auraient accès à des points de vue opposés. Ces études consistent en l'analyse du phénomène de bulle de filtres et de ses conséquences (e.g. polarisation politique), mais aussi en l'analyse de la diversité de contenu proposé par les moteurs de recherche et

2. Une chambre d'écho est une métaphore utilisée dans le domaine des médias et de la communication illustrant le fait qu'une information, une idée ou une croyance est amplifiée et protégée de toute réfutation par la communication et la répétition de celle-ci dans un système fermé (e.g. un journal).

par les réseaux sociaux, celle-ci étant effectivement présente. En effet, comme nous l’avons décrit dans les sections précédentes, la diversification joue un rôle dans la satisfaction des utilisateurs, il n’est donc pas surprenant qu’un tel critère soit pris en compte dans ce type de système. À noter qu’aucune étude ne propose d’étudier l’impact de la variation de la diversification sur le phénomène de bulle de filtres. Pour toutes ces raisons, la vraisemblance et l’impact potentiel du phénomène de bulle de filtres ne font pas consensus dans la communauté scientifique.

Dans cette thèse, nous soutenons que, malgré ce non-consensus, il est évident que l’absence totale de diversification conduirait à la sur-spécialisation du contenu proposé aux utilisateurs, et donc à des phénomènes potentiellement néfastes pour la société comme décrit par [Pariser \(2011\)](#) (e.g. la polarisation politique). Il semblerait que sa prise en compte soit, dans tous les cas, pertinente, que ce soit dans un objectif de satisfaction de l’utilisateur que d’un point de vue éthique lorsqu’il s’agit de limiter le phénomène de bulle de filtres.

La diversité

La diversification est une notion qui est apparue pour la première fois dans le domaine de l’économie en 1952. [Markowitz \(1952\)](#) proposa la méthode de la « sélection portfolio » qui consistait à maximiser les retours d’investissement en minimisant les risques par la diversification. Dans le domaine de la recherche d’information, la diversification fut adoptée pour la première fois par [Carbonell et Goldstein \(1998\)](#). À l’origine, les moteurs de recherche avaient pour objectif de maximiser le nombre d’items pertinents, ce qui avait pour conséquence de rendre ceux-ci trop similaires entre eux. L’introduction de la diversité permettait alors aux moteurs de recherche de maximiser leurs chances d’avoir au moins un item pertinent pour l’utilisateur. En effet, la diversification de la liste d’items renvoyée par un moteur de recherche permettait de pallier l’ambiguïté d’une requête. Si l’emploi d’un terme dans une requête fait référence à deux entités distinctes, il est souvent difficile de prédire avec certitude sur quelle entité l’utilisateur veut trouver des informations. Par exemple, le nom « Renault » peut autant désigner l’entreprise que le chanteur. Diversifier la liste d’items permet de couvrir le plus d’entités possible et ainsi maximiser les chances de renvoyer à l’utilisateur au moins un item pertinent.

Mesurer la diversité d’une liste de recommandations consiste, dans la majorité des cas, à calculer la distance moyenne entre les items constituant la liste ([Smyth et McClave, 2001](#); [Ziegler et al., 2005](#); [Vargas et Castells, 2011](#)) :

$$diversity(R) = \frac{\sum_{i=1}^{|R|} \sum_{j=i+1}^{|R|} 1 - sim(R_i, R_j)}{\frac{|R|^2 - |R|}{2}} \quad (5.14)$$

avec R l’ensemble des items. Plus précisément, l’équation 5.14 calcule la distance moyenne entre toutes les combinaisons des items sans doublons. Mesurer la diversité d’un algorithme consiste, en conséquence, à calculer la diversité moyenne des listes qu’il génère. Il existe d’autres métriques incluant la diversité comme celle proposée par [Clarke et al. \(2008\)](#), appelée α -nDCG, combinant pertinence et diversité.

Dans la littérature, les différences d’utilisations de la diversité se trouvent essentiellement dans le choix de la fonction de similarité qui peut par exemple prendre la forme d’une mesure taxinomique ([Ziegler et al., 2005](#)) ou d’une similarité cosinus sur le vocabulaire des items ([Ekstrand et al., 2014](#)). Une liste de recommandations

Algorithm 4 Algorithme de ré-ordonnement

```
1: procedure GREEDY-RERANKING( $R$  : set,  $n$  : integer)
2:    $R' \leftarrow \emptyset$ 
3:   while  $|R'| < n$  do
4:      $i = \operatorname{argmin}_{i \in R}(\text{r-weight}(i, R'))$ 
5:      $R' = R' \cup \{i\}$ 
6:      $R = R \setminus \{i\}$ 
7:   return  $R'$ 
8: end procedure
```

qualifiée de « diverse » ne le sera pas nécessairement par les utilisateurs. En effet, la diversité dépendra surtout de leur perception propre (Vargas et al., 2014). C’est pourquoi le choix de la fonction de similarité et de la représentation des items est important.

Les métriques de diversité permettent d’évaluer à quel point une liste de recommandations est diverse. Cependant, au-delà de cette simple « évaluation », différents travaux consistent en l’implémentation de méthodes visant à effectuer une diversification des recommandations, correspondant à ce que nous avons appelé l’« amélioration qualitative ». Ces techniques de diversification consistent généralement soit en un ré-ordonnement (ou *reranking*) d’une liste de recommandations, soit en la modification de l’algorithme ayant généré la liste de recommandations. Le ré-ordonnement a pour avantage de ne pas nécessiter la modification de l’algorithme et consiste en un « post-filtrage » des listes de recommandations. Il permet aussi de contrôler le taux de diversification. Le principe est de sélectionner itérativement les items d’une liste de recommandations maximisant la pertinence et la diversité avec ceux déjà sélectionnés. De cette manière, un item proche de la première position (i.e. considéré comme parmi les plus pertinents par l’algorithme) est prioritaire. S’il est sélectionné, les items similaires seront désavantagés dans les itérations suivantes. La liste de recommandations résultante de ce processus de sélection est réduite à une taille inférieure à la liste d’origine puisqu’elle se compose uniquement d’un sous-ensemble des items, ceux maximisant à la fois la pertinence et la diversité.

Cette sélection itérative est généralement effectuée par un algorithme glouton dont le pseudo-code est donné dans l’algorithme 4. Le paramètre R est la liste de recommandations initiale et doit contenir au moins 3 items. Le paramètre n indique le nombre d’items que la liste diversifiée résultante doit contenir. Il doit être inférieur à la taille de R .

Cet algorithme utilise une fonction de pondération dénoté *r-weight* (pour *reranking weight*). Carbonell et Goldstein (1998) sont les premiers à avoir utilisé cet algorithme pour la recommandation et proposent la pondération *Maximal Marginal Relevance* (abrégée MRR) qui consiste à pondérer les items par leur pertinence et l’inverse de leur similarité maximale avec les items déjà sélectionnés. Des travaux plus récents utilisent la moyenne des similarités (Smyth et McClave, 2001; Ziegler et al., 2005; Kelly et Bridge, 2006) :

$$\text{r-weight}(i, R) = \alpha \cdot \text{rel}(i) + (1 - \alpha) \cdot \frac{1}{|R|} \sum_{j \in R} (1 - \text{sim}(i, j)) \quad (5.15)$$

avec *rel* la fonction renvoyant le score de pertinence de l’item donné en paramètre et

sim une fonction de similarité choisie à l’avance.

Par exemple, [Ziegler et al. \(2005\)](#) utilisent la diversification dans la recommandation de livres. La pertinence d’un livre vis-à-vis d’un utilisateur est donnée par un algorithme basé sur le filtrage collaboratif et la fonction de similarité utilisée dans *r-weight* est une fonction basée sur le genre littéraire. Les auteurs montrent que la diversification de leur algorithme basé sur le filtrage collaboratif item augmente la satisfaction des utilisateurs. D’autres méthodes de ré-ordonnement ont été utilisées dans la littérature. [Kaminskas et Bridge \(2016\)](#) proposent une synthèse de ces méthodes. Les auteurs donnent aussi un aperçu des algorithmes diversifiant directement les listes de recommandations sans effectuer de ré-ordonnement par post-filtrage. [Shi et al. \(2012\)](#) utilisent par exemple la « sélection portfolio » associée au filtrage collaboratif pour effectuer une diversification. Comme détaillé en section 5.2.4, certains travaux de recherche visent à optimiser à la fois la diversité et la précision des recommandations par l’hybridation d’algorithmes ([Ribeiro et al., 2012](#)).

La nouveauté

La nouveauté d’un item dans une liste de recommandations est soit :

Absolute L’utilisateur n’a pas encore consommé (e.g. cliqué, visionné) l’item ([Baeza-Yates et al., 1999](#)) ou ne l’a pas consommé depuis longtemps ([Kapoor et al., 2015](#));

Relative L’item est distant (sachant une fonction de similarité donnée) des items que l’utilisateur a déjà consommés ([Zhang et al., 2002](#); [Saranya et Sadasivam, 2017](#)).

Dans le cas « absolu », la nouveauté consiste à compter le nombre d’items que l’utilisateur n’a pas consommés dans la liste. Il est cependant parfois difficile d’identifier ces items, notamment lorsque les seules données d’interaction enregistrées sont des « notes » (ou *ratings*). En effet, les utilisateurs ne notent pas nécessairement tous les items qu’ils ont consommés. Lorsque des retours de pertinence implicites sont aussi enregistrés (e.g. clics, lecture), l’identification des items connus par l’utilisateur est facilitée puisque les données enregistrées sont plus exhaustives.

En revanche, si les données enregistrées ne sont pas exhaustives, il est alors nécessaire de trouver une approximation de la nouveauté absolue de l’item, c’est-à-dire si l’item a déjà été consommé ou non par l’utilisateur. Pour cela, différents auteurs proposent d’utiliser la popularité de l’item en supposant que plus l’item est populaire, plus grande est la probabilité que l’utilisateur ait déjà consommé celui-ci ([Zhou et al., 2010](#); [Vargas et Castells, 2011](#); [Mendoza et Torres, 2019](#)). La popularité d’un item peut être donnée par le nombre d’interactions observées avec les utilisateurs du système ([Zhou et al., 2010](#); [Vargas et Castells, 2011](#)) ou avec des données externes au système ([Oh et al., 2011](#)).

La nouveauté absolue d’un item correspond donc au nombre d’utilisateurs qui ont interagi avec l’item sur le nombre total d’utilisateurs :

$$\text{abs-novelty}(i) = 1 - \frac{|u \in U, (u, i) \in L|}{|U|} \quad (5.16)$$

avec L l’ensemble des tuples utilisateur-item correspondant aux interactions observées. [Zhou et al. \(2010\)](#); [Vargas et Castells \(2011\)](#) proposent d’avantager les items rares

en introduisant le logarithme à l'équation 5.16 :

$$\text{abs-novelty}(i) = -\log_2\left(\frac{|u \in U, (u, i) \in L|}{|U|}\right) \quad (5.17)$$

La nouveauté absolue d'une liste consiste ensuite à effectuer la moyenne des nouveautés absolues de chaque item.

Dans le cas de la « nouveauté relative », tout comme pour la diversité, il s'agit de choisir une fonction de similarité. Cependant, la moyenne des similarités est calculée entre la liste des items historiques et la liste des items recommandés :

$$\text{rel-novelty}(R, H) = \frac{\sum_{i=1}^{|R|} \sum_{j=1}^{|H|} 1 - \text{sim}(R_i, H_j)}{|R| \cdot |H|} \quad (5.18)$$

avec H l'ensemble des items historiques et R les items recommandés. La nouveauté relative peut aussi correspondre à la moyenne des distances minimales entre items historiques et chaque item recommandé comme le proposent [Nakatsuji et al. \(2010\)](#) :

$$\text{rel-novelty}(R, H) = \frac{\sum_{i=1}^{|R|} 1 - \max_{\forall h \in H} \text{sim}(R_i, h)}{|R|} \quad (5.19)$$

Nous donnons une justification à ce choix avec un exemple en section 8.5.7.

L'amélioration de la nouveauté relative peut consister à promouvoir les items non-populaires, dits *long-tail* comme le proposent [Park et Tuzhilin \(2008\)](#). Pour l'amélioration de la nouveauté relative, il est possible d'utiliser le ré-ordonnement des items. Mais, à la différence de la diversité, les items historiques sont utilisés ([Kotkov et al., 2016](#)).

À noter qu'il est plus pertinent d'utiliser la nouveauté relative lorsque l'utilisateur n'a pas de consommation « redondante » et qu'il préférera consommer des items suffisamment distants de son historique comme dans la recommandation d'articles d'actualité. En effet, la lecture de deux articles sur le même sujet d'actualité n'a pas d'intérêt pour l'utilisateur. En revanche, dans la recommandation de musique par exemple, il est souvent pertinent de recommander des morceaux similaires mais « nouveaux » dans le sens absolu (i.e. l'utilisateur ne les a pas encore écoutés). En effet, un morceau nouveau, d'un artiste que l'utilisateur connaît, sera similaire à son historique au sens relatif mais restera nouveau au sens absolu. Il pourra aussi être considéré nouveau si l'utilisateur ne l'a pas écouté depuis un certain temps ([Kapoor et al., 2015](#)).

Malgré les bénéfices que peut apporter l'amélioration de la nouveauté des recommandations, il est courant qu'elle implique une diminution de la précision étant donné que les items « nouveaux » sont plus probablement non pertinents car différents de ce que l'utilisateur a déjà consommé. De même, les items non-populaires seront plus probablement de moindre qualité et donc non pertinents ([Kotkov et al., 2016](#)). Il s'agit donc de trouver un compromis entre nouveauté et précision comme le proposent [Zhang et al. \(2012\)](#).

La sérendipité

Un item recommandé de manière « sérendipe » est un item intéressant (ou pertinent), nouveau et inattendu (ou surprenant) que l'utilisateur n'aurait pas découvert

de lui-même (Herlocker et al., 2004). Tout comme la nouveauté, la sérendipité prend en compte les profils des utilisateurs (e.g. leur historique), lorsque la diversité ne prend en compte qu’une liste d’items recommandée. En revanche, contrairement à la nouveauté, la sérendipité inclut également la notion de pertinence.

Il est difficile de mesurer la sérendipité puisqu’elle se compose d’une notion subjective : la surprise. De même, il est difficile d’estimer si la recommandation d’un item est surprenante sans interroger directement l’utilisateur. Différents auteurs proposent donc des heuristiques visant à donner une estimation de la sérendipité. Par exemple, Murakami et al. (2008) proposent de considérer un item comme sérendipe lorsqu’il n’a pas été recommandé par un « algorithme primitif » (ou « primitive »), l’hypothèse étant qu’un algorithme primitif ne recommande que des items dont la pertinence est évidente alors qu’une recommandation sérendipe est une recommandation dont la pertinence est difficile à prédire. Ainsi, suivant l’idée de Murakami et al. (2008), la sérendipité d’une liste de recommandations R peut être définie par :

$$\text{serendipity}(R, P, T) = \frac{|R \cap (T \setminus P)|}{|T \setminus P|} \text{ such that } T \setminus P \neq \emptyset \quad (5.20)$$

avec R la liste des items recommandés, T l’ensemble des items pertinents et P l’ensemble des items recommandés par la primitive considérée. Kotkov et al. (2016) proposent un aperçu d’autres métriques basées sur des primitives.

Cette formulation de la sérendipité a pour désavantage d’être sensible au choix de la primitive. Celle-ci doit être suffisamment simple en termes algorithmiques (e.g. ordonnancement d’items candidats par similarité) et de représentation des items (e.g. représentation sac de mots) mais aussi suffisamment performante en termes de prédiction des items pertinents. Pour pallier ce problème, certains auteurs emploient une autre heuristique en ne se basant que sur la notion de nouveauté (Nakatsuji et al., 2010; Kaminskis et Bridge, 2014). La formulation est équivalente à la nouveauté relative en considérant, pour chaque item recommandé, l’item historique de distance minimale :

$$\text{serendipity}(R, H) = \frac{\sum_{i=1}^{|R|} 1 - \max_{\forall h \in H} \text{sim}(R_i, h)}{|R|} \quad (5.21)$$

Tout comme pour les deux métriques qualitatives précédentes, il est possible d’utiliser un algorithme de ré-ordonnancement glouton pour l’amélioration de la sérendipité d’une liste de recommandations (Adamopoulos et Tuzhilin, 2014; Zhang et al., 2012). Des optimisations de la sérendipité par de nouveaux algorithmes ou la modification d’algorithmes existants ont aussi été proposées. Par exemple, Said et al. (2013) proposent de modifier la méthode des k plus proches voisins en recommandant à un utilisateur les items que les utilisateurs différents n’ont pas aimés.

5.3.3 Évaluation *online*

La première méthode permettant une évaluation *online* d’un algorithme de recommandation est de proposer à des utilisateurs « volontaires » de répondre à un questionnaire de satisfaction (Knijnenburg et al., 2012) ou de noter des propositions (Pu et al., 2011). Il s’agit donc de collecter des retours de pertinence explicites. Par

exemple, [Pu et al. \(2011\)](#) proposent de donner une note de 1 à 5 (5 correspondant à une totale approbation) aux propositions suivantes :

This interface gave me some really good recommendations.
This interface is competent to help me effectively find products I really like.

Cependant, ce type de questionnaire est sujet à la subjectivité des utilisateurs et ne peut impliquer qu'un nombre limité de volontaires ([Kouki et Said, 2018](#)).

Le test A/B permet de prendre en compte des retours de pertinence implicites. Ce test consiste généralement à employer différents systèmes de recommandation en les sélectionnant de manière aléatoire à chaque requête des utilisateurs. Puis, les différents systèmes sont évalués et comparés en fonction des interactions observées avec les utilisateurs. Le « taux de clics » (ou *click-through rate*, abrégé CTR) est une métrique qui exploite les retours de pertinence implicites des utilisateurs. Il correspond au nombre de clics sur le nombre de recommandations affichées (ou « impressions ») :

$$\text{CTR}(A) = \frac{\text{Nombre de clics que l'algorithme } A \text{ a généré}}{\text{Nombre de recommandations de } A \text{ affichées aux utilisateurs}} \quad (5.22)$$

Un utilisateur aura, à un instant donné, une liste de recommandations générée par un unique algorithme. Les clics de cet utilisateur, à cet instant, seront comptabilisés dans le CTR final attribué à l'algorithme. Ainsi, il est possible de comparer la performance des algorithmes en comparant leur CTR. Par exemple, si l'algorithme A a un CTR de $\frac{50}{1000}$ (i.e. 50 clics sur 1000 recommandations affichées) et l'algorithme B un CTR de $\frac{60}{2000}$ (i.e. 60 clics sur 2000 recommandations affichées), l'algorithme A sera considéré comme plus performant car aura un CTR de 0.05 contre 0.03 pour l'algorithme B .

[Beel et Langer \(2015\)](#) proposent aussi de considérer la moyenne des CTR par liste de recommandations :

$$\text{CTR}_{\text{Set}}(R') = \frac{\sum_{R \in R'} \sum_{i=1}^{|R|} \frac{\text{clicked}(R_i)}{|R|}}{|R'|} \quad (5.23)$$

avec R' l'ensemble des listes de recommandations générées par l'algorithme considéré et *clicked* la fonction renvoyant 1 si l'item donné en paramètre a été cliqué, 0 sinon. D'autres métriques basées sur le fonctionnement de CTR sont proposées dans la littérature. Elles sont généralement adaptées à la plateforme sur laquelle est exécuté l'algorithme de recommandation. Par exemple, [Beel et Langer \(2015\)](#) proposent de calculer le taux de téléchargement des items, ou encore le taux d'ajouts en favoris des items. D'autres mesures plus complexes peuvent aussi être employées comme le taux d'attrition (ou *churn rate*) ([Kouki et Said, 2018](#)) indiquant la fidélité des utilisateurs (e.g. en comptant les désabonnements) ou encore le taux de rebond³ (ou *bounce rate*) qui correspond au nombre de fois qu'un utilisateur n'a effectué qu'une seule action avec le système.

Cependant, tous les utilisateurs n'ont pas les mêmes habitudes de consommation. Certains pourront cliquer plus souvent, sur une plus grande proportion des recommandations qui leur sont affichées, et d'autres moins souvent. Le contexte a aussi

3. Le taux de rebond (ou *bounce rate*) est la proportion d'utilisateurs accédant à un site web (ou, plus généralement, à une application) et le quittant d'emblée (i.e. sans consulter d'autres pages).

un impact sur la consommation (i.e. moment de la journée, de la semaine). Il est donc important de collecter un grand nombre de clics sur une période longue avec plusieurs utilisateurs afin que les scores CTR des systèmes puissent être « comparables » (Hofmann, 2015). Pour pallier ce problème, la méthode de « comparaison par entrelacement » (ou *interleaved comparison*) est souvent employée lorsqu'il s'agit d'évaluer plusieurs systèmes de recommandation ou moteurs de recherche. Cette méthode nécessite moins d'interactions avec l'utilisateur puisque plusieurs systèmes sont directement confrontés à chaque interaction (Schuth et al., 2015; Kharitonov et al., 2015). Au chapitre 9, nous décrivons plus en détail les avantages de cette méthode.

Les différentes méthodes d'entrelacement proposées de la littérature consistent à mélanger plusieurs listes générées par les systèmes de recommandation comparés de sorte à effectuer un entrelacement équitable (i.e. qui n'avantage pas l'un des systèmes). Il s'agit également de faire en sorte que l'utilisateur ne puisse pas identifier quel système a recommandé quel item. La méthode la plus utilisée est la méthode *Team Draft* proposée par Radlinski et al. (2008). D'autres méthodes ont, par la suite, été proposées comme l'entrelacement probabiliste (ou *probabilistic interleaving*) (Hofmann et al., 2011), l'entrelacement optimisé (ou *optimized interleaving*) (Radlinski et Craswell, 2013) ou encore des méthodes d'entrelacement permettant la comparaison de plus de deux systèmes (Schuth et al., 2014).

À noter que les méthodes d'apprentissage par renforcement⁴ peuvent permettre d'automatiquement trouver le meilleur algorithme de recommandation parmi ceux comparés. Il est parfois plus favorable, sur une plateforme en ligne, de ne pas effectuer une évaluation précise des algorithmes en jeu mais de chercher les meilleurs algorithmes par « exploration » tout en minimisant l'utilisation des algorithmes les moins performants par « exploitation » (Hofmann, 2015). En effet, l'utilisation de systèmes de recommandation sous-optimaux lorsque des utilisateurs sont actifs peut conduire à l'altération de leur expérience.

5.3.4 Classification des méthodes et métriques d'évaluation

De nombreuses revues de la littérature font une description approfondie de chaque critère, méthodes et métriques d'évaluation dédiés aux systèmes de recommandation (Hofmann, 2015; Kouki et Said, 2018; Shani et Gunawardana, 2011; Kouki, 2014; Gemmis et al., 2009; Bobadilla et al., 2013; Kaminskas et Bridge, 2016; Kunaver et Požrl, 2017; Kotkov et al., 2016). Certains auteurs proposent un visuel de ces concepts sous forme de taxonomie⁵ (Taghavi et al., 2018; Singh et al., 2017; Silveira et al., 2019). Cependant, à notre connaissance, aucune revue de la littérature ne propose une taxonomie complète de ces concepts. Nous proposons, dans cette section, une taxonomie permettant de positionner chaque concept mentionné dans la littérature du domaine.

Silveira et al. (2019) proposent de différencier six concepts : l'utilité (incluant les métriques d'évaluation *offlines* que nous avons présentées en section 5.3.1 ainsi que les mesures statistiques *onlines* telles que CTR présentées en section 5.3.3), la

4. Dans le domaine de l'apprentissage automatique, l'« apprentissage par renforcement » consiste, pour un agent autonome, en l'apprentissage des meilleures actions à effectuer en fonction de récompenses qu'il obtient par expérience.

5. Une taxonomie est une représentation hiérarchique de concepts.

nouveauté, la diversité, l'inattendu (ou surprise), la sérendipité et la couverture. Ils mentionnent d'autres concepts qu'ils ne traitent pas dans leur revue et qui ont été introduits par Ricci et al. (2010) : la confiance, le risque, la robustesse, le respect de la vie privée, l'adaptabilité et la scalabilité. Les auteurs proposent aussi une étude des relations entre concepts. Par exemple, ils expliquent que la sérendipité est un concept composé de l'inattendu, de l'utilité et de la nouveauté comme nous l'avons détaillé en section 5.3.2. Ils différencient l'utilité des autres concepts en expliquant que l'utilité est un critère « nécessaire » à l'utilisateur, les autres étant des critères « désirés ». Ils placent la diversité entre « nécessaire » et « désiré ».

Taghavi et al. (2018) proposent une classification de différents concepts et métriques. Cependant, ils positionnent différents concepts tels que la confiance, la diversité ou encore la précision au même niveau. Singh et al. (2017), Shani et Gunawardana (2011) et Bobadilla et al. (2013) proposent de classer les métriques *offlines* dans trois catégories comme détaillé en section 5.3.1 : les métriques d'erreur, de précision et d'ordonnement. Kouki et Said (2018) proposent de différencier les « méthodes d'évaluation » (i.e. *offline* et *online* avec les mesures statistiques telles que CTR) des « métriques d'évaluation ». Ces dernières forment quatre catégories : les métriques d'erreur, les métriques d'ordonnement, les métriques qualitatives qu'ils appellent *non-accuracy metrics* et les métriques commerciales (ou *business metrics*) telles que la couverture et le taux d'attrition. Kouki (2014) mentionne uniquement, pour l'évaluation *offline*, les métriques d'erreur et d'ordonnement. L'auteur positionne les métriques qualitatives et d'autres telles que la couverture dans une catégorie « autres métriques ». Bobadilla et al. (2013) différencient les métriques d'erreur, de précision, d'ordonnement et de diversité, cette dernière incluant les métriques qualitatives telles que la diversité et la nouveauté. Les auteurs mentionnent aussi d'autres critères tels que la stabilité et la fiabilité.

Dans notre taxonomie représentée sur la figure 5.1, nous proposons de séparer l'évaluation des systèmes de recommandation en trois grandes catégories :

Pertinence et utilité Cette catégorie regroupe quelques méthodes et métriques que nous avons présentées. D'un côté l'évaluation *offline* se sépare en métriques d'erreur, de précision et d'ordonnement comme le proposent par exemple Singh et al. (2017), Shani et Gunawardana (2011) et Bobadilla et al. (2013). D'un autre côté, l'évaluation *online* se sépare en évaluation par retours de pertinence explicites et par retours de pertinence implicites. Dans les retours implicites, nous n'avons pas ajouté les méthodes d'apprentissage par renforcement car elles ne consistent pas en l'évaluation de systèmes de recommandation, mais sont utilisées dans le but d'avoir une estimation en temps réel des algorithmes les plus performants parmi ceux comparés.

Qualité Cette catégorie regroupe les métriques que nous avons détaillées en section 5.3.2. Elles ont pour particularité de mesurer la qualité des listes de recommandations (Karimi et al., 2018) selon différents critères liés aux items candidats et historiques (e.g. leur contenu, leur popularité), et n'utilisent pas uniquement la pertinence mesurée par les métriques *online* ou estimée par les métriques *offline*.

Autre Enfin, cette catégorie regroupe d'autres critères et mesures liés aux besoins des utilisateurs, aux besoins de la plateforme en termes, par exemple, de « revenus » et aux exigences de « calcul ».

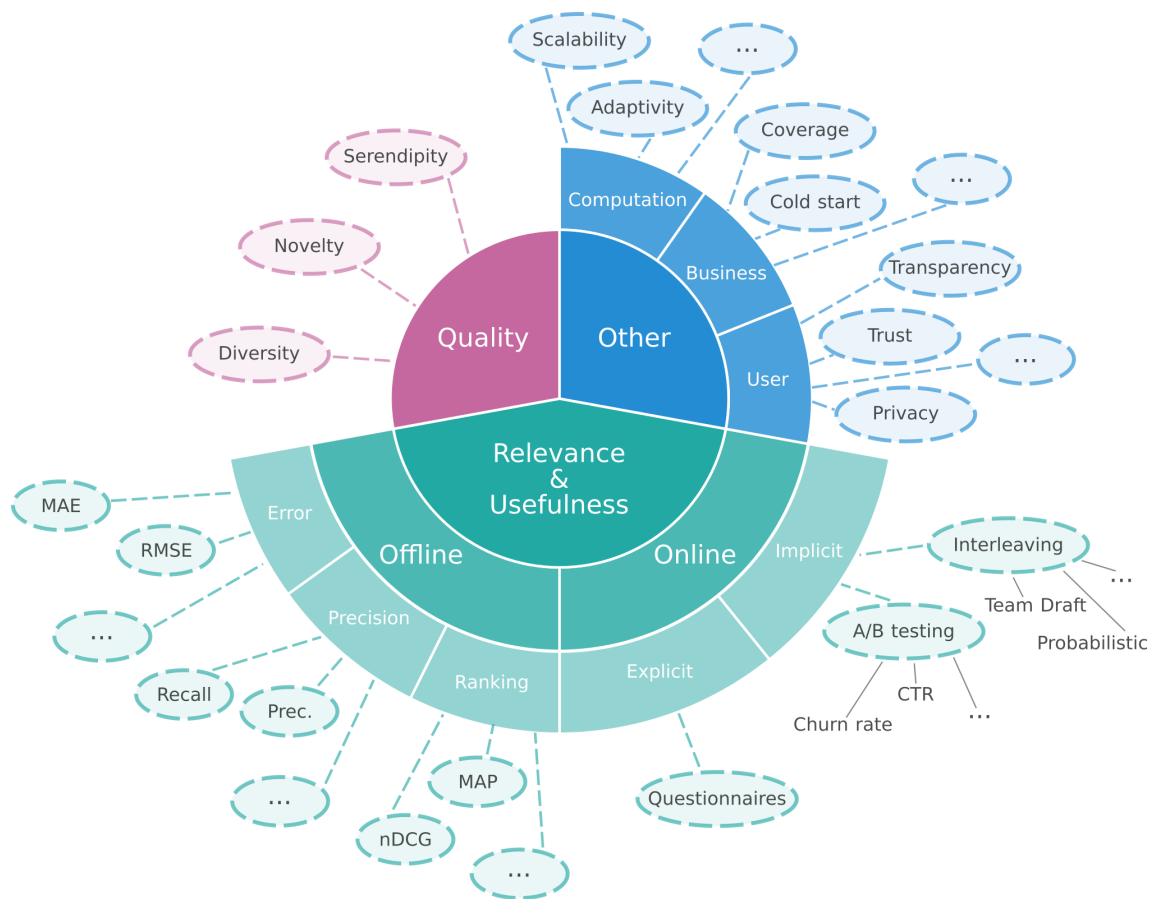


FIGURE 5.1 – Classification des méthodes et métriques pour l'évaluation des systèmes de recommandation

Cette dernière catégorie se compose de différentes notions détaillées par, notamment, [Gemmis et al. \(2009\)](#) et [Taghavi et al. \(2018\)](#). Nous donnons un aperçu de ces concepts :

- La scalabilité (ou *scalability*) est une problématique qui apparaît lorsque, par exemple, le nombre d'utilisateurs sur une plateforme est grand. Il s'agit alors, pour les systèmes de recommandation, d'être capable de modéliser l'ensemble des utilisateurs pour la génération des recommandations, ce qui peut être coûteux en ressources.
- L'adaptabilité (ou *adaptivity*) est la capacité d'un système de recommandation à gérer les changements rapides sur la plateforme (e.g. les intérêts des utilisateurs).
- La couverture (ou *coverage*) est la proportion d'items recommandés aux utilisateurs. Elle correspond à une mesure indiquant à quel point les items ont été recommandés au moins une fois à un utilisateur. Cette mesure est considérée comme une métrique dite *beyond-accuracy* par [Kaminskas et Bridge \(2016\)](#). Cependant, nous la positionnons dans la catégorie « autre », et plus précisément dans la sous-catégorie « *business* » comme considéré par [Kouki et Said \(2018\)](#). En effet, celle-ci ne consiste pas en l'évaluation qualitative des listes de recommandations. Elle est utile lorsqu'il s'agit d'augmenter les revenus d'une plateforme en évitant d'exclure la recommandation des items dits *long-tail*. Cette métrique est aussi liée à la « diversité agrégée » (ou *aggregate diversity*) qui consiste à recommander de façon non redondante les items disponibles sur la plateforme aux utilisateurs ([Adomavicius et Kwon, 2012](#)).
- La transparence (ou *transparency*) correspond à la capacité des systèmes de recommandation à expliquer la cause d'une recommandation aux utilisateurs (e.g. similarité avec des items déjà consommés).
- La confiance (ou *trust*) correspond au degré de fiabilité perçu par les utilisateurs vis-à-vis du système de recommandation (e.g. la capacité du système à recommander les items les plus pertinents, sa capacité à ne pas exclure d'items dont la pertinence est évidente).
- Le respect de la vie privée (ou *privacy*) correspond à la caractéristique du système à ne pas afficher ou distribuer les données liées à un utilisateur. Les plateformes peuvent parfois proposer aux utilisateurs de régler l'affichage (ou distribution) de leurs données.

Cette classification n'est cependant pas exhaustive et omet certains concepts tels que les retours de pertinence utilisés. Par exemple, nous n'avons pas représenté le fait que la comparaison par entrelacement nécessite non seulement l'utilisation d'une méthode telle que *Team Draft* mais aussi une mesure de retour de pertinence (e.g. clic, temps de lecture ou *dwell time*) et une méthode de calcul du classement des algorithmes (e.g. par leur proportion de victoires, par un système d'évaluation « un contre un » tel que le « classement Elo⁶ »). De même, nous n'avons pas inclus les retours de pertinence enregistrés lors de l'utilisation de jeux de données *offlines*. À noter aussi que certains concepts de cette classification tels que le démarrage à froid (ou *cold start*), ou encore la scalabilité (ou *scalability*), peuvent être considérés comme

6. Le « classement Elo » est un système d'évaluation comparatif du niveau de jeu de joueurs participant à des jeux en un contre un. Il est notamment populaire pour son utilisation par différentes fédérations d'échecs.

des « problématiques ». Cependant, la performance des systèmes de recommandation peut être mesurée au regard de ces problématiques. Par exemple, [Zhou et al. \(2011\)](#) séparent les utilisateurs de leur jeu de données en plusieurs ensembles dont un composé d'utilisateurs « nouveaux ». Mesurer la performance de leur algorithme sur cet ensemble (en l'occurrence avec la métrique RMSE) permet ainsi de mesurer la capacité de celui-ci à résoudre le problème du démarrage à froid utilisateur.

5.4 Recommandation d'articles d'actualité

Dans cette section, nous commençons par expliquer dans quelle mesure les domaines d'application de la recommandation peuvent varier. Nous décrivons ensuite en détail les particularités de la recommandation d'articles d'actualité ainsi que les algorithmes et jeux de données proposés dans la littérature. Enfin, nous terminons cette section par décrire les manques identifiés dans l'état de l'art de ce domaine.

5.4.1 Différences entre domaines de recommandation

Dans la recherche sur les systèmes de recommandation, il existe de nombreux domaines d'application (e.g. films, articles, musiques, vidéos, livres, produits) qui ont tous leurs particularités propres. Ces particularités conduisent à différentes problématiques liées à la recommandation des items. Elles peuvent conduire aussi à l'inefficacité de certaines méthodes de recommandation (e.g. filtrage par le contenu, filtrage collaboratif). Dans certains domaines, les items peuvent varier en nombre. Par exemple, le catalogue d'une plateforme de *streaming* de musique (e.g. Deezer, Spotify) sera généralement plus grand que celui d'une plateforme de *streaming* de films (e.g. Netflix, Disney+, Prime Video). Les items pourront être plus nombreux que les utilisateurs, ou parfois moins nombreux (généralement lorsque la plateforme est privée).

La consommation des items pourra être totalement différente entre plateformes. Par exemple, les utilisateurs d'une plateforme de *streaming* de musique pourront avoir une consommation redondante, c'est-à-dire qu'ils écouteront un même morceau plusieurs fois. De plus, ils écouteront plusieurs morceaux en un temps court, sous forme de « listes de lecture » par exemple. Au contraire, les utilisateurs d'une plateforme de *streaming* de films ne visualiseront généralement pas deux fois un même film, et un seul film par jour. Ainsi, dès lors qu'un film a été visionné par un utilisateur, sa recommandation devient non pertinente.

Il pourra exister des items très similaires entre eux (e.g. deux articles d'actualité sur le même évènement, deux morceaux de musique dont l'un a été réédité) ou, au contraire, uniquement des items différents (e.g. films, livres). Les utilisateurs sur une plateforme d'e-commerce pourront naturellement préférer la diversité alors qu'ils préféreront consommer des items similaires sur une plateforme de *streaming* de musique par exemple ([Kapoor et al., 2015](#)). Les besoins de « diversification » varient à la fois entre utilisateurs mais aussi entre domaines de recommandation ([Kaminskas et Bridge, 2016](#)). La recommandation successive d'items de même genre pourra être plus pertinente dans le domaine de la musique que dans le domaine des films. En effet, les utilisateurs d'une plateforme de *streaming* de films pourront préférer passer d'un genre à l'autre lors de leur visionnage ([Shi et al., 2012](#)).

La pertinence d'un item pourra être corrélée à des caractéristiques autres que les intérêts des utilisateurs. Par exemple, les articles d'actualité verront leur pertinence décroître avec le temps puisque les utilisateurs préféreront consommer des informations récentes (Gulla et al., 2016). La pertinence pourra aussi dépendre du contexte tel que la localisation ou les caractéristiques du lieu d'interaction (e.g. domicile, transport en commun). Elle dépendra aussi des connaissances que le système de recommandation est capable de collecter sur l'utilisateur, et à quel point les connaissances sur un utilisateur sont exhaustives. Par exemple, dans le jeu de données *CiteULike*, il est possible que les utilisateurs aient connaissance de beaucoup plus d'items que ceux qu'ils ont ajoutés dans leur liste de favoris. Ils auront pu lire certains des items sans les avoir ajoutés en favoris, et il ne serait pas pertinent de les recommander. De même, dans le jeu de données *MovieLens*, les utilisateurs n'ont pas nécessairement noté tous les films qu'ils ont vus. Au contraire, dans les jeux de données qui consistent en la recommandation de produits, il est probable qu'un utilisateur ne possède pas les items qu'il n'a pas achetés. Autrement dit, contrairement à la recommandation de films et de pages web, si une interaction entre un utilisateur et un produit n'a pas été observée dans le système, alors il est probable que l'utilisateur n'ait réellement pas encore consommé celui-ci, i.e. ne l'a pas encore acheté.

En plus de toutes ces caractéristiques propres aux domaines d'application, il existe aussi des particularités liées aux jeux de données utilisés ou à la plateforme utilisée. Par un exemple, un jeu de données permettant une évaluation *offline* pourra proposer uniquement des interactions entre utilisateurs et items alors que d'autres pourront proposer des informations additionnelles sur les items. Par exemple, Cantador et al. (2011) proposent une extension du jeu de données *MovieLens* (Harper et Konstan, 2015) en ajoutant des informations sur les films issus de *IMDb*⁷. Les retours de pertinence pourront être de nature différente. Par exemple, le temps de visionnage d'un film sera plus long que le temps de lecture d'un morceau de musique. Le fait qu'un utilisateur abandonne le visionnage d'un film pourra être plus significatif d'une recommandation « non pertinente » que le « saut » d'un morceau de musique à un autre, i.e. l'interruption d'un morceau avant la fin afin de passer au suivant.

La nature des retours de pertinence dans le jeu de données que proposent Abel et al. (2011) et celui de Kille et al. (2013) appelé *Plista* sont aussi de nature différente. Ces deux jeux de données sont dédiés à la recommandation d'articles d'actualité. Dans le premier, les retours de pertinence correspondent à des interactions de type « partages » sur Twitter alors que dans le second, les retours de pertinence correspondent à des interactions de type « clics ». Ces interactions correspondent à des actions différentes des utilisateurs, impliquant différents niveaux de pertinence mais aussi l'existence de différents biais d'évaluation. Nous détaillerons plus en profondeur ces différences en section 7.3.1.

Une autre différence entre le jeu de données proposé par Abel et al. (2011) et le jeu de données *Plista* est que Kille et al. (2013) considèrent chaque utilisateur comme étant une « session de navigation »⁸ unique alors qu'un même utilisateur aura pu lire

7. *Internet Movie Database* (abrégée *IMDb*) est une base de données en ligne regroupant des informations dans le domaine du cinéma et de la télévision.

8. Une « session de navigation » désigne la visite d'un ensemble de pages web par l'intermédiaire d'un navigateur web sans interruption pendant un temps donné dépendant de la configuration du serveur.

des articles d'actualité sur plusieurs sessions. Cela implique que différentes sessions de lecture peuvent correspondre à un même utilisateur, mais aucune information ne permet de le détecter. En conséquence, l'évaluation des systèmes de recommandation sur le jeu de données *Plista* est orientée vers la prise en compte d'intérêts courts termes, alors que des intérêts longs termes propres aux utilisateurs peuvent être capturés dans le jeu de données de [Abel et al. \(2011\)](#).

Bien entendu, la nature même des interactions entre les systèmes de recommandation et les utilisateurs modifiera l'implémentation des algorithmes. Par exemple, dans un cadre *online*, il est courant d'utiliser des algorithmes d'apprentissage par renforcement étant donné qu'un choix de l'algorithme de recommandation affectera la consommation de l'utilisateur et qu'il est ainsi possible de modifier les paramètres d'un algorithme en fonction de cette consommation variable ([Zheng et al., 2018](#)). Dans un cadre *offline*, les contraintes de collecte des interactions utilisateurs-items peuvent conduire à l'inefficacité d'une méthode. Il est par exemple exclu d'utiliser le filtrage collaboratif lorsqu'il existe très peu d'interactions communes entre utilisateurs comme dans certains jeux de données de recommandation d'articles d'actualité.

5.4.2 Particularités de la recommandation d'articles d'actualité

Les plateformes intégrant un système de recommandation d'articles d'actualité ont pour objectif de permettre aux lecteurs s'intéressant à l'actualité d'économiser du temps. En effet, les lecteurs font face, chaque jour, à une surcharge d'informations, le catalogue d'articles d'actualité disponibles étant en perpétuelle évolution. Dans une situation idéale, grâce aux algorithmes de recommandation, les lecteurs n'ont plus besoin de chercher d'articles pertinents vis-à-vis de leurs intérêts. Ils n'ont plus besoin non plus de chercher de nouveaux sujets d'intérêt à découvrir, ceux-ci leur étant directement proposés. Sans système de recommandation, les lecteurs consacrent énormément de temps à chercher des articles d'intérêt parmi de nombreux items, ou se contentent d'une sélection parmi un nombre trop limité d'items pour répondre à leurs besoins efficacement ([Özgöbek et al., 2014](#)).

La recommandation d'articles d'actualité est une application directe du domaine des systèmes de recommandation. Cependant, comparée à la recommandation d'autres types de données (e.g. musiques, films, livres, produits), la recommandation d'articles d'actualité présente un certain nombre de particularités qui lui sont propres. Sa principale particularité est le caractère évolutif de la pertinence des items recommandés ([Gulla et al., 2014, 2016](#); [Özgöbek et al., 2014](#)). Les articles d'actualité ont une pertinence intrinsèque qui évolue dans le temps, c'est-à-dire que leur pertinence « générale » (i.e. non spécifique à un utilisateur en particulier) est variable selon le nombre de jours écoulés depuis leur publication. D'après [Gulla et al. \(2016\)](#), les articles ayant été publiés il y a plus de cinq jours ne sont presque plus consultés par les utilisateurs. Cela peut être dû à l'interface des sites d'actualité qui mettent plus en avant les articles d'actualité récents, mais selon [Karimi et al. \(2018\)](#), ce phénomène s'explique avant tout par la volonté des lecteurs à lire les informations les plus récentes sur l'actualité.

Cette particularité conduit au problème de « démarrage à froid item » (ou *item cold-start*). En conséquence, les algorithmes basés sur le filtrage collaboratif ne sont généralement pas en mesure d'effectuer des recommandations pertinentes puisque

les nouveaux articles récemment publiés n’ont pas encore été lus par les utilisateurs (Ilievski et Roy, 2013; Liu et al., 2010; Wang et al., 2017; Sertkan et al., 2019). Or, les algorithmes basés sur le filtrage collaboratif sont souvent considérés comme les plus performants dans le domaine des systèmes de recommandation. Heureusement, les articles d’actualité étant riches d’informations textuelles, il est possible de se baser sur leur contenu pour effectuer des recommandations en comparant les items candidats à l’historique de lecture des utilisateurs (Bai et al., 2017; Abel et al., 2011; Caldarelli et al., 2016).

Les utilisateurs ne sont parfois pas enregistrés sur le site web du journal qu’ils consultent. Leur historique de lecture est donc court puisque correspond à leur session de navigation sur ce site web. Non seulement les algorithmes de recommandation sont continuellement confrontés au problème de démarrage à froid item comme nous l’avons évoqué, mais sont aussi confrontés au démarrage à froid utilisateur étant donné que les historiques de lecteur ne contiennent que très peu d’items. Ainsi, les algorithmes consistent souvent en la prise en compte d’intérêts courts termes et sont basés sur la « session » (Qin et Lu, 2020). Les items sont recommandés en « séquences ». Les algorithmes de recommandations basés sur les sessions modélisent donc des règles en identifiant des dépendances entre les items consultés dans une session (de Souza Pereira Moreira et al., 2018; Zihayat et al., 2019; Jugovac et al., 2018). En effet, les utilisateurs, dans une session, sont plus enclins à chercher des informations sur un même sujet correspondant à leurs intérêts courts termes (Liu et al., 2010; Gulla et al., 2016).

Selon Raza et Ding (2020), la seconde plus grande particularité de la recommandation d’articles d’actualité, aussi considérée comme une difficulté par les auteurs, est que les lecteurs ont des intérêts fluctuant dans le temps. Les lecteurs peuvent à la fois avoir des intérêts courts termes lorsqu’ils veulent s’informer sur un sujet de façon approfondie, c’est-à-dire consulter une série d’articles traitant du même sujet, mais aussi avoir des intérêts longs termes sur des thématiques spécifiques (An et al., 2019).

Ajoutons que, de manière générale, à l’opposé de certains domaines tels que la musique, les utilisateurs ont un besoin constant de diversité, de nouveauté et de sérendipité, les trois critères qualitatifs introduits en section 5.3.2 (Karimi et al., 2018; Maksai et al., 2015). En effet, lorsqu’un utilisateur lit une suite d’articles recommandés par un système, celui-ci préférera éviter les informations redondantes vis-à-vis de ce qu’il a déjà lu dans cette liste, mais aussi vis-à-vis de ses connaissances propres. Il trouvera également pertinents les articles portant sur des sujets inattendus qui pourront lui faire découvrir de nouvelles thématiques d’intérêt. Ces critères de qualité sont également, dans l’absolu, des caractéristiques « souhaitées » pour toutes les raisons évoquées en section 5.3.2 (Pariser, 2011; Helberger, 2019).

Il existe d’autres propriétés spécifiques à la recommandation d’articles d’actualité décrites, entre autres, par Raza et Ding (2020), Qin et Lu (2020) et Li et Wang (2019), les plus importantes étant :

- la durée de consommation d’un item, qui est d’en moyenne quelques minutes lorsque la durée de lecture d’un livre ou le visionnage d’un film prend plusieurs heures ;
- le catalogue d’articles disponibles sur un site web, qui est en constante croissance ;
- le contexte de lecture (e.g. jour de la semaine, moment de la journée, localisa-

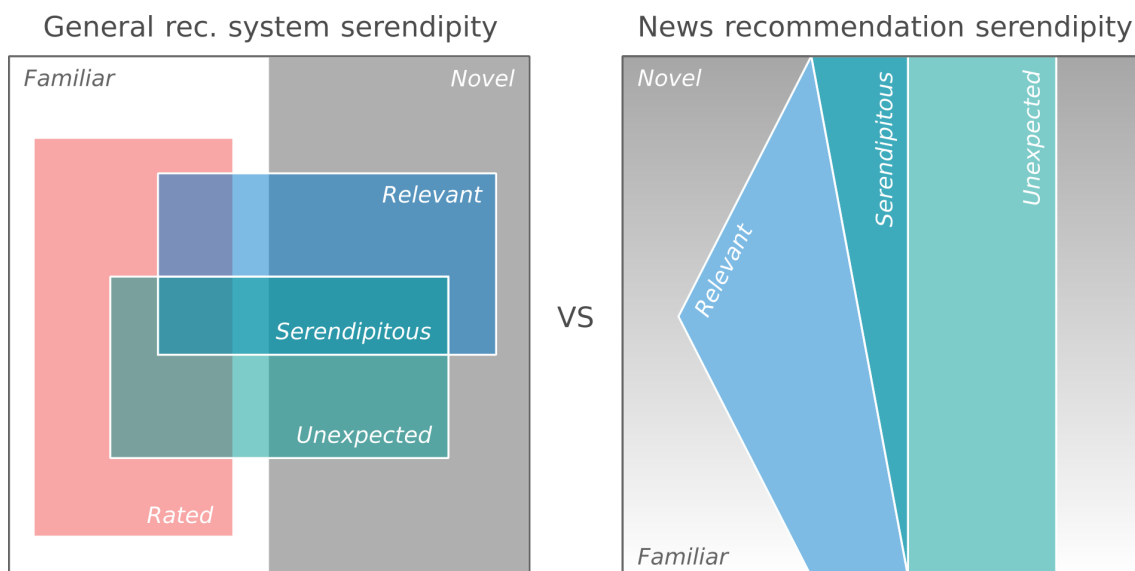


FIGURE 5.2 – Diagramme d’Euler de la serendipité dans les systèmes de recommandation proposé par [Kotkov et al. \(2016\)](#) à gauche et variante proposée dans cette thèse pour la recommandation d’articles d’actualité

tion), qui est une donnée importante à prendre en compte lorsqu’il s’agit de recommander des articles d’actualité ([Lommatzsch et al., 2017](#); [Chen et al., 2017](#); [Ma et al., 2016](#));

- les choix de lectures, qui sont influencés par différents facteurs sociaux (e.g. les amis, les relations et événements sur les réseaux sociaux).

[Kotkov et al. \(2016\)](#) proposent une revue de la littérature sur la sérendipité dans les systèmes de recommandation. Les auteurs proposent un diagramme d’Euler permettant d’illustrer les composantes de la sérendipité entre nouveauté, pertinence, surprise et items déjà consommés. Dans cette section, nous proposons une modification de ce diagramme pour prendre en compte les particularités de la recommandation d’articles d’actualité. Notre diagramme, à droite sur la figure 5.2, introduit la notion de nouveauté relative et non absolue comme détaillé en section 5.3.2. Le diagramme d’Euler proposé par [Kotkov et al. \(2016\)](#) est à gauche sur la figure 5.2. À noter que le concept de « noté » pourrait se traduire par « cliqué » dans le cas de la recommandation d’articles d’actualité.

Les items candidats à une recommandation sont toujours nouveaux (i.e. récemment publiés), le concept d’item « noté » (ou *rated*) introduit par [Kotkov et al. \(2016\)](#) n’est donc pas pertinent pour la recommandation d’articles d’actualité. Dans notre proposition, ce constat conduit à considérer la nouveauté comme une valeur continue et non booléenne. Les items les plus pertinents se situent plus probablement au centre du diagramme, entre totalement familiers et totalement nouveaux. En effet, comme développé par [Kotkov et al. \(2016\)](#), les items nouveaux pour un utilisateur seront plus probablement non pertinents. De l’autre côté, les items familiers seront redondants et donc plus probablement non pertinents. En conséquence, la forme correspondant à la « pertinence » est plus large au centre du diagramme. Ensuite, les items seront plus probablement surprenants (ou inattendus) s’ils sont nouveaux pour l’utilisateur et plus probablement non surprenants s’ils sont familiers. En effet,

la recommandation d'un item familier par rapport à ses lectures passées sera moins surprenante que la recommandation d'un item portant sur un sujet nouveau. En conséquence, la forme correspondante à l'« inattendu » est plus large du côté de la nouveauté totale. La sérendipité est l'intersection de ces deux concepts et est plus large du côté de la nouveauté. En effet, elle consiste en la recommandation des items étant à la fois pertinents, inattendus et nouveaux (i.e. non totalement familiers).

5.4.3 Jeux de données pour la recommandation d'articles d'actualité

Selon la revue de littérature proposée par [Raza et Ding \(2020\)](#), plus de 80% des travaux publiés dans le domaine de la recommandation d'articles d'actualité consistent en l'utilisation de jeux de données privés lorsque l'évaluation des algorithmes est effectuée *offline*. [Karimi et al. \(2018\)](#) font également ce constat. Par exemple, [Maksai et al. \(2015\)](#) utilisent un jeu de données *offline* récolté sur *swissinfo.ch*, et [Chakraborty et al. \(2017\)](#) ainsi que [Cami et al. \(2017\)](#) utilisent des jeux de données récoltés sur des pages Twitter de journaux en ligne (e.g. New York Times, BBC). De plus, la majorité des évaluations *onlines* sont menées sur des plateformes privées ([Liu et al., 2010](#); [Li et al., 2011](#); [Kirshenbaum et al., 2012](#); [Garcin et al., 2014](#); [Maksai et al., 2015](#)), mis à part quelques travaux ayant exploité la plateforme *CLEF NewsREEL* ([Hopfgartner et al., 2016](#)) de 2015 à 2017 ([Liang et al., 2017](#); [Beck et al., 2017](#); [Yuan et al., 2016](#)).

Le constat que les jeux de données *offlines* restent privés peut s'expliquer par le caractère « confidentiel » des données (e.g. les données contiennent les sujets d'intérêt des lecteurs) et qu'il n'est souvent pas possible de les partager. De plus, le constat que les plateformes *onlines* restent privées peut s'expliquer par le fait qu'il est difficile d'ouvrir celles-ci à l'évaluation d'algorithmes de recommandation externes, et qu'il est risqué d'évaluer des algorithmes de recommandation sachant qu'une partie d'entre eux seront sous-optimaux. En effet, ce processus peut altérer l'expérience utilisateur et donc faire décroître les revenus générés par la plateforme. En conséquence, la reproductibilité des expérimentations dans ce domaine est limitée.

Aujourd'hui, il existe néanmoins quelques jeux de données *offlines* mis à disposition à la communauté de recherche. Ceux-ci sont notamment décrits par [Wu et al. \(2020\)](#), [Raza et Ding \(2020\)](#) et [Karimi et al. \(2018\)](#) :

CiteULike Ce jeu de données⁹ regroupe un ensemble d'articles mis en favoris par les utilisateurs de l'application *CiteULike*. Ce jeu de données a permis la publication de différentes études sur la recommandation d'articles d'actualité ([Lee et Brusilovsky, 2017](#); [Bansal et al., 2016](#); [Chen et al., 2017b](#)). Cependant, les articles de ce jeu de données ne sont pas des articles d'actualité mais des ressources bibliographiques. En conséquence, ces données n'exposent pas toutes les problématiques liées aux particularités de la recommandation d'articles d'actualité.

Yahoo! News Ce jeu de données¹⁰ est dédié à la recommandation basée sur la « session ». Il ne comprend pas d'utilisateurs mais des sessions de lecture. Les

9. *CiteULike* est disponible à l'adresse <https://github.com/js05212/citeulike-a>

10. *Yahoo! News* est disponible à l'adresse <https://webscope.sandbox.yahoo.com/catalog.php?datatype=1>

articles sont au nombre de 14 180 et en langue anglaise (Trevisiol et Aiello, 2014). Il a notamment été utilisé par Maksai et al. (2015) et Bai et al. (2017).

Adressa Ce jeu de données¹¹ extrait d'un site d'actualité norvégien comprend 48 486 articles et 3 083 438 utilisateurs (Gulla et al., 2017).

Globo Ce jeu de données¹² assemble des articles du site d'actualité brésilien *globo.com* au nombre de 46 000 (de Souza Pereira Moreira et al., 2018). Il contient 314 000 utilisateurs. Cependant, le contenu des articles n'est pas mis à disposition.

Plista Ce jeu de données¹³ assemble 70 353 articles d'actualité et des clics utilisateurs (Kille et al., 2013) enregistrés de la plateforme *CLEF NewsREEL* (Hopfgartner et al., 2016). Les articles sont en langue allemande. Ce jeu de données est constitué d'historiques utilisateurs courts. Il présente quelques désavantages que nous détaillons au chapitre 9.

MIND Ce jeu de données¹⁴ correspond à des données d'interaction entre le site web Microsoft News et des lecteurs (Wu et al., 2020). Il est aujourd'hui le jeu de données de référence puisqu'il pallie tous les problèmes cités précédemment. Les articles sont en langue anglaise. Les articles d'actualité sont nombreux (161 013) et leur contenu est mis à disposition. Les utilisateurs, au nombre de 1 000 000 et anonymisés, ont tous cliqué sur cinq articles au minimum sur une période d'un mois entre octobre et novembre 2019.

5.4.4 Algorithmes de recommandation

Le domaine de la recommandation de livres et de la recommandation d'articles scientifiques sont proches du domaine de la recommandation d'articles d'actualité dans le sens où le contenu des items est riches d'information qu'il est possible d'exploiter grâce aux méthodes basées sur le contenu. Les problématiques de la recommandation d'articles d'actualité sont, en revanche, différentes : le catalogue d'items évolue rapidement et les items deviennent non pertinents au bout de seulement quelques jours. Les méthodes employées sont néanmoins proches de celles employées pour la recommandation d'articles d'actualité. Par exemple, Vaz et al. (2013a) utilisent le filtrage collaboratif pour la recommandation de livres. Pera et Ng (2015) exploitent des données externes, les avis des lecteurs, afin d'améliorer leur système de recommandation. Plus récemment, Alharthi et al. (2018b) proposent une méthode basée sur le contenu en utilisant un *Support Vector Regression*¹⁵ (abrégé SVR) afin de prédire la pertinence des livres à partir de leur représentation distributionnelle. Nous reviendrons sur ces travaux en section 5.4.4.

Récemment, Bai et al. (2019) proposent une revue de la littérature sur la recommandation d'articles scientifiques. Tout comme pour les autres domaines de la recommandation, la recommandation d'articles scientifiques se repose sur le filtrage collaboratif et le filtrage par le contenu. Les auteurs mentionnent aussi les méthodes

11. *Adressa* est disponible à l'adresse <http://reclab.idi.ntnu.no/dataset>

12. *Globo* est disponible à l'adresse <https://www.kaggle.com/gspmoreira/news-portal-user-interactions-by-globocom>

13. *Plista* est disponible à l'adresse <http://www.newsreelchallenge.org/dataset>

14. *MIND* est disponible à l'adresse <https://msnews.github.io>

15. Un *Support Vector Regression* est une modification du modèle SVM permettant d'effectuer une régression.

basées sur les graphes (Ohta et al., 2011; Zhao et al., 2016; Wang et al., 2016) qui peuvent néanmoins être considérées comme des méthodes basées sur le contenu. Les auteurs expliquent qu’il est pertinent de construire un graphe étant donné que les articles peuvent être reliés par des citations. Le processus de recommandation consiste alors à exploiter ce graphe en le parcourant, par exemple, à partir des nœuds des articles ayant été lus par l’utilisateur.

Malgré les problématiques de la recommandation d’articles d’actualité mentionnées en section 5.4.2, certains travaux utilisent le filtrage collaboratif lorsque suffisamment de données d’interactions communes (i.e. plusieurs utilisateurs ont interagi avec les mêmes items) sont récoltées. Par exemple, Xiao et al. (2015) proposent une méthode de filtrage collaboratif appelée TOCF qui inclut la dimension temporelle. Les auteurs prennent en compte les séquences de lecture des utilisateurs dans leur jeu de données collecté sur un site d’actualité chinois. Plus récemment, Park et al. (2017) utilisent un réseau de neurones récurrents afin d’effectuer un filtrage collaboratif en prenant également en compte les séquences de lecture des utilisateurs.

Avant l’utilisation des réseaux de neurones profonds pour la recommandation basée sur le contenu, la plupart des travaux exploitaient la représentation sémantique des articles d’actualité comme proposé par Hu et al. (2017) et Nath Nandi et al. (2018), ou encore la représentation sac de mots, TFIDF et de *topic modeling* comme proposé par Abel et al. (2011), Bai et al. (2017) ou encore Capelle et al. (2012). Certains travaux exploitent aussi des informations contextuelles comme la popularité (Maksai et al., 2015), le jour de la semaine (Lommatzsch et al., 2017) ou encore la localisation (Chen et al., 2017). Des méthodes hybrides ont aussi été utilisées telles que celle proposée par Liu et al. (2010) pour Google News. Les auteurs utilisent une méthode bayésienne pour prédire l’intérêt des utilisateurs en fonction de leurs clics et des catégories d’articles qu’ils ont lus dans le passé.

Aujourd’hui, la majorité des travaux dans la recommandation d’articles d’actualité exploitent le contenu des articles par des réseaux de neurones profonds. Contrairement aux méthodes classiques de représentation du texte, les réseaux de neurones profonds prennent en compte l’ordre des mots, peuvent capturer des relations complexes entre items et utilisateurs, même lorsque la matrice d’interaction est creuse (ou *sparse*), c’est-à-dire lorsqu’il y a peu d’interactions par utilisateurs. Raza et Ding (2020) décrivent d’autres avantages à l’utilisation des réseaux de neurones profonds pour la recommandation d’articles d’actualité tels que leur capacité à apprendre à partir de données en hautes dimensions, i.e. le texte des articles, ou encore de permettre l’apprentissage par transfert.

Par exemple, de Souza Pereira Moreira et al. (2018), Zihayat et al. (2019), Jugovac et al. (2018) et Park et al. (2017) utilisent des réseaux de neurones récurrents afin d’effectuer des recommandations à partir de l’historique des utilisateurs. Ce type d’architecture est utilisé afin d’effectuer des recommandations de type « session » en exploitant les séquences d’items consommés par les utilisateurs. Il s’agit, à partir de l’historique d’un utilisateur composé d’items qu’il a cliqués, de prédire un score indiquant à quel point un item candidat est pertinent. Kumar et al. (2017), Wang et al. (2018), Qiannan et al. (2019) et Wu et al. (2019) introduisent une couche d’attention aux réseaux de neurones profonds dans l’objectif de pondérer les items historiques en fonction de l’item candidat considéré. En effet, comme nous le verrons en section 8.3, un item candidat n’est pertinent que vis-à-vis d’une portion des items historiques de l’utilisateur.

À noter cependant que, pour la majorité de ces travaux, ces réseaux de neurones profonds n'utilisent que des sessions utilisateurs, donc des historiques de lectures ne reflétant que des intérêts courts termes. De plus, les retours de pertinence considérés sont les « clics ». Or, comme nous le verrons aux chapitres 8 et 9, une action de ce type n'indique pas nécessairement que l'utilisateur a apprécié la lecture de l'item. Nous ajouterons que la majorité de ces travaux n'utilise que les titres des articles et non leur contenu. En effet, lorsqu'un utilisateur clique sur un article, il n'a lu que le titre de cet article. Ainsi, il est pertinent de se baser uniquement sur cette donnée pour effectuer des recommandations puisque le contenu de l'item n'a pas influencé la décision de l'utilisateur. Cependant, dans cette thèse, nous soutenons que la considération du contenu des articles, ainsi que la collecte de retours de pertinence plus complets comme décrit au chapitre 9, est nécessaire dans l'objectif de prendre en compte les réelles préférences des utilisateurs. La seule prise en compte du titre des articles peut avoir comme conséquence que les réseaux de neurones profonds apprennent à recommander les articles dont le titre peut être considéré comme « piège à clics » (ou *clickbait*).

Comme détaillé en section 5.2.3, la majorité des méthodes de filtrage par le contenu exploitent les connaissances qu'apporte l'ensemble des interactions utilisateurs-items. Ainsi, ces méthodes sont proches des méthodes « hybrides » puisqu'exploitent à la fois le contenu et des données de type « collaboratif ». En effet, la pertinence d'un item vis-à-vis d'un utilisateur ne dépendra pas uniquement de son historique mais aussi de l'historique des autres utilisateurs puisqu'un seul et même modèle est entraîné sur l'ensemble des utilisateurs. Ainsi, ce type de modèle parvient à capturer des relations complexes entre items. Autrement dit, lorsqu'il s'agit de recommander des items pertinents à un utilisateur en particulier, ce type de modèle exploite non seulement l'historique de l'utilisateur mais aussi les relations entre items apprises grâce à tous les autres utilisateurs.

5.4.5 Limites identifiées dans l'état de l'art

Le style dans la recommandation d'articles d'actualité

Burgoon et al. (1981) sont, à notre connaissance, les premiers à avoir mis en évidence une corrélation entre le style écrit et les préférences des individus dans leur lecture de l'actualité. Les auteurs ont mené une étude incluant 4 020 lecteurs de six journaux différents. Ces lecteurs devaient répondre à plusieurs questions portant notamment sur leur satisfaction à la lecture des six journaux. L'étude permit de découvrir l'existence de corrélations entre le style employé par les journaux et le jugement des lecteurs envers ces journaux (e.g. impression de compétence, confiance accordée).

a) *the newspaper is perceived as more competent/trustworthy when lexical diversity, number of pauses (or punctuation), and the number of total words are lower and ease of reading is higher (based on readability); b) the newspaper will appear more stimulating as productivity (the frequency of words or sentences) increases; c) it also appears that average sentence length may affect the ease with which the newspaper can be read, and productivity and emotiveness may affect competence.*

Burgoon et al. (1981)

Plus récemment, Wang et al. (2017) mentionnent également que le style peut jouer un rôle dans la sélection des articles par les administrateurs d'applications liées à l'actualité (i.e. application d'agrégation d'articles d'actualité). Les auteurs utilisent un réseau de neurones profond basé sur l'architecture CNN pour effectuer une présélection d'articles pour les administrateurs des applications. Cependant, leur étude n'inclut pas l'utilisation de caractéristiques textuelles de style écrit.

Pon et al. (2007) ont utilisé différentes métadonnées et caractéristiques textuelles dans l'objectif de prédire l'intérêt des utilisateurs pour des articles d'actualité. Les auteurs appellent cet intérêt l'« *interestingness* » qui s'oppose à la notion de pertinence en recherche d'information. En effet, les auteurs expliquent que les résultats fournis par un moteur de recherche, après qu'un utilisateur ait effectué une requête, peuvent ne pas être intéressants pour cet utilisateur mais seulement pertinents vis-à-vis de la requête. Les résultats renvoyés doivent ainsi être personnalisés en fonction des utilisateurs.

Les auteurs de cette étude ne disposaient pas de données d'interactions entre des utilisateurs et des articles d'actualité telles que des clics ou des retours de pertinence explicites. Leur expérimentation se base sur le jeu de données *Yahoo! News RSS feeds* constitué d'articles labélisés dans différentes catégories telles que « *Most Viewed Technology* » ou « *Top Stories Politics* ». Ces catégories servent d'indicateurs pour l'*interestingness*. Par exemple, la catégorie « *Most Viewed Technology* » est considérée comme intéressante pour les profils de lecteurs lisant des articles dans le domaine de la technologie. Afin de prédire l'*interestingness* des articles d'actualité, Pon et al. (2007) proposent d'utiliser différents classifieurs et de combiner différentes caractéristiques telles que la fraîcheur des articles, leur popularité, la réputation des sources ou encore la représentation TFIDF. Les auteurs ont aussi utilisé différentes caractéristiques de style proposées par Chesley et al. (2006). Ces caractéristiques sont cependant des caractéristiques dites *handcrafted* basées sur la présence de mots appartenant à un lexique créé manuellement, aux comptes de différents éléments de ponctuation ainsi qu'aux comptes de différentes étiquettes morpho-syntaxiques. Les résultats de l'étude montrent qu'il existe une corrélation entre le style écrit et l'*interestingness* des articles d'actualité, mais que cette corrélation dépend fortement des catégories considérées.

Nishitarumizu et al. (2010) ont, par la suite, mené une étude sur le même jeu de données mais en proposant un nouveau modèle de classification. Les auteurs ont repris les métadonnées et caractéristiques textuelles proposées par Pon et al. (2007). Ils ont rapporté une amélioration de la précision de leur modèle comparé au modèle de Pon et al. (2007) mais n'ont cependant pas étudié l'impact de la prise en compte

du style écrit.

Plus tard, [Pon et al. \(2011\)](#) ont mené des expérimentations similaires incluant également des données d'interaction entre utilisateurs et articles d'actualité. Le jeu de données utilisé par les auteurs de l'étude n'était cependant constitué que d'un nombre limité d'utilisateurs (moins de 100). De plus, ils n'ont rapporté qu'une faible corrélation entre le style écrit et l'*interestingness* et n'ont pas pu mesurer les bénéfices de ces caractéristiques parmi l'ensemble des caractéristiques utilisées. Les auteurs ont rapporté, tout comme dans l'étude précédente, que la corrélation est fortement dépendante des utilisateurs, i.e. elle n'est pas nécessairement corrélée à chaque utilisateur.

Plus récemment, [Sertkan et al. \(2019\)](#) ont utilisé une mesure stylométrique basée sur la fréquence des mots dans l'objectif de différencier les auteurs des documents d'un corpus d'articles d'actualité. Les auteurs ont montré que leur modèle parvient à identifier les auteurs des articles avec une précision de 97% sur un nombre restreint d'articles d'actualité. Ils ont émis l'hypothèse que, dans le cadre d'un système de recommandation d'articles d'actualité, générer des recommandations sur la base de caractéristiques stylistiques permettrait la diversification des recommandations. Ils n'ont cependant pas validé cette hypothèse car n'ont pas effectué d'expérimentation portant sur la recommandation d'articles d'actualité.

Les travaux du domaine de la recommandation de livres sont liés aux travaux de cette thèse. En effet, plusieurs articles de recherche présentent l'exploitation du style écrit dans la recommandation de livres ([Alharthi et Inkpen, 2019](#); [Vaz et al., 2012b, 2013b](#); [Alharthi et al., 2018b](#)). Par exemple, [Alharthi et al. \(2018b\)](#) ont utilisé un CNN pour apprendre à identifier des auteurs de livres dans le jeu de données *Litrec*. Les représentations apprises permettent la prédiction de la pertinence d'un livre pour un lecteur grâce à un SVR. Les scores prédits par le modèle SVR permettent d'établir la pertinence des livres en fonction d'un historique de lecture. Les auteurs rapportent obtenir de meilleures performances comparées à une autre méthode de représentation distributionnelle du texte, *Doc2Vec* ([Le et Mikolov, 2014](#)), ainsi que des méthodes de *topic modeling*, LDA et LSI. Cependant, comme nous l'avons vu en section 1.3.5, les auteurs de cette étude ne généralisent pas les représentations du style mais utilisent des représentations spécifiques aux auteurs des livres recommandés. Plus récemment, [Alharthi et Inkpen \(2019\)](#) ont utilisé différentes caractéristiques dites *handcrafted* (e.g. utilisation du lexique LIWC, la fréquence des mots, le compte de différents éléments de ponctuation) afin d'effectuer une recommandation de livres sur le jeu de données *Litrec* ([Vaz et al., 2012a](#)). Les auteurs ont analysé l'importance de chacune de ces caractéristiques dans la préférence des lecteurs. Ils ont notamment rapporté que des caractéristiques de style telles que le nombre d'adverbes ou encore la présence de mots appartenant à la catégorie *core drives and needs* du lexique LIWC sont très corrélées aux préférences des lecteurs.

En définitive, à notre connaissance, aucun travail de recherche ne consiste en l'étude de l'intérêt des caractéristiques de style dans la recommandation d'articles d'actualité. Comme nous le verrons au chapitre 2, le style écrit est difficile à extraire du texte par de simples caractéristiques dites *handcrafted*. Il est donc nécessaire d'avoir recours à des méthodes récentes (e.g. réseaux de neurones profonds) capables de capturer automatiquement ce type d'information en se focalisant sur des parties précises du texte, les parties incluant des indices linguistiques relevant du style des auteurs. Dans cette thèse, nous proposons d'employer des représentations stylo-

métriques générales apprises sur un large corpus d'articles d'actualité et de blogs dans l'objectif de mettre en évidence l'intérêt du style écrit dans le processus de recommandation d'articles d'actualité. Nous étudions également d'autres mesures indiquant la qualité des recommandations par le style telles que la diversité, la nouveauté et la sérendipité. En effet, nous soutenons que les lecteurs peuvent avoir des préférences stylistiques dans leur lecture de l'actualité en plus de préférences liées, par exemple, à la thématique. Ainsi, dans les chapitres 6, 7 et 8, nous tenterons de répondre aux questions de recherche QR_3 , QR_4 et QR_5 portant sur le style écrit dans la recommandation d'articles d'actualité.

Données issues des réseaux sociaux pour l'évaluation *offline*

Quelques travaux de recherche ont, par le passé, utilisé des données Twitter pour l'évaluation de systèmes de recommandation d'articles d'actualité (Abel et al., 2011; Phelan et al., 2009; Lee et al., 2014; Chen et al., 2017; Zarrinkalam et al., 2017). Par exemple, Phelan et al. (2009) proposent un modèle basé sur la pondération TFIDF et Lee et al. (2014) proposent un modèle basé sur LDA. Cependant, ces deux études n'incluent qu'un nombre limité d'utilisateurs Twitter. D'autres utilisent les données disponibles sur Twitter afin d'enrichir les informations relatives aux utilisateurs ou aux articles présents dans un jeu de données (Lin et al., 2014a; Morales et al., 2012; Tavakolifard et al., 2013; Lin et al., 2014b; Cucchiarelli et al., 2018). Par exemple, Tavakolifard et al. (2013) utilisent des données Twitter afin de déduire la popularité des articles d'actualité de leur jeu de données initial. Morales et al. (2012) enrichissent le jeu de données *Yahoo! News* et obtiennent de meilleures performances sur ce jeu de données.

Dans cette thèse, nous voulons exploiter des données issues des réseaux sociaux puisque, comme nous l'avons détaillé en introduction de ce manuscrit, les données annexes (i.e. qui ne sont pas indispensables à un algorithme de recommandation) telles que les messages postés et les relations d'amitié sont riches d'informations utiles sur les utilisateurs. Nous soutenons que les données issues des réseaux sociaux permettent une meilleure généralisation « cross-domaines » des algorithmes de recommandation entraînés sur ce type de données. En effet, les messages que postent les utilisateurs des réseaux sociaux peuvent contenir des informations utiles dans la modélisation de leur profil telles que des informations liées aux activités de vie quotidienne des utilisateurs (e.g. activités sportives, passe-temps, métiers) ou liées à leurs opinions politiques (Meguebli et al., 2017). Toutes ces informations sont utiles lorsqu'il s'agit de proposer des services personnalisés aux utilisateurs.

Pour ces raisons ainsi que d'autres liées aux retours de pertinence que nous détaillons au chapitre 7, il est pertinent, pour effectuer des expérimentations sur la tâche de recommandation d'articles d'actualité, de pouvoir disposer de données rassemblant à la fois :

- le contenu des articles d'actualité ;
- des données utilisateurs issues des réseaux sociaux (e.g. messages, relations d'amitié, textes descriptifs (ou *bio*) ;
- des données indiquant l'intérêt des utilisateurs pour une partie des articles d'actualité (e.g. sous forme de partage d'*urls*).

À notre connaissance, le seul jeu de données existant ayant ces caractéristiques était celui proposé par Abel et al. (2011). Ce jeu de données a permis la publication de plusieurs études sur la recommandation d'articles d'actualité (Chen et al., 2017;

Zarrinkalam et al., 2017). Cependant, celui-ci n'est aujourd'hui plus partagé à la communauté de recherche. De plus, il ne rassemblait qu'un nombre limité d'utilisateurs (environ 1 600) et n'incluait pas, par défaut, le contenu des articles d'actualité.

Dans le chapitre 7, nous proposons un nouveau jeu de données, appelé *TwineWS*, ayant ces différentes caractéristiques. Ce jeu de données est suffisamment large pour que les expérimentations effectuées sur celui-ci permettent, tout d'abord, de conclure de manière statistiquement fiable de la pertinence des modèles comparés, mais aussi d'entraîner des modèles sans que ceux-ci sur-apprennent sur un nombre limité de profils utilisateurs, une problématique courante lorsque les méthodes utilisées se basent sur des réseaux de neurones profonds et des représentations distributionnelles du texte. *TwineWS* se compose de plus de 20 000 profils Twitter « actifs » (i.e. appartenant à des utilisateurs ayant partagé plus de 10 articles sur une période de 4 mois) ainsi que d'environ 1 000 000 articles d'actualité. Il se compose d'un ensemble complet de données (e.g. contenu des articles, contenu des *tweets*) et d'un découpage des données effectué à l'avance, évitant ainsi un travail supplémentaire pour les chercheurs voulant l'exploiter.

Plateformes d'évaluation *online* ouvertes à la communauté de recherche

Comme nous l'avons expliqué en section 5.3, et comme nous le verrons au chapitre 7, l'évaluation *offline* souffre de différents biais (Karimi et al., 2018). Pour effectuer une évaluation précise et non biaisée des algorithmes de recommandation, il est nécessaire de prendre en compte les retours de pertinence d'utilisateurs interagissant en temps réel (Garcin et al., 2014; Beel et al., 2013; Chen et al., 2017a; Myttenaere et al., 2015). Il est donc important de disposer de plateformes permettant l'évaluation des systèmes de recommandation en conditions *online*.

De 2015 à 2017, les chercheurs travaillant sur le sujet de la recommandation d'articles d'actualité pouvaient s'inscrire aux compétitions *CLEF NewsREEL* afin d'évaluer et comparer leurs systèmes de recommandation lorsque la majorité des plateformes dédiées à la recommandation d'articles d'actualité étaient privées : Google News (Liu et al., 2010), Yahoo! News (Li et al., 2011), Forbes (Kirshenbaum et al., 2012), swissinfo.ch (Garcin et al., 2014), LePoint (Maksai et al., 2015). Cette plateforme permit la publication de nombreux travaux de recherche sur la tâche de la recommandation d'articles d'actualité (Liang et al., 2017; Beck et al., 2017; Yuan et al., 2016).

Cependant, la plateforme *CLEF NewsREEL* n'est aujourd'hui plus disponible et présente différents désavantages que nous présentons plus en détail au chapitre 9. Au chapitre 9, nous introduisons une nouvelle plateforme, appelée « *Renewal* », permettant de pallier les différents problèmes de *CLEF NewsREEL*. Le projet *Renewal* a démarré durant cette thèse et est actuellement en cours de développement.

5.5 Conclusion

Cet état de l'art a permis de donner un aperçu du domaine des systèmes de recommandation, et notamment des différentes notions dont la compréhension est nécessaire à la lecture des prochains chapitres. Nous avons apporté notre propre point de vue sur différents concepts inhérents à la littérature du domaine des systèmes de recommandation. Ainsi, nous avons proposé :

1. Une analyse détaillée des métriques de qualité pour l'évaluation des systèmes de recommandation ainsi que des biais qu'ils permettent de résoudre et des causes de ceux-ci (e.g. phénomène de bulle de filtres, ignorance rationnelle). Nous détaillerons d'autres causes dans les prochains chapitres, notamment concernant les retours de pertinence utilisés.
2. Une amélioration du diagramme d'Euler proposé par [Kotkov et al. \(2016\)](#) pour la recommandation d'articles d'actualité.
3. Une taxonomie des méthodes et métriques d'évaluation pour les systèmes de recommandation.

Les chapitres 6, 7 et 8 porteront sur les questions de recherche *QR3*, *QR4* et *QR5* et permettront d'apporter des solutions aux limites de l'état de l'art exposées en section 5.4.5.

Chapitre 6

Recommandation d'articles d'actualité par l'exploitation de représentations du texte

6.1 Introduction

Dans ce chapitre, nous commençons par formuler, en section 6.2, le problème de recommandation d'articles d'actualité. Nous décrivons les données nécessaires ainsi que le processus d'évaluation des algorithmes de recommandation. Celui-ci consiste en le découpage temporel d'un jeu de données composé de retours de pertinence préalablement enregistrés. Les algorithmes de recommandation sont évalués, pour chaque utilisateur, sur leur capacité à correctement ordonner les articles candidats à une recommandation.

Nous proposons ensuite, en section 6.3, un nouvel algorithme générique de recommandation se basant sur la notion de « proximité partielle » des items candidats et historiques. Nous formulons l'*hypothèse de proximité partielle* que nous validerons dans les prochains chapitres. L'objectif de l'algorithme proposé en section 6.3 est de permettre la comparaison des différentes représentations vectorielles de texte introduites au chapitre 4 dans la tâche de recommandation d'articles d'actualité. Cet algorithme peut utiliser tout modèle de représentation permettant la projection d'un article en un vecteur représentatif. Il ne se base que sur ces représentations vectorielles ainsi que sur une fonction de similarité pour l'ordonnement des candidats, et permet une comparaison directe et équitable de la capacité de chaque modèle à extraire des indices reflétant les préférences des lecteurs.

Chaque représentation textuelle ne couvrira pas nécessairement tous les axes jouant un rôle dans les préférences des lecteurs. C'est pourquoi nous proposons, en section 6.4, un algorithme d'hybridation permettant d'évaluer la complémentarité des représentations vectorielles utilisées. Cet algorithme se base sur la pondération et le rééchelonnage des ordonnancements. Nous proposons également une nouvelle métrique, la « dominance », permettant de mesurer à quel point un modèle impose son ordonnancement propre des items dans une hybridation.

6.2 Évaluation *offline* par découpage temporel

L'évaluation *offline* d'un algorithme de recommandation d'articles d'actualité doit être paramétrée de telle sorte que les conditions expérimentales reflètent les conditions réelles d'un système de recommandation en ligne. Les systèmes de recommandation en ligne disposent d'un historique d'interaction utilisateurs-items ainsi que d'items candidats généralement « nouveaux » à recommander. Pour simplifier, le système doit faire un choix parmi ces items et recommander différents sous-ensembles aux utilisateurs en exploitant les connaissances qu'il peut tirer de l'historique des utilisateurs. Afin d'être le plus conforme possible aux conditions réelles, l'évaluation *offline* consiste donc, après l'enregistrement d'interactions observées dans un système en production, en le découpage temporel des données. Il s'agit de choisir une date pivot qui correspondrait au « présent » en conditions réelles. Nous considérerons tous les items ayant une date d'ajout postérieure à la date pivot comme étant l'ensemble des items candidats à une recommandation.

Notons $U = \{u_1, \dots, u_n\}$ un ensemble d'utilisateurs et $I = \{i_1, \dots, i_m\}$ un ensemble d'items. Notons $L = \{l_1, \dots, l_q\}$ l'ensemble des interactions utilisateurs-items. Chaque élément de L est un tuple composé d'un utilisateur de u_i et un item i_j indiquant une interaction observée entre u_i et i_j (e.g. un clic ou un partage). Notons $D = \{d_1, \dots, d_m\}$ un ensemble d'entiers traduisant la date d'interaction la plus ancienne entre un item et les utilisateurs, typiquement une heure *Unix*¹ en millisecondes. Pour simplifier, nous supposons que si deux utilisateurs ont interagi avec le même item, cette interaction se sera produite à la même date. En conséquence, D est de même taille que I :

$$|D| = |I| = m$$

Pour tout item $i_i \in I$, la date correspondante à cet item porte le même indice dans D , i.e. la date d'un item i_i est d_i . L'ensemble des dates de D est ordonné de façon croissante selon leur valeur et deux dates ne peuvent pas être égales, en conséquence :

$$\forall i_v, i_w \in I, d_v < d_w \Leftrightarrow v < w$$

Prenons p l'indice d'une date $d_p \in D$ pivot tel que $1 < p < m$. Notons $I^h \subset I$ l'ensemble des items historiques et $I^c \subset I$ l'ensemble des items candidats de telle sorte que :

$$I^h \cap I^c = \emptyset$$

$$\forall i_i \in I, i_i \in I^h \Leftrightarrow d_i \leq d_p$$

$$\forall i_j \in I, i_j \in I^c \Leftrightarrow d_j > d_p$$

Notons $C = \{c_1, \dots, c_n\}$ un ensemble d'ensembles de candidats, chacun de ces ensembles étant attribué à un utilisateur. Les indices dans C correspondent aux indices dans U , donc C et U sont de même taille :

$$|U| = |C| = n$$

Chaque élément de C est un ensemble d'items de I^c de telle sorte que tous les items dans les ensembles de C ont une date postérieure à la date pivot d_p :

$$\forall c_i \in C, \forall i_j \in c_i, i_j \in I^c$$

1. Une heure *Unix* est une mesure du temps basée sur le nombre de secondes écoulées depuis le 1er janvier 1970.

Chaque ensemble d'items c_i dans C est composé au minimum de l'ensemble des items avec lesquels l'utilisateur u_i a interagi ainsi que d'autres items pris au hasard dans I^c . Plus formellement :

$$\forall (u_i, i_j) \in L, i_j \in c_i$$

Tout ensemble de candidats est de taille égale k et est supérieur au nombre d'interactions de l'utilisateur correspondant :

$$\forall c_i \in C, |c_i| = k$$

$$\forall c_i \in C, |c_i| > |I'|$$

$$I' = \{i'_1, \dots, i'_k\}, \forall i'_j \in I', \exists (u_i, i'_j) \in L$$

Nous pourrions par exemple choisir $k = 1000$, ce qui signifiera que les algorithmes doivent ordonner 1000 candidats par utilisateur.

Pour effectuer une recommandation, un système de recommandation doit choisir des sous-ensembles C' des ensembles de candidats C , chacun ayant une taille définie t :

$$C' = \{c'_1, \dots, c'_k\}$$

$$\forall c'_i \in C', |c'_i| = t$$

$$\forall i_i \in c'_j, i_i \in c_j$$

Ces sous-ensembles sont les recommandations du système aux utilisateurs. Différentes métriques telles que *CTR* sont alors utilisées afin d'évaluer l'intérêt que porte un utilisateur à un sous-ensemble de recommandation. Pour une évaluation *offline*, nous connaissons quels items d'un sous-ensemble de candidats sont dits « pertinents » en vérifiant leur existence dans L . Pour obtenir une mesure de précision et de rappel d'un sous-ensemble $c'_i \in C'$, il suffit de calculer :

$$precision(c'_i) = \frac{|relevance(c'_i)|}{|c'_i|} \quad (6.1)$$

$$recall(c'_i) = \frac{|relevance(c'_i)|}{|relevance(c_i)|} \quad (6.2)$$

Avec *relevance* la fonction de pertinence renvoyant les éléments pertinents (i.e. dont il existe une interaction dans L avec l'utilisateur en question) d'un ensemble de candidats :

$$relevance(c_i) = \{i'_j | i'_j \in c_i, \forall i'_j, (u_i, i'_j) \in L\}$$

Cependant les métriques de précision et de rappel ne prennent pas en compte l'ordre des candidats choisis par un algorithme de recommandation. En effet, il est courant qu'un algorithme de recommandation prédise que certains items candidats sont plus pertinents que d'autres. Pour un algorithme de recommandation *offline*, un score est attribué à chaque item candidat, il est donc possible de générer un ordonnancement r_i à partir de c_i . L'exploitation de ces informations permet une évaluation des algorithmes de recommandation plus précise et donc de mieux comparer différents algorithmes (Kumar et al., 2017). Afin d'exploiter ces ordonnancements,

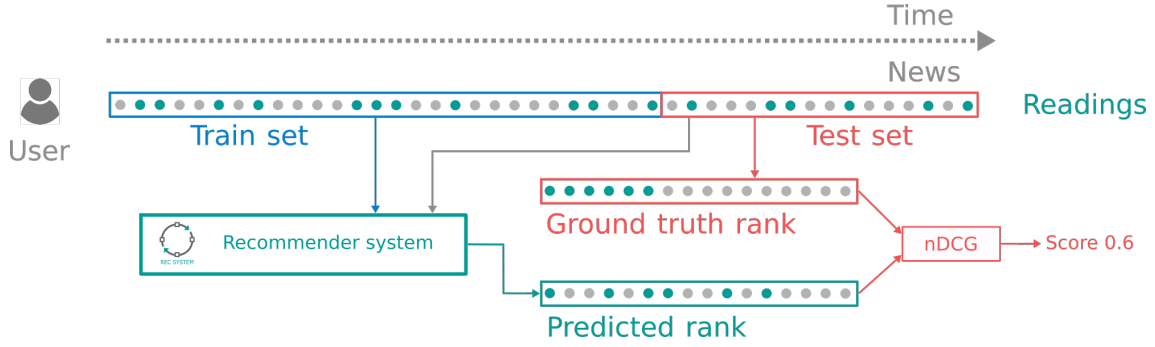


FIGURE 6.1 – Évaluation *offline* d'un système de recommandation

il est nécessaire d'utiliser des métriques mesurant la qualité d'un ordonnancement telles que $nDCG$:

$$nDCG(grel^i) = \frac{DCG(grel^i)}{iDCG(grel^i)} \quad (6.3)$$

$$DCG(grel^i) = \sum_{j=1}^k \frac{2^{grel_j^i} - 1}{\log_2(j + 1)} \quad (6.4)$$

$$iDCG(grel^i) = \sum_{j=1}^{\text{sum}(grel^i)} \frac{1}{\log_2(i + 1)} \quad (6.5)$$

Avec $grel^i$ le vecteur de pertinence graduée de r_i , i.e. un vecteur de même taille que r_i composé de plusieurs 0 et 1 :

$$\forall j \in \{1, \dots, k\}, grel_j^i = 1 \Leftrightarrow \exists i'_j \in r_i, (u_i, i'_j) \in L$$

$$\forall j \in \{1, \dots, k\}, grel_j^i = 0 \Leftrightarrow \nexists i'_j \in r_i, (u_i, i'_j) \in L$$

La métrique $nDCG@k$ est une variante de $nDCG$ qui ne prend en compte que les items jusqu'au k -ième item. Cette coupure des listes de recommandations permet de comparer différents systèmes renvoyant des listes de recommandations de longueurs différentes. Dans nos expérimentations, nous utiliserons $nDCG$ sur tous les items des ensembles de candidats, ce qui est équivalent à utiliser $nDCG@k$ avec $k = 1000$. En conséquence, dans notre cas, les sous-ensembles de C' sont de même taille que les ensembles de C :

$$\forall c'_i \in C', |c'_i| = t = k$$

À noter que lorsque plusieurs algorithmes de recommandation sont évalués et comparés, il est important d'utiliser les mêmes ensembles de candidats C pour que les scores soient comparables.

La figure 6.1 illustre l'évaluation *offline* d'un algorithme de recommandation. Les articles d'actualité sont rangés de gauche à droite selon leur date de publication. Sur cette figure, nous représentons un seul utilisateur et ses lectures. Les lectures sont les points verts. Un item gris est un item n'ayant pas été lu par l'utilisateur et est donc considéré comme non pertinent. L'ensemble des items est séparé en un groupe bleu (*Train set*) et un groupe rouge (*Test set*) par une date pivot.

Le système de recommandation en bas à gauche de la figure doit prendre les items ayant une date postérieure à la date pivot et produire un ordonnancement selon ses observations dans l'historique de lecture de l'utilisateur. L'ordonnancement produit par le système est indiqué dans le groupe vert avec l'indication "*Predicted rank*". L'ordonnancement idéal est représenté dans un groupe rouge avec l'indication "*Ground truth rank*". Le calcul du $nDCG$ pour cet utilisateur consiste donc à comparer ces deux ordonnancements sachant les items pertinents. Pour calculer le $nDCG$ final sur l'ensemble des utilisateurs, il suffit alors de faire une moyenne de tous les $nDCG$. À noter que si deux items ont la même date dans le jeu de données, il est possible de considérer que l'item antérieur est celui ayant l'index le plus petit dans la liste des items, les items de même date étant placés dans un ordre aléatoire entre eux dans la liste.

6.3 Algorithme générique de recommandation exploitant la représentation vectorielle des items

Dans cette section, nous présentons un algorithme générique de recommandation par similarité vectorielle nous permettant d'exploiter tout modèle de représentation du texte. Ce type de modèle prend en entrée un document composé d'une suite de mots et produit en sortie un vecteur *dense* (e.g. pour les modèles de *topic modeling*) ou *sparse* (e.g. pour la pondération TFIDF). L'algorithme présenté généralise l'utilisation de vecteurs représentatifs pour la tâche de recommandation, il est donc à distinguer des modèles ad hoc tels que *BM25* qui permet de calculer des similarités. Nous rappelons que le but d'un système de recommandation est d'ordonner un ensemble de candidats sachant l'historique de lecture d'un utilisateur plus ou moins grand avant la date pivot. L'algorithme consiste à calculer la similarité entre les candidats et l'historique de l'utilisateur sachant leur représentation vectorielle respective.

L'algorithme proposé utilise les vecteurs représentatifs des articles d'actualité historiques et des articles candidats. Il cherche les vecteurs représentatifs des articles candidats les plus similaires aux vecteurs de l'historique de l'utilisateur afin d'effectuer une recommandation. La méthode la plus simple permettant de déterminer la pertinence d'un candidat est de calculer la moyenne des similarités entre le candidat et l'ensemble des articles historiques (Capelle et al., 2012). L'algorithme consiste alors à construire un modèle utilisateur général à partir de l'historique de lectures et à comparer ce modèle à chaque candidat. Plus un article candidat est proche du modèle utilisateur, plus l'algorithme le considérera pertinent et le placera dans les premières positions dans son ordonnancement.

Cependant, l'algorithme que nous proposons se fonde sur l'hypothèse que les articles les plus pertinents à une recommandation le sont au regard d'un sous-ensemble des articles historiques. Nous appelons cette hypothèse l'*hypothèse de proximité partielle* entre un article candidat et des articles historiques. Plus précisément, cela signifie qu'un candidat sera pertinent non pas par similarité avec tous les articles lus par l'utilisateur dans le passé mais uniquement par similarité avec un sous-ensemble de ceux-ci. L'intuition est que les caractéristiques qui sont d'intérêt pour l'utilisateur se retrouveront dans une partie de ses lectures passées, mais pas toutes.

Différentes caractéristiques peuvent être extraites d'un article et correspondre

aux intérêts d'un utilisateur au regard des articles lus dans le passé : le style, le thème, les entités nommées, la localisation, etc. Il n'est pas possible de déterminer avec certitude quelles seront les caractéristiques clés définissant les intérêts d'un utilisateur, ni d'extraire exhaustivement toutes les caractéristiques jouant un rôle dans ses intérêts : ce sont des variables latentes. Cependant, sachant un article candidat, il est possible de connaître la similarité de ce candidat avec l'historique de lectures de l'utilisateur sur un axe choisi au préalable (e.g. le style, le thème), en supposant que l'on dispose d'un modèle capable d'extraire les caractéristiques voulues. Il s'agit généralement d'utiliser des modèles de représentation donnant une « approximation » de ces caractéristiques latentes sous la forme de vecteurs.

Imaginons un modèle d'extraction du thème (*topic modeling*), alors l'*hypothèse de proximité partielle* consiste à dire qu'un article est pertinent parce qu'il a un thème similaire à une partie des lectures passées de l'utilisateur, celui-ci ayant lu des articles sur différents thèmes. Si l'utilisateur a lu par le passé des articles sur les thèmes du « sport » et de la « politique » à proportions égales, alors un article candidat ayant comme thème le « sport » sera pertinent à une recommandation vis-à-vis de seulement la moitié des articles historiques. En effet, si l'on devait effectuer une similarité moyenne sur l'ensemble des articles historiques, alors ceux portant sur le thème de la politique feraient chuter la moyenne des similarités de cet article candidat. Afin de valider l'*hypothèse de proximité partielle*, nous introduisons le paramètre de *ratio historique* (abrégié r) dans notre algorithme. Celui-ci indique la proportion des items historiques qui doivent être pris en compte pour calculer la pertinence d'un candidat.

Plus formellement, nous restreignons la définition de la similarité moyenne entre un item candidat et un ensemble d'items historiques à la proportion des items historiques les plus similaires. Ainsi, pour un item c et un ensemble d'items historiques H , la similarité entre c et H est définie comme suit :

$$hrsim(H, c, r) = \frac{\sum_{i=1}^{\lceil r|H| \rceil} \text{sort}(\{s \mid \forall j \in \{1, \dots, |H|\}, s = \text{sim}(H_j, c)\})_i}{\lceil r|H| \rceil} \quad (6.6)$$

Le paramètre r a pour contrainte $r \in [0, 1]$. Il correspond à la proportion d'items les plus similaires dans l'historique qui permettent de calculer la similarité finale. La fonction *sort* trie le vecteur par ordre croissant. Les éléments de H ainsi que c peuvent se présenter sous la forme de vecteurs. Ils seraient alors de même taille, typiquement des vecteurs représentatifs selon un modèle de représentation donné. Dans ce cas, la fonction *sim* est une fonction de similarité standard que nous aurons choisie à l'avance telle que la similarité cosinus.

L'algorithme 5, appelé *SimRec*, montre le fonctionnement de la recommandation par similarité. Il prend en paramètres l'historique de tous les utilisateurs dans la variable *histories* ainsi que toutes les listes de candidats dans la variable *candidats*. La variable r correspond au *ratio historique*. L'algorithme génère l'ordonnancement des candidats pour chaque utilisateur à partir de la ligne 3. À la ligne 7, pour chaque candidat, nous calculons sa similarité avec l'historique tel que décrit par l'équation 6.6. Ensuite, nous trions les candidats par similarité à la ligne 9. Enfin nous retournons l'ensemble des ordonnancements à la ligne 11.

Dans la littérature, les algorithmes de recommandation basés sur le contenu fonctionnent de deux manières :

1. soit ils construisent un profil de l'utilisateur à partir de ses interactions

Algorithm 5 Algorithme générique de recommandation par similarité

```
1: procedure SimRec(histories, candidats, r)
2:   rankings  $\leftarrow$  empty list
3:   for u in  $\{1, \dots, |\textit{histories}|\}$  do
4:     H, C  $\leftarrow$  historiesu, candidatsu
5:     sims  $\leftarrow$  empty list
6:     for c in C do
7:       s  $\leftarrow$  hrsim(H, c, r)
8:       Append s to sims
9:     C'  $\leftarrow$  (c'1, ..., c'|C|) such that  $\forall c' \in C', c' \in C$  and  $\forall c'_i \in C', \forall c'_j \in C', i <$   

10:    j  $\iff$  simsi  $\geq$  simsj
11:     Append C' to rankings
12:   return rankings
13: end procedure
```

avec le système, généralement à travers l'historique des items avec lesquels l'utilisateur a interagi, puis utilisent ce profil pour calculer la pertinence des items candidats à une recommandation via une fonction de similarité (Capelle et al., 2012) ;

2. soit ils entraînent un modèle à calculer un score de pertinence à partir d'un profil utilisateur ou d'un sous-ensemble fixe de l'historique via des méthodes d'apprentissage automatique (Wang et al., 2018; Garcin et al., 2013).

À notre connaissance, aucune étude dans le domaine des systèmes de recommandation ne propose d'algorithme permettant la considération d'un sous-ensemble variable de l'historique des utilisateurs. Les travaux relatifs assemblent tous les items historiques pour la construction du profil de l'utilisateur (Capelle et al., 2012), ou une portion fixe (Wang et al., 2018). L'introduction de la proximité partielle permet de contextualiser le calcul de pertinence d'un candidat en ne considérant qu'une partie des items historiques. Elle permet d'ignorer les items qui n'apportent aucune information sur les intérêts de l'utilisateur vis-à-vis du candidat. Un autre avantage à l'algorithme *SimRec* est qu'il permet de comparer différents modèles de représentation de façon efficiente. Il ne nécessite pas l'entraînement d'un modèle mais exploite les représentations vectorielles des items et une fonction de similarité. Au chapitre 8, nous validons l'intérêt de la proximité partielle par une étude approfondie.

6.4 Exploitation de la complémentarité des représentations par hybridation des ordonnancements

6.4.1 L'algorithme d'hybridation *Reswhy*

Afin d'observer l'impact de chaque représentation textuelle sur la recommandation d'articles d'actualité, et notamment leur complémentarité et leur capacité à couvrir les préférences des utilisateurs dans leur lecture de l'actualité, nous proposons un nouvel algorithme d'hybridation des ordonnancements. Celui-ci nous permettra d'effectuer

l'hybridation des ordonnancements de *SimRec* générés grâce à différents modèles de représentation. Dans la section 8.4.1, nous reviendrons plus en détail sur les motivations à l'utilisation de ce type d'algorithme dans le cadre de cette thèse.

Cet algorithme, appelé *Reswhy* (*Rescaled and Weighed Hybridization*), consiste à effectuer une hybridation de listes de recommandations avec pondération et rééchelonnage. La pondération permet d'avantager un ordonnancement par rapport aux autres alors que le rééchelonnage permet d'avantager ou désavantager les articles les plus pertinents via une transformation par une fonction monotone paramétrée. Pour un item, l'avantage ou le désavantage qu'il obtient est donc dépendant de sa position, alors que la pondération avantage ou désavantage de la même manière l'ensemble des items quelle que soit leur position. Nous détaillons le rééchelonnage ainsi que ses motivations dans les sections suivantes. Comparée à l'hybridation par combinaison de descripteurs mentionnée en section 5.2.4, l'hybridation par pondération permet de régler le poids de différents algorithmes de recommandation et ainsi avantager plus ou moins les critères de précision ou qualitatifs sur lesquels ces algorithmes sont optimaux.

Cet algorithme est basé sur la méthode d'hybridation par pondération proposée notamment par [Claypool et al. \(1999\)](#) qui consiste à faire une moyenne pondérée des scores afin de déterminer la position finale de chaque item. La recommandation hybride par pondération est la méthode d'hybridation la plus simple à implémenter car elle consiste en la combinaison de plusieurs listes de recommandations. Une combinaison linéaire des positions des items ou du score des items (e.g. similarité, distance, pertinence) permet de réordonner les items ([Burke, 2007](#)). Elle est plus simple que les autres méthodes d'hybridation (e.g. par combinaison des descripteurs, cascade d'ordonnements). Nous donnons plus d'informations sur les différentes techniques d'hybridation en section 5.2.4 de l'état de l'art de cette partie (chapitre 5). La recommandation hybride par pondération a pour avantage de ne nécessiter que les sorties des algorithmes de recommandation à combiner. Les algorithmes n'ont pas besoin d'être adaptés.

Reswhy consiste à réordonner des candidats selon différents ordonnancements originels². Il est nécessaire que ces ordonnancements soient de même taille et contiennent les mêmes items, ce qui impose aux modèles de recommandation de donner un score de pertinence et une position à tous les candidats, sans omission. Les ordonnancements comprennent les mêmes items mais ceux-ci ont, bien entendu, des positions différentes. Un item aura donc un nombre de positions égal au nombre d'ordonnements que l'on doit combiner. Ces prérequis sont cohérents avec le processus de découpage décrit en section 6.2. En effet, les listes de candidats se réduisent à un sous-ensemble de tous les items postérieurs à la date pivot avec, par exemple, $k = 1000$. Pour chaque utilisateur, nous avons les items pertinents ainsi que d'autres pris au hasard après la date pivot jusqu'à atteindre un nombre de 1000 candidats.

Reswhy consiste en quatre étapes principales :

1. Normalisation des ordonnancements originels en considérant que les scores des items sont soit leur position dans l'ordonnement, soit leur score de pertinence (e.g. une distance, une similarité) donné par l'algorithme de recom-

2. Dans ce manuscrit, un « ordonnancement originel » est un ordonnancement issu d'un algorithme de recommandation qui est destiné à être combiné avec d'autres par une hybridation. L'ordonnement résultant est alors dit « combiné ».

mandation ayant produit l'ordonnement.

2. Rééchelonnage des ordonnements originels selon les paramètres α et β attribués à chaque ordonnement.
3. Calcul des nouveaux scores de pertinence des items suivant une pondération donnée.
4. Tri des items selon leur nouveau score de pertinence.

Nous définissons la sortie d'un algorithme de recommandation comme un ensemble d'ordonnements pour chaque utilisateur (dans le découpage de test comprenant les articles candidats). Ainsi, la combinaison de plusieurs algorithmes de recommandation consiste à effectuer une hybridation de tous des ordonnements des algorithmes de recommandation pour chaque utilisateur. L'évaluation de l'hybridation consiste ensuite en la même procédure que pour l'évaluation des algorithmes de recommandation seuls, i.e. nous calculons la moyenne des scores selon les différentes métriques d'ordonnement.

Chaque ordonnement est associé à un utilisateur et se compose d'un ensemble ordonné d'items I représentés par leur identifiant ainsi que d'un ensemble ordonné S de scores de pertinence. Dans notre implémentation, l'ensemble S est facultatif. Si S est omis, alors les scores des items seront leur position dans l'ordonnement. Les identifiants doivent être tous différents et $\forall i \in I, i \in \{1, \dots, |I|\}$, i.e. les identifiants commencent à 1 et terminent à un identifiant correspondant au nombre de candidats considérés. Les identifiants les plus proches de la première position dans l'ensemble ordonné sont ceux considérés comme les plus pertinents par l'algorithme ayant généré l'ordonnement en question. Tous les ordonnements doivent être de même taille, avec, par exemple, $k = 1\,000$.

L'algorithme 6 donne le pseudo code de *Reswhy*. Celui-ci combine les ordonnements de plusieurs algorithmes de recommandation pour un unique utilisateur. Le paramètre *rankings* est composé de plusieurs *tuples*, chaque *tuple* correspond à l'ordonnement originel d'un algorithme de recommandation et est composé des ensembles ordonnés I et S . L'algorithme *Reswhy* retourne un nouvel ensemble ordonné I' correspondant à la combinaison de tous les ensembles ordonnés I donnés dans le paramètre *rankings*.

À la ligne 3, la boucle permet de faire une normalisation, un rééchelonnage et une pondération de tous les ordonnements dans *rankings*. Cet algorithme prend un nombre d'ordonnements variable, i.e. il peut y avoir deux ordonnements ou plus dans *rankings*. Les paramètres de normalisation, rééchelonnage et pondération sont donc multiples, leur nombre est égal à la taille de *rankings*. Ainsi, à la ligne 4, nous prenons les paramètres de l'ordonnement courant. Ce sont les paramètres :

- α et β destinés à la fonction de rééchelonnage que nous décrirons par la suite ;
- rankAsScore*** indiquant si les scores dans S doivent être ignorés et remplacés par la position des items ;
- w pondérant les scores d'un ordonnement par rapport aux autres.

Étant donné que nous ne faisons que des combinaisons par paire, chaque paramètre est au nombre de deux, i.e. il y a deux paramètres α , deux paramètres *rankAsScore* et

Algorithm 6 Algorithme d'hybridation par pondération et rééchelonnage

```
1: procedure RESWHY(rankings, rankAsScores, weights, alphas, betas)
2:    $C \leftarrow \emptyset$  ▷ Combined scores to be filled
3:   for  $p$  in  $\{1, \dots, |I|\}$  do ▷ For each ranking
4:      $rankAsScore, w, \alpha, \beta \leftarrow rankAsScores_p, weights_p, alpha_p, beta_p$ 
5:      $I, S \leftarrow rankings_p$  ▷ Getting items ids and scores
6:     if  $C = \emptyset$  then
7:        $C \leftarrow (0 \text{ for } i \text{ in } \{0, \dots, |I|\})$  ▷ Initializing  $C$  with zeros
8:     if rankAsScore or  $S$  is empty then
9:        $S \leftarrow (i/|I| \text{ for } i \text{ in } \{0, \dots, |I| - 1\})$  ▷ Scores are normalized ranks
10:    if  $S_1 < S_{|S|}$  then
11:       $S \leftarrow (S_i \text{ for } i \text{ in } \{|I|, \dots, 1\})$  ▷ Inversion
12:       $S \leftarrow (\frac{S_i - \min(S)}{\max(S) - \min(S)} \text{ for } i \text{ in } \{1, \dots, |I|\})$  ▷ Min-Max normalization
13:       $S \leftarrow \text{gmr}(S, \alpha, \beta)$  ▷ Rescaling
14:      for  $i$  in  $\{1, \dots, |I|\}$  do
15:         $C_{I_i} \leftarrow C_{I_i} + w.S_i$  ▷ Weighting the score and adding it to  $C$ 
16:       $I' \leftarrow (i'_1, \dots, i'_{|C|})$  such that  $\forall i' \in I', i' \in \{1, \dots, |C|\}$  and  $\forall i'_u \in I', \forall i'_v \in I', u < v \iff C_{i'_u} \geq C_{i'_v}$  ▷ Sorting by weighted scores
17:      return  $I'$ 
18: end procedure
```

ainsi de suite. L'optimisation sur un découpage de validation³ peut donc être effectuée sur sept hyperparamètres, la paire de paramètres *weights* ne comptant en réalité que pour un seul hyperparamètre puisque ses composantes sont complémentaires : leur somme doit être égale à 1.

À la ligne 7, nous initialisons un ensemble ordonné destiné à contenir les nouveaux scores de chaque item. Ainsi, C_i désigne le score combiné de l'item ayant pour identifiant i . Celui-ci est composé de plusieurs 0 puisque l'algorithme additionne les nouveaux scores de chaque item à chaque itération.

Si le premier score dans S est inférieur au dernier, nous considérerons que ceux-ci sont des distances. Au contraire, si le premier score est supérieur, alors nous considérerons que ce sont des scores de similarité ou de pertinence. Afin de ne pas imposer que les scores soient croissants (ou décroissants), nous normalisons ceux-ci pour qu'ils soient toujours décroissants. Cette normalisation ne doit pas changer la position des items. Ainsi, quel que soit le fonctionnement de l'algorithme de recommandation ayant produit l'ordonnancement, nous pouvons utiliser ses scores, qu'ils soient strictement croissants ou strictement décroissants. Cette opération est effectuée à la ligne 11.

À la ligne 12, nous faisons une normalisation *Min-Max* des scores afin qu'ils soient entre 0 et 1. Cette normalisation est nécessaire pour l'utilisation de la fonction *gmr* (*General Monotonic Rescaling Function*) à la ligne 13 qui permet d'effectuer un rééchelonnage. À la ligne 15, nous ajoutons à C les scores des items en les pondérant avec le w courant. Enfin, à la ligne 16 nous effectuons un tri des items selon leurs

3. Un découpage de validation correspond à un ensemble d'items et de retours de pertinence dans un intervalle de temps différent (sans intersection) du jeu de test utilisé pour l'évaluation des algorithmes. Celui-ci permet, comme nous le verrons au chapitre 8, d'optimiser les hyperparamètres des algorithmes de recommandation.

nouveaux scores moyens pondérés et renvoyons l'ordonnancement combiné I' .

6.4.2 La dominance et l'explicitabilité d'une hybridation

Les modèles dans une combinaison ont des poids différents. De plus, les scores attribués aux ensembles de 1 000 candidats ne seront pas nécessairement linéaires. Ainsi, en traçant la courbe des scores en fonction de l'identifiant des items de 1 à 1 000, il est possible qu'une portion de la courbe prenne une forme logarithmique ou exponentielle. Cela signifie que, pour certains intervalles d'items dans l'ordonnancement, il peut exister une grande différence de scores entre deux items consécutifs par rapport à deux autres items consécutifs, ou au contraire, une différence très petite par rapport à deux autres items consécutifs. Prenons un cas simple : dans un ordonnancement de 1 000 candidats, les scores des 500 premiers items pourront par exemple être dans un intervalle entre 0.0 et 0.2 alors que les 500 suivants dans un intervalle entre 0.2 et 1.0, i.e. les scores suivent une courbe de forme exponentielle.

La pondération et la non-linéarité des scores auront pour conséquence qu'une hybridation pourra avantager un algorithme de recommandation aux dépens d'un autre. Ainsi, une combinaison, dans le pire des cas, pourra correspondre aux ordonnancements originels d'un des algorithmes et ne pas tirer parti des ordonnancements de l'autre. Afin d'analyser quantitativement cet effet, nous proposons la métrique de « *dominance* ».

La *dominance* est un score indiquant à quel point le premier algorithme d'une hybridation « impose » son ordonnancement propre des items par rapport au second algorithme. Il correspond à la probabilité qu'un item, dans l'ordonnancement combiné, soit positionné au plus proche de sa position dans l'ordonnancement originel du premier algorithme que de sa position dans l'ordonnancement originel du second algorithme.

Plus formellement, sachant R' l'ensemble des ordonnancements combinés de chaque utilisateur, R^1 et R^2 l'ensemble des ordonnancements originels de chaque utilisateur, la dominance de R^1 par rapport à R^2 est donnée par :

$$\text{domin}(R^1, R^2, R') = \frac{\sum_{u=1}^{|R'|} \sum_{i=1}^{|R'_u|} [abs(i - \text{id}(R'_{u,i}, R_u^1)) < abs(i - \text{id}(R'_{u,i}, R_u^2))]}{\sum_{u=1}^{|R'|} \sum_{i=1}^{|R'_u|} [abs(i - \text{id}(R'_{u,i}, R_u^1)) \neq abs(i - \text{id}(R'_{u,i}, R_u^2))]} \quad (6.7)$$

Dans l'équation 6.7, nous utilisons une notation conditionnelle par l'utilisation de caractères « crochet », ainsi $[c]$ prend la valeur 1 lorsque la condition c est vraie et 0 lorsqu'elle est fausse. Les variables R^1 , R^2 et R' sont des ensembles d'ordonnements. Ainsi, l'ordonnement d'un utilisateur u dans R' se note R'_u . L'item en position i dans cet ordonnancement se note $R'_{u,i}$. La fonction id donne l'indice d'un item dans un ordonnancement. Ainsi, $\text{id}(R'_{u,i}, R_u^1)$ renvoie la position (ou l'indice) de l'item $R'_{u,i}$ dans R_u^1 . Au dénominateur, nous ne faisons pas la somme de tous les items mais comptons le nombre de fois que l'item courant n'est pas à équidistance puisque, dans ce cas, aucun des deux ordonnancements n'est avantagé.

Le calcul de la dominance nécessite de parcourir l'ensemble des utilisateurs et leurs items candidats. Il nécessite également, pour chaque utilisateur, une indexation des items pour un accès en temps constant à leur position. C'est pourquoi nous avons été contraints de calculer une approximation de la dominance pour toutes les

Algorithm 7 Algorithme d'approximation de la dominance

```
1: procedure DOMAPPROX( $R^1, R^2, R', n$ )
2:    $dom \leftarrow 0$ 
3:    $total \leftarrow 0$ 
4:   for  $i$  in  $\{1, \dots, n\}$  do
5:      $u \leftarrow random(\{1, \dots, |R'|\})$ 
6:      $i \leftarrow random(\{1, \dots, |R'_u|\})$ 
7:      $index_1 \leftarrow id(R'_{u,i}, R_u^1)$ 
8:      $index_2 \leftarrow id(R'_{u,i}, R_u^2)$ 
9:      $d \leftarrow abs(i - index_1) - abs(i - index_2)$ 
10:    if  $d > 0$  then
11:       $dom = dom + 1$ 
12:       $total = total + 1$ 
13:    else if  $d < 0$  then
14:       $total = total + 1$ 
15:    return  $dom/total$ 
16: end procedure
```

hybridations proposées dans ce chapitre. Ainsi, l'algorithme 7 nous a permis de donner une valeur approximative de la dominance de deux ordonnancements. Il fonctionne en sélectionnant, à chaque itération, un utilisateur et un item aléatoirement. Il calcule ensuite les écarts de positions en ajoutant 1 la variable dom si l'item du premier ordonnancement est plus proche de sa position dans l'ordonnancement combiné. Cet algorithme effectue un nombre d'itérations correspondant au paramètre n que nous avons fixé à 1 000 000 pour cette expérimentation.

L'algorithme 7 ainsi que l'équation 6.7 peuvent être adaptés au calcul de la dominance d'une combinaison de plus de deux ordonnancements. Cela consisterait alors à compter, pour chaque ordonnancement, le nombre de fois qu'un item est plus proche de sa position dans l'ordonnancement combiné. Il s'agirait donc de calculer plusieurs dominances. Cependant, dans ce chapitre, nous avons combiné uniquement des paires d'ordonnements, nous ne donnons donc que cette version simplifiée.

La dominance est influencée par le paramètre *weights* dans l'algorithme *Reswhy* puisque plus l'un des ordonnancements aura de poids dans l'hybridation, plus la dominance s'éloignera de la valeur neutre 0.5. De plus, dans une hybridation, plus la répartition des scores d'un ordonnancement suit une fonction gaussienne centrée, plus l'ordonnancement sera avantagé par rapport à l'autre (i.e. obtiendra une plus grande dominance). Cette observation est systématique dans les 156 combinaisons que nous générerons au chapitre 8.

Afin de montrer l'influence de la répartition des scores des candidats (en termes de similarité à l'historique de lectures des utilisateurs), nous proposons d'analyser les ordonnancements de quelques modèles qui seront utilisés dans le chapitre 8. La figure 6.2 illustre l'influence de la répartition en donnant trois exemples d'hybridations. Les ordonnancements ont tous des poids de 0.5, la pondération n'aura donc pas d'influence. Les graphiques en bleu correspondent aux histogrammes de la répartition des scores (convertis en distances à l'historique). Nous avons donc collecté les scores des candidats de chaque utilisateur. Chaque score est normalisé entre 0 et 1. Les abscisses de ces histogrammes correspondent aux valeurs de distance à l'historique.

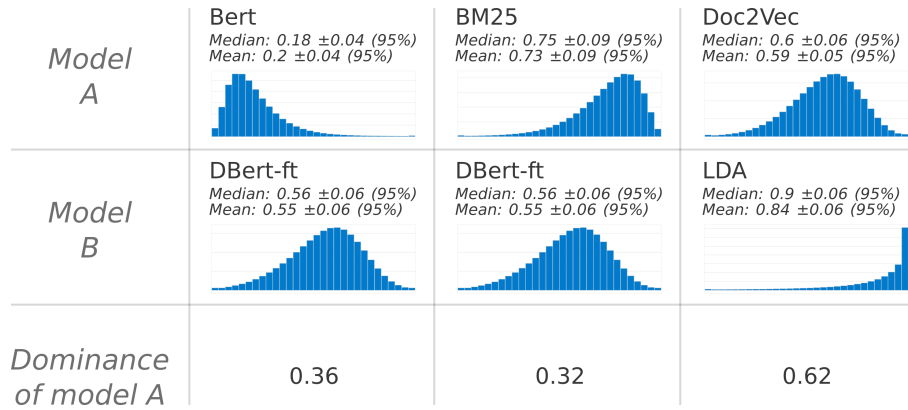


FIGURE 6.2 – Dominance dans trois exemples d’hybridation

La hauteur des barres correspond au nombre d’items candidats ayant le score donné en abscisse (dans un intervalle restreint).

Pour chaque ordonnancement, la taille de l’échantillon permettant l’affichage de ces histogrammes correspond au nombre de candidats (1 000) multiplié par le nombre d’utilisateurs (21 239). Nous décrirons plus en détail les données utilisées au chapitre 7. Avec ces histogrammes, nous donnons également la moyenne des scores ainsi que la médiane et leur écart type. Les écarts types sont calculés sur l’ensemble des moyennes et des médianes des utilisateurs. Nous donnons ensuite la dominance de l’hybridation des paires d’ordonnements effectués grâce à l’algorithme *Reswhy*. À noter que, dans l’algorithme *Reswhy*, si l’on considère les positions comme étant les scores pour chaque item (paramètre *rankAsScores*), alors les dominances deviennent neutres (i.e. autour de 0.5). Autrement dit, le calcul de la dominance n’est pertinent que dans le cas où nous tenons compte des scores des items.

Comme le montre la figure 6.2, l’histogramme de *DBert-ft* est centré. Les ordonnancements de *DBert-ft*, dans les deux premières colonnes, sont avantagés. En effet, les modèles *Bert* et *BM25* qui sont premiers dans l’hybridation, obtiennent une dominance de respectivement 0.36 et 0.32. Pour la troisième colonne, l’ordonnement de *Doc2Vec* obtient une dominance de 0.62 lorsque combiné avec le modèle *LDA* possédant un histogramme décentré vers la droite.

Pour résumer, la dominance permet de mesurer l’avantage d’un ordonnancement par rapport à un autre dans une hybridation. Elle donne une explication sur un ordonnancement combiné, i.e. une liste de recommandations qui est donnée à un utilisateur. Ainsi, si par exemple un ordonnancement basé sur le thème a une grande dominance dans une combinaison avec un ordonnancement basé sur le style, alors il sera possible d’informer l’utilisateur que la liste de recommandations prend en compte, en grande partie, ses préférences thématiques, mais également un peu de ses préférences stylistiques. La dominance est à la fois influencée par :

1. le poids que l’on attribue à chaque ordonnancement (i.e. paramètre *weights*) ;
2. les variations des différences entre les scores de la suite d’items dans l’ordonnement (ce qui implique une répartition des scores variable comme l’illustre les quelques exemples de la figure 6.2).

La méthode de rééchelonnage permet de modifier cette variation. Elle a un impact sur la dominance, et donc sur la performance d’une hybridation comme nous le

Algorithm 8 *General Monotonic Rescaling Function*

```
1: procedure GMRF( $Y, \alpha, \beta$ )
2:   if  $\beta$  then
3:     if  $\alpha < 0.5$  then
4:        $Y = Y^{2\alpha} + 1$  ▷ Blue curve
5:     else
6:        $Y = -Y^{\frac{1}{2(|\alpha-1|+0.5)-1}} + 1$  ▷ Red curve
7:     else
8:       if  $\alpha < 0.5$  then
9:          $Y = (1 - Y)^{\frac{1}{2\alpha}}$  ▷ Green curve
10:      else
11:         $Y = (1 - Y)^{2(|\alpha-1|+0.5)-1}$  ▷ Purple curve
12:      return  $Y$ 
13: end procedure
```

démontrerons au chapitre 8 en section 8.4.2.

6.4.3 Le rééchelonnage des ordonnancements

Le rééchelonnage intervient à la ligne 13 de *Reswhy* (algorithme 6). Il permet de contrôler la dominance dans une hybridation. Les paramètres α et β changent les scores et donc la combinaison des items. Ces paramètres peuvent être optimisés sur un découpage de validation pour chaque combinaison. Ainsi, la modification de la dominance par l’optimisation des paramètres α et β peut permettre d’obtenir une amélioration selon les métriques de précision d’ordonnement telles que nDCG. À noter que si l’on utilise les positions et que l’on ignore les scores de chaque item (paramètre *rankAsScores*), alors le rééchelonnage permet aussi de modifier la dominance lorsque celle-ci est par défaut neutre dans ce cas (proche de 0.5). En effet, les scores de « position » sont linéaires de 0 à 1, mais le rééchelonnage peut rendre ceux-ci non linéaires, pouvant amener à l’amélioration des performances d’une hybridation.

La fonction *gmrif* permet d’effectuer ce rééchelonnage. Elle prend une liste de valeurs et retourne une nouvelle liste de valeurs avec une « courbure » différente. Chaque valeur doit appartenir à l’intervalle $[0, 1]$. Cette fonction est composée de quatre sous-fonctions qui seront utilisées selon les paramètres α et β donnés en entrée. L’algorithme 8 donne le pseudo-code de *gmrif*. La figure 6.3 montre les sous-fonctions de *gmrif* selon différentes valeurs du paramètre α . Les couleurs sur cette figure correspondent aux couleurs données dans l’algorithme 8. À noter que dans notre implémentation, nous détectons automatiquement le sens des valeurs données en paramètre (croissant ou décroissant) afin d’adapter le rééchelonnage pour qu’il corresponde à la même pente.

La figure 6.4 donne deux exemples de rééchelonnage des ordonnancements de *LDA*. Ces courbes correspondent à la moyenne des rééchelonnages des ordonnancements de *LDA* pour tous les utilisateurs. En abscisses, nous donnons les 1 000 positions des items. L’ordonnée correspond aux scores moyens obtenus à chaque position. À noter que les items sont différents dans chaque ordonnancement puisque sont les candidats de différents utilisateurs.

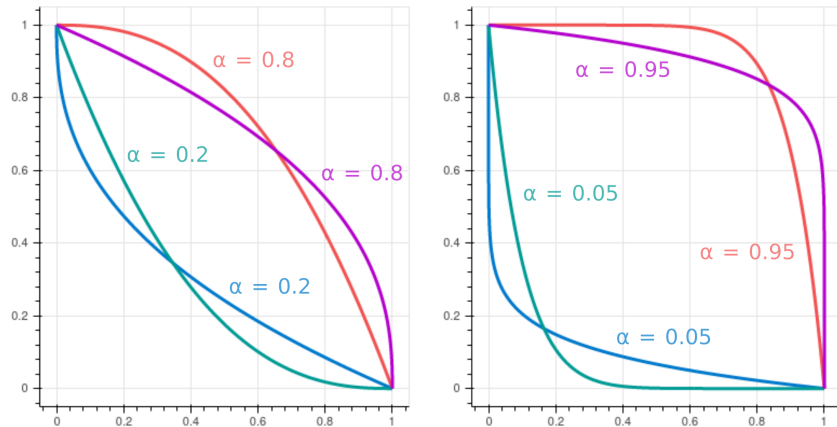


FIGURE 6.3 – Les quatre sous-fonctions de *gmrf* pour différentes valeurs de α

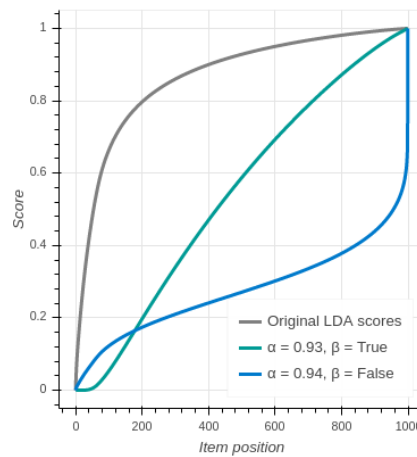


FIGURE 6.4 – Rééchelonnage des ordonnancements de *LDA*

Sur cette figure, nous avons fait varier les paramètres α et β . Ainsi, la courbe verte, correspondant aux affectations $\alpha = 0.93$ et $\beta = true$, permet de centrer la courbe des scores. Les scores ont donc une variation relativement constante : i.e. la courbe devient presque linéaire. Au contraire, les scores originaux en gris sont presque tous égaux à partir de la position 200. La courbe bleue correspond au rééchelonnage (affectations $\alpha = 0.94$ et $\beta = false$) des scores originaux de sorte que la courbure soit inversée, i.e. les scores des items aux premières positions sont plus proches et les scores aux dernières positions sont plus distants. L'avantage de la fonction *gmrf*, comme nous pouvons le voir à travers cette démonstration, est qu'elle permet d'obtenir de nombreuses courbures différentes à partir de diverses listes de scores originaux par l'affectation de seulement deux paramètres : α et β . Elle permet donc une optimisation efficace de la manière de combiner deux ordonnancements.

6.5 Conclusion

Dans ce chapitre, nous avons proposé :

1. L'algorithme *SimRec* permettant d'utiliser tout modèle de représentation

d'item pour effectuer l'ordonnement de candidats à une recommandation sachant l'historique de consommation d'un utilisateur. Cet algorithme se fonde sur l'*hypothèse de proximité partielle* que nous validerons au chapitre 8.

2. Une nouvelle mesure appelée « dominance » permettant de connaître l'importance d'un modèle dans une hybridation. Elle permet donc d'expliquer à l'utilisateur le poids des informations (ou méthodes) utilisées lors d'une recommandation.
3. L'algorithme de pondération *Reswhy* qui introduit le rééchelonnage des ordonnements pour contrôler la répartition des scores générés grâce à un modèle, et donc sa dominance.

Avant d'exploiter ces algorithmes au chapitre 8, nous introduisons le jeu de données *TwineWS* dédié à la recommandation d'articles d'actualité au chapitre 7.

Chapitre 7

Récolte d'un jeu de données large pour la recommandation d'articles d'actualité

7.1 Introduction

Beaucoup d'internautes utilisent les plateformes telles que Twitter et Facebook pour communiquer, s'exprimer et partager du contenu. Ce contenu partagé peut prendre la forme de photos, vidéos ou liens redirigeant vers des pages spécifiques d'articles d'actualité, de vidéos ou d'articles de blog. Les articles partagés par les utilisateurs sur les réseaux sociaux sont souvent des articles qu'ils trouvent pertinents vis-à-vis de leurs intérêts personnels. Dans ce chapitre, nous proposons d'exploiter ce type de données pour simuler, en condition *offline*, l'évaluation d'un système de recommandation d'articles d'actualité et de blog.

Nous introduisons un nouveau jeu de données appelé « *TwineWS* » spécialement récolté pour cette thèse¹. Ce jeu de données est composé de profils Twitter, de leurs *tweets*² ainsi que des articles d'actualité qu'ils ont partagés dans un historique de cinq mois d'octobre 2017 à février 2018. *TwineWS* est particulièrement intéressant pour la recherche menée durant cette thèse car permet de tester le modèle de représentation de style entraîné dans le chapitre précédent et ainsi répondre aux questions de recherche présentées en introduction de ce manuscrit. Il permet également de tester des modèles de représentation des utilisateurs cherchant à capturer des informations dynamiques telles que les intérêts des utilisateurs à travers les articles d'actualité qu'ils partagent dans leurs *tweets*.

Les expérimentations possibles grâce au jeu de données *TwineWS* sur la recommandation d'articles d'actualité permettront une validation et un enrichissement de représentations des utilisateurs pour d'autres services utilisant le même type d'informations dynamiques. De plus, ce jeu de données est mis à disposition à la communauté du domaine de recherche des systèmes de recommandation et comble un manque en termes de disponibilité des données en quantité et en qualité, que ce soit au niveau des articles d'actualité ou au niveau des utilisateurs et de leurs *tweets*.

Dans ce chapitre, nous décrivons tout d'abord le jeu de données *TwineWS* et le

1. Le jeu de données *TwineWS* est disponible à l'adresse <https://github.com/hayj/TwineWS>

2. Les *tweets* sont les commentaires courts que les utilisateurs postent sur la plateforme Twitter. Ils peuvent contenir des liens vers d'autres pages web.

processus qui nous a permis de récolter les profils Twitter et les articles d'actualité. Nous donnons également des pistes de réflexion sur la pertinence et les biais inhérents à ce type d'évaluation.

7.2 Le jeu de données *TwineWS*

Le jeu de données *TwineWS* se compose, tout d'abord, d'un ensemble de profils utilisateurs du réseau social Twitter. Les profils sont constitués d'un ensemble de « *tweets* ». Nous avons récolté les profils de 200 000 utilisateurs en parcourant le graphe Twitter et en suivant une heuristique prédéfinie que nous détaillerons. *TwineWS* se compose également des articles d'actualité que partagent les utilisateurs dans leurs *tweets*. Nous décrivons tout d'abord la récolte des articles d'actualité se trouvant dans les *tweets* des utilisateurs du jeu de données, puis des utilisateurs Twitter.

7.2.1 Collecte des articles d'actualité

Lorsqu'un lien pointe vers un article d'actualité dans un *tweet*, nous en récoltons le contenu, le titre et d'autres attributs tels que l'auteur et le nom de domaine. La récolte de l'ensemble des articles d'actualité de *TwineWS* a été implémentée en *Python* avec une bibliothèque développée durant cette thèse³ dépendante des bibliothèques *Selenium* et *requests*. Nous avons récolté environ un million d'articles d'actualité au total. L'extraction du contenu des articles a été implémentée avec l'aide de la librairie *Newspaper3k*.

La figure 7.1 donne un exemple des liens possibles entre les utilisateurs, leurs *tweets* et les articles d'actualité. Il est à noter que plusieurs *tweets* peuvent pointer vers le même article d'actualité. Un *tweet* peut également pointer vers un autre profil utilisateur lorsque celui-ci est mentionné. Il peut aussi n'être composé que de texte et ne pointer vers aucun article ni aucun profil. Dans ce cas, le tweet est tout de même conservé dans *TwineWS* car peut contenir des indices sur les intérêts de l'utilisateur.

Une des difficultés de cette étape de récolte des articles d'actualité est de s'assurer qu'un lien vers une page web pointe effectivement vers un article d'actualité et non une page web quelconque. Dans ce but, nous avons créé une base de données de noms de domaines correspondant à des sites web de journaux et de blog. Ces noms de domaines ont été collectés à la main puis complétés grâce à d'autres bases de données, et notamment celle fournie par *Google News*. La seconde étape est la validation des *urls* présentes dans les *tweets* : nous avons implémenté une méthode qui, à partir de l'*url*, nous indiquait si celle-ci correspondait à un nom de domaine appartenant à la base de données. La difficulté se situe dans le découpage des *urls* en ces différentes parties :

1. Le *scheme* contenant le protocole (souvent *http* ou *https* avec les caractères " ://").
2. Le *host* avec :

3. La bibliothèque de récolte développée durant cette thèse est disponible à l'adresse <https://github.com/hayj/WebCrawler>.

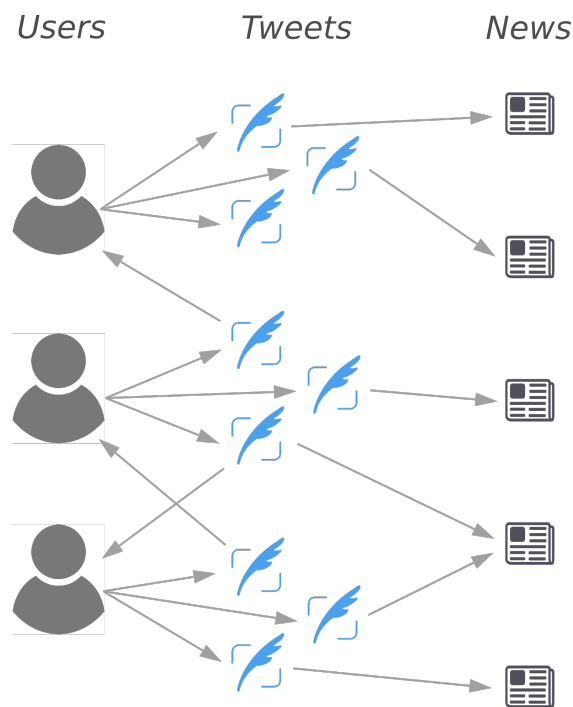


FIGURE 7.1 – Jeu de données *Twinews*

- (a) les sous-domaines⁴ ;
- (b) le nom de domaine principal.

3. Le reste de l'*url* redirigeant vers une page spécifique et pouvant contenir des paramètres et des ancres.

La figure 7.2 donne un exemple d'*url* et ses différentes parties. Il est très simple et courant dans la plupart des langages de programmation, de découper ces différentes parties avec des bibliothèques standards telles qu'*urllib*⁵ en *Python*. La séparation des sous-domaines et du domaine principal est cependant plus délicate, il est généralement nécessaire d'utiliser des bases de données de suffixes publics⁶ telles que *Public Suffix List*⁷ que nous avons utilisée. Prenons comme exemple les *urls* suivantes (pouvant parfois ne pas contenir le *scheme* lorsqu'elles sont extraites de *tweets*) :

bob.blogspot.co.uk/p1 ld.journal.co.uk/p1 http ://www.times.com/p1

À partir de ces *urls* nous pouvons facilement extraire la localisation réseau (*host* ou encore *netloc*) :

bob.blogspot.co.uk sport.journal.co.uk www.times.com

4. Un sous-domaine est subordonné à un domaine et permet de structurer un site web en définissant différentes localisations réseau.

5. La bibliothèque *urllib* est un module standard de *Python* permettant le découpage des *urls* en différentes composantes.

6. Un suffixe public est un suffixe sous lequel un internaute peut enregistrer un nom de domaine. Ce suffixe peut être attaché à un état (e.g. *fr*, *it*, *de*) ou appartenir à un fournisseur (e.g. *blogspot.com*)

7. *Public Suffix List*, disponible à l'adresse <https://publicsuffix.org>, est une liste de tous les suffixes publics connus.

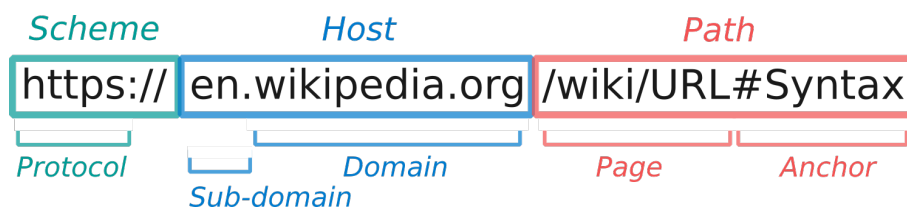


FIGURE 7.2 – Exemple d'*url* et ses composantes

Nous voulons ensuite éliminer les sous-domaines de chaque *url* afin de pouvoir tester leur existence dans notre base de données de domaines correspondant à des sites de blog et de journaux. Pour la première *url*, l'utilisateur *bob* du fournisseur *blogspot* enregistré sous le domaine de premier niveau⁸ *co.uk* n'a pas créé de sous-domaine. Le second exemple utilise le sous-domaine *sport* et le troisième exemple utilise le sous-domaine *www*. Nous voulons donc obtenir le résultat suivant :

bob.blogspot.co.uk journal.co.uk times.com

Or, un découpage naïf à partir de l'avant-dernier label ne permettrait d'obtenir un résultat correct que pour le troisième exemple. C'est pourquoi il est nécessaire de disposer d'une liste de suffixes publics. Nous utilisons *Public Suffix List* pour identifier la partie « suffixe public », puis nous prenons le label précédent en supplément et nous obtenons le nom de domaine correspondant réellement au blog ou au journal en ligne. Nous pouvons ainsi valider ce nom de domaine en testant son existence dans notre base de données de nom de domaine de blog et journaux, et ainsi identifier si l'*url* pointe sur un article d'actualité ou non.

Les articles d'actualité ont été prétraités par la méthode décrite au chapitre 3 en section 3.4. De même, nous avons filtré les articles dont le contenu n'est pas jugé de qualité par le modèle *Linear Support Vector Classification* entraîné sur une annotation manuelle que nous avons introduit en section 3.3. Pour ce jeu de données, nous ne conservons que les articles de langue anglaise. Nous avons donc choisi de nous restreindre à une zone géographique : les États-Unis. Cette restriction nous permet d'obtenir, avec grande probabilité, des articles d'actualité en langue anglaise et dont les sujets abordés et les entités nommées sont proches. De plus, cette localisation nous permet de simuler un système de recommandation qui serait localisé aux États-Unis et dont les articles candidats disponibles portent sur des sujets de même localité. Les modèles entraînés à recommander des articles peuvent ainsi capturer des similarités entre utilisateurs lorsque ceux-ci lisent des articles de thèmes proches. D'un côté, la similarité entre les utilisateurs peut permettre de capturer des caractéristiques communes, réduisant ainsi la complexité d'apprentissage, mais d'un autre côté, il pourra être difficile de recommander les articles pertinents lorsque d'autres articles proches seront candidats.

À noter que deux liens peuvent pointer vers le même article. C'est par exemple le cas des liens qui ne diffèrent que très légèrement (e.g. dans leurs fragments ou paramètres) mais pointant vers la même page. De plus, il peut exister des copies

8. Un domaine de premier niveau (ou extension) correspond généralement au dernier label du nom de domaine d'une *url*. Les domaines de premier niveau sont gérés par une organisation qui alloue leurs sous-domaines avec contrepartie commerciale ou non. Les domaines de premier niveau sont notamment composés des domaines de premier niveau nationaux tels que *fr*, *it* et *de*.

d'articles sur différents sites web. Tous ces cas sont détectés à l'étape d'extraction du contenu des articles. Pour tous les doublons, nous choisissons de ne conserver qu'une *url* et remplaçons toutes les occurrences des autres *urls* du même article dans l'ensemble des *tweets*.

7.2.2 Prise en compte des *urls* réduites

Les profils de *Twineews* sont très largement composés de liens ayant été réduits par des services de réduction d'*url* tels que *bit.ly*, *fb.me*, *ow.ly* et *buff.ly*. Les utilisateurs de Twitter utilisent ces services afin de réduire le nombre de caractères total composant leurs *tweets*. La longueur des *tweets* étant limitée par Twitter, la réduction des *urls* permet aux utilisateurs d'économiser des caractères et ainsi d'écrire plus de mots sans atteindre la limite. Il arrive parfois que ces liens soient automatiquement réduits lorsque les utilisateurs utilisent une fonction de partage sur des sites web ou applications tierce.

Pour la récolte des articles d'actualité, nous devons non seulement constituer une base de données de domaines de journaux et de blog, mais également une base de données de services de réduction d'*urls* afin de ne pas ignorer les *urls* réduites lors de notre collecte. En effet, certaines de ces *urls* réduites peuvent rediriger vers un article d'actualité. Or, nous ne pouvons connaître le vrai nom de domaine de ce type d'*urls* qu'après avoir effectué une requête nous redirigeant vers le site web cible. Nous appelons *JB* (pour *journals and blogs*) la base de données rassemblant les domaines de journaux et de blog. De même, nous appellerons *US* (pour *url shorteners*) la base de données de domaines appartenant à des services de réduction d'*url*. Nous disons qu'une *url* est en collision avec *JB* ou *US* lorsque son domaine extrait par la méthode des suffixes publics est présent dans la base de données. Par exemple, l'*url* `http://bit.ly/2kzeXN2`, en collision avec *US*, redirige vers la page web `https://www.sfchronicle.com/news/article/2-b[...]`, en collision avec *JB*.

Pour que nous puissions obtenir le plus d'articles d'actualité par utilisateurs dans *Twineews*, il a donc été nécessaire de connaître la nature d'une *url* parmi ces trois catégories :

Catégorie 1 les *urls* dont le domaine fait partie de notre base de données de journaux et de blogs, i.e. en collision avec *JB*.

Catégorie 2 les *urls* réduites faisant partie de notre base de données de services de réduction d'*urls*, i.e. en collision avec *US*.

Catégorie 3 les autres *urls* ne correspondant pas à un article d'actualité ou de blog, i.e. n'étant pas en collision avec *JB* ou *US*.

Afin de collecter tous les articles d'actualité de *Twineews*, une possibilité serait de collecter toutes les *urls* des catégories 1 et 2, puis d'éliminer toutes les pages web issues d'une *url* de la catégorie 2 ne redirigeant pas vers une *url* en collision avec *JB*. Cependant, nous ne pouvons pas effectuer de requête pour chaque *url* réduite étant donné leur grand nombre dans *Twineews*. En effet, cette étape aurait nécessité un temps de calcul et un trafic réseau trop important. Au lieu de cela, nous avons cherché à estimer, pour chaque profil de *Twineews*, la proportion d'*urls* réduites redirigeant vers des *urls* en collision avec *JB*. Si cette proportion estimée est supérieure à un seuil défini, nous effectuons alors une collecte complète du profil. Nous appelons « effectuer une collecte complète » d'un profil la récolte de toutes les

urls réduites partagées sur le profil. Afin d'effectuer cette estimation, nous récoltons les pages web d'un échantillon E de 15 *urls* prises au hasard de la catégorie 2 dans chaque profil de *TwineWS*. À noter que pour les profils ayant au plus 15 *urls* de catégorie 2, l'estimation revient finalement à effectuer automatiquement une collecte complète.

Si la proportion d'*urls* redirigeant vers un article d'actualité ou de blog est grande dans E , alors nous effectuons la récolte complète du profil. L'estimation de pertinence pour une récolte complète est également dépendante de la proportion d'*urls* de catégorie 1 dans les *tweets* du profil. En effet, nous supposons intuitivement que plus un utilisateur partage d'articles d'actualité ou de blog, plus il y a de chances pour que les *urls* réduites cachent des pages d'articles d'actualité ou de blog.

Nous estimons donc que l'intérêt pour une récolte complète d'un profil tient compte de :

JB-visible la proportion d'*urls* des catégories 1 et 3 étant en collision avec *JB*, nous appellerons A l'ensemble des *urls* de la catégorie 1 et 3 ;

JB-hidden la proportion estimée d'*urls* de catégorie 2 redirigeant vers une *url* en collision avec *JB*.

Plus formellement, nous effectuons la récolte complète d'un profil si celui-ci respecte la condition suivante :

$$\alpha \frac{|f(A) \cap JB|}{|A|} + \beta \frac{|f(E) \cap JB|}{|E|} \geq \sigma \quad (7.1)$$

Avec f la fonction permettant d'extraire le domaine par la méthode des suffixes publics et σ le seuil de pertinence que nous fixons à 0.5. Les paramètres α et β règlent le poids de, respectivement, la partie gauche correspondant à *JB-visible* et de la partie droite correspondant à *JB-hidden*. Nous fixons ces deux variables à 0.5 mais celles-ci peuvent être adaptées en fonction des données. Elles doivent respecter la condition $\alpha + \beta = 1$.

Pour les profils ne respectant pas la condition définie par l'équation 7.1, nous ne récoltons pas les pages web des *urls* réduites. Les seuls articles d'actualité ou de blog que nous disposons pour ces profils sont ceux de l'échantillon E . Grâce à cette procédure, nous économisons ainsi des ressources en ne collectant que les *urls* redirigeant plus probablement vers des articles d'actualité ou de blog. Les *urls* redirigeant vers une page d'actualité ou de blog déjà collectée à l'étape précédente sont éliminées afin d'éviter les doublons dans *TwineWS*. Comme pour l'étape précédente, nous remplaçons l'*url* doublon par l'*url* de référence dans les profils de *TwineWS*.

7.2.3 Collecte des profils Twitter

Nous considérons le graphe Twitter comme étant l'ensemble des utilisateurs reliés par des références dans leur profil. Lorsqu'un utilisateur mentionne un autre utilisateur dans un de ses *tweets*, ou lorsqu'ils partagent le *tweet* d'autres utilisateurs (i.e. lorsqu'ils « *retweet* »), nous considérons que ceux-ci sont liés. Pour la récolte des profils Twitter, nous avons implémenté un agent de récolte en *Python*⁹. Lorsque celui-ci collecte un profil, il obtient différentes informations :

9. L'implémentation de l'agent de récolte est disponible à l'adresse <https://github.com/hayj/TwineWS>

- l'ensemble des *tweets* du profil depuis sa création ;
- l'ensemble des autres utilisateurs liés sur le profil (parmi les *tweets*, les *retweets*, etc) ;
- la localisation renseignée par l'utilisateur ;
- d'autres informations telles qu'un texte *bio*, le nombre de *followers*, etc.

Nous limitons la récolte de profils par une heuristique de parcours simple. Cette limite nous permet de récolter suffisamment de profils en minimisant le coût en temps et en trafic réseau. L'heuristique de parcours autorise l'agent de collecte à traverser les liens à partir d'un profil utilisateur si ces conditions sont respectées :

- La localisation, si mentionnée sur le profil, doit faire partie d'une liste des régions et états des États-Unis que nous avons créée spécialement pour *Twineews*.
- Le profil doit contenir au minimum 90% de *tweets* de langue anglaise.

Bien entendu, nous avons initialisé l'agent avec une vingtaine de profils choisis manuellement afin qu'il puisse démarrer son parcours. En utilisant l'heuristique décrite, le risque principal pour cette collecte est que l'agent s'arrête prématurément en ayant collecté que très peu de profils. Cela peut arriver lorsque l'heuristique est trop « forte », i.e. que la probabilité de transition ne soit pas assez grande pour que l'agent s'auto-alimente de nouveaux nœuds à parcourir. Dans ce cas, nous aurions alors introduit de façon aléatoire des transitions sans vérification des conditions de l'heuristique. Nous n'avons cependant pas eu à implémenter ces transitions à « autorisations aléatoires » puisque l'agent de récolte a pu collecter 200 000 profils Twitter. Le graphe fut composé de 1.3 million de profils utilisateurs, incluant les 200 000 ayant été effectivement « visités » (i.e. récoltés) par l'agent. L'agent a collecté des profils durant trois mois. Un total de 10 millions de *tweets* ont été collectés durant cette procédure.

7.2.4 Filtrage des profils

Une fois l'ensemble des profils collectés, il nous a fallu déterminer ceux à conserver pour la tâche de recommandation d'articles d'actualité. Ainsi, les profils ont été filtrés sur différents critères. Chaque critère donne des points de pertinence qui sont combinés pour donner un score final à un profil. Les profils peuvent ainsi être ordonnés par pertinence et nous décidons, par la suite, d'un nombre de profils à conserver. Dans ce manuscrit, nous n'entrons pas dans le détail des poids de chaque critère ou du seuil choisi, cependant, nous décrivons tous les critères ainsi que leur motivation :

- Le nombre de *tweets* doit être suffisant pour que le profil présente un intérêt à être conservé dans *Twineews* : supérieur à 30 dans une période de quatre mois définie à l'avance.
- Le profil a au moins 2 *followers*.
- Le nombre de liens pointant vers un article d'actualité ou de blog : supérieur à 2.
- La proportion de *tweets* écrits en anglais : supérieure à 90%.

Nous cherchons à identifier les profils appartenant à des influenceurs. En effet, ce type de profil peut être géré par des agences et ne pas refléter les vrais intérêts, en termes d'actualité, d'un utilisateur. Pour cela nous identifions les critères suivants qui nous permettent d'éliminer ces profils spécifiques :

- Si le compte a le statut *verified* signifiant que le compte appartient généralement à une célébrité.
- Si le nombre de *followers* du profil est supérieur à 10 000.

Nous cherchons également des critères révélateurs d'une gestion automatisée du compte (e.g. organisation et robot web) :

- Nous vérifions que l'utilisateur n'a pas trop de *tweets* : un nombre inférieur à 1 million.
- Nous vérifions que le profil n'a pas trop de *followings* (i.e. de profils suivis).
- Nous vérifions la présence de différents éléments propres à Twitter : médias partagés, favoris, moments, listes, etc.
- Nous vérifions si l'utilisateur a renseigné un site web, une « bio », un avatar.
- Nous vérifions différentes proportions de caractéristiques propres aux *tweets* sur l'ensemble des *tweets* du profil. Si un profil a une trop grande ou trop faible proportion sur l'une de ces caractéristiques, alors il sera désavantagé :
 - la proportion de *tweets*, de *retweets*, de citations et de photos ;
 - la proportion de *tweets* ayant été « *retweetés* », aimés et répondus par d'autres utilisateurs.
- De même nous cherchons la diversité (ou la non-redondance) de ces différentes caractéristiques parmi l'ensemble des *tweets* d'un profil :
 - les *hashtags* ;
 - les mentions (ou *atreplices*) ;
 - les noms de domaine des *urls* présentes dans les *tweets* pour les *urls* n'étant pas en collision avec *US* ;
 - probabilité (approximée avec des exemples pris aléatoirement) pour qu'un 3-gram du texte présent dans un *tweet* soit dans un autre *tweet* (redondances textuelles).

Suite au calcul du score de chaque profil basé sur ces différents critères, nous ne conservons que 20 000 profils.

7.2.5 Particularités de *TwineWS*

Il existe principalement deux types de jeux de données *offline* pour la recommandation d'articles d'actualité :

1. les jeux de données basés sur des indicateurs de pertinence « clic » ;
2. les jeux de données basés sur des indicateurs de pertinence « partage ».

TwineWS est un jeu de données de la deuxième catégorie et est proche du jeu de données de [Abel et al. \(2011\)](#) qui n'est, aujourd'hui, plus mis à la disposition de la communauté scientifique. *TwineWS* est cependant composé de plus d'utilisateurs et de plus d'articles d'actualité, ce qui permet de garantir une plus grande fiabilité des évaluations et de l'entraînement des modèles. Une des différences notables de *TwineWS* comparé aux jeux de données utilisant l'indicateur « clic » est que les actions de l'utilisateur sont très espacées dans le temps. En effet, dans la plupart des jeux de données utilisant l'indicateur « clic », les clics sont souvent liés à des sessions de navigation utilisateur ([Hopfgartner et al., 2016](#)). En conséquence, les modèles les plus employés sont ceux capables de prédire des suites de recommandations ([Qiannan et al., 2019](#); [de Souza Pereira Moreira et al., 2018](#); [Kumar et al., 2017](#); [Hopfgartner et al., 2016](#)). Cette dépendance entre les items d'une même session est justifiée non seulement par la proximité temporelle des actions mais aussi parce que les utilisateurs,

Nombre de partages	1	2	3	4	5	6	> 6
Proportion des articles	76.1%	12.5%	4.4%	2.1%	1.2%	0.8%	2.9%

TABLE 7.1 – Proportion des articles ayant été partagés par un certain nombre d’utilisateurs dans le jeu de données *TwineWS*

sur la période d’une session, ont souvent des intérêts courts termes (An et al., 2019), par exemple lorsqu’ils cherchent des informations sur un sujet précis, ou qu’ils veulent compléter les informations d’une première lecture.

Cette contextualisation n’est pas observée dans les jeux de données tels que *TwineWS* car l’utilisateur partage généralement des articles de façon périodique, sans qu’il y ait de réelle dépendance entre les items autres que les intérêts longs termes de l’utilisateur. La principale conséquence est que les jeux de données basés sur l’indicateur « clic » ont a priori des historiques de lectures utilisateurs moins diversifiés (Park et al., 2017), lorsque la plupart des systèmes de recommandation *online* tirent bénéfice de la diversification des recommandations. Or, en conditions *offline*, les historiques diversifiés permettent d’évaluer les algorithmes au plus proche de leur performance *online*. Comme nous le verrons par la suite, les utilisateurs du jeu de données *TwineWS* lisent des articles assez divers. En effet, lors du calcul de la diversité des historiques de lectures, nous obtenons un score de 76% équivalent à la diversité d’articles pris au hasard. Ces résultats seront présentés en section 8.5.8. En conséquence, les jeux de données basés sur l’indicateur « partage » ont l’avantage de permettre l’évaluation des algorithmes de recommandation sur des métriques qualitatives telles que la diversité.

Le tableau 7.1 montre, pour chaque colonne, le pourcentage d’articles ayant été partagés par un certain nombre d’utilisateurs. La première colonne montre que le pourcentage d’articles ayant été lus par un unique utilisateur est de 76.1%, la deuxième colonne indique que 12.5% des articles n’ont été lus que par deux utilisateurs et ainsi de suite. Ce tableau montre que la plupart des articles ne sont partagés que par un seul utilisateur. L’un des désavantages du jeu de données *TwineWS* est donc que les méthodes de type filtrage collaboratif seront moins efficaces que celles basées sur le contenu. Cependant, comme mentionné dans l’état de l’art de cette partie, l’une des particularités de la recommandation d’articles d’actualité est que les articles les plus récents sont les plus pertinents et qu’un article ancien de seulement quelques jours est très souvent jugé non pertinent (Gulla et al., 2014). Cela implique qu’aucune interaction utilisateur n’est encore observée pour les items les plus pertinents : c’est le problème du « démarrage à froid » des items. Les méthodes basées sur le contenu sont donc souvent privilégiées (Kumar et al., 2017).

Dans *TwineWS*, nous mettons à disposition le contenu des articles d’actualité lorsque d’autres jeux de données n’incluent que le titre, parfois un résumé (Hopfgartner et al., 2016) mais pas l’intégralité de l’article. Nous verrons par la suite que la prise en compte du contenu est pertinente et qu’il est plus pertinent de prendre en compte uniquement le titre dans les jeux de données basés sur l’indicateur « clic ». Une autre différence majeure est que *TwineWS* est composé d’utilisateurs ayant un long historique de partages lorsque certains jeux de données considèrent les utilisateurs comme étant l’ensemble des sessions de navigation, réduisant les historiques à un nombre réduit de clics (Hopfgartner et al., 2016). Dans *TwineWS*, nous donnons également les *tweets* des utilisateurs qui peuvent aider dans la modélisation de ceux-ci

et dans l'extraction de leurs intérêts.

7.3 Évaluation *offline* dans le cadre de *TwineWS*

L'évaluation « *offline* » d'un algorithme de recommandation est son évaluation simulée à l'aide d'un jeu de données statique rassemblant des interactions observées dans le passé. Ce type de jeu ne permet aucune interaction avec des utilisateurs. L'évaluation « *online* » est la méthode duale qui consiste à utiliser des interactions entre des utilisateurs et le système pour évaluer un algorithme de recommandation. L'évaluation *offline* est souvent utilisée car plus simple à mettre œuvre lorsqu'une évaluation *online* nécessite une plus grande infrastructure, du temps et des interactions utilisateur-système. L'évaluation *online* nécessite la mise en place d'une application. Si l'on souhaite que l'application génère des données d'interaction et soit pérenne, elle doit présenter un intérêt pour les utilisateurs. Une autre possibilité est de trouver des utilisateurs volontaires dans le cadre d'une étude. L'évaluation *online* consiste généralement à comparer différents algorithmes selon :

1. soit un test A/B (ou *A/B testing*) avec des mesures statistiques telles que le taux de clics (i.e. métrique « *click-through rate* », abrégée *CTR*) ;
2. soit une confrontation directe des algorithmes en faisant un entrelacement de leurs recommandations comme détaillé en section 5.3.3.

L'évaluation *offline* permet d'obtenir une approximation, avec plus ou moins de biais, de la performance de l'algorithme s'il était en conditions réelles, c'est-à-dire en interaction temps réel avec des utilisateurs. Dans cette section, nous discutons des biais inhérents à l'évaluation *offline* des systèmes de recommandation et notamment ceux de la tâche de recommandation d'articles d'actualité. Les indicateurs « clics » et « partages » ont chacun leurs avantages et inconvénients et sont plus ou moins impactés par les biais que nous détaillons dans cette section selon les jeux de données *offline* utilisés. Dans cette section, nous détaillons également la méthode utilisée et le découpage des données de *TwineWS*.

7.3.1 Biais inhérents à l'évaluation *offline* et aux indicateurs de pertinence

La tâche de recommandation d'articles d'actualité *offline* sur des données de partage Twitter est semblable à celle des jeux de données rassemblant des clics. Dans les jeux de données tels que *Adressa* (Gulla et al., 2017) ou *NewsREEL* (Hopfgartner et al., 2016) dans sa version simulée *offline*, les clics sont considérés comme des indicateurs de pertinence. La tâche de recommandation d'articles d'actualité consiste à prédire si l'utilisateur a cliqué sur un article. Le jeu de données est alors découpé temporellement : avant une date définie, les articles cliqués permettent l'entraînement d'un modèle, après cette date, ce sont les articles effectivement cliqués qui doivent être prédits par l'algorithme de recommandation.

Ainsi, plus l'algorithme aura prédit de clics parmi ceux ayant une date postérieure à la date pivot, plus son score d'évaluation *offline* sera haut. Comme nous le verrons par la suite, l'évaluation consiste en réalité à déterminer la qualité de l'ordonnement des items candidats, i.e. ceux après la date pivot. Il est à noter que, comparées à la tâche de recommandation d'articles d'actualité, certaines tâches

de recommandation ne nécessitent pas de date pivot dans le cas où, par exemple, les items ne sont pas datés, ou qu'il y a peu d'intérêts à les recommander au plus proche de leur date d'ajout dans le système (ou de leur « date de sortie » comme pour la recommandation de films et de livres). Dans le jeu de données *TwineWS*, l'indicateur de pertinence d'un article vis-à-vis d'un utilisateur n'est pas le « clic » mais le « partage ». Lorsqu'un utilisateur partage un article d'actualité en postant le lien de cet article dans un *tweet*, nous supposons que cet article est d'intérêt pour l'utilisateur. La principale différence est l'action de l'utilisateur considérée servant d'indicateur de pertinence pour les jeux de données statiques *offlines*.

La satisfaction de l'utilisateur est l'objectif premier d'un système de recommandation. Cette satisfaction est généralement liée à la capacité du système à générer du revenu puisqu'une satisfaction engendrera une plus grande fidélité de l'utilisateur et donc plus de revenus (e.g. publicitaires ou par abonnement). Les systèmes de recommandation, lorsqu'ils sont en ligne (ou « en production »¹⁰), essaient donc de maximiser des critères tels que le temps d'utilisation ou de minimiser le taux de rebond. Ces mesures permettent de donner une approximation plus ou moins fiable de la satisfaction de l'utilisateur. L'objectif d'une évaluation *offline* est donc de s'approcher le plus possible de ces critères. Cependant, l'hypothèse des évaluations *offlines* est que les indicateurs de pertinence utilisés permettent une bonne approximation des performances de l'algorithme s'il avait été évalué en conditions réelles (i.e. *online*). Or les indicateurs *offline* et *online* sont fondamentalement différents, ce qui entraîne inévitablement des biais dans l'évaluation *offline*. Nous appellerons le biais introduit par l'utilisation d'indicateurs tels que ceux présentés dans ce chapitre (« clic » et « partage ») le biais de « satisfaction utilisateur ». De plus, Kouki et Said (2018) expliquent qu'un item pertinent, dans le cadre d'une évaluation *offline*, ne l'aura pas nécessairement été dans le cadre *online* étant donné que les actions de l'utilisateur sont de natures différentes. Par exemple, si l'on observe une interaction de type « partage », cela ne signifie pas nécessairement que l'utilisateur aurait eu un intérêt à la lecture dans un contexte de recommandation temps réel.

Le biais introduit par l'indicateur « clic » est qu'il ne reflète pas la satisfaction de l'utilisateur vis-à-vis de l'item (Kouki et Said, 2018; Park et al., 2017). Cet indicateur ne donne aucune autre information que l'action de « clic » sur l'item. Or, l'hypothèse des jeux de données basés sur cet indicateur est qu'un clic est révélateur d'une satisfaction vis-à-vis de l'item s'il avait été recommandé dans des conditions réelles *online*. Cet indicateur est alors utilisé afin d'évaluer un algorithme au plus proche de sa performance en conditions réelles. Dans les jeux de données basés sur des indicateurs de pertinence de type « partage », l'hypothèse est qu'un partage sur les réseaux sociaux est révélateur d'une satisfaction. Pour comprendre la différence entre ces deux hypothèses, il faut décrire le contexte de la prise de décision de l'action par l'utilisateur. Pour cela, il est possible d'imaginer quatre catégories de propriétés qui sont variables chez les utilisateurs selon le contexte, et que ces catégories influenceront la prise de décision de celui-ci :

1. sa connaissance de l'item avant l'action (e.g. le contenu, le titre, l'auteur) ;
2. ses connaissances du monde et de l'actualité en général ;
3. sa connaissance des possibilités de lectures (donc liée aux items qui lui sont

10. Un système de recommandation est dit « en production » lorsqu'il est fonctionnel, i.e. en ligne et utilisé par des utilisateurs.

proposés, ou qu'il aura aperçus en naviguant sur le site d'actualité ou la plateforme de recommandation) ;

4. le taux d'engagement nécessaire de l'utilisateur à effectuer l'action (en temps, en ressources, en conséquences suite à l'action).

Dans le cas d'un clic, l'utilisateur a connaissance du titre de l'article, et généralement de l'auteur ou du journal. Il n'a pas connaissance du contenu de l'article. L'action de « clic » est donc motivée par l'intérêt attendu à la lecture de l'article, une approximation de l'intérêt propre de l'utilisateur pour le contenu à la simple lecture du titre. Cette décision est également motivée par ses connaissances du monde et de l'actualité. En effet, l'utilisateur aura peut-être déjà lu des articles traitant du même sujet, ce qui le dissuadera de cliquer. Il aura peut-être connaissance d'un sujet connexe n'étant pas explicité dans le titre de l'article mais d'intérêt pour lui, ce qui le poussera au clic. Un des principaux défauts de l'indicateur « clic » est qu'il n'est qu'une approximation de l'intérêt propre de l'utilisateur pour le contenu de l'article. Or, la tendance au « piège à clics » (ou *clickbait*) sur le web (Chakraborty et al., 2016), consistant à provoquer le clic par une description amplifiée du contenu, tend à biaiser cette approximation propre.

Lorsque l'utilisateur partage un article sur Twitter, celui-ci a connaissance du titre, de l'auteur, et peut avoir connaissance du contenu de l'article s'il l'a lu avant de le partager. Gabielkov et al. (2016) estiment qu'une majorité (59%) des articles partagés sur Twitter ne sont pas cliqués et ne sont donc pas lus. Les partages impliqueront cependant plus de lectures du contenu que les clics qui sont, par définition, des actions antérieures à la lecture. Dans le cas de l'indicateur de pertinence « partage », l'utilisateur a une très faible connaissance des possibilités de lecture car les items considérés sont l'ensemble des items du web. Dans le cas d'un jeu de données basé sur les indicateurs « clics », l'ensemble des items considérés est restreint aux items du site web (e.g. journal, blog) ou de la plateforme. Ce constat implique que le principal désavantage de l'indicateur « partage » est qu'il introduit potentiellement plus de faux négatifs parmi les items candidats, c'est-à-dire des items que l'utilisateur aurait partagés s'il en avait eu connaissance. En effet, beaucoup d'items, s'ils n'ont pas été partagés par l'utilisateur, seront considérés comme non pertinents, alors qu'ils auraient pu être ignorés volontairement ou involontairement par l'utilisateur. Or, l'ignorance d'un item ne signifie pas nécessairement qu'il n'aurait pas été pertinent dans le cadre d'une recommandation en conditions réelles (Beel et al., 2013). Nous appelons ce biais le biais d'« ignorance des alternatives ».

Intuitivement, nous pouvons considérer que ce biais est moins important lorsque les données de clics sont issues de la navigation d'un utilisateur sur un site de journal en ligne par exemple. Dans ce cas, nous pouvons estimer que l'utilisateur a une connaissance large de ses possibilités de lecture parmi les items candidats puisque non seulement il aura cliqué en naviguant sur le site web, donc aura aperçu d'autres items, mais aussi parce que le nombre d'items est limité aux articles du site en question. Cependant, dans certains jeux de données, l'indicateur « clic » est contextualisé, provoquant plus de faux négatifs. Par exemple, les items cliqués peuvent être des items recommandés par un système de recommandation en ligne mis en place lors de la session de lecture de l'utilisateur. C'est le cas du jeu de données *NewsREEL* qui propose un jeu de données de type *offline* reprenant les clics d'un système de recommandation en ligne. Chaque article recommandé est proposé en bas de page d'un article sur lequel l'utilisateur se trouve. Les décisions de clics et la diversité dans

les lectures de l'utilisateur sont donc limitées par les recommandations des systèmes de recommandation mis en place lors de la session de lecture de l'utilisateur. Le biais d'ignorance des alternatives est alors amplifié par la nature des données enregistrées (Chen et al., 2017a). Il est à noter qu'un item faux négatif est non seulement considéré non pertinent à tort, mais est aussi potentiellement plus pertinent que les items vrais positifs. En effet, il est possible qu'un article soit très pertinent pour un utilisateur mais qu'il ait été ignoré volontairement ou involontairement par un utilisateur, rendant l'article non pertinent dans le jeu de données *offline*. L'ignorance « volontaire » peut survenir lorsque l'utilisateur a déjà connaissance d'une actualité et l'ignorance « involontaire » peut survenir lorsque l'utilisateur n'a pas aperçu l'item lors de sa navigation.

Un partage sera beaucoup plus engageant étant donné qu'il nécessite de poster un *tweet* et que celui-ci sera visible publiquement. Ce constat n'implique pas nécessairement qu'un item partagé aura plus d'intérêt pour l'utilisateur qu'un item cliqué. En effet, les utilisateurs ne partageront pas toujours des articles en accord avec tous leurs intérêts, certains pouvant être réservés à la lecture sans pour autant que l'utilisateur veuille les partager. Cependant, un article partagé appartiendra plus assurément aux intérêts d'un utilisateur qu'un article cliqué par erreur ou par curiosité ponctuelle.

D'autres biais sont introduits par l'évaluation *offline* du fait de l'absence d'interaction avec le système de recommandation (Myttenaere et al., 2015). Les interactions avec un système de recommandation en ligne peuvent changer les intérêts d'un utilisateur au cours du temps ainsi que le modèle de l'utilisateur et donc les articles qui lui sont recommandés. L'évaluation *offline* consiste en réalité à évaluer à quel point un algorithme est capable de prédire ce qui a été observé dans le passé, alors que ces observations auraient changé en conditions réelles par les interactions utilisateur-système. En conditions réelles, l'utilisateur a des besoins changeants dans le temps et l'algorithme a un flux d'information différent, ce qui change inévitablement son fonctionnement interne et ses capacités d'apprentissage. Pour un système en ligne, il sera notamment possible d'apprendre par exploration et exploitation avec l'utilisation d'algorithme d'apprentissage par renforcement. Dans le cadre d'une évaluation *offline*, les interactions peuvent être simulées, mais cela amplifie généralement le biais d'ignorance des alternatives lié aux connaissances de l'utilisateur sur ses possibilités de lecture.

Ce biais est aussi lié au manque de diversité et de nouveauté des recommandations. En effet, lorsque nous basons notre évaluation uniquement sur les items candidats pertinents, dans la limite de ce que « pouvait » et « voulait » choisir l'utilisateur lors de ses actions du fait des propriétés détaillées précédemment, nous omettons les items pouvant être nouveaux pour l'utilisateur étant donné son ignorance, mais pertinents dans le cadre d'une recommandation *online*. C'est pourquoi différentes métriques viennent compléter celles basées sur la précision de la prédiction des algorithmes testés en condition *offline*. Dans cette thèse, nous introduirons notamment les métriques de diversité, de nouveauté et de serendipité.

Dans la littérature, la majorité des algorithmes évalués sur des jeux de données avec indicateur « clic » n'utilisent que le titre des articles d'actualité et non le contenu (Qiannan et al., 2019; Lian et al., 2018). En effet, comme expliqué précédemment, la prise de décision du clic d'un utilisateur est motivée, entre autres, par la première propriété : sa connaissance de l'item. Or, l'utilisateur ne connaît que le titre de

Statistics	Validation split		Test split	
	Train	Test	Train	Test
Start date	2017-10-01	2017-12-25	2017-10-01	2018-01-15
End date	2017-12-25	2018-01-15	2018-01-15	2018-02-16
#users	15905	15905	21239	21239
#news	237150	71781	323572	138785
Average #items per user	26.48	7.22	28.0	10.67
Min. #items per user	8	2	8	2
Max. #items per user	379	97	443	164
#candidates items per user	N/A	1000	N/A	1000

TABLE 7.2 – Statistiques du jeu de données *Twineews* et de son découpage temporel

l’item et non le contenu de celui-ci avant l’action de clic. L’exploitation seule du titre des articles par les algorithmes de recommandation est donc cohérente avec cette propriété. La prise en compte du contenu n’a que peu d’impact mis à part d’apporter des indices sur la connaissance de l’actualité (propriété 2). L’inconvénient principal est que l’apprentissage de ce type d’algorithme est fortement impacté par la tendance « piège à clics ». Dans *Twineews*, il est plus pertinent d’utiliser le contenu des articles puisque celui-ci a eu une influence sur la prise de décision du « partage », même si la lecture du contenu n’est pas systématique comme le montrent [Gabiolkov et al. \(2016\)](#).

7.3.2 Découpage de *Twineews*

Après la récolte du jeu de données, nous avons créé un découpage des *tweets* nous permettant de filtrer et sélectionner un certain nombre d’utilisateurs selon différents critères et une fenêtre de temps définie. Étant donné le grand nombre d’utilisateurs présents dans *Twineews*, nous n’avons pas effectué de découpage pour une validation croisée des algorithmes de recommandation mais avons choisi d’utiliser un ensemble de validation et de test. Le découpage du jeu de données est temporel, c’est-à-dire que nous avons conservé tous les utilisateurs entre le jeu d’entraînement et le jeu de test, et avons choisi une date pivot déterminant si un article d’actualité appartient au jeu d’entraînement ou au jeu de test. La date d’un article d’actualité est la date la plus ancienne parmi ses partages dans les *tweets* du jeu de données. Une alternative à ce découpage aurait été de découper le jeu de données également par utilisateurs en regroupant des utilisateurs différents dans le découpage de validation et de test.

Le tableau 7.2 montre quelques statistiques et le découpage des données de *Twineews*. Les *tweets* et items sélectionnés pour ce découpage sont dans une fenêtre de temps de quatre mois et demi : d’octobre 2017 à mi-février 2018. Un premier découpage de validation (*Validation split*) permet d’optimiser les hyperparamètres des algorithmes de recommandation et un second de test (*Test split*) permet d’évaluer les modèles ainsi que les meilleurs hyperparamètres trouvés. À noter que les parties d’entraînement et de test du découpage de validation servent d’entraînement au découpage de test. Autrement dit, nous utilisons la partie validation pour l’entraînement des modèles finaux et nous les évaluons sur la partie test correspondant à la dernière colonne du tableau. Les dates pivots sont les dates de fin des parties d’entraînement et les dates de début des parties de test.

Lors du découpage des données, nous avons imposé une contrainte de minimum 8 items historiques dans la partie d’entraînement et de 2 dans la partie test. Le découpage de test comprend plus d’utilisateurs que le découpage de validation étant donné que plus d’utilisateurs satisfaisaient ces contraintes grâce à une fenêtre de temps plus grande. Comme le montre le tableau 7.2, les 21 239 utilisateurs sélectionnés dans le découpage de test de *Twineews* ont en moyenne 28 items partagés pour l’entraînement et 10 items partagés pour la partie test. L’ensemble de tous les utilisateurs de *Twineews* est donc pertinent pour l’entraînement et l’évaluation des modèles, i.e. il n’est pas nécessaire de filtrer le jeu de données.

Concernant les items candidats, nous avons choisi de garder, pour chaque utilisateur, leurs items partagés dans le jeu de test ainsi que d’autres pris au hasard jusqu’à atteindre 1 000 items. Ainsi, les algorithmes de recommandation ont pour objectif de calculer un ordonnancement des 1 000 items candidats de chaque utilisateur en fonction de leur historique de partages. D’autres méthodes de génération des candidats auraient pu être choisies, chacune ayant leurs avantages et inconvénients. Nous aurions pu, par exemple, conserver, pour chaque utilisateur, l’ensemble des items de la partie test. Le temps d’exécution du calcul d’un ordonnancement aurait cependant été long étant donné le nombre d’items : plus de 70 000 pour le découpage de validation et plus de 138 000 pour le découpage de test. Nous aurions également pu générer plusieurs ensembles réduits d’items candidats par utilisateurs en ajoutant au hasard des items réellement partagés par l’utilisateur. Par exemple plusieurs ensembles de 10 items candidats avec 5 items pris au hasard parmi ceux réellement partagés par l’utilisateur et 5 autres pris au hasard.

En définitive, chaque découpage est composé de :

1. La liste des identifiants des utilisateurs sélectionnés avec :
 - les identifiants de leurs items partagés avant la date pivot ;
 - les identifiants de leurs items partagés après la date pivot.
2. Une liste d’identifiants d’items candidats à ordonner pour chaque utilisateur.
3. Le contenu des *tweets*, des items (articles d’actualité et de blog) ainsi que diverses informations comme les auteurs des items, la première date de partage des items, la localisation des utilisateurs, etc.

7.4 Conclusion

Dans ce chapitre, nous avons décrit la récolte ainsi que les caractéristiques du jeu de données *Twineews*. Avec le jeu de données *Twineews*, l’évaluation *offline* est identique à celle décrite en section 6.2 : les algorithmes de recommandation doivent prédire quels articles d’actualité sont partagés dans une fenêtre de temps définie. Plus précisément, ils doivent générer un ordonnancement correct (comprenant les items pertinents aux premières positions dans l’ordonnancement) des items candidats pour chaque utilisateur. Nous avons discuté des différents biais inhérents à ce type d’évaluation ainsi qu’aux indicateurs de pertinence considérés.

Dans la suite de ce manuscrit, nous allons utiliser *Twineews* afin d’évaluer si l’exploitation du style écrit peut avoir une influence sur la recommandation d’articles d’actualité. Pour cela, nous allons tout d’abord évaluer les différents modèles de représentation présentés dans le chapitre 4 sur le découpage de validation, puis nous allons effectuer une évaluation finale des meilleurs modèles sur le découpage de

test composé de plus de 21 000 utilisateurs Twitter et de plus de 300 000 articles d'actualité et de blog.

Chapitre 8

Expérimentation de la recommandation basée sur le style

8.1 Introduction

Dans ce chapitre, nous évaluons les modèles proposés dans le chapitre 4 sur le jeu de données *TwineWS*. Nous proposons d’analyser les bénéfices à l’utilisation du modèle *DBert-ft* sur la tâche de recommandation d’articles d’actualité afin de déterminer si le style joue un rôle dans les préférences de lecture des utilisateurs. Pour cela, nous utilisons les algorithmes *SimRec* (section 6.3) et *Reswhy* (section 6.4).

En section 8.2, nous comparons les différents modèles grâce à l’algorithme *SimRec*. Nous validons l’*hypothèse de proximité partielle* en section 8.3. Puis, nous introduisons deux autres hypothèses qui seront détaillées dans les sections suivantes : l’*hypothèse de complémentarité du style* et l’*hypothèse de corrélation entre complémentarité et qualité de recommandation*. En section 8.4, nous étudions la complémentarité des représentations stylo-métriques pour la recommandation d’articles d’actualité grâce à l’algorithme *Reswhy* afin de valider l’*hypothèse de complémentarité du style*. En section 8.5, nous analysons les modèles proposés sur les critères qualitatifs que sont la « diversité », la « nouveauté » ainsi que la « serendipité » afin de valider l’*hypothèse de corrélation entre complémentarité et qualité de recommandation*. Enfin, nous terminons ce chapitre par une conclusion répondant aux différentes questions de recherche formulées dans l’introduction de ce manuscrit.

8.2 Exploitation de modèles de représentation pour la recommandation d’articles d’actualité

Dans cette section, nous utilisons l’algorithme *SimRec* proposé au chapitre 6 permettant de comparer différentes représentations vectorielles de texte. Nous utilisons tous les modèles de représentation introduits au chapitre 4. Il s’agit donc de comparer les performances du modèle *DBert-ft*, permettant d’extraire des caractéristiques de style, dans la tâche de recommandation d’articles d’actualité.

Dans *TwineWS*, il est possible d’exploiter d’autres informations telles que les *tweets* et la *bio* de l’utilisateur, mais dans ce chapitre, nous n’utiliserons que le contenu des articles d’actualité historiques comme source d’information pour l’ordonnement

des candidats. Nous aurions également pu exploiter le titre des articles pour améliorer la performance des algorithmes, mais l’objectif de ce chapitre est avant tout de déterminer si les caractéristiques stylométriques peuvent aider dans la recommandation d’articles d’actualité, i.e. il s’agit de comparer les différents modèles de représentation vectorielle à notre disposition. Dans *TwineWS*, nous devons ordonner un ensemble de 1 000 candidats pour chaque utilisateur dans le découpage de validation et dans le découpage de test. Le score d’un modèle (e.g. le score nDCG) sera la moyenne des scores obtenus grâce à une métrique d’ordonnement sur tous les utilisateurs.

8.2.1 La pondération *BM25*

Dans l’objectif de comparer les résultats obtenus avec l’utilisation de *SimRec* et des modèles de représentation vectorielle présentés au chapitre 4, nous introduisons une méthode de pondération performante lorsqu’il s’agit de trier des documents par pertinence en se basant sur le contenu : *BM25* (Robertson et al., 1993). *BM25* est basée sur la représentation sac de mots. Cette méthode permet de pondérer la pertinence d’un document vis-à-vis d’une requête donnée.

Dans cette expérimentation, nous considérons tous les documents historiques d’un utilisateur comme étant la requête :

$$\text{BM25}(c, H) = \sum_{i=1}^{|H|} \text{IDF}(H_i) \cdot \frac{f(H_i, c) \cdot (k_1 + 1)}{f(H_i, c) + k_1 \cdot \left(1 - b + b \cdot \frac{|c|}{\text{avgdl}}\right)} \quad (8.1)$$

Le paramètre c correspond à l’ensemble des mots du candidat. Le paramètre H correspond à l’ensemble des mots des documents historiques d’un utilisateur, il est la concaténation de tous les documents historiques. La fonction *IDF* correspond à l’*inverse document frequency* utilisée pour la pondération TFIDF. La fonction f donne la fréquence d’un terme dans un document donné. Les constantes k_1 et b sont des hyperparamètres. Étant donné que la variable H est particulièrement large dans notre cas, ces hyperparamètres sont optimisés, garantissant ainsi une pondération adaptée à notre cas particulier. La constante *avgdl* est calculée sur l’ensemble du corpus des candidats et correspond à la taille moyenne des documents.

Ainsi, les ordonnancements sont calculés pour chaque utilisateur en effectuant un tri décroissant des documents candidats sachant leur score *BM25*. En conséquence, l’algorithme correspondant est identique à *SimRec* (algorithme 5), à la différence que nous calculons la similarité entre l’historique et le candidat avec le score *BM25* de leur représentation sac de mots à la ligne 7 et qu’il n’y a pas de paramètre r correspondant au *ratio historique*.

8.2.2 Les modèles *baselines*

Afin d’avoir une base comparative dans cette expérimentation, nous introduisons trois baselines :

Random Consiste à ordonner les candidats de façon aléatoire. Les scores correspondant à cette baseline sont les scores moyens de 10 évaluations aléatoires.

Worst Consiste à ordonner les candidats de la façon la plus mauvaise, i.e. en plaçant les candidats réellement pertinents en fin de liste.

Ideal Consiste à ordonner les candidats de la meilleure façon, i.e. en plaçant les candidats réellement pertinents en début de liste.

Nous introduisons également la *baseline Jaccard* consistant à calculer la similarité entre un historique et un candidat par la distance de Jaccard (Jaccard, 1901) entre vocabulaire historique et le vocabulaire d'un candidat :

$$\text{jaccsim}(H, c) = 1 - \frac{|(\bigcup_{i=1}^{|H|} \text{voc}(H_i)) \cap \text{voc}(c)|}{|(\bigcup_{i=1}^{|H|} \text{voc}(H_i)) \cup \text{voc}(c)|} \quad (8.2)$$

Avec *voc* la fonction renvoyant l'ensemble des mots d'un document sans doublons. De même que pour BM25, l'ordonnement des candidats consiste à trier ceux-ci selon le score de similarité mais en utilisant l'équation 8.2.

8.2.3 Optimisation des hyperparamètres

L'optimisation des hyperparamètres est effectuée sur le découpage de validation. Dans cette expérimentation, pour les modèles *BM25* et *Jaccard* ainsi que *TFIDF*, nous optimisons les hyperparamètres de prétraitement du corpus suivant :

lowercase (booléen) Les majuscules des documents sont converties en minuscules. Nous avons retenu la valeur *false* pour *BM25* et *TFIDF*, et *true* pour *Jaccard*.

lemmatization (booléen) Les mots sont lemmatisés. Nous avons retenu la valeur *false* pour les trois modèles.

minDF (réel entre 0 et 1 ou entier) La proportion de documents minimum du corpus pour laquelle un mot donné doit se trouver pour qu'il soit conservé dans le vocabulaire considéré. Si cette valeur est entière, alors ce n'est pas une proportion mais un compte de documents. Ce paramètre signifie *minimum document frequency*. Nous avons retenu 1/2000 pour *BM25* et *TFIDF*, et 2 pour *Jaccard*.

maxDF (réel entre 0 et 1 ou entier) La proportion de documents maximum du corpus pour laquelle un mot donné doit se trouver pour qu'il soit conservé dans le vocabulaire considéré. Si cette valeur est entière, alors ce n'est pas une proportion mais un compte de documents. Ce paramètre signifie *maximum document frequency*. Nous avons retenu une valeur de 300 pour les trois modèles.

Pour le modèle *BM25*, les hyperparamètres suivant ont également été optimisés :

k_1 (réel) Nous avons effectué une optimisation de type recherche exhaustive (ou *grid search*) dans l'intervalle [1, 3] avec des différences de 0.1 entre les valeurs. La valeur 2.4 a été retenue.

b (réel) Nous avons effectué une optimisation de type recherche exhaustive dans l'intervalle [0.5, 2] avec des différences de 0.05 entre les valeurs. La valeur 1.2 a été retenue.

Les modèles présentés dans le chapitre 4 sont des modèles pré-entraînés sauf *TFIDF* qui a été ré-entraîné sur une partie des documents de *Twinews*. Pour *TFIDF*, la réduction en dimension avec la méthode SVD ne permettait pas d'obtenir de meilleurs résultats sur le découpage de validation. Pour les modèles du chapitre 4, dont *TFIDF*, nous avons seulement optimisé les hyperparamètres relatifs à *SimRec* :

r (réel entre 0 et 1) Le *ratio historique* présenté en section 6.3. Le choix de ce paramètre dépend du modèle. Une étude approfondie de cet hyperparamètre sera proposée en section 8.3.

distance (réel) La fonction de distance (opposée de la similarité). Entre la distance euclidienne, la distance cosinus et d'autres distances standards implémentées dans *Scikit-learn*, nous avons retenu la distance cosinus pour tous les modèles.

8.2.4 Métriques de précision d'ordonnement

Les métriques de précision d'ordonnement utilisées dans ce chapitre sont quatre métriques présentées en état de l'art de cette partie (chapitre 5) :

nDCG Donne un score de précision d'ordonnement avec une pondération décroissante en fonction de la position des items.

MAP Correspond à la précision moyenne en considérant les précisions partielles de la première position à la dernière.

MRR Donne un score indiquant à quel point les premiers items pertinents des ordonnements sont proches de la première position.

P@100 Donne la précision partielle à 100 éléments, i.e. la proportion d'items pertinents dans les 100 premiers items de l'ordonnement.

Dans cette expérimentation ainsi que dans les suivantes, nous avons également utilisé la métrique *nDCG@100*. Cependant, nous avons observé que celle-ci est équivalente à la métrique *nDCG* lorsqu'il s'agit de comparer la performance des modèles. En conséquence, nous ne l'avons pas ajoutée dans les tableaux de résultats.

8.2.5 Résultats obtenus sur *TwineWS*

Le tableau 8.1 donne les résultats obtenus sur le découpage de test. La première partie du tableau 8.1 donne les scores des *baselines*. Les *baselines Ideal* et *Worst* sont des modèles de recommandation qui ont connaissance des labels de pertinence des candidats. Leur rôle est d'indiquer les valeurs minimums et maximums que les autres modèles peuvent obtenir. Comme nous pouvons le voir, le modèle *Jaccard*, qui consiste à comparer les documents représentés par de simples sacs de mots, obtient de meilleurs scores que la *baseline Random*, ce qui semble indiquer qu'une prise en compte du contenu, même simple, est pertinente. À noter également que le score nDCG minimum que l'on puisse obtenir est 0.205 (modèle *Worst*). Comme l'indique la *baseline Ideal*, la meilleure précision partielle à 100 (*P@100*) possible est de 0.106 et non de 1 étant donné que la majorité des utilisateurs ont moins de 100 articles pertinents dans leurs articles candidats. Au contraire, la pire précision partielle à 100 (*P@100*) possible est de 0, ce qui signifie qu'il y a toujours au moins 100 items non pertinents dans les candidats, une information vérifiable avec le tableau de statistiques 7.2 donné en section 7.3.2.

Dans la deuxième partie du tableau, les modèles du chapitre 4 exploités par l'algorithme *SimRec* sont comparés ainsi que le modèle *BM25*. La présentation des modèles est triée par score nDCG de façon décroissante, ainsi les modèles les plus hauts dans le tableau sont les modèles les plus performants. Le modèle *BM25* obtient les meilleurs scores. Les modèles *TFIDF* et *Doc2Vec* obtiennent des scores comparables.

	Model	↓ nDCG	MAP	MRR	P@100
<i>Baselines</i>	<i>Ideal</i>	1.000	1.000	1.000	0.106
	<i>Jaccard</i>	0.303	0.043	0.114	0.025
	<i>Random</i>	0.240	0.012	0.017	0.006
	<i>Worst</i>	0.205	0.005	0.001	0.000
<i>Assessed models</i>	<i>BM25</i>	0.608	0.363	0.617	0.076
	<i>TFIDF</i>	0.596	0.347	0.604	0.074
	<i>Doc2Vec</i>	0.582	0.323	0.587	0.074
	<i>NMF</i>	0.540	0.269	0.516	0.072
	<i>DBert-ft</i>	0.536	0.271	0.530	0.068
	<i>USent</i>	0.517	0.248	0.491	0.064
	<i>LDA</i>	0.474	0.195	0.408	0.064
	<i>DBert</i>	0.463	0.182	0.437	0.054
	<i>InferSent</i>	0.461	0.181	0.422	0.054
	<i>BERT</i>	0.431	0.152	0.379	0.048
	<i>Sent2Vec</i>	0.396	0.120	0.311	0.041
	<i>Stylo</i>	0.287	0.035	0.099	0.021
	<i>Word2Vec</i>	0.252	0.016	0.045	0.010

TABLE 8.1 – Scores de précision d’ordonnement triés par nDCG des modèles du chapitre 4, du modèle *BM25* ainsi que de baselines

Nous remarquerons que le modèle *DBert-ft*, qui représente les documents par des caractéristiques de style, obtient des scores plus bas. Ce résultat n’est pas surprenant étant donné que ce modèle est entraîné de manière à ne pas focaliser son attention sur les termes fortement sémantiques comme nous l’avons montré en section 4.3.6 au chapitre 4. Cependant, le modèle parvient tout de même à obtenir des scores supérieurs aux autres modèles en fin de tableau, ce qui semble indiquer que le style pourrait jouer un rôle dans les préférences de lecture des utilisateurs. Pour chaque métrique proposée, les mêmes modèles sont les plus performants. Ces métriques sont donc cohérentes dans l’évaluation des modèles. Pour la suite de ce chapitre, nous utiliserons nDCG comme métrique principale. De plus, nous éliminons le modèle *Word2Vec* qui obtient des scores plus bas que les autres modèles.

8.3 Validation de l’*hypothèse de proximité partielle*

Dans cette section, nous tentons de valider l’*hypothèse de proximité partielle* détaillée en section 6.3. Comme expliqué dans les sections précédentes, les hyperparamètres principaux de *SimRec* (algorithme 5) sont *distance* et *r*. La distance cosinus fut la plus efficace lors de l’optimisation faite sur le découpage de validation. L’optimisation de l’hyperparamètre *r* correspondant au *ratio historique* fut cependant plus coûteuse en temps puisque nous devons trouver le *r* optimal dans un intervalle de 0 à 1. Nous avons pour cela fait une recherche exhaustive (ou *grid search*) avec une différence de 0.05 entre chaque valeur.

Nous rappelons que, sachant un candidat, un *ratio historique* de 0 signifie que l’on

ne considère que le document historique le plus similaire au candidat pour déterminer le score de pertinence de celui-ci. A contrario, si le *ratio historique* est 1, alors tous les documents historiques sont utilisés pour calculer la pertinence d'un candidat, ainsi la pertinence d'un candidat est la moyenne de toutes les similarités avec les documents historiques. Si l'on considère un *ratio historique* de 0.5, alors *SimRec* utilisera la moitié des documents de l'historique, ceux étant les plus similaires, pour calculer la pertinence du candidat vis-à-vis de l'historique.

L'étude de l'impact de cet hyperparamètre sur les résultats obtenus nous permet de valider ou d'invalider l'*hypothèse de proximité partielle*. Ainsi, si cet hyperparamètre est optimal dans le même intervalle restreint pour l'ensemble des modèles étudiés, alors cela indiquera qu'une proportion consistante des documents historiques permet d'effectivement garantir un calcul de pertinence fiable pour une recommandation, validant ainsi l'*hypothèse de proximité partielle*. Au contraire, si l'intervalle optimal de cet hyperparamètre est variant selon les modèles utilisés, alors cela pourra signifier qu'il ne joue pas de rôle majeur dans la qualité des ordonnancements, invalidant l'*hypothèse de proximité partielle*.

Enfin, un autre indicateur validant l'*hypothèse de proximité partielle* est la différence de score entre la valeur de *ratio historique* la moins optimale et la valeur la plus optimale. Ainsi, si cette différence est significative, cela indiquera qu'il est effectivement pertinent de ne considérer qu'une proportion donnée de l'historique pour le calcul de pertinence d'un candidat.

Ainsi, les deux indicateurs que nous cherchons à étudier sont :

1. La consistance des intervalles optimaux du *ratio historique*.
2. La significativité de l'amélioration des performances par la variation du *ratio historique*.

Afin de vérifier ces deux indicateurs, nous proposons de tracer la courbe nDCG de tous les modèles en fonction du *ratio historique*. Nous avons considéré les neuf modèles les plus performants du tableau 8.1, à l'exception de *BM25* qui n'est pas combiné à l'algorithme 5.

La figure 8.1 montre en rouge la moyenne des scores nDCG en fonction des valeurs de *ratio historique* qui ont été optimisées par une recherche exhaustive. Nous avons ajouté les courbes des modèles *DBert-ft* ainsi que *TFIDF*. Afin d'observer la similitude des tracés entre les trois courbes, nous avons décalé les courbes de *DBert-ft* et de *TFIDF* proche de la courbe moyenne. Ainsi, l'intervalle $[0.3, 0.42]$ semble être l'intervalle optimal de cet hyperparamètre, ce qui signifie que, pour chaque item candidat, environ 35% des documents historiques des utilisateurs sont utiles dans le calcul de pertinence du candidat. Autrement dit, seulement 35% des articles d'actualité de l'historique de l'utilisateur se composent d'indices sur les préférences de l'utilisateur qui permettent de considérer le candidat pertinent.

Ces courbes montrent que l'optimisation de l'hyperparamètre r permet un gain d'environ 2.5% comparé à un algorithme ne considérant que le document historique le plus similaire (i.e. avec un *ratio historique* proche de 0) et environ 1.5% comparé à un algorithme considérant l'ensemble des documents historiques (i.e. avec un *ratio historique* de 1). Finalement, les deux indicateurs 1 et 2 considérés valident l'*hypothèse de proximité partielle*.

L'intervalle optimal du *ratio historique* trouvé n'indique pas qu'il le sera pour tout jeu de données de recommandation, i.e. le *ratio historique* n'est pas nécessairement

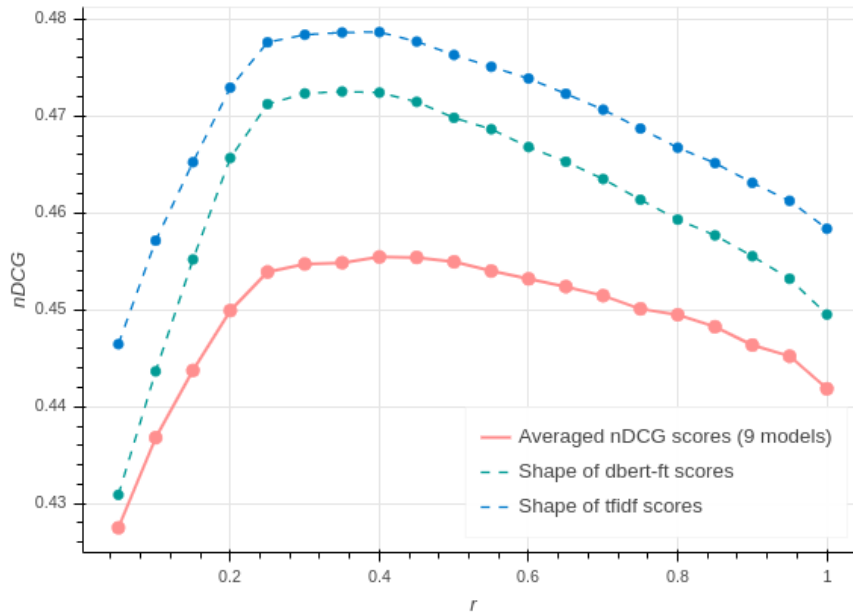


FIGURE 8.1 – Courbes du score nDCG en fonction du *ratio historique*

universel. Cependant, l'intervalle de *ratio historique* trouvé est effectivement l'intervalle optimal dans le contexte du jeu de donnée *Twineews* pour le découpage utilisé. De plus, il est plausible de considérer que tout jeu de données de recommandation d'articles d'actualité aura un *ratio historique* optimal, ceci pouvant être vérifié en effectuant le même type d'analyse que celle proposée dans cette section. Une autre perspective intéressante à cette analyse serait de faire varier le *ratio historique* en fonction de caractéristiques propres aux utilisateurs et d'observer le *ratio historique* optimal selon, par exemple, le nombre d'articles partagés par utilisateur ou la diversité de ses lectures.

8.4 Combinaison des ordonnancements et complémentarité des représentations

8.4.1 Motivations

L'objectif premier de ce chapitre est d'étudier dans quelle mesure le style peut aider à mieux recommander des articles d'actualité, i.e. s'il existe des préférences de style chez les lecteurs et si ces préférences jouent un rôle dans leur consommation des articles d'actualité. En section 8.2, nous avons vu que le style, à travers les représentations vectorielles du modèle *DBert-ft*, ne permet pas d'obtenir des ordonnancements de meilleure qualité en termes de prédictions d'items pertinents. Cette observation était prévisible étant donné que, intuitivement, le style n'a aucune raison d'être l'axe majeur dans les préférences d'un lecteur. Une des directions possibles est donc d'associer une recommandation basée sur le contenu et une recommandation basée sur le style afin d'observer les gains potentiels dans la qualité des ordonnancements.

Comme nous l'avons vu en section 8.2 avec la validation de l'*hypothèse de proximité partielle*, les articles candidats sont pertinents uniquement vis-à-vis d'une sous-

partie de l’historique de lecture d’un utilisateur. Dans cette section, nous supposons maintenant que, selon l’axe de caractéristiques choisi (e.g. thème, style, vocabulaire), un même article candidat ne sera pas nécessairement le plus similaire avec les mêmes articles historiques. La similarité entre deux articles serait donc relative à l’axe de caractéristiques choisi. Cela signifie que, non seulement un article serait pertinent vis-à-vis d’un sous-ensemble de l’historique, mais aussi que cette pertinence serait liée à différents articles historiques selon l’axe choisi. Cette réflexion nous amène à supposer qu’une recommandation selon plusieurs axes nous permettrait d’augmenter nos chances d’identifier les articles les plus pertinents en créant des combinaisons complémentaires. En effet, un article donné peut être très pertinent uniquement vis-à-vis d’un seul axe, et sans une combinaison des axes clés, sa pertinence pourrait ne pas être établie.

We believe that there is no one single approach that would satisfy all users’ needs, but a set of complementary approaches. Considering this, we hypothesize that it is interesting to combine them.

Candillier et al. (2011)

Le modèle *DBert-ft*, comme nous l’avons vu dans le chapitre 4, est en mesure d’extraire des caractéristiques de style et de se focaliser sur des mots à faible valeur sémantique (propriété de la *non-spécificité sémantique*). Il capture donc des caractéristiques complémentaires aux autres modèles, qui, pour chacun d’entre eux, sont destinés à extraire des caractéristiques sémantiques du texte : le vocabulaire (e.g. *TFIDF*), le thème (e.g. *NMF*, *LDA*), le sens des phrases par leur représentation sémantique (e.g. *USent*, *InferSent*), ou encore le sens des documents (e.g. *Doc2Vec*).

L’hypothèse que nous formulons dans cette section est que le style joue un rôle dans les préférences de l’utilisateur et que *DBert-ft* est suffisamment complémentaire (en termes de caractéristiques qu’il extrait) avec les autres modèles pour permettre un gain en précision d’ordonnancement des items. Nous appelons cette hypothèse l’« *hypothèse de complémentarité du style* » dans la recommandation d’articles d’actualité.

Prenons l’exemple d’un utilisateur lisant essentiellement des articles sur le thème de la politique et d’autres (sur divers thèmes) ayant tous en commun le style « critique », i.e. exposant l’opinion de son auteur. Imaginons deux articles candidats étant pertinents à une recommandation dans le jeu de test :

- L’article candidat *A* ayant comme thème la politique et un style « descriptif », c’est-à-dire exposant des faits sans donner d’opinion.
- L’article candidat *B* ayant comme thème le sport et étant de style « critique ».

Considérant l’algorithme générique de recommandation par similarité vectorielle présenté en section 8.2, le candidat *A* ne serait pertinent vis-à-vis de l’historique de l’utilisateur que sur l’axe thématique. À l’opposé, le candidat *B* ne serait pertinent que sur l’axe du style. Ainsi, en combinant les deux représentations vectorielles, i.e. à la fois le thème et le style, ces deux articles pourraient être recommandés à l’utilisateur. En ne considérant que l’axe du thème, seul l’article *A* aurait été recommandé.

Dans cette section, nous étudions l’association de tous les modèles avec *DBert-ft* afin de valider l’*hypothèse de complémentarité du style*. De plus, dans l’objectif de consolider l’expérimentation, nous évaluons également la combinaison de tous les

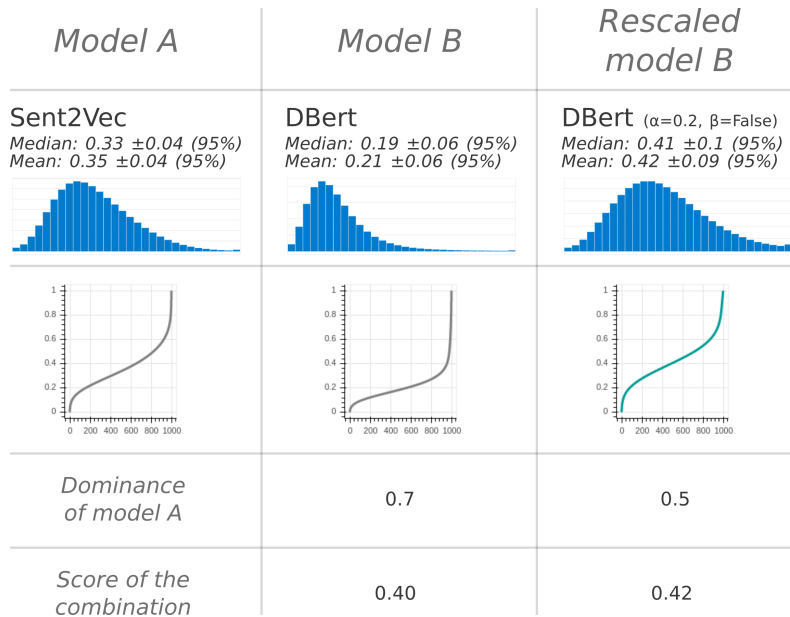


FIGURE 8.2 – Impact du rééchantonnage sur la dominance et le score nDCG

autres modèles par paires. Ceci nous permet d’établir si un potentiel gain observé avec *DBert-ft* est statistiquement fiable. Nous proposons d’utiliser l’algorithme *Reswhy* permettant d’effectuer une hybridation par pondération et d’optimiser la répartition des scores de chaque item dans les ordonnancements par un rééchantonnage.

8.4.2 Impact du rééchantonnage sur la dominance et les métriques d’ordonnement

Pour observer concrètement les effets du rééchantonnage, prenons une combinaison effectuée sur le découpage de validation : *DBert* et *Sent2Vec*. Nous calculons la dominance et le score nDCG de l’hybridation par défaut et une hybridation avec différents paramètres α et β . La figure 8.2 donne les résultats de cette démonstration. Lorsque les paramètres de rééchantonnage ne sont pas modifiés, i.e. $\alpha = 0.5$ et $\beta = true$ pour les deux ordonnancements, alors le score nDCG de l’hybridation est de 0.4. La dominance de l’ordonnement du modèle *Sent2Vec* est de 0.7. Ainsi, il est possible de voir dans la deuxième colonne de la figure 8.2 que l’histogramme de *DBert* est décentré. Cette observation est validée par la courbe moyenne des scores se trouvant en dessous de l’histogramme.

En effectuant une hybridation avec les paramètres $\alpha = 0.2$ et $\beta = false$, l’histogramme et la courbe des scores sont recentrés, comme le montre la troisième colonne de la figure. La dominance du modèle *Sent2Vec* devient alors 0.5, i.e. aucun des deux modèles n’est avantagé. De plus, le score nDCG de l’hybridation gagne 2% en passant de 0.4 à 0.42. Ainsi, nous avons montré, à travers un exemple, que le rééchantonnage a un effet sur la dominance et le score de l’hybridation. L’optimisation des hyperparamètres α et β sur le découpage de validation permet donc une optimisation des scores nDCG.

Dans la littérature, l’hybridation par pondération consiste en une simple combinaison linéaire des scores des items issus de différents algorithmes de recommandation

Model	<i>BM25</i>	<i>TFIDF</i>	<i>Doc2Vec</i>	<i>NMF</i>	<i>DBert-ft</i>	<i>USent</i>	<i>LDA</i>	<i>DBert</i>	<i>InferSent</i>	<i>BERT</i>	<i>Sent2Vec</i>	<i>Stylo</i>
<i>BM25</i>	.608	.616	.621	.607	.635	.605	.581	.608	.601	.602	.584	.606
<i>TFIDF</i>		.596	.610	.606	.626	.607	.584	.595	.587	.588	.559	.594
<i>Doc2Vec</i>			.582	.596	.610	.582	.563	.578	.571	.571	.544	.578
<i>NMF</i>				.540	.606	.567	.533	.555	.550	.548	.535	.540
<i>DBert-ft</i>					.536	.592	.571	.552	.553	.548	.532	.533
<i>USent</i>						.517	.529	.532	.532	.527	.521	.517
<i>LDA</i>							.474	.505	.503	.498	.495	.477
<i>DBert</i>								.463	.481	.464	.440	.461
<i>InferSent</i>									.461	.454	.430	.459
<i>BERT</i>										.431	.416	.430
<i>Sent2Vec</i>											.396	.398
<i>Stylo</i>												.287

TABLE 8.2 – Scores nDCG des combinaisons d’ordonnements

(Suriati et al., 2017). Elle est la méthode d’hybridation la plus utilisée car est facile à implémenter (Burke, 2002). En effet, pour fonctionner, elle n’a besoin que de la sortie d’algorithmes de recommandation déjà établis et il n’est pas nécessaire de modifier ceux-ci. Le rééchelonnage permet d’étendre la pondération en modifiant les scores des items par une fonction de rééchelonnage s’appliquant indifféremment à chaque ordonnancement. La possibilité d’optimiser les hyperparamètres du rééchelonnage, comme nous l’avons vu, permet d’améliorer le score des hybridations. Les poids des hybridations par pondération sont parfois optimisés (Ribeiro et al., 2012; Claypool et al., 1999) mais aucun travail de recherche n’a encore proposé de rééchelonner les scores des ordonnancements.

À notre connaissance, *Reswhy* est le premier algorithme d’hybridation par pondération corrigeant la distribution des scores des ordonnancements par un rééchelonnage. Comme nous l’avons vu, différents algorithmes peuvent avoir des distributions de scores variables dues au fonctionnement interne des modèles de représentation. Le rééchelonnage permet de modifier les différences variables entre deux items proches. Il permet non seulement de contrôler la dominance, mais aussi d’améliorer la combinaison des scores de pertinence des items par l’optimisation des deux hyperparamètres du rééchelonnage.

8.4.3 Résultats obtenus par hybridation des ordonnancements sur *TwineWS*

Nous avons optimisé les paramètres de *Reswhy* par une méthode de type recherche exhaustive (ou *grid search*) sur le découpage de validation. Puis, en reprenant les meilleurs paramètres des hybridations, nous avons évalué chacune d’entre elles sur le découpage de test. Le tableau 8.2 donne le score nDCG de chaque combinaison ainsi que le score de chaque modèle seul dans la diagonale.

Le tableau 8.3 donne les scores et la dominance des meilleures hybridations de

	Model A	Model B	Dominance	↓ nDCG	MAP	MRR	P@100
<i>Top combinations</i>	<i>BM25</i>	<i>DBert-ft</i>	0.32	0.635*	0.395	0.661	0.079
	<i>DBert-ft</i>	<i>TFIDF</i>	0.66	0.626*	0.382	0.652	0.078
	<i>BM25</i>	<i>Doc2Vec</i>	0.32	0.621*	0.376	0.639	0.078
	<i>BM25</i>	<i>TFIDF</i>	0.5	0.616*	0.371	0.635	0.076
	<i>Doc2Vec</i>	<i>TFIDF</i>	0.66	0.610	0.363	0.623	0.077
	<i>DBert-ft</i>	<i>Doc2Vec</i>	0.52	0.610	0.358	0.628	0.078
	<i>BM25</i>	<i>DBert</i>	0.53	0.608	0.360	0.623	0.075
	<i>BM25</i>	<i>NMF</i>	0.37	0.607	0.353	0.625	0.077
	<i>TFIDF</i>	<i>USent</i>	0.31	0.607	0.358	0.623	0.075
	<i>NMF</i>	<i>TFIDF</i>	0.6	0.606	0.353	0.620	0.077

TABLE 8.3 – Scores et dominance des meilleures hybridations. Les astérisques (*) indiquent un score nDCG significatif en comparaison au score nDCG du modèle se trouvant sur la ligne du dessous.

modèles. Pour chacune d’entre elles, l’affectation optimale de *rankAsScores* est (*false, false*), i.e. nous avons conservé les scores des items dans les ordonnancements originels. Les autres paramètres ont été optimisés pour chaque hybridation. Nous n’avons cependant pas optimisé le paramètre *weights*. Nous avons calculé l’intervalle de confiance (95%) des quatre premières combinaisons. Ces intervalles sont à ± 0.0035 de la moyenne des nDCG et permettent de confirmer la significativité de ces scores.

Comme le montre les tableaux 8.2 et 8.3, la meilleure combinaison est celle comprenant le meilleur modèle seul *BM25* ainsi que notre modèle qui projette les documents dans un espace stylométrique : *DBert-ft*. Malgré les faiblesses du modèle *DBert-ft* lorsqu’il est utilisé seul, il parvient à être suffisamment complémentaire avec *BM25* pour que les ordonnancements issus de leur hybridation obtiennent le meilleur nDCG. Une des explications à ce phénomène est que l’hybridation parvient à couvrir plus de caractéristiques jouant un rôle dans les préférences des utilisateurs : en l’occurrence le vocabulaire et le style écrit.

Le score nDCG gagne 1% comparé à la seconde meilleure hybridation. Cette hybridation obtient également les meilleurs scores pour les autres métriques. La dominance montre que *DBert-ft* est globalement avantageux. Nous remarquerons aussi que *DBert-ft* est complémentaire avec le modèle *TFIDF* qui est basé, tout comme *BM25*, sur le vocabulaire des documents. À noter que *Doc2Vec*, basé sur la représentation sémantique des documents, est aussi complémentaire avec *BM25*.

En définitive, l’hybridation des modèles a permis un gain de 3% en nDCG. Ainsi, nous avons, dans un premier temps, montré que différents ordonnancements peuvent être complémentaires et améliorer la précision des recommandations, et, dans un second temps, nous avons validé l’hypothèse de complémentarité du style dans la recommandation d’articles d’actualité. En effet, si les meilleurs modèles utilisés seuls permettaient d’obtenir la meilleure hybridation, alors celle-ci serait composée des modèles *TFIDF* et *BM25*. Dans cette expérimentation, c’est *DBert-ft* qui fait partie des deux meilleures hybridations, ce qui montre qu’il est plus complémentaire que d’autres modèles plus performants seuls.

8.5 Au-delà de la précision d'ordonnancement

Dans cette section, nous présentons de nouvelles métriques ainsi que leurs intérêts dans le cadre des expérimentations sur *TwineWS*. Nous choisissons une formulation particulière des différentes « mesures qualitatives » détaillées dans l'état de l'art de cette partie en section 5.3.2. Ces mesures qualitatives vont au-delà de la mesure de précision de l'ordonnancement des recommandations en mesurant une certaine « qualité » selon différents critères qui ont montré permettre une amélioration de la satisfaction de l'utilisateur. Tous les choix effectués dans cette section sont justifiés par les contraintes de la recommandation d'articles d'actualité ainsi que par les différents biais que nous détaillons. La fin de cette section est consacrée à la présentation des résultats obtenus par les modèles présentés précédemment ainsi que les résultats obtenus par leurs combinaisons.

8.5.1 Intérêts des mesures qualitatives

Lorsqu'il s'agit d'évaluer un système de recommandation en situation *offline*, la plupart des travaux se concentrent sur l'optimisation de la prédiction de pertinence des items. Les métriques utilisées sont alors celles présentées dans la section précédente (e.g. nDCG, MAP). Cependant, la seule prise en compte de ces métriques a pour conséquence de générer des systèmes ne recommandant que des items :

1. similaires entre eux (absence de diversité) ;
2. similaires aux items historiques (absence de nouveauté) ;
3. ne permettant pas de découverte surprenante et inattendue de nouveaux intérêts (absence de sérendipité).

Comme nous l'avons détaillé dans l'état de l'art de cette partie (chapitre 5), cela peut s'expliquer par la restriction de l'utilisateur dans sa consommation des items dans le passé. L'utilisateur a un temps de lecture limité et se concentrera souvent sur ses préférences les plus importantes correspondant à des items faciles d'accès. Il ne couvrira que très peu de ses intérêts sur la période de l'historique considéré. Avec plus de temps et une lecture exhaustive, l'utilisateur pourrait consommer des items correspondant à d'autres intérêts, et même en découvrir de nouveaux. Or, les items consommés dans cette période constituent le profil de l'utilisateur. Les systèmes de recommandation se basant sur ces informations sont restreints à un sous-ensemble des intérêts existants et potentiels de l'utilisateur.

Dans la littérature récente du domaine, différents travaux ont montré que la seule considération du pouvoir prédictif des systèmes de recommandation conduit à une faible satisfaction des utilisateurs lorsque les systèmes sont testés en conditions *online* (Kotkov. et al., 2016; Maksai et al., 2015; Ziegler et al., 2005). De nouvelles métriques qui ne sont pas basées sur la précision de la prédiction de pertinence des items ont montré être corrélées à la satisfaction de l'utilisateur dans plusieurs « études utilisateurs » (ou *user studies*) (Pu et al., 2011; Knijnenburg et al., 2012; Ekstrand et al., 2014) malgré le fait que l'augmentation de l'une des mesures qualitatives conduise généralement à une baisse de précision en conditions *offline* (Kaminskas et Bridge, 2016; Aslanian et al., 2016; Kunaver et Požrl, 2017; Javari et al., 2016). Ces métriques sont la diversité, la nouveauté, la sérendipité que nous appelons « métriques qualitatives » (à opposer aux métriques de pertinence).

La diversification consiste à recommander des items non redondants à l'utilisateur. L'objectif est d'augmenter la satisfaction de l'utilisateur dans sa lecture de l'actualité en évitant qu'il soit confronté à la même information plusieurs fois. En effet, lorsque l'objectif d'un lecteur est de se renseigner sur l'actualité, il ne verra généralement aucun intérêt à la lecture de deux articles donnant les mêmes informations. Ainsi, il préférera gagner du temps en lisant des articles sur des sujets différents ou complémentaires. Pour le système de recommandation, la diversification consiste donc en la maximisation du nombre d'items pertinents dans une liste de recommandations, sachant que la lecture d'items a pour contexte la présence d'autres items dans la liste. En conséquence, la redondance des éléments d'un ensemble implique la non-pertinence d'un sous-ensemble de celui-ci.

De la même manière, la nouveauté d'une recommandation est sa non-redondance vis-à-vis des lectures passées de l'utilisateur. Elle permet d'éviter à l'utilisateur de lire des articles restreints aux intérêts qu'il a exprimés dans le passé par ses interactions. Ainsi, le lecteur peut découvrir de nouveaux sujets d'intérêts et compléter ses connaissances de l'actualité. À noter que, dans le contexte de la recommandation d'articles d'actualité, la nouveauté n'est pas binaire. En effet, il ne s'agit pas de considérer un item comme nouveau simplement parce que l'utilisateur n'a pas interagi avec. Comme expliqué dans l'état de l'art de cette partie (chapitre 5), il s'agit d'une quantité plus ou moins grande indiquant la similarité entre les informations apportées par un item et les connaissances antérieures du lecteur.

Ajoutons également le fait que le biais d'ignorance des alternatives décrit dans le chapitre 7 en section 7.3 implique qu'il peut exister des faux négatifs dans les interactions enregistrées. Ces faux négatifs sont les articles avec lesquels l'utilisateur n'a pas interagi parce qu'il ignorait leur existence, malgré leur pertinence vis-à-vis de ses intérêts de lecture. Ce biais constitue également une raison de chercher à diversifier des recommandations et à les rendre nouvelles.

Pour résumer, la diversité correspond donc à une non-redondance d'items dans une liste de recommandations alors que la nouveauté correspond à une non-redondance d'une liste de recommandations avec un historique de lecture. Ces deux objectifs qualitatifs, en plus d'augmenter la satisfaction des lecteurs, permettent d'éviter le phénomène de bulle de filtres (ou *filter bubble*) formalisé par [Pariser \(2011\)](#) et détaillé en section 5.3.2.

La diversité est une qualité mesurable, la limite étant la fonction de mesure utilisée qui peut ne pas correspondre aux critères des lecteurs, ou y correspondre partiellement. Prenons l'exemple de deux articles d'actualité relatifs à une même personnalité politique, mais parlant de deux événements distincts. Un lecteur pourra trouver ces deux articles similaires lorsqu'il considérera les personnalités mentionnées comme critère principal. À l'opposé, un système de recommandation pourra considérer ces deux articles comme distants lorsque les seuls critères utilisés sont les termes employés (e.g. distance de Jaccard). En effet, les termes employés seront majoritairement relatifs à l'évènement et non à la personnalité impliquée, ce qui orientera la fonction de similarité vers une différenciation plus « thématique ». Si le système de recommandation était basé sur la similarité des entités nommées, alors la diversité des recommandations pour ce lecteur aurait été plus appropriée.

Non seulement la fonction de similarité ne correspondra pas nécessairement aux critères d'un lecteur, mais les critères entre lecteurs pourront également varier. Ainsi, rien ne prouve que la liste la plus diverse identifiée grâce à une mesure le soit

également pour le lecteur final. Les mesures *offline* utilisées que nous détaillerons dans les sections suivantes ne donnent qu’une approximation de la diversité ressentie par l’utilisateur s’il interagissait en temps réel avec le système en conditions *online*.

La nouveauté souffre aussi de ce biais puisqu’il est nécessaire d’utiliser une fonction de similarité entre les items historiques et candidats. Mais elle est aussi biaisée par l’ignorance du système sur les connaissances du lecteur. Celui-ci aura pu avoir connaissance d’une actualité en particulier sans que cela ait été observé dans le système par une interaction (e.g. un partage, un clic). Il aura pu avoir connaissance de cette actualité en utilisant une autre plateforme de lecture ou par d’autres intermédiaires (e.g. interactions sociales). Ainsi, la seule prise en compte des lectures passées dans le contexte du système de recommandation utilisé peut amener, à tort, à considérer un item comme nouveau pour le lecteur.

La sérendipité est plus difficile à définir. Il est communément admis qu’elle se compose des notions de pertinence, de nouveauté et de surprise (Kaminskas et Bridge, 2016). Ainsi, une recommandation « sérendipe » consistera non seulement en la recommandation d’un item nouveau et pertinent, mais aussi inattendu dans la mesure où l’utilisateur ne l’aurait pas découvert de lui-même (Candillier et al., 2011). Les deux premières notions sont concrètes et bien définies. Elles sont mesurables malgré les biais que nous venons de détailler. Cependant, la surprise, parfois appelée l’« inattendu » (ou *unexpectedness*) est plus difficile à mesurer. Elle inclut une dimension émotionnelle subjective et il est difficile de connaître les raisons pour lesquelles un utilisateur trouvera une recommandation sérendipe (Kotkov et al., 2016). Les mesures proposées dans la littérature sont donc, tout comme les mesures de diversité et de nouveauté, des approximations du ressenti réel de l’utilisateur.

Une des solutions possibles permettant de réduire les biais relatifs au ressenti des lecteurs est d’utiliser plusieurs mesures de similarité. Ainsi, différentes mesures couvriront plus d’aspects différents en cohérence avec le jugement des lecteurs. Pour les trois métriques qualitatives que sont la diversité, la nouveauté et la sérendipité, nous utiliserons plusieurs mesures de similarité que nous détaillons dans les sections suivantes.

Dans ce chapitre, nous n’utiliserons pas la mesure de couverture car elle n’est pas adaptée au jeu de données *TwineWS*. Cette métrique mesure à quel point les items candidats ont été recommandés de façon exhaustive sur l’ensemble des recommandations (pour tous les utilisateurs), i.e. si les recommandations « couvrent » tous les items. Or, les items candidats de *TwineWS* qui ne sont pas pertinents sont pris au hasard dans un large ensemble. L’ignorance d’une partie de ceux-ci ne sera pas considérée néfaste comme pour la recommandation de films ou de produits puisque la pertinence intrinsèque des articles d’actualité décroît rapidement avec le temps.

8.5.2 Rôle de l’hybridation

Il existe des méthodes permettant d’améliorer directement la diversité, la nouveauté ou encore la sérendipité. Parmi ces méthodes, la méthode du « ré-ordonnement » (ou *reranking*) est la plus utilisée et facile à mettre en œuvre (Ziegler et al., 2005; Barraza-Urbina et al., 2015). En effet, elle ne modifie pas un algorithme de recommandation mais consiste à prendre l’ordonnement produit par celui-ci et de réordonner les items de manière à optimiser un critère. Par exemple, si l’on souhaite améliorer la diversité d’une liste de recommandations, il s’agira de sélectionner itéra-

tivement les items dissimilaires avec ceux déjà sélectionnés grâce à un algorithme glouton. Généralement, cette sélection résulte en la diminution de la taille de la liste de recommandations.

Cependant, il est souvent difficile d’optimiser un critère qualitatif, surtout lorsque les algorithmes de recommandation sont basés sur le contenu. Par exemple, il est courant que ce type d’algorithmes produise des ordonnancements d’items partiels et intrinsèquement faiblement diversifiés. Cela s’explique par le fait qu’il n’utilise qu’un seul axe lorsqu’il s’agit de calculer des similarités entre items. Les items sélectionnés sont alors ceux les plus proches de l’historique selon l’unique axe considéré. Une diversification par ré-ordonnement ne permettra pas une amélioration significative. Elle sera faible par rapport à l’amélioration à laquelle il est possible de s’attendre sachant l’ensemble des items candidats à disposition. Ainsi, l’une des méthodes utilisées lorsqu’il s’agit d’améliorer les recommandations d’un point de vue qualitatif est l’hybridation.

L’hybridation est parfois utilisée pour améliorer la précision (Claypool et al., 1999). Mais elle est surtout utilisée lorsqu’il s’agit d’améliorer les critères qualitatifs. En effet, il est admis qu’un algorithme donné ne sera jamais optimal sur l’ensemble des critères liés à la satisfaction des utilisateurs (Candillier et al., 2011). Une des solutions est alors de réussir à tirer parti de différents algorithmes ayant chacun un avantage sur un ou plusieurs critères, en espérant que leur combinaison puisse permettre un compromis intéressant couvrant tous les critères.

L’avantage de l’hybridation est donc qu’elle peut potentiellement cumuler les avantages de plusieurs algorithmes, mais aussi qu’elle permet de régler la balance en faveur d’un ou plusieurs critères choisis (Bellogín Kouki et al., 2013; Javari et Jalili, 2015). Peu de travaux démontrent la possibilité d’une optimisation multi-critères (Javari et al., 2016; Cai et al., 2020) car il est admis que lorsque l’on cherche à optimiser le critère de précision, les critères qualitatifs en pâtissent, et vice versa (Kaminskas et Bridge, 2016; Aslanian et al., 2016; Kunaver et Požrl, 2017; Javari et al., 2016). Il s’agit alors de faire une optimisation multi-critères et de choisir une hybridation sur la frontière de Pareto comme proposé par Ribeiro et al. (2012).

Nous avons montré dans la section précédente que la complémentarité du style permettait une augmentation de la précision des recommandations. En effet, des caractéristiques complémentaires peuvent être corrélées à différents aspects dans les préférences du lecteur. Le style (via *DBert-ft*) et le vocabulaire (via *BM25*) sont les deux représentations d’items les plus complémentaires que nous avons réussi à combiner pour une augmentation de la précision des recommandations. L’hypothèse que nous formulons dans cette section est la suivante : différentes représentations d’items, lorsqu’elles sont suffisamment complémentaires pour augmenter la précision, permettent également une diversification et une nouveauté par la nature même des caractéristiques qu’elles capturent, celles-ci étant différentes. Nous appelons cette hypothèse l’*hypothèse de corrélation entre complémentarité et qualité de recommandation*.

Reprenons l’exemple de la section 8.4 d’un utilisateur lisant essentiellement des articles sur le thème de la politique et d’autres (sur divers thèmes) ayant tous en commun le style « critique ». Nous avons deux articles candidats étant pertinents :

- L’article *A* ayant pour thème la politique et un style descriptif.
- L’article *B* ayant pour thème le sport et un style critique.

L’hybridation par le thème et le style permet de recommander à la fois *A* et *B* alors

que la seule prise en compte des thèmes ne permettra pas de recommander l'article B (ou le positionnera très bas dans l'ordonnement). Dans cette section, l'hypothèse formulée consiste à dire que la recommandation faite par la même hybridation, sachant qu'elle cible deux axes de caractéristiques différentes (le thème et le style), fait que les items effectivement recommandés sont plus diversifiés. En effet, la recommandation des articles A et B sera plus diversifiée que la recommandation de l'article A avec un article C de thématique similaire. Il en ira de même pour l'article B combiné à un article D de style similaire.

L'exemple donné ci-dessus est trivial mais ne reflète pas la réalité des données. En effet, *TwineWS* ne se compose pas uniquement d'articles strictement « politiques » ou strictement de style « descriptif ». De même, les lecteurs n'ont pas de préférences strictes. Ces caractéristiques relatives aux items et aux utilisateurs sont des quantités latentes et non des qualités strictes. Il n'est donc pas certain que l'intuition donnée dans cet exemple se vérifie lors de l'évaluation des hybridations. Nous consacrons cette section à l'étude de cette intuition et à la validation de l'*hypothèse de corrélation entre complémentarité et qualité de recommandation*.

8.5.3 Fonctions de similarité

Dans ce chapitre, nous avons choisi de retenir deux fonctions de similarité qui seront utilisées par la suite :

1. la similarité thématique (par *topic modeling*) ;
2. la similarité TFIDF.

Dans les équations des métriques qualitatives, nous utiliserons soit l'une, soit l'autre, ce qui permettra d'avoir deux versions d'une même métrique qualitative couvrant deux aspects : les thèmes et le vocabulaire (avec pondération).

La similarité thématique (ou « topicale ») utilise le modèle *NMF* que nous avons entraîné sur 300 000 articles d'actualité et de blogs qui ne font pas partie des items candidats ou historiques dans *TwineWS*. Nous avons effectué une « inférence » des vecteurs *NMF* de tous les items candidats et historiques afin de pouvoir effectuer des similarités sur la base d'un même modèle *NMF*. Entre deux documents, la similarité thématique correspond à la similarité cosinus de leur vecteur *NMF* composé de 100 dimensions correspondant à 100 thèmes. Chacune des 100 valeurs correspond à la probabilité que le document appartienne au thème à la position correspondante. Le tableau 8.4 donne quelques exemples de thèmes que nous avons interprétés à partir des mots composants ceux-ci.

Pour les hyperparamètres de ce modèle, nous nous sommes inspirés des meilleurs trouvés sur le découpage de validation de *TwineWS*, mais avons également vérifié manuellement les *topics*. Cette analyse qualitative avait pour but de vérifier que les thèmes générés correspondaient à de vrais thèmes au regard d'un lecteur de l'actualité (e.g. sans bruits tels que des thèmes composés uniquement de ponctuations ou des thèmes doublons). Dans le vocabulaire, nous avons conservé uniquement les mots qui apparaissent au moins dans 1/2000 documents et nous avons retiré de ce vocabulaire les 300 termes les plus utilisés. Le nombre de *topics* est fixé à 100. Le nombre d'itérations est de 200. Les mots n'ont pas été lemmatisés. Les majuscules ont été réduites. Nous avons utilisé l'implémentation de *NMF* disponible dans *Scikit-learn* (Buitinck et al., 2013).

Thèmes	Termes composants
Science	medical, health, study, disease, researchers, activity, brain, mental, hypothesis
Business	chamber, startup, entrepreneurs, area, commerce, incubator, fund, collective
Justice	court, county, defendants, attorney, lawyer, lawsuit, drug, justice, legal, filed
Technology	technology, companies, platform, services, customer, strategy, cloud, industry
Politic	tax, republicans, senate, bill, republican, gop, democrats, vote, sen
Violence	shooting, gun, las, vegas, guns, paddock, violence, attack, killed, danley

TABLE 8.4 – Exemples de thèmes interprétés du modèle NMF et les termes qui les composent

La similarité TFIDF est la similarité cosinus des vecteurs TFIDF de chaque paire de documents parmi les items candidats et historiques. Pour l'entraînement du modèle TFIDF, nous avons également utilisé l'implémentation *Scikit-learn* (Buitinck et al., 2013). Ce modèle fonctionne de la même manière, i.e. il est entraîné sur 300 000 documents (autres que ceux choisis pour NMF) et l'« inférence » des items candidats et historiques est effectuée sur la base de ce même modèle. Concernant le prétraitement du corpus, nous avons choisi les mêmes hyperparamètres. Aucune réduction en dimension n'a été effectuée.

8.5.4 La diversité

Lorsqu'il s'agit d'évaluer un ordonnancement par une métrique qualitative, l'ordre des items n'a pas d'influence sur la mesure. En effet, pour un même utilisateur, un modèle A et un modèle B auront leur propre ordonnancement des mêmes 1 000 items. Il en va de même pour les hybridations qui consistent à effectuer un ré-ordonnancement de deux ordonnancements originels des mêmes 1 000 items. Les mesures qualitatives des ordonnancements d'un modèle A, d'un modèle B et de leur hybridation seront les mêmes étant donné que ce sont les mêmes 1 000 items candidats qui seront considérés. Il est donc nécessaire d'effectuer la mesure sur un sous-ensemble des items. Pour cela, nous choisissons d'effectuer, pour chacune des métriques qualitatives, la mesure sur les 100 premiers items des ordonnancements, ceux-ci correspondant aux items que le modèle ou l'hybridation considère comme les plus pertinents à une recommandation.

Nous avons choisi le nombre de 100 car correspond à un bon compromis entre réduction des ordonnancements et quantité d'items à lire pour les utilisateurs. En effet, nous effectuons une réduction des ordonnancements d'un facteur 10, ce qui permet de ne considérer que très peu des items originels, ceux les plus pertinents. Par ailleurs, considérer 100 items reste suffisant lorsqu'il s'agit d'envoyer une liste de recommandations aux lecteurs (e.g. dans le cas d'une lecture journalière de l'utilisateur avec possibilité de choisir ses articles parmi 100 candidats recommandés).

La diversité que nous choisissons d'utiliser pour *TwineWS* est la diversité intra-liste comme décrit par Smyth et McClave (2001). Elle prend R l'ensemble des items recommandés et calcule la différence moyenne entre toutes les combinaisons des items (par paires) dans une liste :

$$\text{diversity}(R) = \frac{\sum_{i=1}^{|R|} \sum_{j=i+1}^{|R|} 1 - \text{sim}(R_i, R_j)}{\frac{|R|^2 - |R|}{2}} \quad (8.3)$$

Dans notre expérimentation, cette diversité s’applique à la liste des 100 premiers items d’un ordonnancement, en conséquence $|R| = 100$. La première version notée *div@100* (abrégée *div*) reprend l’équation 8.3 et utilise la similarité TFIDF décrite en section 8.5.3. La métrique *topic-div@100* (abrégée *t-div*) reprend l’équation 8.3 et utilise la similarité thématique décrite en section 8.5.3.

8.5.5 La nouveauté

Comme discuté dans l’état de l’art de cette partie (chapitre 5), la recommandation d’articles d’actualité possède quelques particularités. Tout d’abord, les items candidats sont toujours nouveaux. Autrement dit, les articles pertinents à une recommandation sont les nouveaux articles récemment publiés. En effet, les utilisateurs chercheront d’abord à lire des articles relatifs à l’actualité. De plus, ces candidats perdront en pertinence en seulement quelques jours, ce qui signifie que même *nouveaux* (au sens de la nouveauté vis-à-vis de l’historique d’un utilisateur), les anciens items ne seront pas pertinents.

Les conséquences de ces observations sont que les articles ne doivent pas être considérés nouveaux seulement parce qu’ils n’ont pas été lus par un utilisateur. La nouveauté n’est pas une qualité mais une quantité indiquant la similarité entre les informations disponibles dans l’article et celles dont l’utilisateur a connaissance. Ainsi, pour déterminer si un item est nouveau, il est plus important de considérer la redondance des informations qu’il prodigue vis-à-vis de ce que l’utilisateur a déjà lu dans le passé. En effet, un item candidat sera, dans tous les cas, « nouveau » dans le sens où il n’aura pas été partagé (et donc pas lu). Cette idée est illustrée par la figure 5.2 en section 5.4.2 qui montre une nouveauté en dégradé de gris lorsqu’elle est généralement représentée comme une qualité binaire (gris ou blanc).

Pour toutes ces raisons, l’approximation de la « nouveauté » d’un item par sa « popularité » comme proposé par (Zhou et al., 2010) n’est pas pertinente dans notre expérimentation. En effet, nous n’utiliserons pas la popularité car un article candidat peut être pertinent sans qu’il ait eu le temps d’être populaire étant donné que les articles seront tous récemment publiés. De plus, dans *Twineews*, un item ne sera généralement partagé que par un seul utilisateur comme nous l’avons vu au chapitre 7.

Cette approximation se fonde sur l’hypothèse que si un item est populaire, alors l’utilisateur aura plus de chances d’avoir déjà « interagi » avec (e.g. via un partage, un clic) (Kaminskas et Bridge, 2016). Elle est notamment utile lorsque l’on n’a pas de certitude sur les interactions de l’utilisateur avec les items, par exemple sur les plateformes qui n’enregistrent que les items notés explicitement mais n’enregistrent pas les informations de clics.

Nous basons donc notre mesure de nouveauté sur celle proposée par Zhang et al. (2002). Cette nouveauté est aussi utilisée par Hurley et Zhang (2011) et par Mendoza et Torres (2019) sous le nom de « nouveauté de contenu » (ou *content novelty*). Elle consiste à calculer une similarité par paires entre les items historiques et les items candidats :

$$\text{novelty}(R, H) = \frac{\sum_{i=1}^{|R|} \sum_{j=1}^{|H|} 1 - \text{sim}(R_i, H_j)}{|R| \cdot |H|} \quad (8.4)$$

La variable R est l’ensemble des items recommandés et la variable H est l’ensemble des items historiques, i.e. que l’utilisateur a partagé avant la date pivot. Pour les

mêmes raisons que pour la diversité, nous effectuons la mesure de nouveauté sur les 100 premiers items d'un ordonnancement. La variable R est donc de taille 100. La taille de H dépend de l'utilisateur considéré. Pour le découpage de validation : $8 \leq |H| \leq 379$ avec une moyenne de 26.48.

8.5.6 La sérendipité

La mesure de sérendipité que nous choisissons dans ce chapitre se base sur un algorithme primitif (ou « primitive ») comme détaillé dans l'état de l'art de cette partie en section 5.3.2. L'hypothèse est que l'on peut déterminer la « surprise » dans la recommandation d'un item par le fait que cet item n'ait pas été recommandé par un modèle primitif. Ce modèle est primitif dans le sens où il est simple et qu'il permet de générer des listes de recommandations composées d'items « évidents ». Ces items sont « évidents » dans le sens où il est évident qu'ils doivent être recommandés sachant le profil de l'utilisateur. La sérendipité inclut également la notion de pertinence comme expliqué par Kaminskis et Bridge (2016) et Kotkov et al. (2016). Ainsi, plus une liste de recommandations contiendra d'items pertinents qui n'ont pas été recommandés par le modèle primitif, plus elle sera considérée sérendipe.

Comme décrit par Kaminskis et Bridge (2016); Kotkov et al. (2016), la sérendipité, pour un utilisateur, correspond à la proportion d'items pertinents qui n'ont pas été recommandés par le modèle primitif :

$$\text{serendipity}(R, P, T) = \frac{|R \cap (T \setminus P)|}{|T \setminus P|} \text{ such that } T \setminus P \neq \emptyset \quad (8.5)$$

La variable R est la liste des items recommandés qui est, dans notre cas, de taille 100 comme pour les autres métriques qualitatives. La variable T est l'ensemble des items qui sont pertinents (i.e. ceux qui ont été partagés par l'utilisateur après la date pivot). La variable P est l'ensemble des items qui ont été recommandés par la primitive. Cette variable doit être de même taille que la taille de R . Dans notre cas : $|R| = |P| = 100$.

Lorsque $T \setminus P = \emptyset$, i.e. lorsque la primitive a déjà prédit tous les items pertinents, nous considérons que la sérendipité de R n'est pas calculable. Ces cas particuliers ne sont donc pas pris en compte lors du calcul de la serendipité moyenne sur les ordonnancements de l'ensemble des utilisateurs.

Dans le cas de la diversité et de la nouveauté, la mesure est dépendante du choix de la fonction de similarité. La sérendipité est dépendante du choix du modèle primitif. Dans tous ces cas, la dépendance est, finalement, liée à la manière de représenter les items. Pour éviter le biais de ressenti utilisateur, nous utiliserons plusieurs versions de la serendipité en utilisant trois modèles primitifs différents.

Nous avons tout d'abord choisi de considérer le modèle le plus performant en tant que modèle primitif : *BM25*. Ce modèle est « simple » puisque correspond à une pondération qui dépend du vocabulaire du corpus considéré. Lorsque l'on utilise cette pondération, il n'est pas nécessaire de disposer d'un corpus externe comme pour, par exemple, *Doc2Vec* et *DBert-ft*. De plus, ce modèle est très populaire dans le domaine de la recherche d'information et a montré être performant dans de nombreuses tâches (Liu, 2011). Nous utilisons les hyperparamètres de *BM25* qui ont obtenu les meilleurs scores sur le découpage de validation. Nous appelons *bm25-ser@100* (abrégé *b-ser*) la version de la sérendipité basée sur ce modèle primitif.

Le second modèle considéré est celui qui ordonne les candidats par leur similarité TFIDF moyenne avec les items historiques. Il correspond à l'utilisation de la représentation TFIDF et de *SimRec* (algorithme 5) de la section 6.3 avec comme paramètre $r = 1.0$ (un ratio correspondant à un nombre réel). Le troisième et dernier modèle considéré est un modèle qui mesure la similarité TFIDF entre un candidat et l'historique en ne considérant que l'item historique le plus similaire. Puis, il ordonne les candidats par leur score de similarité. Cet algorithme correspond donc à l'utilisation de la représentation TFIDF et de l'algorithme *SimRec* avec $r = 1$ (un entier indiquant que l'on ne considère qu'un seul document historique, le plus similaire).

Ces deux modèles sont simples dans le sens où ils se basent sur la représentation TFIDF, donc sur le vocabulaire des documents. De plus, le premier considérera un item comme pertinent lorsqu'il sera proche de l'ensemble des lectures passées de l'utilisateur alors que le second considérera un item comme pertinent lorsqu'il sera très proche de seulement l'un des items historiques. Ces deux heuristiques sont naïves et simples à implémenter lorsque la « proximité partielle » introduite en section 8.2 a montré être bénéfique, mais nécessitant l'optimisation de l'hyperparamètre r . Nous choisissons de faire l'hybridation de ces deux derniers modèles pour construire un modèle primitif hybride basé sur TFIDF. La mesure de sérendipité basée sur cette hybridation sera notée *avg-ser@100* (abrégée *ser*).

En plus de mesurer la sérendipité d'un modèle, ces mesures permettront de mesurer la complémentarité d'un modèle avec, dans un premier temps, le modèle *BM25*, et, dans un second temps, une version naïve du modèle TFIDF, sans la notion de proximité partielle. En effet, si un modèle produit un ordonnancement qui prédit beaucoup d'items pertinents qui n'ont pas été prédits par la primitive, cela signifie que ce modèle et la primitive sont complémentaires puisqu'ils parviennent, par leur combinaison, à recommander une grande proportion des items pertinents.

8.5.7 La nouveauté stricte

Dans la littérature, la sérendipité contient parfois également la notion de « nouveauté » en plus des notions de « surprise » et de « pertinence ». C'est pourquoi certains travaux utilisent une formulation identique à celle donnée pour la nouveauté lorsqu'il s'agit d'évaluer la sérendipité d'une liste de recommandations. D'autres proposent une variante en ne considérant non pas une moyenne des similarités entre un candidat et les items historiques mais la similarité maximale entre le candidat et les items historiques (Kaminskas et Bridge, 2016; Nakatsuji et al., 2010). Dans ce chapitre, nous utilisons également cette variante, mais l'appellerons la « nouveauté stricte » :

$$\text{strictnovelty}(R, H) = \frac{\sum_{i=1}^{|R|} 1 - \max_{\forall h \in H} \text{sim}(R_i, h)}{|R|} \quad (8.6)$$

Les variables R et H , tout comme pour les précédentes métriques, correspondent respectivement à l'ensemble des items recommandés et à l'ensemble des items historiques. Dans cette formule, pour chaque candidat, nous choisissons l'item $h \in H$ qui est le plus similaire et nous faisons la moyenne pour tous les candidats. L'idée développée par Nakatsuji et al. (2010), justifiant ce choix, est que la moyenne des similarités conduit à une perte d'information surtout lorsque les utilisateurs lisent de

façon diversifiée. Ainsi, selon Nakatsuji et al. (2010), il est préférable d'utiliser cette variante de la nouveauté.

Pour expliquer cela, prenons l'exemple de deux utilisateurs U_1 et U_2 ayant chacun lu un article H_1 sur le sujet de la victoire de Rafael Nadal au tournoi de Roland-Garros. D'après son historique, l'utilisateur U_1 lit généralement des articles sur le sujet du sport, de la politique et des articles de blog sur le voyage. Son historique est donc diversifié. L'utilisateur U_2 , quant à lui, ne lit que des articles sur le sujet du sport. Imaginons un nouvel article C_1 candidat à une recommandation portant sur le même sujet que l'article historique H_1 , i.e. la victoire de Rafael Nadal au tournoi de Roland-Garros. La « nouveauté » de C_1 mesurée selon la formulation donnée en section 8.5.5, c'est-à-dire celle basée sur une moyenne des similarités entre items candidats et historiques, sera très importante pour l'utilisateur U_1 étant donné que la plupart des articles historiques seront sur des sujets différents à C_1 . Pour l'utilisateur U_2 , la « nouveauté » de C_1 sera, en revanche, basse étant donné que les items historiques portent tous sur des sujets similaires à C_1 . Or, cette différence dans la « nouveauté » de C_1 est erronée étant donné que chacun de ces utilisateurs a lu H_1 . La réelle « nouveauté » de C_1 est donc très basse pour ces deux utilisateurs. La formulation de la nouveauté stricte donnée dans cette section résout ce biais en ne considérant que l'article le plus similaire. Plus concrètement, pour le calcul de la nouveauté de C_1 , nous ne considérons que l'article le plus similaire : H_1 . Ainsi, la nouveauté stricte de C_1 est basse pour l'utilisateur U_1 , tout comme pour l'utilisateur U_2 .

8.5.8 Résultats qualitatifs des ordonnancements

L'expérimentation de cette section consiste à reprendre les mêmes ordonnancements évalués sur les métriques de précision, et de les évaluer sur les nouvelles métriques qualitatives suivantes :

- div@100 (div)** la diversité TFIDF ;
- topic-div@100 (t-div)** la diversité thématique ;
- nov@100 (nov)** la nouveauté TFIDF ;
- topic-nov@100 (t-nov)** la nouveauté thématique ;
- snov@100 (snov)** la nouveauté stricte TFIDF ;
- topic-snov@100 (t-snov)** la nouveauté stricte thématique ;
- avg-ser@100 (ser)** la sérendipité TFIDF hybride ;
- bm25-ser@100 (b-ser)** la sérendipité *BM25*.

Le tableau 8.5 donne les résultats obtenus par tous les modèles sur les métriques qualitatives. La première partie du tableau correspond aux *baselines*, la seconde aux modèles seuls et la dernière aux combinaisons de modèles. Dans la dernière partie, nous ne donnons pas toutes les combinaisons mais uniquement les 10 meilleures en termes de nDCG ainsi que celles obtenant de meilleurs scores sur différentes métriques qualitatives.

Dans la première partie du tableau, la *baseline Ideal* obtient le meilleur nDCG ainsi que les meilleurs scores de sérendipité. Ces scores sont optimaux pour le tableau entier. Le score nDCG de 1 est attendu étant donné que les items sont toujours ordonnés en fonction de leur pertinence : les plus pertinents en premier.

Model	↓ nDCG	div	t-div	nov	t-nov	snov	t-snov	ser	b-ser	
<i>Baselines</i>	<i>Ideal</i>	1	.765	.642	.714	.589	.653	.699	.999	.999
	<i>Jaccard</i>	.303	.392	.404	.556	.480	.532	.601	.071	.118
	<i>Random</i>	.240	.767	.651	.741	.639	.695	.751	.044	.045
	<i>Worst</i>	.205	.778	.657	.752	.650	.705	.762	0	0
<i>Individual models</i>	<i>BM25</i>	.608	.530	.321	.528	.266	.439	.316	.290	0
	<i>TFIDF</i>	.596	.438	.338	.494	.290	.409	.336	.181	.142
	<i>Doc2Vec</i>	.582	.541	.382	.548	.323	.469	.382	.319	.245
	<i>NMF</i>	.540	.591	.213	.562	.185	.498	.264	.350	.253
	<i>DBert-ft</i>	.536	.607	.477	.602	.422	.541	.506	.394	.354
	<i>USent</i>	.517	.614	.382	.599	.351	.518	.404	.317	.246
	<i>LDA</i>	.474	.618	.282	.591	.275	.517	.323	.293	.215
	<i>DBert</i>	.463	.570	.417	.583	.391	.526	.479	.210	.179
	<i>InferSent</i>	.461	.507	.341	.552	.345	.502	.439	.175	.136
	<i>BERT</i>	.431	.535	.386	.573	.387	.528	.494	.170	.141
	<i>Sent2Vec</i>	.396	.474	.351	.558	.398	.524	.510	.122	.123
<i>Stylo</i>	.287	.649	.574	.665	.570	.626	.688	.116	.113	
<i>Combinations</i>	<i>BM25+DBert-ft</i>	.635	.544	.376	.545	.313	.467	.375	.395	.285
	<i>DBert-ft+TFIDF</i>	.626	.498	.380	.529	.327	.453	.388	.350	.298
	<i>BM25+Doc2Vec</i>	.621	.515	.328	.525	.275	.439	.325	.325	.166
	<i>BM25+TFIDF</i>	.616	.473	.317	.503	.267	.415	.313	.245	.092
	<i>Doc2Vec+TFIDF</i>	.610	.472	.337	.509	.287	.425	.336	.275	.201
	<i>DBert-ft+Doc2Vec</i>	.610	.547	.400	.556	.341	.483	.408	.409	.347
	<i>BM25+DBert</i>	.608	.515	.319	.526	.276	.445	.334	.277	.096
	<i>TFIDF+USent</i>	.607	.508	.321	.532	.289	.443	.330	.297	.212
	<i>BM25+NMF</i>	.607	.549	.230	.535	.193	.460	.258	.344	.189
	<i>NMF+TFIDF</i>	.606	.508	.232	.517	.198	.442	.260	.313	.216
	<i>DBert-ft+NMF</i>	.606	.570	.283	.554	.234	.488	.308	.434	.350

	<i>DBert-ft+Stylo</i>	.393	.608	.520	.619	.482	.571	.581	.221	.206

TABLE 8.5 – Scores de précision (nDCG) et qualitatifs des *baselines*, des modèles et de leurs combinaisons

En conséquence, la sérendipité de ce modèle est haute étant donné qu’il parvient à bien positionner tous les items pertinents que les primitives n’ont pas réussi à bien positionner. La sérendipité n’est pas de 1 car certains utilisateurs ont plus de 100 items pertinents.

Comme remarqué par Möller et al. (2018) dans leur étude sur la diversité, la *baseline Random* obtient les meilleurs scores en termes de diversité TFIDF et thématique. Dans notre tableau de scores, cette *baseline* ainsi que la *baseline Worst* obtiennent les meilleurs scores également sur la nouveauté et la nouveauté stricte. Les bons scores qualitatifs de ces deux *baselines* montrent que les utilisateurs ont en effet une certaine consistance dans leur lecture de l’actualité, et que sélectionner strictement d’autres items (*baseline Worst*) ou des items au hasard (*baseline Random*) conduit à une meilleure diversité et nouveauté. Cependant, nous remarquerons que la diversité et la nouveauté de la *baseline Ideal* sont hautes par rapport aux modèles évalués, et presque égales à la diversité et la nouveauté de la *baseline Random*. Les utilisateurs ont donc, intrinsèquement, une diversité de lecture et une lecture de « nouveaux » sujets d’intérêt qui sont difficiles à atteindre sans sacrifier considérablement la précision des recommandations. Les scores qualitatifs de la *baseline Ideal* sont des valeurs références que l’on peut considérer comme des objectifs à atteindre lorsqu’il s’agit d’évaluer un algorithme de recommandation.

Parmi les modèles évalués seuls, le modèle *Stylo* obtient les meilleurs scores qualitatifs. Cependant, ce modèle obtient un score de précision qui est très bas (0.287) par rapport aux autres modèles. Après le modèle *Stylo*, le modèle *DBert-ft* obtient les meilleurs scores sur toutes les métriques qualitatives sauf la métrique *div@100* (première colonne) pour laquelle *USent* et *LDA* obtiennent de meilleurs scores.

La sérendipité *bm25-ser@100* du modèle *BM25* est de 0 car ce modèle a servi de primitive, les items recommandés sont donc les mêmes. De même pour le modèle *TFIDF* qui obtient un score de sérendipité *avg-ser@100* très bas étant donné qu’il est proche de la primitive hybride utilisée pour *avg-ser@100*. Concernant *DBert-ft*, nous remarquerons qu’il obtient les meilleures sérendipités *avg-ser@100* et *bm25-ser@100*. Cela peut s’expliquer par la précision de ce modèle et le fait qu’il capture des caractéristiques du texte qui diffèrent beaucoup des autres modèles : le style. Il semble donc que la prise en compte du style permet une recommandation « inattendue » et que le modèle *DBert-ft* est, potentiellement, très complémentaire avec des modèles tels que *BM25* et *TFIDF* qui se concentrent sur le vocabulaire des items.

La dernière partie du tableau confirme cette hypothèse. En effet, le modèle *DBert-ft*, lorsqu’il est combiné avec le meilleur modèle seul, i.e. *BM25*, obtient le meilleur score de précision comme nous l’avons vu dans la section précédente. Il obtient également de bons scores qualitatifs comparé aux autres combinaisons. Cependant, les combinaisons les plus performantes sur les mesures qualitatives sont l’association de *DBert-ft* et *NMF* qui obtient les meilleures sérendipités et l’association *DBert-ft* et *Stylo* qui obtient la meilleure diversité, nouveauté et nouveauté stricte. Il semble donc que l’utilisation conjointe du style et des autres modèles permette d’obtenir un bon compromis entre précision et qualité.

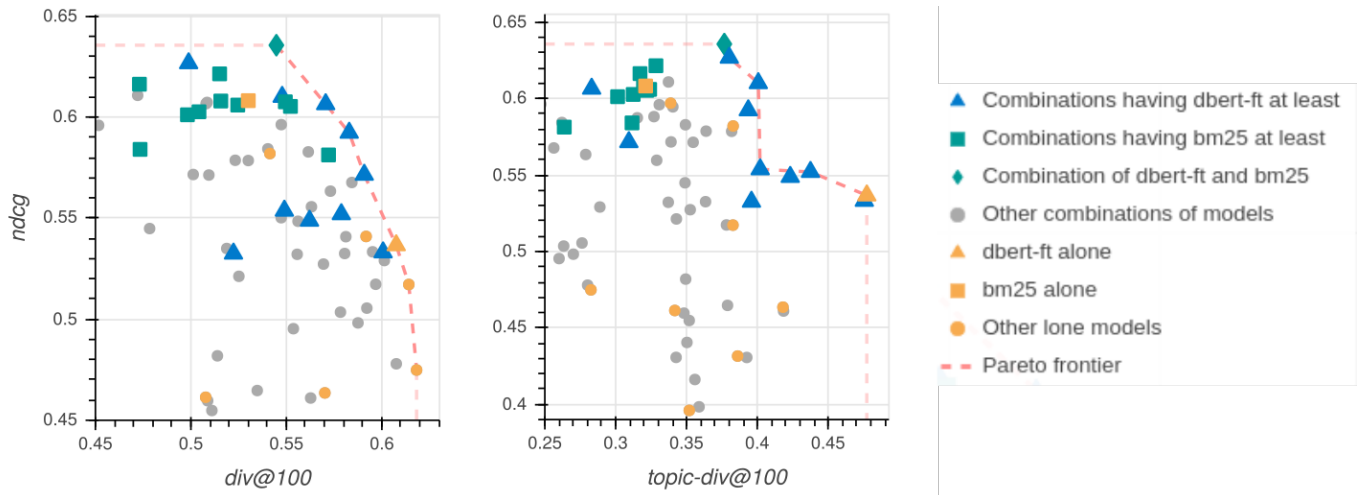


FIGURE 8.3 – Frontière d’efficacité de Pareto entre précision et diversité

8.5.9 Performance multi-critères

Comme le montre le tableau 8.5, l’hybridation permet une augmentation de la précision mais ne permet pas d’améliorer les scores obtenus par les mesures qualitatives par rapport aux modèles utilisés seuls. En effet, l’augmentation de la précision conduit généralement à une baisse des mesures qualitatives comme montré à de nombreuses reprises dans la littérature (Kaminskas et Bridge, 2016; Aslanian et al., 2016; Kunaver et Požrl, 2017; Javari et al., 2016). L’objectif est donc de trouver un compromis entre précision et qualité comme proposé par Ribeiro et al. (2012). Cependant, le tableau 8.5 ne permet pas d’avoir une idée des performances multi-critères des modèles.

Dans cette section, nous proposons donc d’afficher la performance multi-critères de chaque modèle et de leur combinaison afin d’identifier quels modèles et quelles combinaisons sont optimaux, et plus précisément quels modèles sont sur la frontière de Pareto. Pour cela, nous positionnons les modèles dans un espace à deux dimensions ayant pour abscisse une métrique qualitative et pour ordonnée la précision traduite par la métrique nDCG.

Dans toutes les figures d’efficacité de Pareto, nous afficherons la position de la combinaison de *DBert-ft* et de *BM25* avec un losange vert. Le triangle jaune correspondra à *DBert-ft* seul et le carré jaune correspondra à *BM25* seul. Les combinaisons de modèles contenant au moins *DBert-ft* correspondront aux triangles bleus. Les combinaisons de modèles contenant au moins *BM25* correspondront aux carrés verts. Les autres combinaisons seront des points gris. Nous donnons la frontière de Pareto en rouge pointillé.

La figure 8.3 affiche la frontière d’efficacité de Pareto entre précision à travers la métrique nDCG et la diversité. Le graphique de gauche correspond à la diversité *TFIDF* (*div@100*) et celui de droite la diversité thématique (*topic-div@100*). Comme dans le tableau 8.5, le modèle le plus performant est la combinaison *DBert-ft* et *BM25* en termes de précision puisqu’elle se situe le plus haut sur les graphiques. Pour ces deux types de diversité, nous remarquerons que l’ensemble des modèles sur la frontière de Pareto sont des modèles contenant *DBert-ft*, sauf deux modèles seuls pour la diversité *TFIDF* mais ayant une précision basse. Le modèle *DBert-ft* seul est

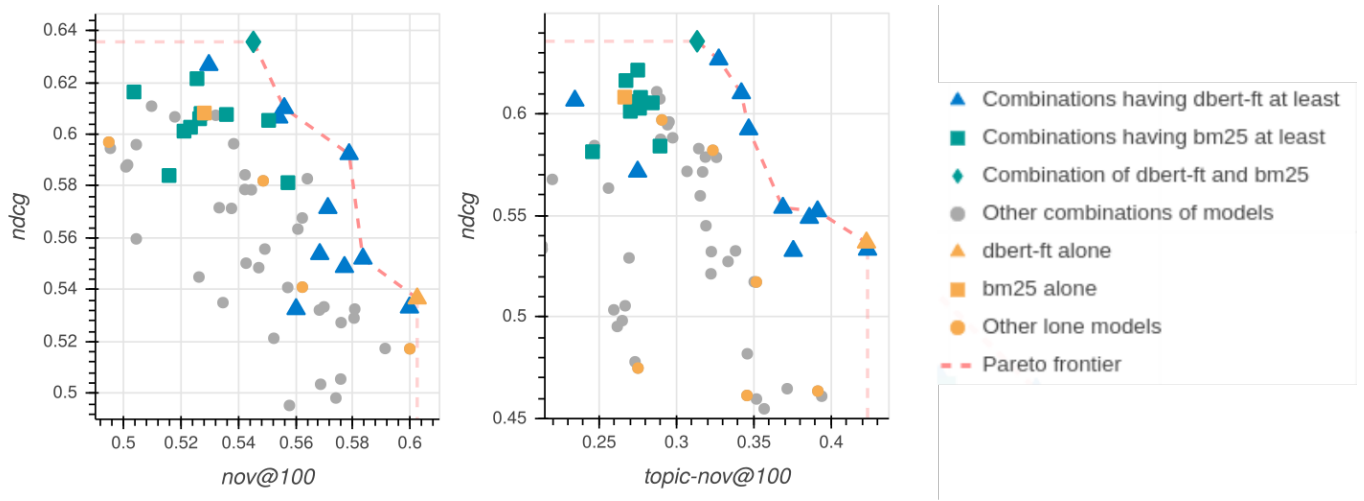


FIGURE 8.4 – Frontière d'efficacité de Pareto entre précision et nouveauté

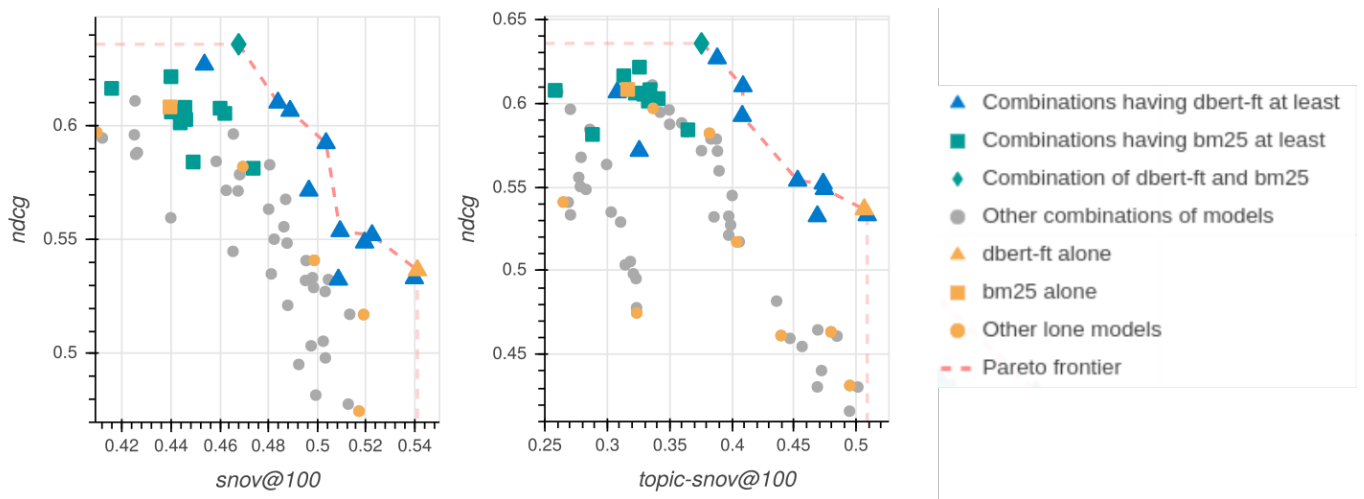


FIGURE 8.5 – Frontière d'efficacité de Pareto entre précision et nouveauté stricte

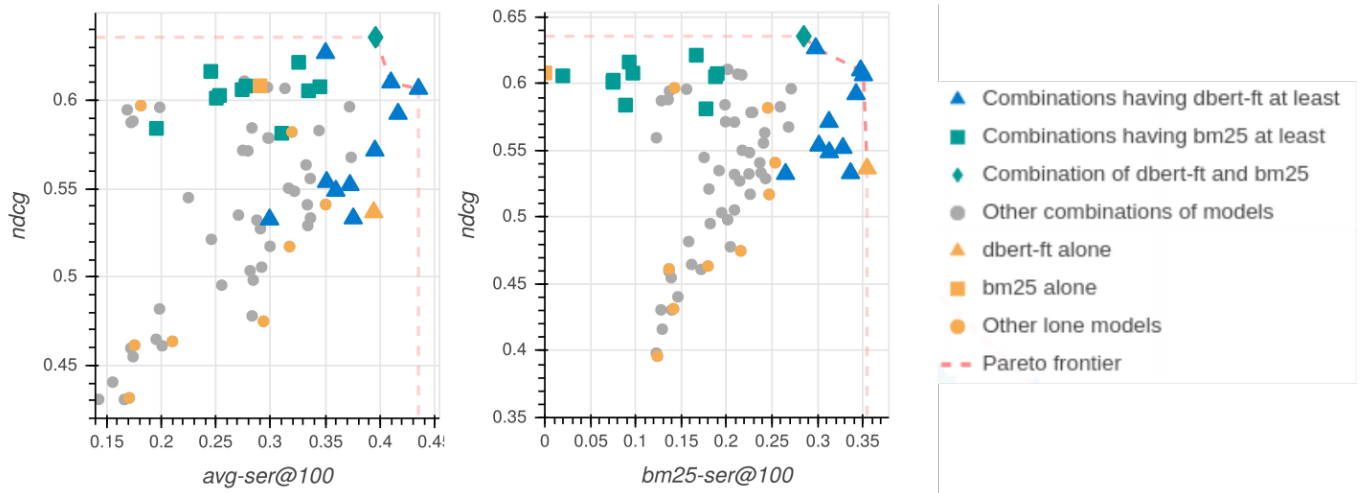


FIGURE 8.6 – Frontière d’efficacité de Pareto entre précision et sérendipité

également sur la frontière de Pareto. Les autres combinaisons de modèles ne sont pas sur la frontière de Pareto. En définitive, l’utilisation de *DBert-ft* semble permettre d’obtenir, dans presque tous les cas, une diversité acceptable sans perdre énormément de précision. Les ordonnancements de ces modèles sont donc à la fois pertinents et composés d’items divers d’un point de vue thématique et du vocabulaire employé.

Les figures 8.4 et 8.5 affichent la frontière d’efficacité de Pareto entre précision et nouveauté ainsi qu’entre précision et nouveauté stricte. Sur ces figures, il est possible de tirer les mêmes conclusions concernant le modèle *DBert-ft*. De plus, sur ces quatre figures, il est possible de remarquer que les modèles contenant *DBert-ft* et ceux ne le contenant pas sont nettement délimités. Le modèle *Stylo* est trop bas en termes de précision et n’a pas été inclus dans ces figures. En conséquence, le modèle *DBert-ft* seul est optimal en termes de nouveauté et nouveauté stricte. L’introduction de *DBert-ft* semble donc permettre également de recommander des items différents de l’historique des lecteurs.

La figure 8.6 affiche la frontière d’efficacité de Pareto entre précision et sérendipité. De même que pour les autres métriques, les modèles sur la frontière de Pareto sont des modèles composés de *DBert-ft*. À noter que la figure de droite montre que *DBert-ft* semble beaucoup plus complémentaire avec le modèle *BM25* qu’avec la primitive hybride basée sur TFIDF.

En définitive, ces analyses montrent que :

1. L’association de *DBert-ft* avec le meilleur modèle seul (i.e. *BM25*) permet une augmentation de non seulement la précision mais aussi de tous les scores obtenus par les métriques qualitatives.
2. Le modèle *DBert-ft* lorsqu’il est seul ou combiné, se situe sur la frontière de Pareto pour les quatre métriques qualitatives.

Les résultats obtenus, grâce aux analyses sur la sérendipité, permettent de confirmer l’hypothèse de complémentarité du style. De plus, nous avons montré que la prise en compte du style dans la recommandation d’articles d’actualité permet aux algorithmes de recommandation de se positionner sur la frontière de Pareto pour les quatre métriques qualitatives utilisées dans ce chapitre, validant ainsi l’hypothèse de corrélation entre complémentarité et qualité de recommandation.

8.6 Conclusion

Dans ce chapitre, nous avons proposé d’exploiter l’algorithme de recommandation générique *SimRec* qui prend en entrée la représentation vectorielle des items candidats et historiques, et qui renvoie un ordonnancement des items candidats. Cet algorithme générique peut prendre toute représentation vectorielle des items et permet donc la comparaison de différents modèles de représentation. Nous avons ainsi pu comparer les modèles présentés au chapitre 4. Dans la littérature, les algorithmes de recommandation basés sur les représentations vectorielles considèrent qu’un item est pertinent lorsqu’il est similaire à l’historique en moyenne, ou lorsqu’il est très similaire à l’un des items historiques. Au lieu de cela, nous avons proposé d’introduire la notion de proximité partielle qui permet de considérer une portion variable de l’historique pour le calcul de la pertinence des items candidats. Nous avons démontré l’intérêt de cette technique et validé l’hypothèse de proximité partielle en analysant en profondeur l’impact de la modification du ratio historique sur les performances des modèles. La valeur optimale de ce ratio a montré être consistante pour les différents modèles testés.

Afin d’analyser les bénéfices à l’introduction du style dans le processus de recommandation, nous avons effectué l’hybridation des modèles. Pour cela, nous avons exploité le nouvel algorithme d’hybridation, appelé *Reswhy*, introduit au chapitre 6. Le rééchelonnage dans l’hybridation fut motivé par les différences de distribution des scores de similarité des ordonnancements issus des différents modèles testés. La fonction *gmrf*, introduite dans l’algorithme d’hybridation *Reswhy*, a permis d’effectuer ce rééchelonnage. Cette fonction permet d’entendre les possibilités d’optimisation de l’hybridation par pondération qui est la méthode la plus utilisée dans la littérature du domaine (Burke, 2002). Son utilisation a montré être pertinente lorsque l’optimisation de ses hyperparamètres que nous avons effectuée permet une amélioration de la précision des ordonnancements, mais aussi un contrôle de la dominance des hybridations. La dominance est une nouvelle mesure que nous proposons dans cette thèse et qui permet d’expliquer à quel point un ordonnancement originel domine un autre en termes de positionnement des items dans une hybridation.

Grâce aux résultats de précision d’ordonnement obtenus avec l’hybridation de *DBert-ft* et de *BM25*, nous avons répondu à la question de recherche *QR3* formulée en introduction de ce manuscrit et montré que la dimension stylistique joue un rôle lorsque les utilisateurs lisent l’actualité, i.e. ceux-ci ont des préférences stylistiques. La sérendipité de *DBert-ft* et sa présence dans les meilleures combinaisons de modèle, malgré ses performances moindres lorsqu’utilisé seul, a montré que le style est une caractéristique complémentaire aux modèles standards dans le cadre de la recommandation d’articles d’actualité. Ainsi, nous avons validé l’hypothèse de complémentarité du style et avons répondu à la question de recherche *QR4*.

Dans ce chapitre, nous avons proposé une formulation de métriques qualitatives adaptées à la recommandation d’articles d’actualité après une discussion des biais inhérents à l’évaluation *offline*, aux indicateurs de pertinence et aux mesures de qualité d’ordonnement dans cette tâche. À travers la question de recherche *QR5*, nous supposons que le style permettait une amélioration des recommandations sur un axe qualitatif. Cependant, les conclusions de ce chapitre montrent que l’exploitation du style dans la recommandation d’articles d’actualité permet d’aller plus loin que cette première intuition. En effet, l’utilisation du modèle *DBert-ft* introduit au

chapitre 4 dans des combinaisons de modèles, notamment avec le plus « précis » (i.e. *BM25*), permet non seulement d’augmenter la pertinence des recommandations, mais permet également d’augmenter leur qualité. Les expérimentations proposées ont donc permis de démontrer, à travers une analyse d’efficacité de Pareto des hybridations, que le style permet une amélioration significative de la qualité des recommandations en termes de diversité, de nouveauté et de sérendipité. Ainsi nous avons validé l’hypothèse de *corrélation entre complémentarité et qualité de recommandation* et répondu à la question de recherche *QR5*.

Récemment, [Sertkan et al. \(2019\)](#) ont émis l’hypothèse que, dans le cadre d’un système de recommandation d’articles d’actualité, générer des recommandations sur la base de caractéristiques stylistiques permettrait la diversification des recommandations. Notre travail de recherche de ce chapitre sur la recommandation par le style valide cette hypothèse.

Used in a news recommender system, the author-level analysis has potential to deliver the most diverse recommendations compared to the other approaches, since it is not constraint by the actual content of the documents. For example, a reader can get a surprising recommendation with a totally different thematic simply because his or her current read is similarly written (i.e., linguistic similarity) as the recommended item.

[Sertkan et al. \(2019\)](#)

Dans de futurs travaux, nous pourrions proposer une optimisation similaire à celle de [Ribeiro et al. \(2012\)](#) qui cherchent, par l’utilisation d’algorithmes génétiques, les meilleurs poids d’hybridation permettant d’obtenir un optimal de Pareto entre pertinence, nouveauté et diversité. Il est également possible de choisir la balance entre pertinence et qualité de recommandation comme le proposent [Bellogín Kouki et al. \(2013\)](#) ou encore appliquer des méthodes de ré-ordonnement afin d’améliorer différents critères qualitatifs ([Kaminskas et Bridge, 2016](#)). Nous pourrions faire varier le ratio historique en fonction des caractéristiques de chaque utilisateur afin d’améliorer la pertinence pour chacun d’entre eux, ou encore faire varier les paramètres d’hybridation de l’algorithme *Reswhy* selon les caractéristiques de chaque utilisateur afin d’adapter la diversité et la nouveauté en fonction de leurs besoins propres.

Chapitre 9

La plateforme d'évaluation Renewal

9.1 Introduction

L'évaluation *offline* se base sur des données statiques, sans interaction avec l'utilisateur. L'enregistrement de ces interactions peut avoir été effectué sur un site web ou une application intégrant un système de recommandation. Il peut aussi n'y avoir eu aucun système lors de l'enregistrement des interactions. Celles-ci peuvent par exemple correspondre à une consommation d'items parmi un choix exhaustif d'items, au partage d'items sur un réseau social, comme dans le cas de *Twineews*, ou encore à la recherche d'items via un moteur de recherche. L'évaluation *offline* vise à comparer des algorithmes de recommandation sur leur capacité à prédire quels sont les items que l'utilisateur a consommés ou aimés dans le passé. Elle consiste à émettre l'hypothèse que si l'on avait recommandé, dans le cadre d'une interaction en temps réel, un item que l'utilisateur a consommé dans le passé, il aurait trouvé cette recommandation pertinente. Cependant, dans une interaction en temps réel entre un système de recommandation et un utilisateur, la nature même de l'interaction (e.g. consommation parmi un nombre limité d'items) et son contexte (e.g. utilisation du système dans une démarche de « découverte » de nouveaux intérêts) n'aboutiront pas nécessairement aux décisions « supposées » de l'utilisateur lors de l'exploitation de données statiques. En définitive, l'évaluation *online* consiste à mesurer la satisfaction observée de l'utilisateur à partir de ses interactions sur une liste de recommandations, lorsque l'évaluation *offline* consiste à estimer la satisfaction de l'utilisateur à partir de ses interactions supposées sur une liste de recommandations. Et comme l'ont montré [Garcin et al. \(2014\)](#) pour la recommandation d'articles d'actualité, les performances *offline* d'un algorithme de recommandation ne sont pas toujours corrélées à ses performances *online*.

Plus concrètement, dans un jeu de données *offline* constitué d'enregistrements de clics (ou de partages), un item ayant été cliqué (ou partagé) ne serait pas nécessairement pertinent s'il était recommandé par un système de recommandation *online* ([Kouki et Said, 2018](#)). Inversement, un item qui n'a pas été cliqué ne serait pas nécessairement non pertinent s'il était recommandé *online* ([Beel et al., 2013](#); [Chen et al., 2017a](#)). De plus, l'évaluation *offline* ne prend en compte aucune interaction utilisateur-système, alors que les interactions d'un utilisateur peuvent changer ses intérêts et les choix effectués par le système ([Myttenaere et al., 2015](#)). De nombreux biais d'évaluation *offline* ont été détaillés dans différentes sections de ce manuscrit. En section 5.3.2, nous avons expliqué que l'historique de consommation d'un utilisateur



FIGURE 9.1 – Premier logo du projet *Renewal* proposé par Dalal Zerhoun (Octopeek)

ne correspond qu'à un sous-ensemble de ses intérêts existants et potentiels. En section 7.3.1, nous avons introduit les biais de « satisfaction utilisateur » et d'« ignorance des alternatives ». En section 8.5.1, nous avons expliqué que, pour pallier les biais cités précédemment, il est possible d'évaluer qualitativement (e.g. diversité, nouveauté) les listes de recommandations générées en *offline*. Cependant, les fonctions de similarité (e.g. lexicale, thématique, sémantique) utilisées pour l'évaluation qualitative peuvent ne pas être en adéquation avec le ressenti des utilisateurs : nous parlons alors d'un biais de « ressenti » des utilisateurs lié aux mesures qualitatives.

Pour toutes ces raisons, mesurer de manière fiable l'utilité d'un algorithme de recommandation implique donc nécessairement son évaluation *online*, avec l'interaction d'utilisateurs en temps réel. Dans cet objectif, nous proposons la plateforme *Renewal* dédiée à l'organisation de compétitions autour de la tâche de recommandation d'articles d'actualité. Cette plateforme est aujourd'hui en cours de développement au Laboratoire de Recherche en Informatique (LRI) et vise à répondre aux besoins, en termes d'évaluation, de la communauté de recherche des systèmes de recommandation. Le nom du projet, « *Renewal* », est construit à partir des termes « *recommandation* », « *news* » et « *evaluation* ». Ce chapitre est dédié à la présentation des caractéristiques de *Renewal* comparé aux plateformes existantes ainsi qu'à la simulation de compétitions nous permettant de prédire dans quelles conditions il est possible d'organiser une compétition *Renewal*.

9.2 Nouveautés apportées par *Renewal*

Non seulement la majorité des auteurs de travaux de recherche effectuant l'évaluation *offline* de leur algorithme de recommandation utilisent leur propre jeu de données comme l'ont rapporté Raza et Ding (2020) (plus de 80% des papiers relus), mais aussi la plupart des plateformes permettant d'effectuer des évaluations *online* sont privées : Google News (Liu et al., 2010), Yahoo! News (Li et al., 2011), Forbes (Kirshenbaum et al., 2012), swissinfo.ch (Garcin et al., 2014), LePoint (Maksai et al., 2015).

CLEF NewsREEL (Hopfgartner et al., 2016) fut la seule plateforme de compétition (de 2015 à 2017) permettant à la communauté de recherche en recommandation d'articles d'actualité d'évaluer leur système de recommandation en conditions *online* en les connectant à un service distant. La figure 9.2 illustre l'architecture proposée par Hopfgartner et al. (2016). Les systèmes de recommandation concurrents (en vert en haut à gauche de la figure) se connectent à un service fourni par la compagnie Plista afin d'effectuer des recommandations en temps réel. Ces recommandations sont

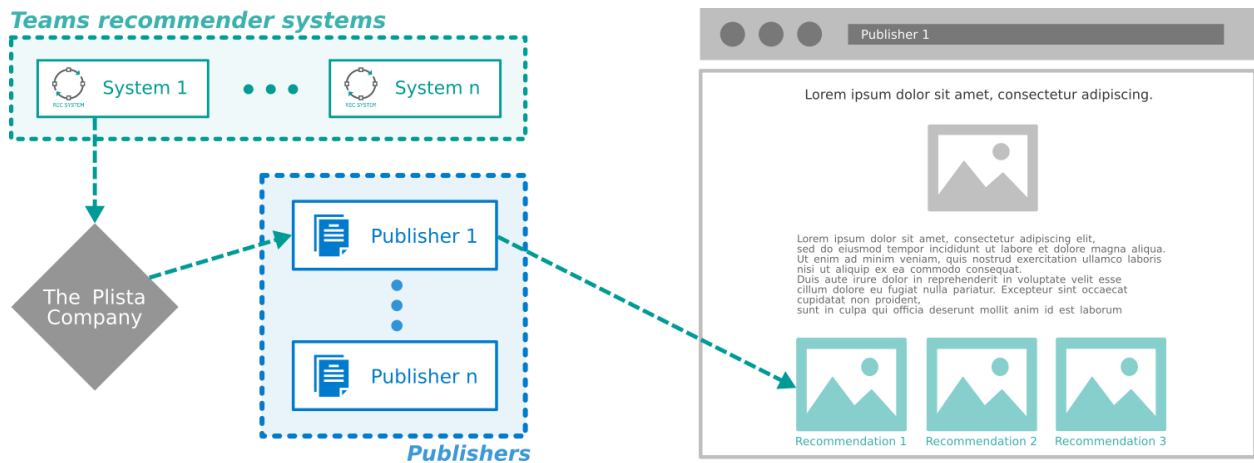


FIGURE 9.2 – Architecture de la plateforme CLEF NewsREEL (Hopfgartner et al., 2016)

effectuées sur des journaux en ligne allemands et sont donc redirigées sur les sites web de ces journaux (en bleu sur la figure). La partie droite de la figure illustre l’affichage des recommandations en bas de la page web d’un article cliqué par un utilisateur. Ces recommandations redirigent vers d’autres articles du même journal. L’avantage majeur de cette plateforme était qu’elle permettait d’effectuer des recommandations à un large nombre d’utilisateurs en temps réel à travers plusieurs journaux en ligne allemands.

By providing both large scale transaction data set and access to a large user base, we argue that NewsREEL can play an important role in closing this gap between academia and industry.

Hopfgartner et al. (2016)

Elle permet la publication de nombreux travaux de recherche sur la tâche de la recommandation d’articles d’actualité (Liang et al., 2017; Beck et al., 2017; Yuan et al., 2016). Cependant, celle-ci présentait différentes limitations que nous proposons de surmonter dans *Renewal*.

9.2.1 Architecture indépendante

La figure 9.3 illustre l’architecture en trois parties de *Renewal*. Contrairement à *CLEF NewsREEL*, *Renewal* propose une application mobile dédiée à la lecture des articles recommandés aux utilisateurs (à gauche sur la figure). La plateforme *Renewal* (au milieu) fait le lien entre les systèmes de recommandation concurrents (à droite) et les utilisateurs. Elle effectue l’évaluation en temps réel et collecte les articles d’actualité qui seront recommandés aux utilisateurs. Ainsi, les données recommandées et leur affichage ne dépendent pas de fournisseurs tiers.

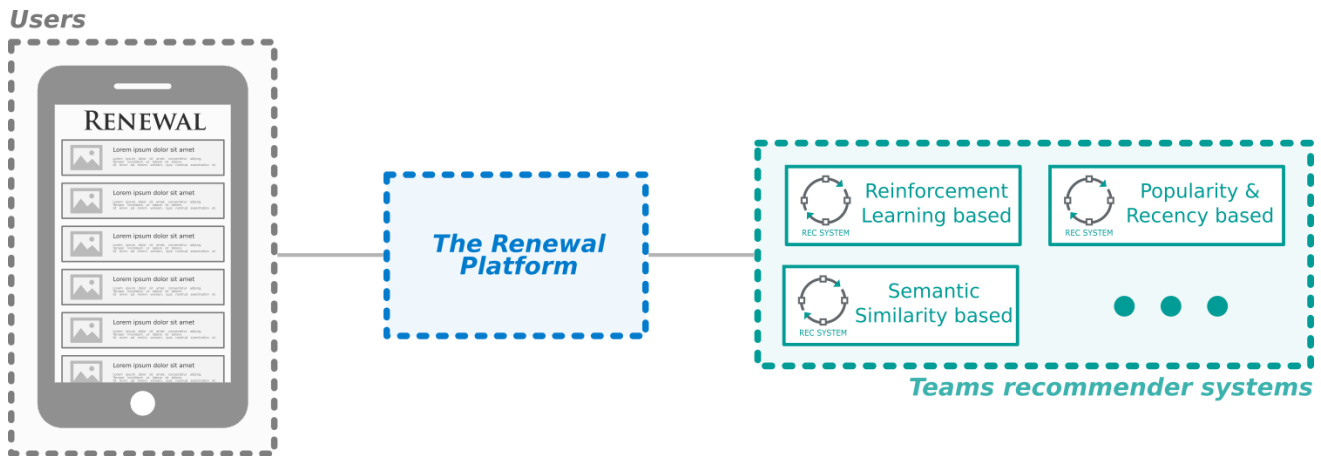


FIGURE 9.3 – Architecture de la plateforme *Renewal*

9.2.2 Sources variées d’articles d’actualité et indépendance du contexte de recommandation

Dans *CLEF NewsREEL*, les articles d’actualité ne proviennent que d’un nombre limité de journaux en ligne inscrits sur le service Plista. Dans *Renewal*, nous collectons les articles d’actualité de toutes les sources que nous avons à disposition par une collecte temps réel sur le web (e.g. à partir d’une liste de journaux en ligne, de flux RSS¹) et sur les réseaux sociaux (e.g. Twitter). Il est possible d’augmenter le nombre de sources et le nombre d’articles collectés en ajoutant des sources dans les agents de collecte de *Renewal*. La figure 9.4 illustre le flux d’information entre les agents de collecte (en bleu), les systèmes de recommandation concurrents (en vert) et les utilisateurs de l’application mobile *Renewal* (en gris). Les articles recommandés sur les smartphones des utilisateurs (via l’application mobile *Renewal*) sont directement requêtés sur les sites web des journaux grâce aux *urls* des articles.

L’inconvénient majeur de l’architecture de *CLEF NewsREEL* est que les recommandations d’articles d’actualité sont effectuées en interne aux journaux en ligne, c’est-à-dire que chaque recommandation effectuée par les systèmes de recommandation concurrents doit nécessairement rediriger vers le même journal en ligne. Ainsi, il n’est pas possible d’effectuer de recommandation entre différents journaux en ligne. Dans *Renewal*, il est possible d’effectuer des recommandations à partir de plusieurs sources grâce à la mise à disposition d’une unique application mobile dédiée.

Le second inconvénient majeur de l’architecture de *CLEF NewsREEL* est que les recommandations envoyées aux utilisateurs sont « contextualisées » : lorsque l’utilisateur reçoit des recommandations d’articles, celui-ci se trouve sur la page web d’un article qu’il a cliqué. Les recommandations sont affichées en bas des pages web des articles d’actualité comme l’illustre la partie droite de la figure 9.2. Dans *Renewal*, les utilisateurs utilisent l’application uniquement dans une démarche consistant à chercher la recommandation la plus pertinente vis-à-vis de leurs intérêts et à découvrir de nouveaux intérêts. Ainsi, l’application est « dédiée » à la recommandation et ne nécessite pas que les utilisateurs aient cliqué sur un premier

1. Un « flux RSS » est une ressource web mise à jour périodiquement selon les données (e.g. articles, produits) mises à disposition sur un site web.

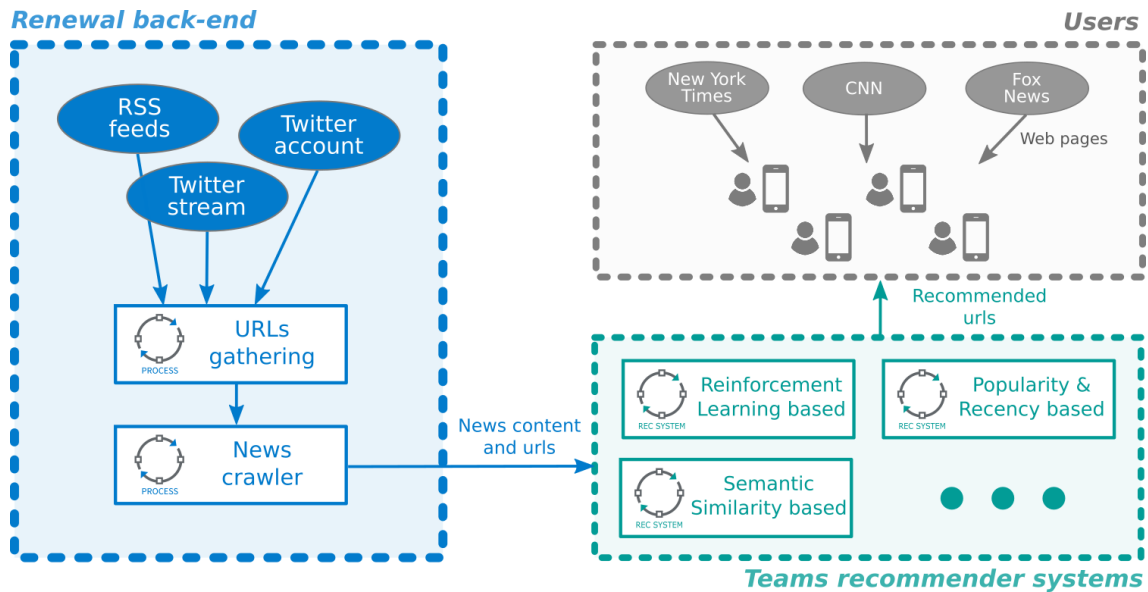


FIGURE 9.4 – Flux d’information entre les agents de collecte, les systèmes de recommandation concurrents et les utilisateurs dans la plateforme *Renewal*

article. Les recommandations dans l’application *Renewal* sont affichées sur la page d’accueil. L’utilisateur peut alors naviguer de haut en bas pour choisir un article à lire. Il peut rafraîchir la liste d’articles afin d’obtenir de nouvelles recommandations.

9.2.3 Historique utilisateur long terme

Dans *CLEF NewsREEL*, l’historique utilisateur que les systèmes concurrents peuvent utiliser pour générer leur recommandation est, pour la plupart des utilisateurs, limité aux sessions de navigation sur les journaux en ligne. Dans *Renewal*, les utilisateurs sont inscrits sur l’application soit anonymement par le simple téléchargement de l’application, soit en ayant fourni leur adresse e-mail. Ainsi, il est possible d’exploiter un historique de lecture long terme, sur potentiellement plusieurs mois, ce qui n’est pas possible dans *CLEF NewsREEL*. Les systèmes de recommandation peuvent alors prendre en compte les intérêts longs termes des utilisateurs tout comme leurs intérêts courts termes tel que proposé par [An et al. \(2019\)](#).

9.2.4 Mesurer la réelle satisfaction de l’utilisateur

La mesure statistique basée sur le taux de clics (ou *click-through rate*, abrégée CTR) utilisée dans *CLEF NewsREEL* et dans la plupart des plateformes permettant l’évaluation de systèmes de recommandation d’articles d’actualité en conditions *online* ([Liu et al., 2010](#); [Li et al., 2011](#); [Garcin et al., 2014](#); [Kirshenbaum et al., 2012](#); [Maksai et al., 2015](#)) ne prend pas en compte si les utilisateurs lisent les articles cliqués ou non. Elle ne permet pas non plus de prendre en compte si les utilisateurs ont trouvé les recommandations pertinentes ([Kille et al., 2013](#); [Karimi et al., 2018](#)).

In addition, the click-through-rate might only be an indicator of the user's attention, but not necessarily a sign of genuine interest in the topic and thus proof of the recommender system's success.

Karimi et al. (2018)

Le temps de lecture (ou *dwell time*) des articles est plus adéquat lorsqu'il s'agit d'évaluer la satisfaction des utilisateurs comme démontré par Yi et al. (2014) pour la recommandation d'articles d'actualité. Cependant, cette mesure n'a pas été introduite dans *CLEF NewsREEL*.

Dwell times could offer a more accurate picture of users' preferences. Unfortunately, we cannot measure dwell times reliably. Most web sessions tend to be short and include only few articles. We cannot assure that users actually read the articles. Nonetheless, we expect users not to click on articles whose snippets they deem irrelevant.

Kille et al. (2015)

Dans la littérature du domaine de la recherche d'information et des systèmes de recommandation, peu de travaux prennent en compte l'indicateur *dwell time*. En effet, cet indicateur est plus difficile à collecter puisqu'il nécessite un traitement particulier lorsque l'utilisateur consulte un item alors que le clic est enregistré en amont et correspond à un événement ponctuel. Historiquement, Foltz (1990) fut parmi les premiers auteurs à avoir associé le *dwell time* à un retour de pertinence positif.

Other methods, such as measuring the amount of time spent per article or the number of pages read may provide less obtrusive ways of determining the user's interest.

Foltz (1990)

Cet indicateur fut ensuite utilisé en recherche d'information par, entre autres, Morita et Shinoda (1994).

Again, from our experience of reading news articles in Japanese text, as we reject uninteresting articles using our first impression of the articles, we can easily assume that articles which took considerable amount of time to read can be treated as potentially interesting articles. If we can determine whether a reader is interested in an article or not by measuring the time to read it, we might be able to capture the readers profile automatically.

Morita et Shinoda (1994)

Pour la recommandation d'articles d'actualité, le jeu de données *Adressa* (Gulla et al., 2017) permet la publication de travaux intégrant cet indicateur dans l'évaluation

des algorithmes (Wang et al., 2020). D’autres travaux exploitent cet indicateur sur des jeux de données privées comme Lu et al. (2019). Cependant, très peu de travaux de recherche (Gulla et al., 2014; Yi et al., 2014) intègrent cet indicateur dans l’évaluation *online* d’algorithmes de recommandation d’articles d’actualité.

Dans *Renewal*, nous considérons qu’un article a été apprécié lorsqu’il a été cliqué et lu suffisamment longtemps selon un seuil défini à l’avance. Afin d’éviter les faux positifs lorsque, par exemple, l’utilisateur reste sur une page web d’article sans la consulter, nous considérons uniquement le temps d’activité, c’est-à-dire le temps de *scrolling*² sur la page de l’article. Nous appelons ce retour de pertinence positif le « *Click-and-Read* ».

9.2.5 Données mises à disposition aux systèmes de recommandation concurrents

Dans la tâche de recommandation d’articles d’actualité, plusieurs travaux ont montré l’importance du contexte dans le processus de recommandation tel que le jour de la semaine (Lommatzsch et al., 2017) ou encore la localisation (Chen et al., 2017). L’application mobile *Renewal*, avec l’accord de l’utilisateur, permettra la collecte de ces informations en plus de données statiques relatives à l’utilisateur telles que l’âge et le genre. Ces données pourront être exploitées par les systèmes concurrents dans l’objectif d’améliorer la pertinence de leurs recommandations.

Dans *CLEF NewsREEL*, les systèmes concurrents n’ont accès qu’aux titres des articles d’actualité candidats à une recommandation, parfois aux résumés. Comme nous l’avons détaillé dans l’état de l’art de ce manuscrit, les articles d’actualité sont riches d’informations utiles aux méthodes basées sur le contenu. Dans *Renewal*, comme illustré par la figure 9.4, nous collectons le contenu de chaque article afin de permettre aux systèmes concurrents de se baser sur le contenu pour effectuer leur recommandation.

9.2.6 Allègement de contraintes

Pour une expérience utilisateur positive en recommandation d’articles d’actualité, il est important de répondre aux requêtes de l’utilisateur en un temps court (Kille et al., 2013). La plateforme *CLEF NewsREEL* impose aux systèmes concurrents de répondre en moins de 100 millisecondes aux requêtes des utilisateurs. Plus précisément, les systèmes ont 100 millisecondes pour envoyer les recommandations qui s’afficheront en bas de page à partir de l’instant où l’utilisateur a cliqué sur l’article consulté (Hopfgartner et al., 2016).

Afin d’éviter cette contrainte, et pour garantir une bonne scalabilité des systèmes concurrents connectés à la plateforme *Renewal*, nous proposons ces différentes solutions :

1. La liste de recommandations « suivante » (i.e. celle qui sera affichée si l’utilisateur rafraîchit la page d’accueil de l’application) est pré-chargée dans l’instance de l’application mobile de l’utilisateur. Ce mécanisme permet d’éviter d’imposer aux systèmes concurrents de répondre aux requêtes en un temps court.

2. Le « *scrolling* » est l’action de naviguer de haut en bas sur une page web.

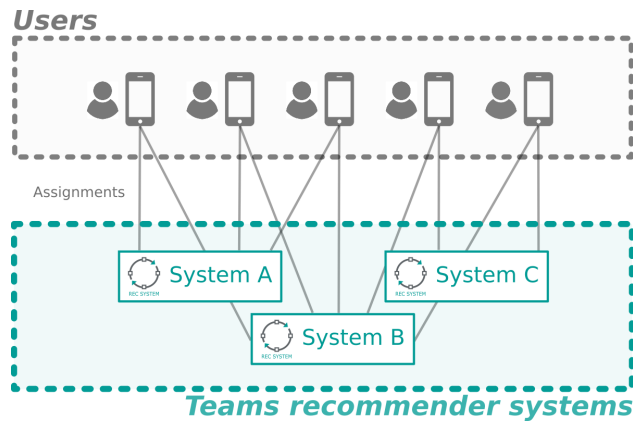


FIGURE 9.5 – Répartition des utilisateurs de l’application mobile *Renewal* aux systèmes concurrents

2. Nous implémentons plusieurs *baselines* de systèmes de recommandation dans l’objectif de remplacer les systèmes concurrents lorsqu’ils ne répondent plus aux requêtes. À noter que ces *baselines* serviront également de modèle (ou *template*) pour les équipes de recherche voulant participer aux compétitions *Renewal*.
3. Les systèmes concurrents pourront pré-calculer leurs listes de recommandations destinées aux utilisateurs. Ce mécanisme permet de ne pas contraindre les systèmes à répondre immédiatement à chaque fois qu’un utilisateur demande une nouvelle liste de recommandations. Les systèmes peuvent ainsi envoyer leurs listes de recommandations en avance et les affiner périodiquement.
4. Pour que les systèmes ne prennent pas en compte tous les utilisateurs de *Renewal*, ce qui peut être coûteux en temps de calcul si le nombre d’utilisateurs est grand, nous distribuons équitablement l’ensemble des utilisateurs aux systèmes concurrents. Nous réaffectons aléatoirement les utilisateurs aux systèmes tous les n jours, n étant ajustable. La figure 9.5 illustre la répartition des utilisateurs aux systèmes en jeu.

La figure 9.6 illustre le mécanisme de communication entre les systèmes concurrents et la plateforme *Renewal*. L’API permet aux systèmes d’obtenir des informations telles que la localisation des utilisateurs, leur âge ou encore le contenu des articles d’actualité. La queue appelée « *event stream* » permet d’informer le système en temps réel de différents événements tels que les nouvelles affectations d’utilisateurs, les nouveaux articles collectés, les retours de pertinence *Click-and-Reads* des utilisateurs, etc. Les queues « *rec requests* » et « *news rec* » permettent de recevoir des demandes de liste de recommandations pour un utilisateur en particulier et de renvoyer une liste d’articles d’actualité qui correspondra aux recommandations effectuées par le système à l’utilisateur en question.

9.3 Évaluation des systèmes concurrents

Dans les compétitions *Renewal*, nous utiliserons l’évaluation par entrelacement des recommandations comme décrit dans l’état de l’art de cette partie (chapitre 5).

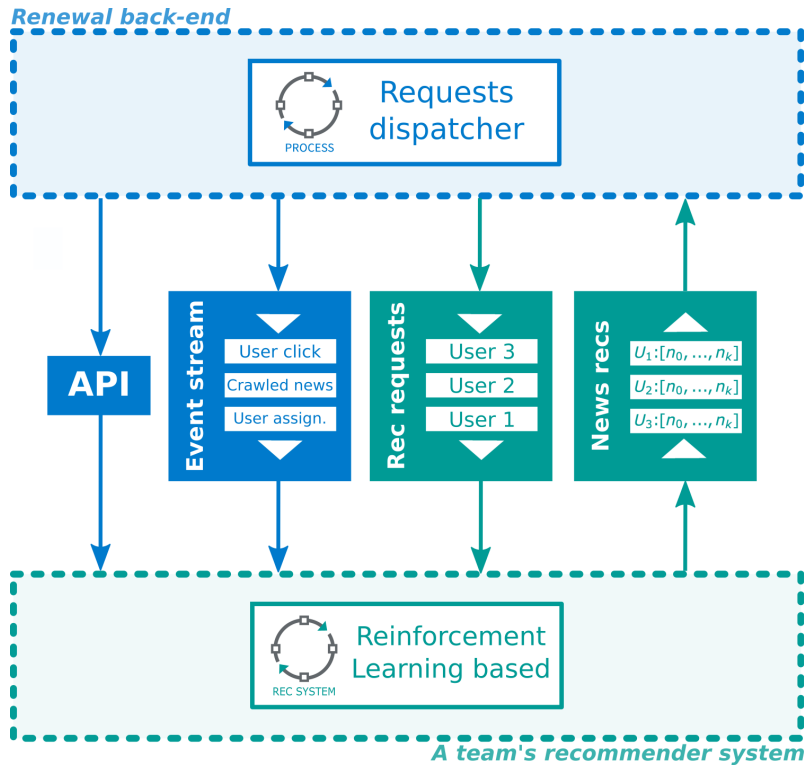


FIGURE 9.6 – Communication entre les systèmes concurrents et la plateforme *Renewal*

Cette méthode a montré être plus efficace que la méthode de test A/B utilisant des mesures statistiques telles que CTR lorsqu’il s’agit de comparer la performance de différents systèmes (Schuth et al., 2015; Kharitonov et al., 2015). En effet, l’évaluation par entrelacement est moins impactée, entre autres, par :

Les habitudes des utilisateurs Un utilisateur pourra cliquer sur une grande proportion des articles qui lui sont recommandés (comportement « actif »), et d’autres sur une petite proportion (comportement « passif »).

Le contexte d’interaction Le moment de la journée, le jour de la semaine, les lieux (domicile, travail) feront fluctuer la consommation des utilisateurs.

Pour que l’évaluation des systèmes soit statistiquement fiable, un scénario de test A/B nécessitera un plus grand nombre d’interactions utilisateurs-système étant donné ces biais (Hofmann, 2015). Par exemple, si un système recommande plus souvent des articles à des utilisateurs qui ont un comportement « actif », il sera avantagé puisque générera plus de clics. Il est donc nécessaire, pour limiter ces effets, d’augmenter le nombre de recommandations générées et d’assigner aléatoirement les utilisateurs aux systèmes, à des moments aléatoires. L’évaluation par entrelacement est moins impactée par ces biais étant donné que les systèmes sont évalués conjointement et partagent le contexte d’interaction et les habitudes de l’utilisateur lorsqu’une liste entrelacée lui est présentée.

Comme l’illustre la figure 9.7, l’entrelacement consiste à afficher une recommandation d’un système *A*, puis une autre d’un système *B* et ainsi de suite. Sur cette figure, le système *A* reçoit plus de *Click-and-Reads* que le système *B* et sera donc considéré comme plus performant dans cette confrontation.

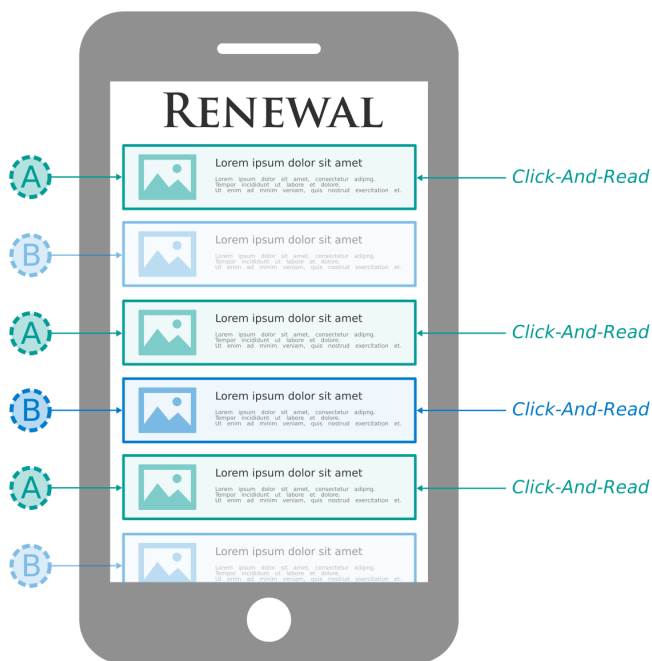


FIGURE 9.7 – Illustration de l’application mobile de *Renewal* et de l’entrelacement de recommandations provenant de deux systèmes concurrents

Étant donné que l’évaluation par entrelacement consiste en un enchaînement de « confrontations » (i.e. un système est confronté à un autre), nous pouvons utiliser un « système d’évaluation un contre un ». Un « système d’évaluation » permet de calculer le classement des concurrents en jeu, celui-ci étant modifié à chaque résultat d’une confrontation. Pour *Renewal*, nous utiliserons la bibliothèque *TrueSkill*³ (Herbrich et al., 2007) qui implémente un système d’évaluation proche du « classement Elo ».

Plus concrètement, une confrontation correspondra à ces différentes étapes :

1. Un utilisateur demande une liste de recommandations via l’application mobile *Renewal*.
2. Les systèmes *A* et *B* lui étant affectés dans la fenêtre de jours courante envoient chacun un nombre n d’articles d’actualité (représentés par leur identifiant).
3. Nous effectuons un entrelacement E des deux listes de recommandations par la méthode *Team Draft* proposée par Radlinski et al. (2008) et implémentée dans la bibliothèque *Interleaving*⁴. Les listes peuvent avoir une intersection non nulle. En conséquence, la liste résultante générée par la méthode *Team Draft* est de taille n' avec $n \leq n' \leq 2n$.
4. Nous récoltons les retours de pertinence *Click-and-Reads* de l’utilisateur. Ces retours de pertinence sont enregistrés en temps réel dans la base de données

3. La bibliothèque *TrueSkill*, créée par Microsoft et disponible à l’adresse <https://trueskill.org>, implémente un système d’évaluation généralisant le classement Elo. Il permet de prendre en compte des confrontations à plusieurs joueurs, de calculer des classements individuels à partir de confrontations en équipes, etc.

4. La bibliothèque *Interleaving*, disponible à l’adresse <https://github.com/mpkato/interleaving>, permet d’entrelacer des items renvoyés par différents systèmes et l’évaluation de ceux-ci selon les clics des utilisateurs. Elle utilise différentes stratégies, et notamment la stratégie *Team Draft* proposée par Radlinski et al. (2008).

de *Renewal*.

5. Lorsque la plateforme *Renewal* observe que l'utilisateur a terminé son interaction avec la liste de recommandations entrelacées E , nous calculons le résultat de la confrontation A contre B . Pour déterminer si l'utilisateur a terminé d'interagir avec la liste E , nous utilisons des heuristiques telles que le nombre de nouvelles listes entrelacées générées après E , ou encore le nombre de jours écoulés depuis le dernier « survol » de la liste E par l'utilisateur. Afin de calculer le résultat d'une confrontation à partir des *Click-and-Reads*, nous utilisons la méthode *Team Draft*. L'issue de la confrontation peut être en faveur de A , de B , ou correspondre à une égalité.
6. Nous ajoutons l'issue de la confrontation dans le système d'évaluation *TrueSkill* afin de générer un nouveau classement des systèmes concurrents.

Outre le classement généré par le système d'évaluation *TrueSkill*, il est possible d'obtenir une valeur de « compétence » (ou *skill*) et une valeur de confiance (qui dépend du nombre de confrontations effectuées) pour chaque système. Ainsi, nous pouvons afficher conjointement le classement des systèmes, leur compétence et la confiance accordée à ces valeurs. Un site web *Renewal* sera dédié à l'affichage de ces statistiques afin que les équipes de recherche puissent observer les performances de leur système en temps réel.

9.4 Prédiction des conditions nécessaires par simulation de compétitions

9.4.1 Motivation

L'intérêt d'une compétition organisée grâce à la plateforme *Renewal* dépend de deux facteurs :

1. notre capacité à établir un classement fiable des systèmes de recommandation concurrents ;
2. notre capacité à établir ce classement en un temps limité, par exemple dans le temps imparti avant l'affichage du classement dans le cadre d'une conférence scientifique.

Ce classement est déterminé par la performance des systèmes de chaque équipe. Intuitivement, le temps nécessaire pour obtenir un classement fiable dépendra du nombre de confrontations observées par la plateforme *Renewal*. Ainsi, nous proposons de répondre aux questions suivantes :

1. Afin d'effectuer le classement d'un certain nombre de systèmes, quel est le nombre minimum d'utilisateurs nécessaires ? Une réponse à cette question nous permettra d'avoir une idée du nombre d'utilisateurs actifs nécessaires avant l'organisation d'une compétition. Par exemple, si 100 systèmes sont en jeu et que seulement 100 utilisateurs sont actifs sur l'application mobile, il est probable que le classement obtenu ne soit pas fiable, ou ne soit fiable qu'avec un grand nombre de jours de compétition.
2. Combien de jours de compétition sont nécessaires pour une convergence du classement au plus proche du classement véritable terrain (i.e. le véritable classement) ?

3. Quel est l'impact de la différence de performance des systèmes ? Autrement dit : si les systèmes ont des performances proches, est-il possible de converger vers un classement proche de la vérité terrain ? Une réponse à cette question nous permettra de connaître dans quelles conditions il est possible de converger vers un classement fiable. En effet, si tous les systèmes ont des performances proches, alors il sera difficile d'établir un classement fiable et fixe dans le temps. Au contraire, si les différences de performance des systèmes sont significatives, alors il est probable que le classement généré converge vers le classement vérité terrain.
4. Quel est l'impact de l'affectation partielle (liée à l'allègement de contraintes décrit en section 9.3) des utilisateurs aux systèmes ? En effet, avant chaque réaffectation des utilisateurs, il n'est pas garanti d'obtenir un classement fiable des systèmes étant donné que :
 - certains n'auront pas été confrontés (i.e. nous n'affecterons pas nécessairement toutes les paires de systèmes possibles aux utilisateurs) ;
 - chaque système n'aura pas recommandé d'items à tous les utilisateurs.

Pour répondre à ces questions, nous proposons d'effectuer différentes simulations de compétitions *Renewal*⁵, chacune ayant des conditions initiales différentes.

9.4.2 Paramètres initiaux des simulations

Pour cette simulation, nous simplifions la procédure d'évaluation détaillée en section 9.3 : nous ne simulerons pas l'entrelacement d'items mais générerons uniquement des « résultats de confrontation », c'est-à-dire une victoire du premier système, du deuxième ou une égalité. La première étape d'une simulation de compétition est de générer un ensemble de systèmes de recommandation concurrents. Chaque système est représenté par sa probabilité de gagner contre un autre système pris au hasard. Lors de l'exécution d'une simulation, nous devons choisir :

- n*** Le nombre de systèmes concurrents.
- w*** Un intervalle de valeurs qui indiquera les bornes inférieures et supérieures de leur probabilité de gagner. Ces valeurs doivent être prises au hasard dans cet intervalle appelé « intervalle de probabilité de victoire » (ou *win probability interval*, abrégé WPI). Cet intervalle prendra par exemple la valeur 0.6, ce qui signifiera que les probabilités de victoire des systèmes seront toutes entre 0.2 et 0.8.
- d*** Un intervalle de probabilité d'égalité (ou *draw probability interval*) déterminant l'égalité entre deux systèmes concurrents.

Il s'agit alors, à partir des paramètres initiaux n , w et d , de générer une « matrice de confrontations » P (ou *pairwise win probability matrix*) aléatoire. Cette matrice indique, pour chaque paire d'indices de systèmes i et j , la probabilité pour que le système dont l'indice est i gagne contre le système dont l'indice est j . Ainsi, si par exemple P_{ij} a pour valeur 0.8, cela signifiera que le système ayant l'indice i a une probabilité de 0.8 de gagner contre le système ayant l'indice j . Nous proposons l'algorithme 9 permettant la génération de cette matrice. Il consiste à effectuer une

5. Le code source de la simulation est disponible à l'adresse <https://github.com/hayj/RenewalSimulator>.

Algorithm 9 Algorithme de génération de matrice de confrontations

```
1: procedure GENERATEWPM( $n$  : float,  $w$  : float,  $d$  : float)
2:    $pw \leftarrow null$  ▷ Predicted win probability interval
3:    $cw \leftarrow 0.5$  ▷ Current win probability interval
4:    $shift \leftarrow 0.25$  ▷ Current dichotomic search shift
5:    $\epsilon \leftarrow 0.01$  ▷  $\epsilon$  for the terminating condition
6:   while  $pw$  is null or  $|w - pw| > \epsilon$  do
7:     if  $pw$  is not null then
8:       if  $pw > w$  then
9:          $cw \leftarrow cw - shift$ 
10:      else
11:         $cw \leftarrow cw + shift$ 
12:         $shift \leftarrow \frac{shift}{2}$ 
13:       $wp \leftarrow$  list of zeros such that  $|wp| = n$ 
14:      for  $i \leftarrow 1$  to  $n$  do
15:         $wp_i \leftarrow$  random value  $r$  such that  $0.5 - \frac{cw}{2} \leq r \leq 0.5 + \frac{cw}{2}$ 
16:       $P \leftarrow$  matrix of size  $n \times n$  such that  $P_{ij} = \frac{wp_i \cdot (1 - wp_j)}{wp_i \cdot (1 - wp_j) + (1 - wp_i) \cdot wp_j}$ 
17:       $wins, defs \leftarrow$  lists of zeros of size  $n$ 
18:      for  $i \leftarrow 1$  to  $100n$  do
19:         $a, b \leftarrow$  random indexes in  $[1, n]$ 
20:         $outcome \leftarrow match(a, b, P, d)$ 
21:        if  $outcome = 1$  then
22:           $wins_a, defs_b \leftarrow wins_a + 1, defs_b + 1$ 
23:        else if  $outcome = -1$  then
24:           $wins_b, defs_a \leftarrow wins_b + 1, defs_a + 1$ 
25:       $pwp \leftarrow$  list of zeros such that  $|pwp| = n$ 
26:      for  $i \leftarrow 1$  to  $n$  do
27:         $pwp_i \leftarrow \frac{wins_i}{wins_i + defs_i}$ 
28:       $pw \leftarrow \max(pwp) - \min(pwp)$ 
29:    return  $P$ 
30: end procedure
```

recherche dichotomique entre 0 et 1 d'une valeur cw (initialisée à 0.5) jusqu'à trouver une matrice de confrontations P respectant un intervalle de probabilité de victoire observé pw proche de l'intervalle de probabilité de victoire w donné en paramètres.

La fonction *match* utilisée dans l'algorithme 9 est une fonction permettant, à partir d'une matrice de confrontations P et d'une valeur d (l'intervalle de probabilité d'égalité), d'obtenir le résultat d'une confrontation entre deux systèmes. L'algorithme 10 donne le pseudo-code de cette fonction. Une valeur retournée de 1 signifie que le premier système gagne la confrontation, une valeur de -1 signifie que le deuxième système gagne et une valeur de 0 signifie une égalité entre les deux systèmes.

Pour exécuter une simulation, après avoir choisi les paramètres initiaux n , w et d , il est nécessaire de choisir les paramètres initiaux suivants :

nmatches Le nombre de confrontations par jour par utilisateur. Pour nos simulations, nous avons fixé cette valeur à 2.

u Le nombre d'utilisateurs actifs. Un utilisateur actif est un utilisateur qui

Algorithm 10 Fonction de confrontation

```
1: procedure MATCH( $a$  : integer,  $b$  : integer,  $P$  : matrix,  $d$  : float)
2:    $r \leftarrow$  random float in  $[0, 1]$ 
3:   if  $P_{ab} - \frac{d}{2} \leq r \leq P_{ab} + \frac{d}{2}$  then
4:     return 0
5:   else if  $r < P_{ab}$  then
6:     return 1
7:   else
8:     return  $-1$ 
9: end procedure
```

Algorithm 11 Algorithme d'affectation aléatoire des utilisateurs aux systèmes

```
1: procedure ASSIGN( $u$  : integer,  $n$  : integer)
2:    $assign \leftarrow$  empty dictionary
3:    $sys \leftarrow$  shuffle( $(i$  for  $i$  in  $1$  to  $n$ ))  $\triangleright$  Shuffled list of systems' indices
4:    $sys \leftarrow sys +$  shuffle( $(i$  for  $i$  in  $1$  to  $n$ ))
5:   for  $j \leftarrow 1$  to  $u$  do
6:      $s1 \leftarrow$  pop an element from  $sys$ 
7:      $s2 \leftarrow$  pop an element from  $sys$ 
8:     while  $s1 = s2$  do
9:        $s2 \leftarrow$  pop an element from  $sys$ 
10:     $assign[j] = \{s1, s2\}$ 
11:    if  $|sys| < 3n$  then
12:       $sys \leftarrow sys +$  shuffle( $(i$  for  $i$  in  $1$  to  $n$ ))
13:  return  $assign$ 
14: end procedure
```

interagit avec $nmatchs$ listes entrelacées par jours, conduisant à $nmatchs$ confrontations par jours.

window Le nombre de jours avant une réaffectation des utilisateurs aux systèmes. Pour nos simulations, nous avons fixé cette valeur à 4. L'algorithme 11 donne le pseudo-code permettant une affectation des utilisateurs aux systèmes. Les propriétés de cet algorithme sont les suivantes : les affectations sont aléatoires et les utilisateurs sont répartis équitablement entre systèmes. Ainsi, les systèmes auront un nombre d'utilisateurs proche de $\frac{2u}{n}$. À noter que cette fonction n'est pas seulement utilisée pour les simulations mais aussi dans la plateforme *Renewal*.

9.4.3 Simulation de compétitions

La simulation d'une compétition consiste à effectuer des confrontations aléatoires en réutilisant la fonction de confrontation dont le pseudo-code est donné dans l'algorithme 10 et en respectant la procédure décrite dans les sections précédentes. Après chaque confrontation, nous ajoutons le résultat au système d'évaluation *TrueSkill*. Celui-ci nous permet d'obtenir un classement que nous comparons au classement vérité terrain obtenu grâce aux valeurs de probabilité de victoire de chaque système en jeu.

Il existe deux manières de représenter un classement :

Vecteur d'ordres Un « vecteur d'ordres » (ou *rank vector*) est un vecteur qui associe à chaque item (en coordonnée du vecteur) son rang (en valeur dans le vecteur). Par exemple, le vecteur (2, 1, 3) indique que l'item 0 (i.e. dont la coordonnée est 0) est au rang 2, l'item 1 au premier rang et l'item 2 au dernier rang.

Ordonnement Un « ordonnancement » (ou *ranking*) est un vecteur qui associe à chaque rang (en coordonnée du vecteur) l'identifiant d'un item (en valeur dans le vecteur). Par exemple, le vecteur (1, 0, 2) indique que l'item ayant l'identifiant 1 est au premier rang, l'item ayant l'identifiant 0 est au second rang et l'item ayant l'identifiant 2 est au dernier rang.

Dans cette section ainsi que dans notre implémentation, nous utiliserons uniquement des vecteurs d'ordres.

Lors des simulations, afin de calculer la similitude entre les vecteurs d'ordres prédits et le vecteur d'ordres vérité terrain, nous utilisons le tau de Kendall (Kendall, 1938). Le tau de Kendall nous permet de mesurer la corrélation entre deux vecteurs d'ordres et s'obtient en soustrayant le nombre de paires discordantes au nombre de paires concordantes sur le nombre total de paires :

$$\text{K.'s } \tau = \frac{\# \text{concordants} - \# \text{discordants}}{\frac{1}{2} \cdot n \cdot (n - 1)} \quad (9.1)$$

Une paire concordante est une paire d'items (en l'occurrence de « systèmes » dans le cas de *Renewal*) qui ont le même « ordre » dans les deux vecteurs d'ordre. Par exemple, pour que la paire de système *A* et *B* soit concordante, si le système *A* a un meilleur rang que le système *B* dans le premier vecteur d'ordre, alors le système *A* doit aussi avoir un meilleur rang que le système *B* dans le deuxième vecteur d'ordres.

Nous utilisons l'implémentation du tau de Kendall disponible dans la bibliothèque *SciPy*⁶ prenant en compte les égalités comme proposé par Kendall (1945) suite à son premier travail sur cette mesure de corrélation. Le tableau 9.1 donne quelques exemples de tau de Kendall calculés sur un vecteur d'ordres de référence (première partie du tableau) et différents vecteurs d'ordres (seconde partie du tableau). L'égalité de classement entre deux items se traduit, dans un vecteur d'ordres, par une valeur identique. Par exemple, dans le deuxième vecteur d'ordres comparé ayant un tau de Kendall de 0.9, deux items ont le rang 5.

Ces exemples nous permettront d'interpréter les résultats de simulations afin d'avoir une idée de la précision du classement qu'il est possible d'obtenir en conditions réelles. Comme le montre ce tableau, un tau de Kendall de 1 correspond à un ordre d'items identique et un tau de Kendall de -1 correspond à un ordre inverse. Sur ces exemples, un tau de Kendall de 0.8 semble correspondre à un classement des items proche du classement de référence. En dessous d'un tau de Kendall de 0.6, le classement est fortement dégradé. Nous pouvons alors considérer qu'un tau de Kendall inférieur à 0.6 ne permet pas d'obtenir un classement fiable des systèmes concurrents.

Ground truth rank vector										
1	2	3	4	5	6	7	8	9	10	
Assessed rank vectors										K.'s τ
1	2	3	4	5	6	7	8	9	10	1
1	2	3	5	4	6	5	7	8	9	0.9
1	3	2	6	4	5	5	7	8	9	0.8
1	5	2	3	4	4	7	8	6	8	0.7
1	2	3	4	10	7	5	8	9	6	0.6
1	7	2	4	3	2	5	9	6	8	0.5
2	3	1	8	7	7	4	5	6	9	0.4
2	8	3	5	4	4	1	6	7	7	0.3
7	1	6	2	9	3	10	4	8	5	0.2
9	1	3	5	8	2	6	7	4	4	0
8	5	1	6	2	8	4	3	8	7	0.1
9	6	8	3	6	4	5	2	7	1	-0.5
10	9	8	7	6	5	4	3	2	1	-1

TABLE 9.1 – Exemples de vecteurs d’ordres et leur tau de Kendall lorsque comparés à un vecteur d’ordres de référence

Id	<i>Initial settings</i>				<i>Average simulation results</i>		
	#simu	#users	#sys	WPI	K.'s τ	#days	#matches
1	400	100	10	0.6	0.87	2.7	541
2	400	1000	10	0.6	0.88	0.28	560
3	400	100	100	0.6	0.84	30.61	6122
4	400	1000	100	0.6	0.84	2.78	5570
5	400	100	10	0.4	0.8	3.08	617
6	400	1000	10	0.4	0.81	0.31	629
7	400	100	100	0.4	0.76	35.07	7014
8	400	1000	100	0.4	0.76	3.36	6721
9	400	100	10	0.2	0.63	4.25	851
10	400	1000	10	0.2	0.63	0.45	906
11	400	100	100	0.2	0.52	38.88	7777
12	400	1000	100	0.2	0.52	3.78	7568

TABLE 9.2 – Tau de Kendall, nombre de jours de simulation et nombre de confrontations obtenus après convergence du tau de Kendall selon différents paramètres initiaux

9.4.4 Résultats des simulations

Le tableau 9.2 montre les résultats obtenus pour 12 ensembles de paramètres initiaux différents. Chaque ligne du tableau correspond à un paramétrage spécifique. Pour chaque ligne, les scores et statistiques obtenus sont des moyennes pour 400 simulations. Les simulations de 1 à 4 ont un WPI de 0.6. Avec ce WPI, le classement des systèmes concurrents est plus facile à obtenir étant donné que ceux-ci ont des différences de performances très significatives. Les simulations de 5 à 8 ont un WPI de 0.4 et les simulations de 9 à 12 ont un WPI de 0.2. Nous avons fait varier le nombre de systèmes de 10 à 100 et le nombre d'utilisateurs de 100 à 1000. Les tau de Kendall en partie droite de tableau sont la moyenne des tau de Kendall obtenus à convergence, c'est-à-dire lorsque le tau de Kendall est supérieur au tau de Kendall maximum obtenu soustrait à un epsilon (fixé à 0.05 dans notre cas). Le nombre de jours et le nombre de confrontations sont aussi affichés en partie droite du tableau et correspondent aux valeurs obtenues à convergence.

La figure 9.8 affiche, pour chaque ensemble de simulations, une courbe bleue correspondant à la moyenne des tau de Kendall en fonction du jour de simulation. Les lignes verticales en gris et pointillées correspondent aux réaffectations des utilisateurs. Pour différentes simulations ayant les mêmes conditions initiales, la distribution des tau de Kendall, à un jour donné, suit une loi normale. Nous affichons donc les courbes rouges supérieures et inférieures correspondant aux moyennes à plus ou moins deux écarts-types. Elles indiquent l'intervalle de valeurs possibles de la majorité (95%) des tau de Kendall pour le jour considéré en abscisse.

Ces résultats montrent que le temps nécessaire à une convergence du tau de Kendall, c'est-à-dire le temps nécessaire à l'obtention d'un classement fiable, semble dépendant du ratio $\frac{u}{n}$. Par exemple, si ce ratio est de 10, signifiant qu'il y a 10 fois plus d'utilisateurs actifs que de systèmes de recommandation en jeu, la convergence est rapide : entre 1 et 4 jours. En revanche, dans le cas extrême d'un ratio de 1, signifiant qu'il y a autant d'utilisateurs actifs que de systèmes de recommandation en jeu, la convergence sera longue : entre 30 et 50 jours. Ainsi, nous avons répondu à la première et deuxième question posées en section 9.4.1 : la convergence des classements est possible dans un temps raisonnable. Il est possible d'organiser des compétitions *Renewal* même dans les cas où les conditions expérimentales sont extrêmement défavorables.

L'obtention d'un classement fiable semble dépendre uniquement de la différence de performance des systèmes (i.e. la valeur WPI). Un WPI de 0.4 et de 0.6 permet d'établir un classement fiable avec un tau de Kendall supérieur à 0.8. Dans le cas extrême d'un WPI de 0.2, signifiant que les systèmes ont des performances proches, le classement des systèmes ne peut pas être établi de manière fiable : le tau de Kendall ne dépasse pas 0.63. Nous avons donc répondu à la troisième question : la différence de performance des systèmes a un impact sur la fiabilité des classements. Une convergence est possible mais le tau de Kendall obtenu ne correspond pas au classement vérité terrain dans le cas d'un WPI de 0.2.

Cependant, en conditions réelles, si la différence de performance des systèmes est faible, alors il n'est pas pertinent d'établir un classement de ceux-ci. Il s'agit alors d'identifier ces cas particuliers en analysant, par exemple, les fluctuations du classement dans le temps. Une grande fluctuation du classement peut être indicateur

6. *SciPy* est une bibliothèque *Python* gratuite et open-source utilisée pour le calcul scientifique.

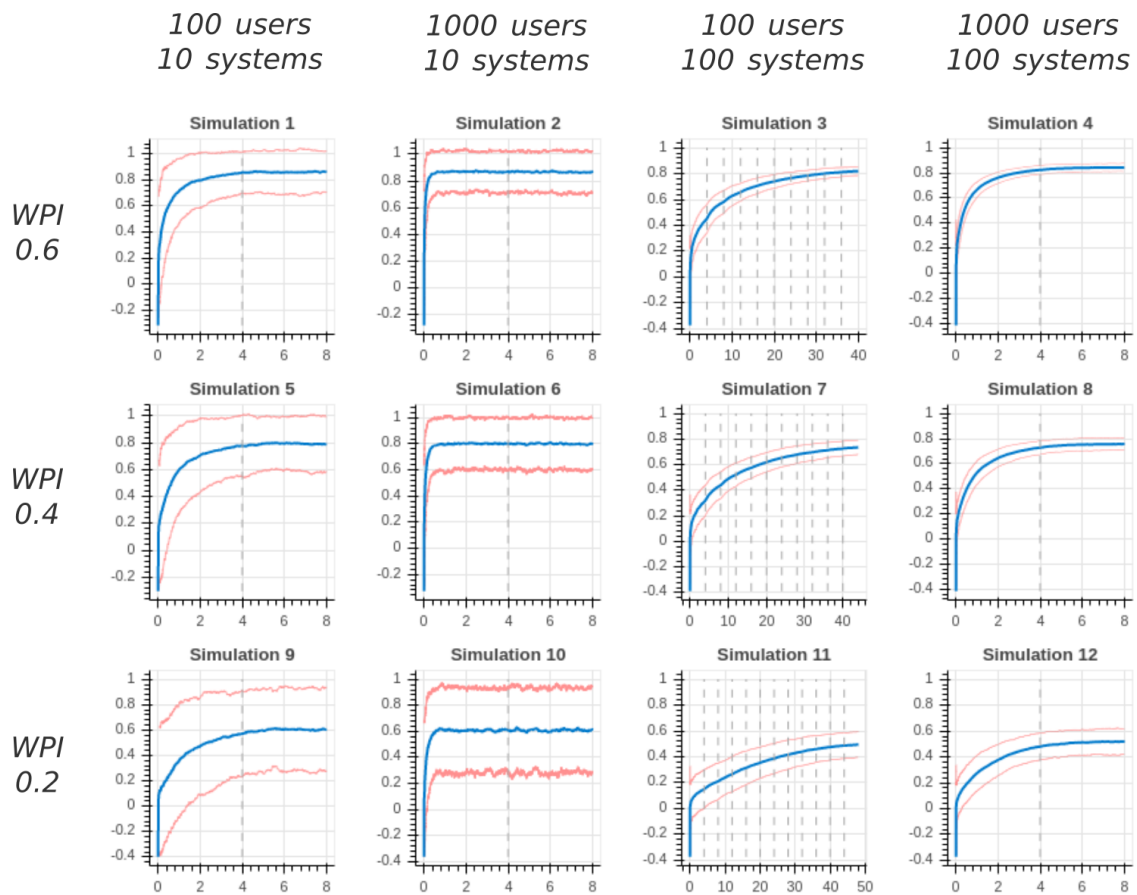


FIGURE 9.8 – Courbes moyennes du tau de Kendall en fonction du nombre de jours de simulation pour 12 simulations ayant des paramètres initiaux différents

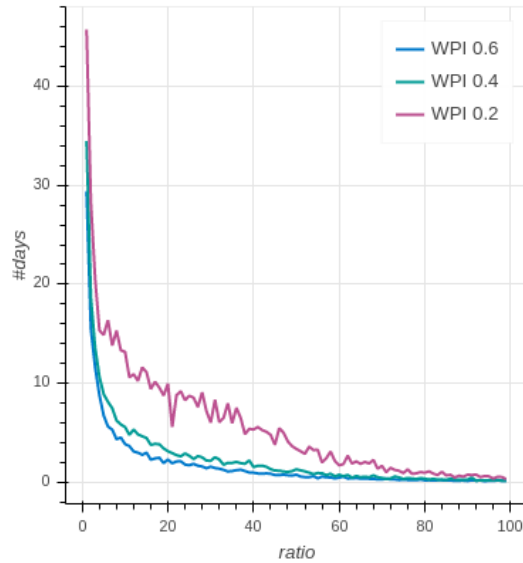


FIGURE 9.9 – Nombre de jours nécessaires à l’obtention d’une convergence du classement en fonction du ratio $\frac{u}{n}$ pour trois WPI différents

d’un classement de faible fiabilité. En perspective de ce travail, il pourrait être intéressant d’étudier la corrélation entre le tau de Kendall et la fluctuation du classement afin de pouvoir prédire une fiabilité en conditions réelles (le tau de Kendall étant inconnu et uniquement utilisé dans le cadre d’une simulation).

Pour terminer, les affectations partielles des utilisateurs aux systèmes n’ont que peu d’impact sur le tau de Kendall. Il est possible de voir une légère augmentation du tau de Kendall après réaffectation dans les simulations 3, 7 et 11 sur la figure 9.8. Ainsi, nous avons répondu à la quatrième et dernière question : la fluctuation du tau de Kendall est prévisible comme détaillé en section 9.4.1, mais celle-ci est faible.

9.4.5 Nombre de jours nécessaires en fonction du ratio utilisateurs-systèmes

La figure 9.9 montre le nombre de jours nécessaires à l’obtention d’une convergence du classement en fonction du ratio utilisateurs-systèmes, i.e. $\frac{u}{n}$. Chaque point sur ces trois courbes correspond à la moyenne de 100 simulations dont le nombre d’utilisateurs a été choisi aléatoirement entre 10 et 1 000. Le ratio en abscisse permet d’obtenir le nombre de systèmes utilisés dans la simulation.

Comme nous l’avons vu dans la section précédente, le nombre de jours de convergence semble être corrélé au ratio $\frac{u}{n}$. Cette figure confirme cette intuition puisque montre une décroissance rapide du nombre de jours de convergence lorsque ce ratio croît. Un nombre d’utilisateurs 5 fois supérieur au nombre de systèmes en jeu permet une convergence en moins de 15 jours, quel que soit le WPI. Un nombre d’utilisateurs égal au nombre de systèmes en jeu, c’est-à-dire un ratio $\frac{u}{n} = 1$, permet une convergence entre 25 et 45 jours en fonction du WPI. Cette figure montre qu’il semble préférable de se limiter à un nombre de systèmes 5 fois plus faible que le nombre d’utilisateurs actifs sur l’application, mais aussi que, dans le cas de conditions extrêmement défavorables, l’organisation d’une compétition *Renewal* reste possible.

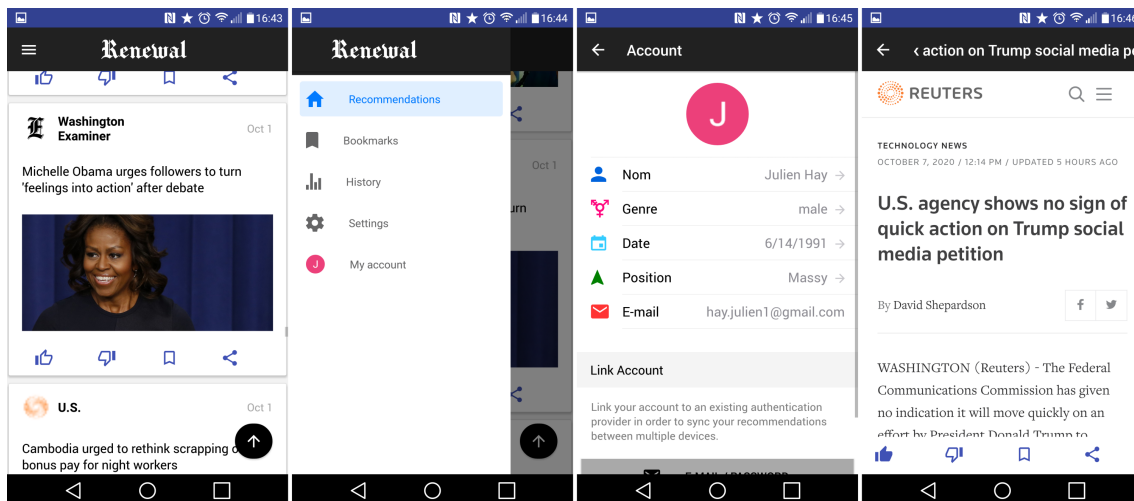


FIGURE 9.10 – Captures d'écran de l'application mobile *Renewal*

9.5 Conclusion

En plus de permettre à la communauté de recherche de tester leurs algorithmes de recommandation, cette plateforme nous permettra d'évaluer l'efficacité, en conditions réelles, des algorithmes et des représentations vectorielles proposés dans cette thèse. Cette plateforme permet aussi de pallier tous les biais que nous avons détaillés dans ce manuscrit liés à l'évaluation *offline* et l'évaluation qualitative des systèmes de recommandation.

La figure 9.10 montre des captures d'écrans de l'application mobile *Renewal*. À ce jour, l'application est en version bêta⁷ et testée en interne au Laboratoire Interdisciplinaire des Sciences du Numérique (LISN). Le code source de la plateforme et de tous ses composants est rendu disponible sur GitHub⁸.

Ce projet offre aussi l'opportunité d'étudier différentes problématiques :

- l'étude des fausses nouvelles ou de la qualité des articles d'actualité par étiquetage participatif (ou *crowd-labeling*) sur l'application mobile *Renewal* ;
- l'étude du problème de démarrage à froid utilisateur et item ;
- la recommandation d'articles d'actualité complémentaires lorsque l'utilisateur termine la lecture d'un article.

7. La version « bêta » d'une application correspond à une version d'essai avant sa publication.

8. Le dépôt de *Renewal* est disponible à l'adresse <https://github.com/RenewalResearch>.

Conclusion

Pour conclure, nous proposons de revenir sur les travaux de recherche effectués durant cette thèse ainsi que les cinq questions de recherches auxquelles nous avons répondu. Dans la première partie de cette thèse, nous avons montré qu’il est possible d’exploiter un corpus de référence pour l’apprentissage de la représentation du style écrit. Nous avons, pour cela, défini la méthode de ciblage des indices intra-auteurs consistants au chapitre 2. Cette méthode consiste à entraîner un réseau de neurones profond intégrant un mécanisme d’attention. Ce mécanisme permet au réseau de neurones de se concentrer sur un sous-ensemble des mots pour l’encodage des documents. En se basant sur la *consistance intra-auteur*, une propriété du style indiquant que les structures linguistiques relevant du style sont consistantes dans les écrits d’un même auteur, nous avons entraîné ce réseau de neurones sur un ensemble d’auteurs de référence. Nous avons émis l’hypothèse qu’un tel réseau de neurones, s’il peut identifier les auteurs d’un large corpus de référence, permet la représentation « stylométrique » de documents « nouveaux », c’est-à-dire ayant été écrits par des auteurs inconnus : c’est l’hypothèse de *généralisation du style*. Ces documents seraient alors représentés dans un espace stylométrique de référence suffisamment général pour que deux documents nouveaux d’un auteur inconnu soient proches dans cet espace. L’intuition ayant motivé cette méthode est que chaque auteur « nouveau » écrit en employant un style proche d’un sous-ensemble consistant des auteurs de référence. Ainsi, il est possible de généraliser l’extraction de caractéristiques de style en projetant les écrits d’auteurs inconnus dans un espace appris sur la base d’auteurs de référence.

Les expérimentations effectuées au chapitre 4 valident cette hypothèse. Nous avons montré que notre méthode, lorsqu’appliquée au modèle *DBert-ft*, est capable de partitionner des jeux de données composés de nouveaux documents de manière plus précise que des modèles de représentation standards tels que *Doc2Vec*, *TFIDF*, *InferSent* ou encore des modèles basés sur la modélisation de thème (ou *topic models*). Dans ce chapitre, nous avons également montré que les représentations vectorielles issues de nos modèles permettent d’obtenir de meilleures performances sur la tâche d’identification d’auteur. La seconde propriété des structures linguistiques relevant du style est la *non-spécificité sémantique*. Cette propriété peut être vérifiée grâce à la mesure *TFIDF focus* ainsi que la couche d’attention des modèles entraînés par la méthode de ciblage des indices intra-auteurs consistants. Nous avons montré que notre modèle focalise son attention sur des mots ayant un poids TFIDF faible, c’est-à-dire les mots « fonctions ». L’extraction de la représentation des documents par nos modèles se base donc effectivement sur le style écrit, i.e. les mots non-spécifiques aux documents qui n’apportent que peu d’information sur le thème, les entités, etc.

Au chapitre 2, nous avons également proposé d’améliorer la méthode de ciblage des indices intra-auteurs consistants par le filtrage des indices trop « révélateurs » des

auteurs de référence. Nous avons émis l’hypothèse que rendre difficile la reconnaissance des auteurs dans le corpus de référence stylistique permettrait aux réseaux de neurones profonds entraînés par la méthode de ciblage des indices intra-auteurs consistants de se focaliser sur des indices plus subtils en accord avec la *consistance intra-auteur* et la *non-spécificité sémantique* : c’est l’hypothèse de *filtrage*. Au chapitre 3, nous avons proposé une méthode permettant une élimination exhaustive et efficiente des phrases contenant des *n-grams* révélateurs d’un auteur de référence. Puis, au chapitre 4, nous avons démontré que cette méthode permet d’améliorer la qualité des représentations pour le partitionnement d’auteur, l’identification d’auteurs, mais aussi permet d’augmenter l’adéquation des modèles avec la propriété de *non-spécificité sémantique*.

QR1 *Est-il possible d’utiliser le volume conséquent de données disponibles sur le web pour améliorer l’extraction de caractéristiques stylométriques en exploitant les annotations de type auteur et source (e.g. journal en ligne, blog) ?*

QR2 *Peut-on entraîner un modèle de représentation à projeter un article d’actualité ou de blog dans un espace stylométrique général ?*

Le corpus de référence que nous avons utilisé dans cette thèse est composé d’articles d’actualité et de blog. Nous avons exploité à la fois le texte de ces articles, les labels auteur et les noms de domaine disponibles. Dans la littérature, et comme expliqué en section 1.3.5, aucun travail de recherche ne propose l’exploitation de ce type de données ainsi que de leurs labels pour l’apprentissage de la représentation du style écrit. Ces documents labélisés ont pour avantage d’être abondants sur le web, ce qui nous a permis d’écarter le processus coûteux d’annotation manuelle. Nous avons pu éviter un surapprentissage lors de l’entraînement de nos modèles en exploitant un jeu de données suffisamment large avec un grand nombre de classes et un grand nombre d’exemples par classe. Les expérimentations du chapitre 4 répondent donc à la question de recherche *QR1*. Les résultats obtenus par le modèle *DBert-ft* au chapitre 4 répondent également à la question de recherche *QR2* car permet un meilleur partitionnement des documents d’auteurs inconnus comparé aux modèles standards. À noter que la collecte et le traitement des données présentés au chapitre 3 ainsi que la méthode de filtrage proposée sont suffisamment génériques pour être utilisés sur d’autres sources de données (e.g. littérature, fanfictions).

L’une des principales hypothèses de cette thèse ayant motivé nos travaux est que les caractéristiques stylométriques peuvent apporter une dimension supplémentaire dans la représentation des utilisateurs par leurs lectures, et que ces représentations permettraient une amélioration de la performance des algorithmes de recommandation d’articles d’actualité. La seconde partie de cette thèse valide cette hypothèse. Dans cette partie, nous avons exploité la représentation stylométrique pour la recommandation d’articles d’actualité, une limite de l’état de l’art que nous avons détaillée en section 5.4.5. Aux chapitres 6 et 8, nous avons proposé les algorithmes :

SimRec Algorithme permettant de recommander des items à partir de leur représentation vectorielle ainsi que la prise en compte du *ratio historique*.

Reswhy Algorithme permettant de combiner des ordonnancements en effectuant une pondération et un rééchelonnage. Le rééchelonnage *gmrf* proposé dans cette thèse permet de contrôler la « dominance » d’une hybridation sur une autre.

Nous avons validé l'*hypothèse de proximité partielle* stipulant que seule une partie de l'historique de lecture d'un utilisateur est pertinente pour calculer le score d'un article candidat à une recommandation.

QR3 *Les utilisateurs ont-ils des préférences stylistiques dans leur lecture de l'actualité et donc est-il possible d'améliorer la recommandation d'articles d'actualité en exploitant la similarité stylométrique ?*

QR4 *Les caractéristiques de style sont-elles complémentaires avec les caractéristiques textuelles standards employées en recommandation d'articles d'actualité ?*

QR5 *Cette complémentarité permet-elle une amélioration qualitative des recommandations en termes de diversité, nouveauté et sérendipité ?*

Les résultats obtenus par la combinaison du modèle standard BM25 ainsi que de la représentation stylométrique d'items utilisée dans l'algorithme *SimRec* permis d'obtenir les meilleurs ordonnancements d'articles d'actualité pour les utilisateurs dans *Twineews*, un jeu de données introduit au chapitre 7. La représentation stylométrique utilisée seule ne permet pas un ordonnancement optimal des articles car ne joue pas un rôle majeur dans les préférences des utilisateurs. Cependant, les résultats obtenus montrent la complémentarité du style écrit dans cette tâche grâce à sa combinaison avec le modèle de pondération BM25. Cette combinaison est supérieure à toutes les autres combinaisons utilisées. Ainsi nous avons répondu aux questions de recherche *QR3* et *QR4*. De surcroît, l'intérêt de ces représentations a dépassé nos attentes : non seulement le style écrit joue un rôle dans les préférences des lecteurs, mais la prise en compte de cette dimension permet une amélioration de la pertinence et de la qualité des recommandations en termes de diversité, de nouveauté et de sérendipité. Ce constat répond à la dernière question de recherche *QR5*. La section 8.6 de ce manuscrit donne une analyse approfondie des réponses que nous avons apportées à ces différentes questions.

Dans cette thèse, nous avons proposé un point de vue particulier sur l'évaluation des systèmes de recommandation ainsi que sur le sujet des biais d'évaluation *offline* et *online*. En section 5.3.4, nous avons présenté une nouvelle taxonomie des méthodes et métriques d'évaluation pour les systèmes de recommandation. Celle-ci expose une vue d'ensemble de ces métriques et méthodes lorsque les classifications proposées dans certaines revues de la littérature (e.g. Hofmann (2015), Kouki et Said (2018), Silveira et al. (2019)), desquelles nous nous sommes inspirés, sont individuellement moins exhaustives. Nous avons ensuite proposé une discussion des biais liés aux contextes d'interaction en section 5.4.1. Nous avons notamment évoqué les différences entre domaines de recommandation, la présence ou non d'un système de recommandation lors de la collecte d'un jeu de données *offline*, la consommation redondante ou diverse, les interactions en « sessions » ou périodiques, l'influence du temps de visionnage des items ou encore les retours de pertinence disponibles, qu'ils soient explicites ou implicites. Ensuite, du fait de la nature des items et des préférences de consommation des lecteurs dans un système de recommandation d'articles d'actualité, nous avons proposé, en section 5.4.2, un nouveau point de vue sur la notion de « sérendipité » et de ses liens avec les notions de pertinence, nouveauté et surprise à travers une évolution du diagramme d'Euler proposé par Kotkov et al. (2016). En section 7.3.1, nous avons proposé de discuter de la signification de deux retours de pertinence pour la recommandation d'articles d'actualité : les « clics » et les « partages ». Nous avons

mentionné le biais de « satisfaction utilisateur » et introduit un nouveau que nous appelons le biais d'« ignorance des alternatives ». Tous ces biais nous ont amené à proposer la plateforme d'évaluation *Renewal* présentée au chapitre 9. Cette plateforme est dédiée à l'organisation de compétitions pour la tâche de recommandation d'articles d'actualité. Dans ce chapitre, nous avons détaillé les nouveautés de la plateforme au regard des plateformes existantes. Nous avons également proposé une simulation de compétition indiquant la faisabilité d'une compétition en fonction du nombre de lecteurs et de compétiteurs.

Pour terminer, les perspectives de cette thèse sont les suivantes :

1. Nous projetons de travailler sur une analyse approfondie de la métrique *SimRank*. Dans cette thèse, cette métrique a montré être pertinente dans l'évaluation *interne* des algorithmes de partitionnement lorsqu'il s'agit d'évaluer la qualité de représentations vectorielles sachant leur label vérité terrain. Plus généralement, nous voulons mettre en place une série d'expérimentations visant à comparer la métrique *SimRank* à différentes métriques disponibles dans la littérature. Ces expérimentations impliqueraient plusieurs jeux de données ainsi que différentes tâches externes et algorithmes de partitionnements comme le proposent par exemple [Lamirel et al. \(2016\)](#) et [Rendón et al. \(2011\)](#). L'objectif serait d'identifier si cette métrique permet une meilleure prédiction, comparée aux métriques existantes, de la qualité de représentation vectorielles lorsque celles-ci sont utilisées dans des tâches externes (e.g. classification, partitionnement « externe »).
2. Dans cette thèse, nous avons aussi travaillé sur la prédiction de personnalité à partir des messages écrits par des utilisateurs Twitter. Pour cela, nous avons collecté un jeu de données appelé *TwitterMBTI* contenant 80 000 utilisateurs Twitter ayant partagé leur personnalité « MBTI »⁹ sur leur profil. Nous avons implémenté un premier modèle capable de prédire la personnalité entre « *Extraversion* » et « *Introversion* ». Nous voulons améliorer ce modèle, et notamment sa précision, par l'utilisation des récentes avancées en traitement automatique du langage naturel, e.g. le modèle *BERT* ([Devlin et al., 2019](#)). L'objectif final de ce projet est d'entraîner un modèle capable de représenter la personnalité des utilisateurs sur Twitter et d'exploiter ce modèle dans *Twinews* pour la recommandation d'articles d'actualité, i.e. prendre en compte la personnalité de l'utilisateur lors de la recommandation d'articles d'actualité. Ainsi, en plus des articles d'actualité partagés par les utilisateurs, nous exploiterons leurs messages postés sur leur profil.
3. Nous prévoyons de publier le travail effectué dans les chapitres 6, 7 et 8 portant sur l'exploitation du style écrit dans la recommandation d'articles d'actualité. Par la suite, nous voulons intégrer, dans les évaluations sur le jeu de données *Twinews*, des modèles basés sur des réseaux de neurones profonds présentés dans la littérature du domaine tels que celui proposé par [Wang et al. \(2018\)](#).

9. Le *Myers Briggs Type Indicator* (MBTI) est un outil d'évaluation psychologique déterminant le type psychologique d'un sujet, suivant une méthode proposée en 1962 par Isabel Briggs Myers et Katherine Cook Briggs. Cet outil consiste en une suite de questions déterminant la personnalité MBTI des sujets. Les personnalités possibles sont au nombre de 16. Une personnalité se compose de 4 lettres : *E* ou *I* pour « *Extraversion* » ou « *Introversion* », *S* ou *N* pour « *Sensation* » ou « *Intuition* », *T* ou *F* pour « *Pensée* » ou « *Sentiment* » et *J* ou *P* pour « *Jugement* » ou « *Perception* ».

L'objectif serait d'unifier le travail effectué sur la représentation du style dans un réseau de neurones profond adapté à la génération de recommandations à partir d'un historique de lecture. Ce type de modèle pourra aussi nous permettre d'introduire des représentations complémentaires, apprises par des réseaux de neurones profonds, telles que la représentation de la personnalité des utilisateurs.

4. Nous prévoyons l'organisation de compétitions autour de la recommandation d'articles d'actualité grâce à la plateforme *Renewal*, d'abord avec un nombre limité de candidats, puis à grande échelle dans le cadre d'un *workshop*. À travers l'organisation de compétitions, nous voulons également évaluer les modèles proposés dans cette thèse consistant à recommander des articles sur la base de préférences stylistiques.

Bibliographie

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, et Xiaoqiang Zheng. TensorFlow : Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- Fabian Abel, Qi Gao, Geert-Jan Houben, et Ke Tao. Analyzing user modeling on twitter for personalized news recommendations. In Joseph A. Konstan, Ricardo Conejo, José L. Marzo, et Nuria Oliver, editors, *User Modeling, Adaption and Personalization*, pages 1–12, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-22362-4.
- Panagiotis Adamopoulos et Alexander Tuzhilin. On unexpectedness in recommender systems : Or how to better expect the unexpected. *ACM Trans. Intell. Syst. Technol.*, 5(4), December 2014. ISSN 2157-6904. doi : 10.1145/2559952. URL <https://doi.org/10.1145/2559952>.
- G. Adomavicius et Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5) :896–911, 2012.
- Haifa Alharthi et Diana Inkpen. Study of linguistic features incorporated in a literary book recommender system. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19*, page 1027–1034, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359337. doi : 10.1145/3297280.3297382. URL <https://doi.org/10.1145/3297280.3297382>.
- Haifa Alharthi, Diana Inkpen, et Stan Szpakowicz. A survey of book recommender systems. *Journal of Intelligent Information Systems*, 51(1) :139–160, August 2018a. ISSN 0925-9902. doi : 10.1007/s10844-017-0489-9. URL <https://doi.org/10.1007/s10844-017-0489-9>.
- Haifa Alharthi, Diana Inkpen, et Stan Szpakowicz. Authorship identification for literary book recommendations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 390–400, Santa Fe, New Mexico, USA, August

- 2018b. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/C18-1033>.
- Nawaf Ali. *Text stylometry for chat bot identification and intelligence estimation*. PhD thesis, The American University of the Middle East, 05 2014.
- Janek Amann. Authorship attribution in fan-fictional texts given variable length character and word n-grams, 2019.
- Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, et Xing Xie. Neural news recommendation with long- and short-term user representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 336–345, Florence, Italy, July 2019. Association for Computational Linguistics. doi : 10.18653/v1/P19-1033. URL <https://www.aclweb.org/anthology/P19-1033>.
- Cecilie Schou Andreassen. Online social network site addiction : A comprehensive review. *Current Addiction Reports*, 2(2) :175–184, 2015.
- S. Argamon. Interpreting burrows’s delta : Geometric and probabilistic foundations. *Lit. Linguistic Comput.*, 23 :131–147, 2008.
- Shlomo Argamon, Sushant Dhawle, Moshe Koppel, et James W. Pennebaker. Lexical predictors of personality type. In *Proceedings of the Joint Annual Meeting of the Interface and the Classification Society of North America*, 2005.
- Sanjeev Arora et Andrej Risteski. Provable benefits of representation learning. *CoRR*, abs/1706.04601, 2017. URL <http://arxiv.org/abs/1706.04601>.
- R. Arun, V. Suresh, et C. E. V. Madhavan. Stopword graphs and authorship attribution in text corpora. In *2009 IEEE International Conference on Semantic Computing*, pages 192–196, 2009. doi : 10.1109/ICSC.2009.101.
- E. Aslanian, M. Radmanesh, et M. Jalili. Hybrid recommender systems based on content feature relationship. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2016.
- Andrea Bacciu, Massimo La Morgia, Alessandro Mei, Eugenio Nerio Nemmi, Valerio Neri, et Julinda Stefa. Cross-domain authorship attribution combining instance based and profile-based features. In *CLEF (Working Notes)*, 2019.
- Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.
- Douglas Bagnall. Author identification using multi-headed recurrent neural networks. *CoRR*, abs/1506.04891, 2015. URL <http://arxiv.org/abs/1506.04891>.
- Dzmitry Bahdanau, Kyunghyun Cho, et Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.
- X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, et F. Xia. Scientific paper recommendation : A survey. *IEEE Access*, 7 :9324–9339, 2019.

- Xiao Bai, B. Barla Cambazoglu, Francesco Gullo, Amin Mantrach, et Fabrizio Silvestri. Exploiting search history of users for news personalization. *Information Sciences*, 385-386 :125 – 137, 2017. ISSN 0020-0255. doi : <https://doi.org/10.1016/j.ins.2016.12.038>. URL <http://www.sciencedirect.com/science/article/pii/S0020025516322617>.
- Eytan Bakshy, Solomon Messing, et Lada A Adamic. Exposure to ideologically diverse news and opinion on facebook. *Science*, 348(6239) :1130–1132, 2015.
- Matthias Baldauf, Schahram Dustdar, et Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) : 263–277, 2007.
- Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, et Preslav Nakov. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, Brussels, Belgium, 2018.
- Donald Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, 12(4) :387 – 415, 1975. ISSN 0022-2496. doi : [https://doi.org/10.1016/0022-2496\(75\)90001-2](https://doi.org/10.1016/0022-2496(75)90001-2). URL <http://www.sciencedirect.com/science/article/pii/0022249675900012>.
- Trapit Bansal, David Belanger, et Andrew McCallum. Ask the gru : Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 107–114, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi : 10.1145/2959100.2959180. URL <https://doi.org/10.1145/2959100.2959180>.
- Pablo Barberá. How social media reduces mass political polarization. evidence from germany, spain, and the u.s., 2014.
- Andrea Barraza-Urbina, Benjamin Heitmann, Conor Hayes, et Angela Carrillo Ramos. Xplodiv : An exploitation-exploration aware diversification approach for recommender systems. In *FLAIRS Conference*, 2015.
- Paul David Beck, Manuel Blaser, Adrian Michalke, et Andreas Lommatzsch. A system for online news recommendations in real-time with apache mahout. In *CLEF (Working Notes)*, 2017.
- Joeran Beel et Stefan Langer. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In Sarantos Kapidakis, Cezary Mazurek, et Marcin Werla, editors, *Research and Advanced Technology for Digital Libraries*, pages 153–168, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24592-8.
- Joeran Beel, Marcel Genzmehr, Stefan Langer, Andreas Nürnberger, et Bela Gipp. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys

- '13, pages 7–14, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2465-6. doi : 10.1145/2532508.2532511. URL <http://doi.acm.org/10.1145/2532508.2532511>.
- Alejandro Bellogín Kouki, Iván Cantador, et Pablo Castells. A comparative study of heterogeneous item recommendations in social systems. *Information Sciences*, 221 :142 – 169, 2013. ISSN 0020-0255. doi : <https://doi.org/10.1016/j.ins.2012.09.039>. URL <http://www.sciencedirect.com/science/article/pii/S0020025512006329>.
- Sonia Ben Ticha. *Recommandation personnalisée hybride*. Theses, Université de Lorraine, November 2015. URL <https://hal.univ-lorraine.fr/tel-01752090>.
- Y. Bengio, A. Courville, et P. Vincent. Representation learning : A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8) :1798–1828, Aug 2013. ISSN 0162-8828. doi : 10.1109/TPAMI.2013.50.
- J. Bennett et S. Lanning. The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, pages 3–6, New York, August 2007. ACM. URL <http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf>.
- Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, et Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- Sebastian Bischoff, Niklas Deckers, Marcel Schliebs, Ben Thies, Matthias Hagen, Efstathios Stamatatos, Benno Stein, et Martin Potthast. The importance of suppressing domain style in authorship analysis, 2020.
- David M. Blei, Andrew Y. Ng, et Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null) :993–1022, March 2003. ISSN 1532-4435.
- J. Bobadilla, F. Ortega, A. Hernando, et A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46 :109 – 132, 2013. ISSN 0950-7051. doi : <https://doi.org/10.1016/j.knosys.2013.03.012>. URL <http://www.sciencedirect.com/science/article/pii/S0950705113001044>.
- Hugo L. Borges et Ana C. Lorena. *A Survey on Recommender Systems for News Data*, pages 129–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-04584-4. doi : 10.1007/978-3-642-04584-4_6. URL https://doi.org/10.1007/978-3-642-04584-4_6.
- Dainis Boumber, Yifan Zhang, et Arjun Mukherjee. Experiments with convolutional neural networks for multi-label authorship attribution. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan, May 2018. European Languages Resources Association (ELRA). URL <https://www.aclweb.org/anthology/L18-1409>.
- Dainis Boumber, Yifan Zhang, Marjan Hosseinia, Arjun Mukherjee, et Ricardo Vilalta. Robust authorship verification with transfer learning. EasyChair Preprint no. 865, 2019. doi : 10.29007/9nf3.

- Marcelo Luiz Brocardo, Issa Traore, Isaac Woungang, et Mohammad S. Obaidat. Authorship verification using deep belief network systems. *International Journal of Communication Systems*, 30(12) :e3259, 2017. doi : 10.1002/dac.3259. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3259>. e3259 dac.3259.
- Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, et Gaël Varoquaux. API design for machine learning software : experiences from the scikit-learn project. In *ECML PKDD Workshop : Languages for Data Mining and Machine Learning*, pages 108–122, 2013.
- Judee K Burgoon, Michael Burgoon, et Miriam Wilkinson. Writing style as predictor of newspaper readership, satisfaction and image. *Journalism and Mass Communication Quarterly*, 58(2) :225–231, 1981. ISSN 1077-6990. doi : 10.1177/107769908105800207.
- Robin Burke. Hybrid recommender systems : Survey and experiments. *User modeling and user-adapted interaction*, 12(4) :331–370, 2002.
- Robin Burke. *Hybrid Web Recommender Systems*, pages 377–408. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9. doi : 10.1007/978-3-540-72079-9_12. URL https://doi.org/10.1007/978-3-540-72079-9_12.
- J. Burrows. 'delta' : a measure of stylistic difference and a guide to likely authorship. *Lit. Linguistic Comput.*, 17 :267–287, 2002.
- Kevin Burton, Akshay Java, Ian Soboroff, et al. The icwsm 2009 spinn3r dataset. In *Third Annual Conference on Weblogs and Social Media (ICWSM 2009)*, 2009.
- Kevin Burton, Niels Kasch, et Ian Soboroff. The icwsm 2011 spinn3r dataset. In *Proceedings of the Annual Conference on Weblogs and Social Media (ICWSM 2011)*, 2011.
- Xingjuan Cai, Zhaoming Hu, et Jinjun Chen. A many-objective optimization recommendation algorithm based on knowledge mining. *Information Sciences*, 537 :148 – 161, 2020. ISSN 0020-0255. doi : <https://doi.org/10.1016/j.ins.2020.05.067>. URL <http://www.sciencedirect.com/science/article/pii/S0020025520304709>.
- Sirian Caldarelli, Davide Feltoni Gurini, A. Micarelli, et G. Sansonetti. A signal-based approach to news recommendation. In *UMAP*, 2016.
- T. Caliński et J Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1) :1–27, 1974. doi : 10.1080/03610927408827101. URL <https://www.tandfonline.com/doi/abs/10.1080/03610927408827101>.
- Deborah Cameron. Style policy and style politics : a neglected aspect of the language of the news. *Media, Culture & Society*, 18(2) :315–333, 1996. doi : 10.1177/016344396018002008.

- Bagher Rahimpour Cami, Hamid Hassanpour, et Hoda Mashayekhi. User Trends Modeling for a Content-based Recommender System. *Expert Systems with Applications*, 87 :209–219, 2017. ISSN 09574174. doi : 10.1016/j.eswa.2017.06.020. URL <http://www.sciencedirect.com/science/article/pii/S0957417417304384>.
- Laurent Candillier, Max Chevalier, Damien Dudognon, et Josiane Mothe. Diversity in recommender systems : Bridging the gap between users and systems. In *4th International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies and Services (CENTRIC'2011)*, Barcelona, Spain, october 2011. URL <http://lcandillier.free.fr/publis/CENTRIC11.pdf>.
- Iván Cantador, Peter Brusilovsky, et Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.
- Michel Capelle, Flavius Frasinca, Marnix Moerland, et Frederik Hogenboom. Semantics-based news recommendation. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450309158. doi : 10.1145/2254129.2254163. URL <https://doi.org/10.1145/2254129.2254163>.
- Jaime Carbonell et Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130155. doi : 10.1145/290941.291025. URL <https://doi.org/10.1145/290941.291025>.
- Claudio Carpineto et Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1) :1 :1–1 :50, January 2012. ISSN 0360-0300. doi : 10.1145/2071389.2071390. URL <http://doi.acm.org/10.1145/2071389.2071390>.
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, et Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018. URL <http://arxiv.org/abs/1803.11175>.
- A. Chakraborty, B. Paranjape, S. Kakarla, et N. Ganguly. Stop clickbait : Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16, Aug 2016. doi : 10.1109/ASONAM.2016.7752207.
- Abhijnan Chakraborty, Saptarshi Ghosh, Niloy Ganguly, et Krishna P. Gummadi. Optimizing the recency-relevancy trade-off in online news recommendations. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 837–846, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi : 10.1145/3038912.3052656. URL <https://doi.org/10.1145/3038912.3052656>.

- C. Chen, X. Meng, Z. Xu, et T. Lukasiewicz. Location-aware personalized news recommendation with deep semantic analysis. *IEEE Access*, 5 :1624–1638, 2017.
- Hung-Hsuan Chen, Chu-An Chung, Hsin-Chien Huang, et Wen Tsui. Common pitfalls in training and evaluating recommender systems. *SIGKDD Explor. Newsl.*, 19(1) :37–45, September 2017a. ISSN 1931-0145. doi : 10.1145/3137597.3137601. URL <https://doi.org/10.1145/3137597.3137601>.
- Ting Chen, Liangjie Hong, Yue Shi, et Yizhou Sun. Joint text embedding for personalized content-based recommendation, 2017b.
- Paula Chesley, B. Vincent, Li Xu, et R. Srihari. Using verbs and adjectives to automatically classify blog sentiment. In *AAAI Spring Symposium : Computational Approaches to Analyzing Weblogs*, 2006.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, et Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi : 10.3115/v1/D14-1179. URL <https://www.aclweb.org/anthology/D14-1179>.
- François Chollet et al. Keras. <https://keras.io>, 2015.
- Andrzej Cichocki et Anh Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions*, 92-A(3) :708–721, 2009. doi : 10.1587/transfun.E92.A.708. URL <https://doi.org/10.1587/transfun.E92.A.708>.
- Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, et Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, page 659–666, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581644. doi : 10.1145/1390334.1390446. URL <https://doi.org/10.1145/1390334.1390446>.
- Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, et M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *SIGIR 1999*, 1999.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, et Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/D17-1070>.
- Corinna Cortes et Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

- Paolo Cremonesi, Yehuda Koren, et Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10*, page 39–46, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589060. doi : 10.1145/1864708.1864721. URL <https://doi.org/10.1145/1864708.1864721>.
- Paolo Cremonesi, Antonio Tripodi, et Roberto Turrin. Cross-domain recommender systems. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 496–503. Ieee, 2011.
- Alessandro Cucchiarelli, Christian Morbidoni, Giovanni Stilo, et Paola Velardi. What to write and why : A recommender for news media. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18*, page 1321–1330, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351911. doi : 10.1145/3167132.3167274. URL <https://doi.org/10.1145/3167132.3167274>.
- Viktoriia Danilova et Andrew Ponomarev. Hybrid recommender systems : The review of state-of-the-art research and applications, 2017.
- D. L. Davies et D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2) :224–227, April 1979. ISSN 1939-3539. doi : 10.1109/TPAMI.1979.4766909.
- Gabriel de Souza Pereira Moreira, Felipe Ferreira, et Adilson Marques da Cunha. News session-based recommendations using deep neural networks. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, DLRS 2018*, page 15–23, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450366175. doi : 10.1145/3270323.3270328. URL <https://doi.org/10.1145/3270323.3270328>.
- J. Devlin, Ming-Wei Chang, Kenton Lee, et Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- Paul Dickson et Robert Skole. *Journalese : A Dictionary for Deciphering the News*. Marion Street Press, 2012.
- S. H. H. Ding, B. C. M. Fung, F. Iqbal, et W. K. Cheung. Learning stylometric representations for authorship analysis. *IEEE Transactions on Cybernetics*, 49(1) : 107–121, Jan 2019. doi : 10.1109/TCYB.2017.2766189.
- Anthony Downs. *An Economic Theory of Democracy*. Harper, 1957.
- Michael D. Ekstrand, F. Maxwell Harper, Martijn C. Willemsen, et Joseph A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14*, page 161–168, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326681. doi : 10.1145/2645710.2645737. URL <https://doi.org/10.1145/2645710.2645737>.
- Hugo Jair Escalante, Tamar Solorio, et Manuel Montes-y Gómez. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual*

Meeting of the Association for Computational Linguistics : Human Language Technologies, pages 288–298, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/P11-1030>.

Seth Flaxman, Sharad Goel, et Justin M. Rao. Filter Bubbles, Echo Chambers, and Online News Consumption. *Public Opinion Quarterly*, 80(S1) :298–320, 03 2016. ISSN 0033-362X. doi : 10.1093/poq/nfw006. URL <https://doi.org/10.1093/poq/nfw006>.

P. W. Foltz. Using latent semantic indexing for information filtering. In *Proceedings of the ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems*, COCS '90, page 40–47, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0897913582. doi : 10.1145/91474.91486. URL <https://doi.org/10.1145/91474.91486>.

Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, et Arnaud Legout. Social Clicks : What and Who Gets Read on Twitter ? In *ACM SIGMETRICS / IFIP Performance 2016*, Antibes Juan-les-Pins, France, June 2016. URL <https://hal.inria.fr/hal-01281190>.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2) :23–38, 1994.

Florent Garcin, Christos Dimitrakakis, et Boi Faltings. Personalized news recommendation with context trees. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 105–112, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324090. doi : 10.1145/2507157.2507166. URL <https://doi.org/10.1145/2507157.2507166>.

Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, et Amr Huber. Offline and online evaluation of news recommender systems at swissinfo.ch. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 169–176, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326681. doi : 10.1145/2645710.2645745. URL <https://doi.org/10.1145/2645710.2645745>.

Marco De Gemmis, Leo Iaquinta, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, et Giovanni Semeraro. Preference learning in recommender systems. In *In Preference Learning (PL-09) ECML/PKDD-09 Workshop*, 2009.

Jade Goldstein-Stewart, Ransom Winder, et Roberta Sabin. Person identification from text and speech genre samples. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 336–344, Athens, Greece, March 2009. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/E09-1039>.

A. Granados, M. Cebrian, D. Camacho, et F. d. B. Rodriguez. Reducing the loss of information through annealing text distortion. *IEEE Transactions on Knowledge and Data Engineering*, 23(7) :1090–1102, 2011.

- Jon Gulla, Arne Dag Fidjestøl, X. Su, et Humberto Castejón. Implicit user profiling in news recommender systems. *WEBIST 2014 - Proceedings of the 10th International Conference on Web Information Systems and Technologies*, 1 :185–192, 01 2014.
- Jon Atle Gulla, Jon Espen Ingvaldsen, Cristina Marco, et Xiaomeng Su. The Intricacies of Time in News Recommendation. *INRA 2016*, 2016. URL http://ceur-ws.org/Vol-1618/INRA_paper4.pdf.
- Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, et Xiaomeng Su. The adressa dataset for news recommendation. In *Proceedings of the International Conference on Web Intelligence, WI '17*, page 1042–1048, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349512. doi : 10.1145/3106426.3109436. URL <https://doi.org/10.1145/3106426.3109436>.
- Shriya TP Gupta, Jajati Keshari Sahoo, et Rajendra Kumar Roul. Authorship identification using recurrent neural networks. In *Proceedings of the 2019 3rd International Conference on Information System and Data Mining, ICISDM 2019*, pages 133–137, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6635-9. doi : 10.1145/3325917.3325935. URL <http://doi.acm.org/10.1145/3325917.3325935>.
- Nathan Halko, Per-Gunnar Martinsson, et Joel A. Tropp. Finding structure with randomness : Probabilistic algorithms for constructing approximate matrix decompositions, 2009.
- Oren Halvani, Lukas Graner, Roey Regev, et Philipp Marquardt. An improved topic masking technique for authorship analysis, 2020.
- F. Maxwell Harper et Joseph A. Konstan. The movielens datasets : History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi : 10.1145/2827872. URL <https://doi.org/10.1145/2827872>.
- Zellig S. Harris. Distributional structure. *<i>WORD</i>*, 10(2-3) :146–162, 1954. doi : 10.1080/00437956.1954.11659520. URL <https://doi.org/10.1080/00437956.1954.11659520>.
- Julien Hay, Tim Van de Cruys, Philippe Muller, Bich-Liên Doan, Fabrice Popineau, et Lyes Benamsili. Complémentarités de représentations lexicales pour la similarité sémantique. In *Proceedings of 18e Conférence sur l'Extraction et la Gestion des Connaissances (EGC 2018)*, January 2018.
- Julien Hay, Tim Van de Cruys, Philippe Muller, Bich-Liên Doan, Fabrice Popineau, et Ouassim Ait-Elhara. *Automatically Selecting Complementary Vector Representations for Semantic Textual Similarity*, pages 45–60. Springer International Publishing, Cham, 2019. ISBN 978-3-030-18129-1. doi : 10.1007/978-3-030-18129-1_3. URL https://doi.org/10.1007/978-3-030-18129-1_3.
- Julien Hay, Bich-Liên Doan, Fabrice Popineau, et Ouassim Ait Elhara. Filtering a reference corpus to generalize stylometric representations. In *Proceedings of the 12th Conference on Knowledge Discovery and Information Retrieval (KDIR 2020)*, November 2020a.

- Julien Hay, Bich-Liên Doan, Fabrice Popineau, et Ouassim Ait Elhara. Renewal : an online competition platform for news recommender systems. In *Proceedings of Challenges in Machine Learning at NeurIPS 2020*, December 2020b.
- Julien Hay, Bich-Liên Doan, Fabrice Popineau, et Ouassim Ait Elhara. Representation learning of writing style. In *Proceedings of the 6th Workshop on Noisy User-generated Text (W-NUT 2020)*, November 2020c.
- Natali Helberger. On the democratic role of news recommenders. *Digital Journalism*, 7(8) :993–1012, 2019. doi : 10.1080/21670811.2019.1623700. URL <https://doi.org/10.1080/21670811.2019.1623700>.
- Ralf Herbrich, Tom Minka, et Thore Graepel. Trueskill™ : A bayesian skill rating system. In B. Schölkopf, J. C. Platt, et T. Hoffmann, editors, *Advances in Neural Information Processing Systems 19*, pages 569–576. MIT Press, 2007. URL <http://papers.nips.cc/paper/3079-trueskilltm-a-bayesian-skill-rating-system.pdf>.
- Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, et John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1) :5–53, January 2004. ISSN 1046-8188. doi : 10.1145/963770.963772. URL <https://doi.org/10.1145/963770.963772>.
- Wynford Hicks. *Subediting for Journalists (Media Skills)*. Routledge, aug 2002. ISBN 0415240859.
- Sepp Hochreiter et Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8) :1735–1780, November 1997. ISSN 0899-7667. doi : 10.1162/neco.1997.9.8.1735. URL <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- Elad Hoffer et Nir Ailon. Deep metric learning using triplet network. In Aasa Feragen, Marcello Pelillo, et Marco Loog, editors, *Similarity-Based Pattern Recognition*, pages 84–92, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24261-3.
- Katja Hofmann. *Online Experimentation for Information Retrieval*, pages 21–41. Springer International Publishing, Cham, 2015. ISBN 978-3-319-25485-2. doi : 10.1007/978-3-319-25485-2_2. URL https://doi.org/10.1007/978-3-319-25485-2_2.
- Katja Hofmann, Shimon Whiteson, et Maarten de Rijke. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, page 249–258, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307178. doi : 10.1145/2063576.2063618. URL <https://doi.org/10.1145/2063576.2063618>.
- David I. Holmes. The Evolution of Stylometry in Humanities Scholarship. *Literary and Linguistic Computing*, 13(3) :111–117, 09 1998. ISSN 0268-1145. doi : 10.1093/lc/13.3.111. URL <https://doi.org/10.1093/lc/13.3.111>.

- Frank Hopfgartner, Torben Brodt, Jonas Seiler, Benjamin Kille, Andreas Lommatzsch, Martha Larson, Roberto Turrin, et András Serény. Benchmarking news recommendations : The clef newsreel use case. *SIGIR Forum*, 49(2) : 129–136, January 2016. ISSN 0163-5840. doi : 10.1145/2888422.2888443. URL <https://doi.org/10.1145/2888422.2888443>.
- John Houvardas et Efstathios Stamatatos. N-gram feature selection for authorship identification. In Jérôme Euzenat et John Domingue, editors, *Artificial Intelligence : Methodology, Systems, and Applications*, pages 77–86, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-40931-1.
- J. Hu, F. Jin, G. Zhang, J. Wang, et Y. Yang. A user profile modeling method based on word2vec. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 410–414, 2017.
- Neil Hurley et Mi Zhang. Novelty and diversity in top-n recommendation – analysis and evaluation. *ACM Trans. Internet Technol.*, 10(4), March 2011. ISSN 1533-5399. doi : 10.1145/1944339.1944341. URL <https://doi.org/10.1145/1944339.1944341>.
- Ilija Ilievski et S. Roy. Personalized news recommendation based on implicit feedback. In *NRS '13*, 2013.
- Giacomo Inches et Fabio Crestani. Overview of the international sexual predator identification competition at pan-2012. In *CLEF (Online working notes/labs/workshop)*, volume 30, 2012.
- Paul Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37 :241–72, 01 1901. doi : 10.5169/seals-266440.
- Fotis Jannidis, Steffen Pielström, Christof Schöch, et Thorsten Vitt. Improving burrows’ delta – an empirical evaluation of text distance measures. In *Book of Abstracts of the Digital Humanities Conference 2015*. ADHO, UWS, 2015. URL http://dh2015.org/abstracts/xml/JANNIDIS_Fotis_Improving_Burrows__Delta___An_empi/JANNIDIS_Fotis_Improving_Burrows__Delta___An_empirical_.html.
- Kalervo Järvelin et Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4) :422–446, October 2002. ISSN 1046-8188. doi : 10.1145/582415.582418. URL <http://doi.acm.org/10.1145/582415.582418>.
- Johannes Jasper, Philipp Berger, Patrick Hennig, et Christoph Meinel. Authorship verification on short text samples using stylometric embeddings. In Wil M. P. van der Aalst, Vladimir Batagelj, Goran Glavaš, Dmitry I. Ignatov, Michael Khachay, Sergei O. Kuznetsov, Olessia Koltsova, Irina A. Lomazova, Natalia Loukachevitch, Amedeo Napoli, Alexander Panchenko, Panos M. Pardalos, Marcello Pelillo, et Andrey V. Savchenko, editors, *Analysis of Images, Social Networks and Texts*, pages 64–75, Cham, 2018. Springer International Publishing. ISBN 978-3-030-11027-7.

- Amin Javari et Mahdi Jalili. A probabilistic model to resolve diversity—accuracy challenge of recommendation systems. *Knowl. Inf. Syst.*, 44(3) :609–627, September 2015. ISSN 0219-1377. doi : 10.1007/s10115-014-0779-2. URL <https://doi.org/10.1007/s10115-014-0779-2>.
- Amin Javari, Malihe Izadi, et Mahdi Jalili. *Recommender Systems for Social Networks Analysis and Mining : Precision Versus Diversity*, pages 423–438. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-47824-0. doi : 10.1007/978-3-662-47824-0_16. URL https://doi.org/10.1007/978-3-662-47824-0_16.
- Rong Jin, Joyce Y. Chai, et Luo Si. An automatic weighting scheme for collaborative filtering. In *SIGIR*, pages 337–344, 2004. URL <https://doi.org/10.1145/1008992.1009051>.
- Michael Jugovac, Dietmar Jannach, et Mozghan Karimi. Streamingrec : A framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18*, page 269–273, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi : 10.1145/3240323.3240384. URL <https://doi.org/10.1145/3240323.3240384>.
- Marius Kaminskas et Derek Bridge. Measuring surprise in recommender systems. In *Proceedings of the workshop on recommender systems evaluation : dimensions and design (Workshop programme of the 8th ACM conference on recommender systems)*. Citeseer, 2014.
- Marius Kaminskas et Derek Bridge. Diversity, serendipity, novelty, and coverage : A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, 7(1), December 2016. ISSN 2160-6455. doi : 10.1145/2926720. URL <https://doi.org/10.1145/2926720>.
- Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A. Konstan, et Paul Schrater. "i like to explore sometimes" : Adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15*, page 19–26, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336925. doi : 10.1145/2792838.2800172. URL <https://doi.org/10.1145/2792838.2800172>.
- Mozghan Karimi, Dietmar Jannach, et Michael Jugovac. News recommender systems – survey and roads ahead. *Information Processing and Management*, 54(6) :1203 – 1227, 2018. ISSN 0306-4573. doi : <https://doi.org/10.1016/j.ipm.2018.04.008>. URL <http://www.sciencedirect.com/science/article/pii/S030645731730153X>.
- Jussi Karlgren. The wheres and whyfores for studying text genre computationally. In *Workshop on Style and Meaning in Language, Art, Music and Design. National Conference on Artificial Intelligence*, 2004.
- John Paul Kelly et Derek Bridge. Enhancing the diversity of conversational collaborative recommendations : A comparison. *Artif. Intell. Rev.*, 25(1–2) : 79–95, April 2006. ISSN 0269-2821. doi : 10.1007/s10462-007-9023-8. URL <https://doi.org/10.1007/s10462-007-9023-8>.

- M. G. Kendall. A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30 (1-2) :81–93, 06 1938. ISSN 0006-3444. doi : 10.1093/biomet/30.1-2.81. URL <https://doi.org/10.1093/biomet/30.1-2.81>.
- M. G. Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3) : 239–251, 1945. ISSN 00063444. URL <http://www.jstor.org/stable/2332303>.
- Mike Kestemont, Efsthathios Stamatatos, Enrique Manjavacas, Walter Daelemans, Martin Potthast, et Benno Stein. Overview of the cross-domain authorship attribution task at {PAN} 2019. In *Working Notes of CLEF 2019-Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019*, pages 1–15, 2019.
- E. Kharitonov, C. MacDonald, P. Serdyukov, et I. Ounis. Generalized team draft interleaving. In *CIKM '15*, 2015.
- Benjamin Kille, Frank Hopfgartner, Torben Brodt, et Tobias Heintz. The plista dataset. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge*, NRS '13, page 16–23, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450323024. doi : 10.1145/2516641.2516643. URL <https://doi.org/10.1145/2516641.2516643>.
- Benjamin Kille, Andreas Lommatzsch, Roberto Turrin, András Serény, Martha Larson, Torben Brodt, Jonas Seiler, et Frank Hopfgartner. Stream-based recommendations : Online and offline evaluation as a service. In Josanne Mothe, Jacques Savoy, Jaap Kamps, Karen Pinel-Sauvagnat, Gareth Jones, Eric San Juan, Linda Capellato, et Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 497–517, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24027-5.
- Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, et Sanja Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3294–3302, Cambridge, MA, USA, 2015. MIT Press.
- Evan Kirshenbaum, George Forman, et Michael Dugan. A live comparison of methods for personalized article recommendation at forbes.com. In Peter A. Flach, Tijl De Bie, et Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 51–66, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33486-3.
- Bart P. Knijnenburg, M. C. Willemsen, Zeno Gantner, Hakan Soncu, et C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22 :441–504, 2012.
- Yehuda Koren, Robert Bell, et Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8) :30–37, August 2009. ISSN 0018-9162. doi : 10.1109/MC.2009.263. URL <http://dx.doi.org/10.1109/MC.2009.263>.
- Denis Kotkov., Jari Veijalainen., et Shuaiqiang Wang. Challenges of serendipity in recommender systems. In *Proceedings of the 12th International Conference on Web Information Systems and Technologies - Volume 2 : WEBIST,*, pages

251–256. INSTICC, SciTePress, 2016. ISBN 978-989-758-186-1. doi : 10.5220/0005879802510256.

Denis Kotkov, Shuaiqiang Wang, et Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111 :180 – 192, 2016. ISSN 0950-7051. doi : <https://doi.org/10.1016/j.knosys.2016.08.014>. URL <http://www.sciencedirect.com/science/article/pii/S0950705116302763>.

Alejandro Bellogin Kouki. *Recommender system performance evaluation and prediction an information retrieval perspective*. PhD thesis, Université autonome de Madrid, 2014.

Alejandro Bellogin Kouki et Alan Said. Recommender systems evaluation. In Reda Alhajj et Jon G. Rokne, editors, *Encyclopedia of Social Network Analysis and Mining, 2nd Edition*. Springer, 2018. doi : 10.1007/978-1-4939-7131-2_110162. URL https://doi.org/10.1007/978-1-4939-7131-2_110162.

Vaibhav Kumar, Dhruv Khattar, Shashank Gupta, Manish Gupta, et Vasudeva Varma. Deep neural architecture for news recommendation. In *CLEF*, 2017.

Matevž Kunaver et Tomaž Požrl. Diversity in recommender systems – a survey. *Knowledge-Based Systems*, 123 :154 – 162, 2017. ISSN 0950-7051. doi : <https://doi.org/10.1016/j.knosys.2017.02.009>. URL <http://www.sciencedirect.com/science/article/pii/S0950705117300680>.

Jean-Charles Lamirel, Nicolas Dugué, et Pascal Cuxac. New efficient clustering quality indexes. In *International Joint Conference on Neural Networks (IJCNN 2016)*, Vancouver, Canada, July 2016. URL <https://hal.archives-ouvertes.fr/hal-01350509>.

Quoc Le et Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML'14*, page II–1188–II–1196. JMLR.org, 2014.

Danielle Hyunsook Lee et Peter Brusilovsky. Improving personalized recommendations using community membership information. *Information Processing & Management*, 53(5) :1201 – 1214, 2017. ISSN 0306-4573. doi : <https://doi.org/10.1016/j.ipm.2017.05.005>. URL <http://www.sciencedirect.com/science/article/pii/S0306457316305477>.

W. Lee, K. Oh, C. Lim, et H. Choi. User profile extraction from twitter for personalized news recommendation. In *16th International Conference on Advanced Communication Technology*, pages 779–783, 2014.

David D. Lewis, Yiming Yang, Tony G. Rose, et Fan Li. Rcv1 : A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5 :361–397, 2004.

Lihong Li, Wei Chu, John Langford, et Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11*, page 297–306, New York, NY, USA, 2011. Association for Computing

- Machinery. ISBN 9781450304931. doi : 10.1145/1935826.1935878. URL <https://doi.org/10.1145/1935826.1935878>.
- M. Li et L. Wang. A survey on personalized news recommendation technology. *IEEE Access*, 7 :145861–145879, 2019.
- Jianxun Lian, Fuzheng Zhang, Xing Xie, et Guangzhong Sun. Towards better representation learning for personalized news recommendation : a multi-channel deep fusion approach. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3805–3811. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi : 10.24963/ijcai.2018/529. URL <https://doi.org/10.24963/ijcai.2018/529>.
- Yu Liang, Babak Loni, et Martha A Larson. Clef newsreel 2017 : Contextual bandit news recommendation. In *CLEF (Working Notes)*, 2017.
- Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, et Tao Li. Personalized news recommendation via implicit social experts. *Inf. Sci.*, 254 :1–18, January 2014a. ISSN 0020-0255. doi : 10.1016/j.ins.2013.08.034. URL <https://doi.org/10.1016/j.ins.2013.08.034>.
- Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, et Tao Li. Personalized news recommendation via implicit social experts. *Information Sciences*, 254 :1 – 18, 2014b. ISSN 0020-0255. doi : <https://doi.org/10.1016/j.ins.2013.08.034>. URL <http://www.sciencedirect.com/science/article/pii/S002002551300594X>.
- Jiahui Liu, Peter Dolan, et Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI '10*, page 31–40, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605585154. doi : 10.1145/1719970.1719976. URL <https://doi.org/10.1145/1719970.1719976>.
- Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.
- Zhiyuan Liu, Yankai Lin, et Maosong Sun. *Representation Learning and NLP*, pages 1–11. Springer Singapore, Singapore, 2020. ISBN 978-981-15-5573-2. doi : 10.1007/978-981-15-5573-2_1. URL https://doi.org/10.1007/978-981-15-5573-2_1.
- Andreas Lommatzsch, Benjamin Kille, et Sahin Albayrak. Incorporating context and trends in news recommender systems. In *Proceedings of the International Conference on Web Intelligence, WI '17*, page 1062–1068, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349512. doi : 10.1145/3106426.3109433. URL <https://doi.org/10.1145/3106426.3109433>.
- Pasquale Lops, Marco De Gemmis, et Giovanni Semeraro. Content-based recommender systems : State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.
- Ravi Lourdasamy et Stanislaus Abraham. A survey on text pre-processing techniques and tools. *International Journal of Computer Sciences and Engineering*, 6(3) : 148–157, 2018.

- Hongyu Lu, M. Zhang, W. Ma, Ce Wang, F. Xia, Yiqun Liu, Leyu Lin, et S. Ma. Effects of user negative experience in mobile news streaming. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.
- Hao Ma, Xueqing Liu, et Zhihong Shen. User fatigue in online news recommendation. In *Proceedings of the 25th International Conference on World Wide Web, WWW '16*, page 1363–1372, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi : 10.1145/2872427.2874813. URL <https://doi.org/10.1145/2872427.2874813>.
- Michelle Madejski, Maritza Lupe Johnson, et Steven Michael Bellovin. The failure of online social network privacy settings, 2011.
- François Mairesse, Marilyn A. Walker, Matthias R. Mehl, et Roger K. Moore. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research*, 30 :457–500, 2007. ISSN 10769757. doi : 10.1613/jair.2349.
- Andrii Maksai, Florent Garcin, et Boi Faltings. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15*, page 179–186, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336925. doi : 10.1145/2792838.2800184. URL <https://doi.org/10.1145/2792838.2800184>.
- Christopher D Manning, Hinrich Schütze, et Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge university press, 2008.
- Harry Markowitz. Portfolio selection*. *The Journal of Finance*, 7(1) :77–91, 1952. doi : 10.1111/j.1540-6261.1952.tb01525.x. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x>.
- Rangsipan Marukatat, Robroo Somkiadcharoen, Ratthanan Nalintasnai, et Tappasarn Aramboonpong. Authorship attribution analysis of thai online messages. *2014 International Conference on Information Science and Applications (ICISA)*, pages 1–4, 2014.
- Sean M. McNee, John Riedl, et Joseph A. Konstan. Being accurate is not enough : How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems, CHI EA '06*, page 1097–1101, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595932984. doi : 10.1145/1125451.1125659. URL <https://doi.org/10.1145/1125451.1125659>.
- Youssef Meguebli, Mouna Kacimi, Bich-Liên Doan, et Fabrice Popineau. Towards better news article recommendation. *World Wide Web*, 20(6) :1293–1312, February 2017. doi : 10.1007/s11280-017-0436-2. URL <https://hal.archives-ouvertes.fr/hal-01502672>.
- Marcelo Mendoza et Nicolás Torres. Evaluating content novelty in recommender systems. *Journal of Intelligent Information Systems*, 03 2019. doi : 10.1007/s10844-019-00548-x.

- Rohith Menon et Yejin Choi. Domain independent authorship attribution without domain adaptation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 309–315, Hissar, Bulgaria, September 2011. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/R11-1043>.
- Tomas Mikolov, Kai Chen, Greg Corrado, et Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.
- Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhresch, et Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
- A. M. Mohsen, N. M. El-Makky, et N. Ghanem. Author identification using deep learning. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 898–903, Dec 2016. doi : 10.1109/ICMLA.2016.0161.
- G. Morales, A. Gionis, et C. Lucchese. From chatter to headlines : harnessing the real-time web for personalized news recommendation. In *WSDM '12*, 2012.
- Masahiro Morita et Yoichi Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 272–281, Berlin, Heidelberg, 1994. Springer-Verlag. ISBN 038719889X.
- Tomoko Murakami, Koichiro Mori, et Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In Ken Satoh, Akihiro Inokuchi, Katashi Nagao, et Takahiro Kawamura, editors, *New Frontiers in Artificial Intelligence*, pages 40–46, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78197-4.
- Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, et Fabrice Rossi. Study of a bias in the offline evaluation of a recommendation algorithm. *CoRR*, abs/1511.01280, 2015. URL <http://arxiv.org/abs/1511.01280>.
- Judith Möller, Damian Trilling, Natali Helberger, et Bram van Es. Do not blame it on the algorithm : an empirical assessment of multiple recommender systems and their impact on content diversity. *Information, Communication & Society*, 21(7) : 959–977, 2018. doi : 10.1080/1369118X.2018.1444076. URL <https://doi.org/10.1080/1369118X.2018.1444076>.
- Makoto Nakatsuji, Yasuhiro Fujiwara, Akimichi Tanaka, Toshio Uchiyama, Ko Fujimura, et Toru Ishida. Classical music for rock fans? novel recommendations for expanding user interests. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, page 949–958, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300995. doi : 10.1145/1871437.1871558. URL <https://doi.org/10.1145/1871437.1871558>.

- R. Nath Nandi, M. M. Arefin Zaman, T. Al Muntasir, S. Hosain Sumit, T. Sourov, et M. Jamil-Ur Rahman. Bangla news recommendation using doc2vec. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5, 2018.
- Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, et Damon Woodard. Surveying Stylometry Techniques and Applications. *ACM Comput. Surv. Article*, 50(86), 2017a. ISSN 03600300. doi : 10.1145/3132039.
- Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, et Damon Woodard. Surveying stylometry techniques and applications. *ACM Comput. Surv.*, 50(6) :86 :1–86 :36, November 2017b. ISSN 0360-0300. doi : 10.1145/3132039. URL <http://doi.acm.org/10.1145/3132039>.
- A. Nishitarumizu, T. Itokawa, T. Kitasuka, et M. Aritsugi. Improving a news recommendation system in adapting to interests of a user with storage of a constant size. In *2010 12th International Asia-Pacific Web Conference*, pages 109–115, 2010.
- J. Oh, S. Park, H. Yu, M. Song, et S. Park. Novel recommendation based on personal popularity tendency. In *2011 IEEE 11th International Conference on Data Mining*, pages 507–516, 2011.
- M. Ohta, T. Hachiki, et A. Takasu. Related paper recommendation to support online-browsing of research papers. In *Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011)*, pages 130–136, 2011.
- Özlem Özgöbek, J. Gulla, et R. C. Erdur. A survey on challenges and methods in news recommendation. In *WEBIST*, 2014.
- Matteo Pagliardini, Prakhar Gupta, et Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi : 10.18653/v1/N18-1049. URL <https://www.aclweb.org/anthology/N18-1049>.
- Eli Pariser. *The filter bubble : What the Internet is hiding from you*. Penguin UK, 2011.
- Keunchan Park, Jisoo Lee, et Jaeho Choi. Deep neural networks for news recommendations. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*, page 2255–2258, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349185. doi : 10.1145/3132847.3133154. URL <https://doi.org/10.1145/3132847.3133154>.
- Yoon-Joo Park et Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys '08*, page 11–18, New York, NY, USA, 2008. Association for

- Computing Machinery. ISBN 9781605580937. doi : 10.1145/1454008.1454012. URL <https://doi.org/10.1145/1454008.1454012>.
- Jagadeesh Patchala et Raj Bhatnagar. Authorship attribution by consensus among multiple features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2766–2777, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.
- James W Pennebaker, Martha E Francis, et Roger J Booth. Linguistic inquiry and word count : Liwc 2001. *Mahway : Lawrence Erlbaum Associates*, 71, 2001.
- Jeffrey Pennington, Richard Socher, et Christopher D. Manning. Glove : Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL <http://www.aclweb.org/anthology/D14-1162>.
- Maria Soledad Pera et Yiu-Kai Ng. Analyzing book-related features to recommend books for emergent readers. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, HT '15, page 221–230, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450333955. doi : 10.1145/2700171.2791037. URL <https://doi.org/10.1145/2700171.2791037>.
- Owen Phelan, Kevin McCarthy, et Barry Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, page 385–388, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584355. doi : 10.1145/1639714.1639794. URL <https://doi.org/10.1145/1639714.1639794>.
- Danae Pla Karidi, Yannis Stavarakas, et Yannis Vassiliou. Tweet and followee personalized recommendations based on knowledge graphs. *Journal of Ambient Intelligence and Humanized Computing*, 0(0) :0, 2017. ISSN 1868-5137. doi : 10.1007/s12652-017-0491-7. URL <https://link.springer.com/article/10.1007/s12652-017-0491-7>.
- R. K. Pon, A. F. Cardenas, D. J. Buttler, et T. J. Critchlow. iscore : Measuring the interestingness of articles in a limited user environment. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 354–361, 2007.
- R.K. Pon, A.F. Cardenas, D.J. Buttler, et T.J. Critchlow. Measuring the interestingness of articles in a limited user environment. *Information Processing and Management*, 47(1) :97 – 116, 2011. ISSN 0306-4573. doi : <https://doi.org/10.1016/j.ipm.2010.03.001>. URL <http://www.sciencedirect.com/science/article/pii/S030645731000021X>.
- Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, et Benno Stein. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 231–240, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi : 10.18653/v1/P18-1022. URL <https://www.aclweb.org/anthology/P18-1022>.

- Pearl Pu, Li Chen, et Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11*, page 157–164, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306836. doi : 10.1145/2043932.2043962. URL <https://doi.org/10.1145/2043932.2043962>.
- Chen Qian, Ting He, et Rao Zhang. Deep learning based authorship identification, 2017.
- Tie-Yun Qian, Bing Liu, Qing Li, et Jianfeng Si. Review authorship attribution in a similarity space. *Journal of Computer Science and Technology*, 30(1) :200–213, Jan 2015. ISSN 1860-4749. doi : 10.1007/s11390-015-1513-6. URL <https://doi.org/10.1007/s11390-015-1513-6>.
- Zhu Qiannan, Xiaofei Zhou, Zeliang Song, Jianlong Tan, et Li Guo. Dan : Deep attention neural network for news recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33 :5973–5980, 07 2019. doi : 10.1609/aaai.v33i01.33015973.
- Jing Qin et Peng Lu. Application of news features in news recommendation methods : A survey. In Pinle Qin, Hongzhi Wang, Guanglu Sun, et Zeguang Lu, editors, *Data Science*, pages 113–125, Singapore, 2020. Springer Singapore. ISBN 978-981-15-7984-4.
- Filip Radlinski et Nick Craswell. Optimized interleaving for online retrieval evaluation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, WSDM '13*, page 245–254, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318693. doi : 10.1145/2433396.2433429. URL <https://doi.org/10.1145/2433396.2433429>.
- Filip Radlinski, Madhu Kurup, et T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, 2008.
- Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, et Yejin Choi. Truth of varying shades : Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi : 10.18653/v1/D17-1317. URL <https://www.aclweb.org/anthology/D17-1317>.
- Shaina Raza et Chen Ding. A survey on news recommender system – dealing with timeliness, dynamic user interest and content quality, and effects of recommendation on news readers, 2020.
- Radim Řehůřek et Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. <http://is.muni.cz/publication/884893/en>.
- E. Rendón, Itzel Abundez, C. Gutierrez, S. Zagal, A. Arizmendi, E. M. Quiroz, et H. Arzate. A comparison of internal and external cluster validation indexes, 2011.

- Eréndira Rendón, Itzel M. Abundez, Citlalih Gutierrez, Sergio Díaz Zagal, Alejandra Arizmendi, Elvia M. Quiroz, et H. Elsa Arzate. A comparison of internal and external cluster validation indexes. In *Proceedings of the 2011 American Conference on Applied Mathematics and the 5th WSEAS International Conference on Computer Engineering and Applications*, AMERICAN-MATH'11/CEA'11, page 158–163, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS). ISBN 9789604742707.
- Paul Resnick et Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3) : 56–58, March 1997. ISSN 0001-0782. doi : 10.1145/245108.245121. URL <http://doi.acm.org/10.1145/245108.245121>.
- Marco Tulio Ribeiro, Anisio Lacerda, Adriano Veloso, et Nivio Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, page 19–26, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312707. doi : 10.1145/2365952.2365962. URL <https://doi.org/10.1145/2365952.2365962>.
- Francesco Ricci, Lior Rokach, et Bracha Shapira. *Recommender Systems Handbook*. Springer, Boston, MA, 10 2010. doi : 10.1007/978-0-387-85820-3_1.
- Stephen Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, et M. Lau. Okapi at trec. In *The First Text REtrieval Conference (TREC-1)*, pages 21–30. Gaithersburg, MD : NIST, January 1993. URL <https://www.microsoft.com/en-us/research/publication/okapi-at-trec/>.
- A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, et E. Stamatatos. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1) :5–33, 2017. doi : 10.1109/TIFS.2016.2603960.
- Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, et Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 487–494, Arlington, Virginia, USA, 2004. AUAI Press. ISBN 0974903906.
- Ryan A. Rossi et Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL <http://networkrepository.com>.
- Alan Said, Ben Fields, Brijnesh J. Jain, et Sahin Albayrak. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, page 1399–1408, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450313315. doi : 10.1145/2441776.2441933. URL <https://doi.org/10.1145/2441776.2441933>.
- Ruslan Salakhutdinov, Andriy Mnih, et Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

- Victor Sanh, Lysandre Debut, Julien Chaumond, et Thomas Wolf. Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*, 2019.
- Upendra Sapkota et Thamar Solorio. Sub-profiling by linguistic dimensions to solve the authorship attribution task. In *CLEF (Online Working Notes/Labs/Workshop)*. Citeseer, 2012.
- Upendra Sapkota, Steven Bethard, Manuel Montes, et Thamar Solorio. Not all character n-grams are created equal : A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 93–102, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi : 10.3115/v1/N15-1010. URL <https://www.aclweb.org/anthology/N15-1010>.
- K. Saranya et G. S. Sadasivam. Personalized news article recommendation with novelty using collaborative filtering based rough set theory. *Mobile Networks and Applications*, 22 :719–729, 2017.
- Badrul Sarwar, George Karypis, Joseph Konstan, et John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web, WWW '01*, page 285–295, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133480. doi : 10.1145/371920.372071. URL <https://doi.org/10.1145/371920.372071>.
- Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, et David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.
- Jonathan Schler, Moshe Koppel, Shlomo Argamon, et James Pennebaker. Effects of age and gender on blogging. In *Computational Approaches to Analyzing Weblogs - Papers from the AAAI Spring Symposium, Technical Report*, volume SS-06-03, pages 191–197, 8 2006. ISBN 1577352645.
- Anne Schuth, Floor Sietsma, Shimon Whiteson, Damien Lefortier, et Maarten de Rijke. Multileaved comparisons for fast online evaluation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14*, page 71–80, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325981. doi : 10.1145/2661829.2661952. URL <https://doi.org/10.1145/2661829.2661952>.
- Anne Schuth, Katja Hofmann, et Filip Radlinski. Predicting search satisfaction metrics with interleaved comparisons. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15*, page 463–472, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336215. doi : 10.1145/2766462.2767695. URL <https://doi.org/10.1145/2766462.2767695>.
- Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, et Lexing Xie. Autorec : Autoencoders meet collaborative filtering. In *Proceedings of the 24th International*

- Conference on World Wide Web*, WWW '15 Companion, page 111–112, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334730. doi : 10.1145/2740908.2742726. URL <https://doi.org/10.1145/2740908.2742726>.
- Rico Sennrich, Barry Haddow, et Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi : 10.18653/v1/P16-1162. URL <https://www.aclweb.org/anthology/P16-1162>.
- Yanir Seroussi, Ingrid Zukerman, et Fabian Bohnert. Authorship attribution with topic models. *Computational Linguistics*, 40(2) :269–310, June 2014. doi : 10.1162/COLI_a_00173. URL <https://www.aclweb.org/anthology/J14-2003>.
- M. Sertkan, J. Neidhardt, et H. Werthner. Documents, topics, and authors : Text mining of online news. In *2019 IEEE 21st Conference on Business Informatics (CBI)*, volume 01, pages 405–413, 2019.
- Guy Shani et Asela Gunawardana. *Evaluating Recommendation Systems*, pages 257–297. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi : 10.1007/978-0-387-85820-3_8. URL https://doi.org/10.1007/978-0-387-85820-3_8.
- Yue Shi, Xiaoxue Zhao, Jun Wang, Martha Larson, et Alan Hanjalic. Adaptive diversification of recommendation results via latent factor portfolio. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, page 175–184, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314725. doi : 10.1145/2348283.2348310. URL <https://doi.org/10.1145/2348283.2348310>.
- Thiago Silveira, Min Zhang, X. Lin, Yiqun Liu, et S. Ma. How good your recommender system is ? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, 10 :813–831, 2019.
- R. K. Singh, Kanika Chuchra, et Akshama Rani. A survey on the generation of recommender systems. *International Journal of Information Engineering and Electronic Business*, 9 :26–35, 2017.
- Barry Smyth et Paul McClave. Similarity vs. diversity. In David W. Aha et Ian Watson, editors, *Case-Based Reasoning Research and Development*, pages 347–361, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44593-7.
- L. Son. Dealing with the new user cold-start problem in recommender systems : A comparative review. *Inf. Syst.*, 58 :87–104, 2016.
- Dominic Spohr. Fake news and ideological polarization : Filter bubbles and selective exposure on social media. *Business Information Review*, 34(3) :150–160, 2017.
- E. Stamatatos. Author identification using imbalanced and limited training texts. In *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*, pages 237–241, 2007.
- E. Stamatatos. On the robustness of authorship attribution based on character n-gram features. *Journal of law and policy*, 21 :7, 2013.

- Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3) :538–556, 2009. ISSN 15322882. doi : 10.1002/asi.21001.
- Efstathios Stamatatos. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*, pages 1138–1149, Valencia, Spain, April 2017. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/E17-1107>.
- Efstathios Stamatatos. Masking topic-related information to enhance authorship attribution. *Journal of the Association for Information Science and Technology*, 69(3) : 461–473, 2018. doi : 10.1002/asi.23968. URL <https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.23968>.
- Efstathios Stamatatos, Francisco Rangel, Michael Tschuggnall, Benno Stein, Mike Kestemont, Paolo Rosso, et Martin Potthast. Overview of pan 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 267–285. Springer, 2018.
- Stein Sterling et Shlomo Argamon. A mathematical explanation of burrows’s delta, 01 2006.
- V Subramaniaswamy, R Logesh, M Chandrashekar, Anirudh Challa, et V Vijayakumar. A personalised movie recommendation system based on collaborative filtering. *International Journal of High Performance Computing and Networking*, 10(1-2) :54–63, 2017.
- Md Arafat Sultan, Steven Bethard, et Tamara Sumner. Dls@cu : Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 148–153, Denver, Colorado, June 2015. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/S15-2027>.
- Chi Sun, Xipeng Qiu, Yige Xu, et Xuanjing Huang. How to fine-tune bert for text classification ? In Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, et Yang Liu, editors, *Chinese Computational Linguistics*, pages 194–206, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32381-3.
- K. Surendran, O. P. Harilal, P. Hrudya, Prabakaran Poornachandran, et N. K. Suchetha. Stylometry detection using deep learning. In Himansu Sekhar Behera et Durga Prasad Mohapatra, editors, *Computational Intelligence in Data Mining*, pages 749–757, Singapore, 2017. Springer Singapore. ISBN 978-981-10-3874-7.
- S. Suriati, Meisyarah Dwiastuti, et T. Tulus. Weighted hybrid technique for recommender system. *Journal of Physics : Conference Series*, 930 :012050, dec 2017. doi : 10.1088/1742-6596/930/1/012050. URL <https://doi.org/10.1088/1742-6596/930/1/012050>.
- Mona Taghavi, Jamal Bentahar, Kaveh Bakhtiyari, et Chihab Hanachi. New insights towards developing recommender systems. *Comput. J.*, 61 :319–348, 2018.

- Yla R. Tausczik et James W. Pennebaker. The psychological meaning of words : LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1) :24–54, 2010. ISSN 0261927X. doi : 10.1177/0261927X09351676.
- Mozhgan Tavakolifard, Jon Atle Gulla, Kevin C. Almeroth, Jon Espen Ingvaldesn, Gaute Nygreen, et Erik Berg. Tailored news in the palm of your hand : A multi-perspective transparent approach to news recommendation. In *Proceedings of the 22nd International Conference on World Wide Web, WWW '13 Companion*, page 305–308, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320382. doi : 10.1145/2487788.2487930. URL <https://doi.org/10.1145/2487788.2487930>.
- Michele Trevisiol et Luca Maria Aiello. Cold-start news recommendation with domain-dependent browse graph. *Proceedings of the 8th ACM Conference on Recommender systems - RecSys '14*, pages 81–88, 2014. doi : 10.1145/2645710.2645726. URL http://www.micheletrevisiol.com/papers/recsys2014_trevisiol.pdf.
- Robin Van Meteren et Maarten Van Someren. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age : MLnet/ECML2000 Workshop*, volume 30, pages 47–56, 2000.
- Saúl Vargas et Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11*, page 109–116, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306836. doi : 10.1145/2043932.2043955. URL <https://doi.org/10.1145/2043932.2043955>.
- Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, et Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14*, page 209–216, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326681. doi : 10.1145/2645710.2645743. URL <https://doi.org/10.1145/2645710.2645743>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, et Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.
- P. Vaz, R. Ribeiro, et David Martins de Matos. Litrec vs. movielens - a comparative study. In *KDIR*, 2012a.
- Paula Cristina Vaz, David Martins de Matos, et Bruno Martins. Stylometric relevance-feedback towards a hybrid book recommendation algorithm. In *Proceedings of the fifth ACM workshop on Research advances in large digital book repositories and complementary media*, pages 13–16. ACM, 2012b.
- Paula Cristina Vaz, Ricardo Ribeiro, et David Martins de Matos. Understanding temporal dynamics of ratings in the book recommendation scenario. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication, ISDOC '13*, page 11–15, New York, NY, USA, 2013a. Association for Computing Machinery. ISBN 9781450322997. doi : 10.1145/2503859.2503862. URL <https://doi.org/10.1145/2503859.2503862>.

- Paula Cristina Vaz, Ricardo Ribeiro, et David Martins de Matos. Book recommender prototype based on author’s writing style. In *Proceedings of the 10th conference on open research areas in information retrieval*, pages 227–228, 2013b.
- Maksims Volkovs, Guang Wei Yu, et Tomi Poutanen. Content-based neighbor models for cold start in recommender systems. In *Proceedings of the Recommender Systems Challenge 2017*, RecSys Challenge ’17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450353915. doi : 10.1145/3124791.3124792. URL <https://doi.org/10.1145/3124791.3124792>.
- Hao Wang, Binyi Chen, et Wu-Jun Li. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*, 2013.
- Hongwei Wang, Fuzheng Zhang, Xing Xie, et Minyi Guo. Dkn : Deep knowledge-aware network for news recommendation, 2018.
- Jun Wang, Arjen P. de Vries, et Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’06, page 501–508, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933697. doi : 10.1145/1148170.1148257. URL <https://doi.org/10.1145/1148170.1148257>.
- Qisen Wang, Wenzhong Li, Xiao Zhang, et Sanglu Lu. Academic paper recommendation based on community detection in citation-collaboration networks. In Feifei Li, Kyuseok Shim, Kai Zheng, et Guanfeng Liu, editors, *Web Technologies and Applications*, pages 124–136, Cham, 2016. Springer International Publishing. ISBN 978-3-319-45817-5.
- Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, et Tat-Seng Chua. Denoising implicit feedback for recommendation, 2020.
- Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, et Jun Wang. Dynamic attention deep model for article recommendation by learning human editors’ demonstration. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’17, page 2051–2059, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi : 10.1145/3097983.3098096. URL <https://doi.org/10.1145/3097983.3098096>.
- Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, et Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML ’09, page 1113–1120, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi : 10.1145/1553374.1553516. URL <https://doi.org/10.1145/1553374.1553516>.
- Andrew Weir. Article drop in english headlines. *London : University College MA thesis*, 2009.

- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, et Jamie Brew. Huggingface’s transformers : State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.
- Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, et Xing Xie. Neural news recommendation with multi-head self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6389–6394, Hong Kong, China, November 2019. Association for Computational Linguistics. doi : 10.18653/v1/D19-1671. URL <https://www.aclweb.org/anthology/D19-1671>.
- Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et Ming Zhou. MIND : A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606, Online, July 2020. Association for Computational Linguistics. doi : 10.18653/v1/2020.acl-main.331. URL <https://www.aclweb.org/anthology/2020.acl-main.331>.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, et Jeffrey Dean. Google’s neural machine translation system : Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL <http://arxiv.org/abs/1609.08144>.
- Han Xiao. bert-as-service. <https://github.com/hanxiao/bert-as-service>, 2018.
- Y. Xiao, P. Ai, C. Hsu, H. Wang, et X. Jiao. Time-ordered collaborative filtering for news recommendation. *China Communications*, 12(12) :53–62, 2015.
- Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, et Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’05*, page 114–121, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930345. doi : 10.1145/1076034.1076056. URL <https://doi.org/10.1145/1076034.1076056>.
- Min Yang, Xiaojun Chen, Wenting Tu, Ziyu Lu, Jia Zhu, et Qiang Qu. A topic drift model for authorship attribution. *Neurocomput.*, 273(C) :133–140, January 2018. ISSN 0925-2312. doi : 10.1016/j.neucom.2017.08.022. URL <https://doi.org/10.1016/j.neucom.2017.08.022>.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, et Quoc V Le. Xlnet : Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc., 2019.

- Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, et Suju Rajan. Beyond clicks : Dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 113–120, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326681. doi : 10.1145/2645710.2645724. URL <https://doi.org/10.1145/2645710.2645724>.
- Jing Yuan, Andreas Lommatzsch, et Benjamin Kille. Clicks pattern analysis for online news recommendation systems. In *CLEF (Working Notes)*, pages 679–690, 2016.
- Hossein Rahmatizadeh Zagheli, Mozhdeh Ariannezhad, et Azadeh Shakery. Negative feedback in the language modeling framework for text recommendation. In Joemon M Jose, Claudia Hauff, Ismail Sengor Altingovde, Dawei Song, Dyaa Albakour, Stuart Watt, et John Tait, editors, *Advances in Information Retrieval*, pages 662–668, Cham, 2017. Springer International Publishing. ISBN 978-3-319-56608-5.
- Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, et Ion Stoica. Apache spark : A unified engine for big data processing. *Commun. ACM*, 59(11) :56–65, October 2016. ISSN 0001-0782. doi : 10.1145/2934664. URL <https://doi.org/10.1145/2934664>.
- Fattane Zarrinkalam, Hossein Fani, Ebrahim Bagheri, et Mohsen Kahani. Predicting users' future interests on twitter. In Joemon M Jose, Claudia Hauff, Ismail Sengor Altingovde, Dawei Song, Dyaa Albakour, Stuart Watt, et John Tait, editors, *Advances in Information Retrieval*, pages 464–476, Cham, 2017. Springer International Publishing. ISBN 978-3-319-56608-5.
- Yi Zhang, Jamie Callan, et Thomas Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, page 81–88, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135610. doi : 10.1145/564376.564393. URL <https://doi.org/10.1145/564376.564393>.
- Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, et Tamas Jambor. Auralist : Introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, page 13–22, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450307475. doi : 10.1145/2124295.2124300. URL <https://doi.org/10.1145/2124295.2124300>.
- Pengfei Zhao et Dik Lun Lee. How much novelty is relevant ? it depends on your curiosity. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 315–324, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340694. doi : 10.1145/2911451.2911488. URL <https://doi.org/10.1145/2911451.2911488>.
- Weidong Zhao, Ran Wu, et Haitao Liu. Paper recommendation based on the knowledge gap between a researcher's background knowledge and research target. *Information Processing & Management*, 52(5) :976 – 988, 2016. ISSN 0306-4573. doi :

<https://doi.org/10.1016/j.ipm.2016.04.004>. URL <http://www.sciencedirect.com/science/article/pii/S030645731630070X>.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, et Zhenhui Li. Drn : A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, page 167–176, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi : 10.1145/3178876.3185994. URL <https://doi.org/10.1145/3178876.3185994>.

Rong Zheng, Jiexun Li, Hsinchun Chen, et Zan Huang. A framework for authorship identification of online messages : Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.*, 57(3) :378–393, February 2006. ISSN 1532-2882.

Ke Zhou, Shuang-Hong Yang, et Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11*, page 315–324, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307574. doi : 10.1145/2009916.2009961. URL <https://doi.org/10.1145/2009916.2009961>.

L. Zhou et Huafei Wang. News authorship identification with deep learning, 2016.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, et Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics. doi : 10.18653/v1/P16-2034. URL <https://www.aclweb.org/anthology/P16-2034>.

Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, et Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10) :4511–4515, 2010. ISSN 0027-8424. doi : 10.1073/pnas.1000488107. URL <https://www.pnas.org/content/107/10/4511>.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, et Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web, WWW '05*, page 22–32, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930469. doi : 10.1145/1060745.1060754. URL <https://doi.org/10.1145/1060745.1060754>.

Morteza Zihayat, Anteneh Ayanso, Xing Zhao, Heidar Davoudi, et Aijun An. A utility-based news recommendation system. *Decision Support Systems*, 117 :14 – 27, 2019. ISSN 0167-9236. doi : <https://doi.org/10.1016/j.dss.2018.12.001>. URL <http://www.sciencedirect.com/science/article/pii/S0167923618301970>.