Métrologie des graphes de terrain, application à la construction de ressources lexicales et à la recherche d'information

Emmanuel Navarro

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :
Informatique

Présentée et soutenue par
Emmanuel NAVARRO
le : 4 novembre 2013

Titre :
Métrologie des graphes de terrain, application à la construction de ressources lexicales et à la recherche d'information

---

Directeurs de thèse :
Bruno GAUME
Henri PRADE

Rapporteurs :
Pierre FRAIGNIAUD
Clémence MAGNIEN
Pierre ZWEIGENBAUM

Autres membres du jury :
Frédéric AMBLARD
Eric GAUSSIER

École doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
UMR 5505 IRIT - Institut de Recherche en Informatique de Toulouse

Résumé

Les graphes construits à partir de données réelles d'origines variées (les graphes de synonymie, les réseaux sociaux, les graphes d'interaction de protéines ou encore les graphes issus d'internet, ...) comportent des propriétés communes non triviales. Cela a été découvert il y a maintenant près de quinze ans. Ces graphes sont appelés *complex networks* ou graphes de terrain. Depuis ce nouvel éclairage, il a émergé un nombre considérable de travaux qui tendent à former une nouvelle discipline. On parle de « science des réseaux » (« *network science* ») ou encore de l'étude des grands graphes de terrain. Bien sûr cette discipline hérite de nombreuses théories et résultats plus anciens provenant en particulier de la théorie des graphes, des statistiques ou encore de l'analyse de données. Toutefois, l'objet d'étude et la prise en compte de ces propriétés particulières font que cette discipline se distingue de plus en plus clairement de ses « parents ». Les retombées de l'étude des grands graphes de terrain concernent potentiellement un nombre important de domaines. Il faut cependant remarquer que les applications se développent plus fortement dans certaines disciplines. L'étude des réseaux sociaux est, aujourd'hui, le champ d'application privilégié des travaux sur les graphes de terrain. À l'inverse, par exemple, les travaux en traitement automatique du langage sont beaucoup moins nombreux à intégrer les résultats de l'analyse des graphes de terrain [1].

C'est dans ce cadre général que s'inscrit cette thèse. Entre théories et applications, nous développons et étudions des outils d'analyse de graphe de terrain, et les utilisons ensuite pour résoudre des problèmes de traitement automatique du langage naturel et de recherche d'information. Après un chapitre d'introduction qui présente le contexte scientifique de l'étude des graphes de terrain et définit un certain nombre de notions, ce rapport s'organise en deux parties. Nous nous intéressons tout d'abord aux mesures de proximité pouvant être définies entre les sommets d'un graphe, puis dans une seconde partie aux méthodes de clustering de graphe, et en particulier de graphe biparti.

Une première partie traite donc des mesures de proximité (ou mesures de similarité) entre sommets d'un graphe. Nous dressons tout d'abord un état de l'art de

1. Précisons qu'il existe tout de même de tels travaux, en témoigne le workshop annuel TextGraphs (http://www.textgraphs.org/).

ces mesures tant locales que globales (soit ne dépendant que d'un voisinage immédiat des sommets, soit utilisant la topologie complète du graphe). Nous proposons ensuite une nouvelle méthode, la *confluence*, basée sur des marches aléatoires en temps courts. Cette méthode vérifie plusieurs caractéristiques profitables, en particulier elle se montre indépendante de la densité du graphe. Enfin nous proposons une importante comparaison expérimentale de ces mesures de proximité. Cette comparaison permet de mieux comprendre les différences et ressemblances entre toutes ces mesures, et cela permet de mettre en avant différentes propriétés les caractérisant.

Nous utilisons ensuite ces mesures de proximité entre sommets pour nous pencher sur un autre problème de l'étude des graphes de terrain : mesurer une similarité entre deux graphes. Nous nous concentrons sur les méthodes de comparaison de graphes partageant le même ensemble de sommets. Les méthodes classiques, par exemple la distance d'édition, en considérant les graphes comme des ensembles d'arêtes indépendantes, ne mesurent pas, ou mal, la similarité de structure que peuvent présenter deux graphes. Nous proposons une généralisation de la distance d'édition classique entre graphes qui se montre plus robuste et répond au problème. Cette distance d'édition est mise en application pour comparer et fusionner des graphes de synonymie. Nous montrons notamment que ces ressources peuvent avoir une structure similaire malgré le désaccord apparent qui existe entre leurs ensembles d'arêtes. Cette dernière conclusion a des implications linguistiques quant à la définition de la notion de synonymie.

Nous développons ensuite une application des mesures de proximité entre sommets d'un graphe pour construire un système d'aide à la création de ressources lexicales. L'idée est de pouvoir proposer des candidats synonymes à un contributeur à partir de l'ensemble des relations déjà établies dans la ressource. Une preuve de concept de ce système a été développée sur le dictionnaire collaboratif, le *Wiktionnaire*. En effet, nous présentons une étude de cette ressource qui montre que le nombre d'entrées progresse plus rapidement que le nombre de relations (en particulier de synonymies). Cela indique que les contributions consistent plus facilement en des ajouts d'entrées (et définitions) qu'en des ajouts de relations (en particulier de synonymie). Un système d'assistance à l'ajout de relations répond donc manifestement à un besoin.

Dans une seconde partie nous nous intéressons au problème de clustering de graphe et en particulier de graphe biparti. Nous établissons, dans un premier temps, un parallèle entre l'analyse formelle de concepts et le clustering de graphe biparti. Ce parallèle apporte un éclairage pertinent pour comprendre les différentes méthodes de clustering de graphe. Il permet aussi de nous pencher sur le cas particulier où l'on souhaite partitionner les objets d'un graphe biparti sans imposer de partitionnement correspondant sur les propriétés. Nous proposons et évaluons une méthode simple qui répond à ce problème.

Nous présentons ensuite *Kodex*, un système de classification automatique des résultats d'une recherche. Ce système est une application en recherche d'information des méthodes de clustering de graphe biparti vues précédemment. Une architecture logicielle modulable a été mise en place, elle est présentée dans ses grandes lignes. Elle permet à la fois de construire des scripts pour lancer des campagnes d'évaluation, et de façonner facilement des applications « finales ». Nous avons évalué *Kodex* sur une collection d'un peu plus de deux millions de pages web. Cette évaluation montre les avantages de l'approche et permet en outre de mieux comprendre certaines différences entre méthodes de clustering. Aussi deux applications de démonstration de *Kodex* sont disponibles en ligne. Elles fonctionnent en utilisant les APIs de recherche du Guardian et de DBLP. Nous les présentons brièvement.

Remerciements

Je clôture avec émotion la rédaction de ce rapport par cette traditionnelle et indispensable section de remerciements. Ce n'est pas la plus facile des parties à rédiger tant il me tient à cœur de saluer, avec des mots justes, les nombreuses personnes qui ont parcouru avec moi l'un ou l'autre des chemins qui ont relié mes premiers travaux de recherche à ce présent rapport.

Merci tout d'abord à Bruno et à Henri. Merci pour m'avoir fait confiance, pour m'avoir toujours encouragé. Ce fut une chance incroyable d'avoir deux directeurs de thèse si complémentaires.

Merci à Pierre Fraigniaud, Clémence Magnien et Pierre Zweigenbaum d'avoir accepté d'être rapporteurs de mon travail. Ce n'est pas une tâche aisée que de lire et évaluer les 230 pages qui suivent. Merci pour vos riches retours. De même, merci à Frédéric Amblard et à Eric Gaussier d'avoir accepté d'être membre du jury.

J'ai eu l'opportunité durant ma thèse de profiter de nombreuses collaborations avec plusieurs chercheurs, étudiants et ingénieurs. Sans ce travail en équipe, beaucoup des travaux présentés dans ce rapport n'auraient été possibles.

De toutes ces collaborations, la plus longue et la plus enrichissante aura été celle avec Yannick, collègue et ami précieux. C'est toujours un plaisir de coder, débattre, réfléchir, puis *refactorer* ensemble ! Merci pour tous ces captivants moments partagés avec toi. Je suis sûr que nous allons continuer à faire équipe et que nos milles idées vont nous mener à de belles réalisations.

Merci aussi à Pierre pour nos enrichissantes discussions entre Paris, Toulouse, Taiwan et Beytan, par IRC ou IRL (mais jamais vraiment AFK). Je ne doute pas que nos collaborations vont s'intensifier : la segmentation et le clustering ont beaucoup à faire sans supervision.

Merci à Benoît, en particulier pour le travail sur la comparaison de graphes. Merci à Franck sans qui le travail sur Wiktionary n'aurait pas été possible. Salut et merci à Yann (alors on se le fait ce SLAM v2 ?). Merci à Karen et Guillaume pour leur précieuse collaboration sur l'évaluation de Kodex.

Merci enfin à tous les collègues de l'ENSEEIHT, de l'IRIT ou de CLLE-ERSS. Il est difficile de citer toutes les personnes auxquelles je peux penser, et ce l'est encore plus pour celles que j'oublie. Mais s'il est si plaisant de travailler dans ces lieux, votre bonne humeur et « vos humours » y sont certainement pour beaucoup.

Ce rapport doit aussi beaucoup à nombre de proches et d'amis sans qui ces années de travail n'auraient pas été si agréables.

Merci à mes parents pour leurs encouragements et leur soutien tout au long de mes études. De la prépa jusqu'à cette thèse, en passant par l'école d'ingénieur... la route est longue.

Merci en particulier à mon frère, Étienne, pour tous ses retours (très) pertinents, et je lui souhaite plein de courage pour la rédaction qu'il entame à son tour.

Merci à Choucas et Agnès pour les (plus que) précieuses relectures de dernières minutes.

Une immense embrassade de mercis à toute l'équipe de colocataires : Nico, Nata, Coralie et Seb. Merci d'avoir toujours supporté le Manu tant absorbé dans cette rédaction. Cette embrassade vaut aussi pour Natcho, Yannick, Jojo et Vincent.

Et enfin un immense merci à Sarah, la plus belle des (sages-)femmes avec qui nous vivons cette année « deux » fabuleuses grossesses, dont les accouchements vont se suivre de près. Merci de m'avoir soutenu au quotidien tout en faisant grandir « notre petite crevette » dans ton ventre. Nous nous construisons de merveilleux souvenirs, mais le plus heureux est que ces aboutissements ne sont que le début de nouvelles aventures. Merci d'être là.

Table des matières

Chapitre 1

Introduction

Ce chapitre a pour objectif de définir le contexte scientifique dans lequel s'inscrit ce rapport de thèse. Ce chapitre nous permet aussi de donner un certain nombre de concepts qui seront utilisés par la suite.

Tout d'abord, la section 1.1 retrace l'histoire de l'étude des graphes : de la théorie des graphes aux travaux actuels sur les grands graphes de terrain en passant par le développement de l'étude des graphes sociaux. Nous présentons en section 1.2 l'ensemble des notations et concepts de la théorie des graphes qui seront utiles dans les sections et chapitres suivants. Enfin la section 1.3 définit les caractéristiques particulières communes aux graphes de terrain.

1.1 De la théorie des graphes à l'étude des graphes de terrain

Les graphes sont des outils de modélisation et de raisonnement puissants et néanmoins simples dans leur formalisation. Le développement des théories mathématiques sous-jacentes est relativement ancien, il connaît cependant un renouveau important depuis environ une quinzaine d'années.

On indique en général que l'étude des graphes, comme objets mathématiques, commence en 1735 quand Leonhard Euler présente à l'académie de Saint Pétersbourg le problème des sept ponts de Königsberg (l'article a ensuite été publié en 1741, [Euler, 1741]). Ce problème consiste à trouver un chemin qui permette de revenir au lieu de départ après être passé une seule fois par chacun des sept ponts de la ville. Ce type de chemin dans un graphe est maintenant nommé *chemin eulérien*. Depuis ce premier travail s'est développée une branche des mathématiques discrètes : la *théorie des graphes* [Berge, 1970]. Cette théorie s'intéresse à un certain nombre de questions pratiques, par exemple : quel est le plus court chemin reliant deux sommets ? quelle est la taille du plus court chemin passant par tous les

sommets ? comment, avec le moins de couleurs possibles, colorier chaque sommet d'un graphe de façon à ce que deux sommets adjacents soient toujours de couleurs différentes ? connaissant la capacité de chaque arête, quel est le flux maximal possible entre deux sommets ? ou encore, comment séparer les sommets d'un graphe en deux parties de même taille en retirant un nombre minimal d'arêtes entre ces parties ? Tous ces développements ont eu, et ont encore, d'importantes applications en chimie [Trinajstić, 1983; Faulon, 1998], en physique [Harary, 1967], en ingénierie (électronique, télécommunication, etc.) [Ohlrich et al., 1993; Kahng et al., 2011] et en informatique [Bollobás, 1998; Diestel, 2000]. Il convient aussi de noter que les systèmes de représentation des connaissances constituent une autre application importante de la théorie des graphes. Nous pouvons citer le formalisme des graphes conceptuels [Chein et Mugnier, 2008], ou encore celui des réseaux bayésiens [Naïm et al., 2011].

À partir des années 1920, et parallèlement au développement de la théorie des graphes, un autre usage des graphes se développe en sciences humaines par l'analyse des *réseaux sociaux* [1]. En effet, bien que cette discipline se soit fortement développée depuis moins d'une trentaine d'années, les premiers travaux datent de la première moitié du XXème siècle, voir [Freeman, 1996], [Wasserman et Faust, 1994, chap. 1.2] et [Carrington et al., 2005, chap. 1]. L'analyse des réseaux sociaux, bien que s'appuyant en partie sur la théorie des graphes, pose des problèmes différents de ceux étudiés classiquement dans cette discipline. Par exemple : quels sont les acteurs (i.e. sommets) centraux dans un groupe social ? quelles sont les différentes communautés ? comment modéliser tel type de structure sociale ? Les travaux d'analyse des réseaux sociaux s'appuient donc aussi fortement sur les statistiques et la théorie des probabilités [Wasserman et Faust, 1994; Carrington et al., 2005]. De plus il faut remarquer que les graphes étudiés proviennent de relevés « de terrain ».

Dans ce contexte, plusieurs découvertes faites il y a un peu plus d'une décennie ont engendré un intérêt nouveau à l'étude des graphes au sein de différentes communautés scientifiques. Watts et Strogatz [1998] ainsi que Barabási et Albert [1999] ont montré que des graphes issus de systèmes réels variés (interaction de protéines, liens hypertextes, réseaux sociaux, etc.) possèdent des propriétés non-triviales communes. Dans ces graphes, deux sommets sont toujours reliés par un chemin de longueur faible (on parle du phénomène de *petit monde*). Aussi, alors qu'une large majorité des sommets ont peu de voisins, un certain nombre de sommets sont très fortement liés. Nous présentons plus en détails ces propriétés en section 1.3. Cela a donné un nouvel intérêt à l'étude et à l'utilisation des graphes comme outils de modélisation et a engendré un nombre considérable de travaux qui en sortant du

1. Il faut bien remarquer que l'analyse des réseaux sociaux en sciences humaines a peu à voir avec le développement récent de services de « réseautage » social sur internet. Ces services ouvrent cependant de nouvelles perspectives à l'analyse des réseaux sociaux.

champ de la théorie des graphes classique tendent à créer une nouvelle discipline. On parle de « *network science* » ou de « *network theory* » (science ou théorie des réseaux, en français). Les graphes, issus de systèmes réels et possédant des propriétés non-triviales particulières, sont souvent nommés *complex networks*[2] dans la littérature anglophone (i.e. *réseaux complexes*). On parle parfois aussi de *grands réseaux d'interaction*. Pour notre part, nous préférons le terme *réseaux de terrain* ou *graphes de terrain*[3], pour bien souligner le fait que ces graphes sont issus de données « réelles », de données de « terrain ». Nous parlerons donc de l'*étude de graphes de terrain*, plutôt que de science des réseaux.

Notons que beaucoup de travaux en analyse des réseaux sociaux sont précurseurs de cette nouvelle discipline, tant par les problèmes abordés, les méthodes utilisées, que par les objets d'étude (graphes issus de systèmes réels). D'une autre manière, cette nouvelle discipline hérite aussi directement des travaux de la théorie des graphes s'intéressant aux graphes aléatoires, en particulier les travaux d'Erdős et Rényi [1959, 1960] et ceux qui ont suivi[4].

L'étude des graphes de terrain ou la science des réseaux est donc la discipline qui s'intéresse à développer des outils et méthodes tâchant de modéliser et d'exploiter des données ou systèmes réels prenant la forme de graphes. La simplicité de cette définition indique bien le caractère très général de cette discipline. Effectivement, malgré sa jeunesse, une riche variété de travaux la compose (ce qui ne rend pas évident le dessin de frontières nettes). De plus ce domaine d'étude est pluridisciplinaire à plusieurs niveaux : tout d'abord du fait des applications (en sociologie, biologie, ingénierie, recherche d'information, traitement automatique du langage naturel, etc.), et ensuite du fait de l'origine des chercheurs s'y intéressant (mathématiques discrètes et théorie des graphes, étude des réseaux sociaux, physique statistique, analyse de données, intelligence artificielle, systèmes multi-agents, systèmes complexes, etc.).

Notons, pour finir ce rapide historique, que « l'écosystème » scientifique de cette nouvelle discipline commence à se structurer. Au niveau international, on peut citer la conférence annuelle *NetSci*[5] qui existe depuis 2006. Aussi, en France la conférence *MARAMI*[6] existe maintenant depuis 2010.

2. Le terme *complex networks* provient du parallèle existant avec les *systèmes complexes*. Succinctement, un système est dit « complexe » si on ne peut pas prévoir directement son évolution à partir des règles de fonctionnement de chacun de ses composants.
3. Nous proposons d'utiliser la traduction *terrain network* en anglais.
4. Nous recommandons à ce propos le support de cours de van der Hofstad [2013].
5. http://netsci2013.net
6. http://lipn.fr/marami12/MARAMI2012

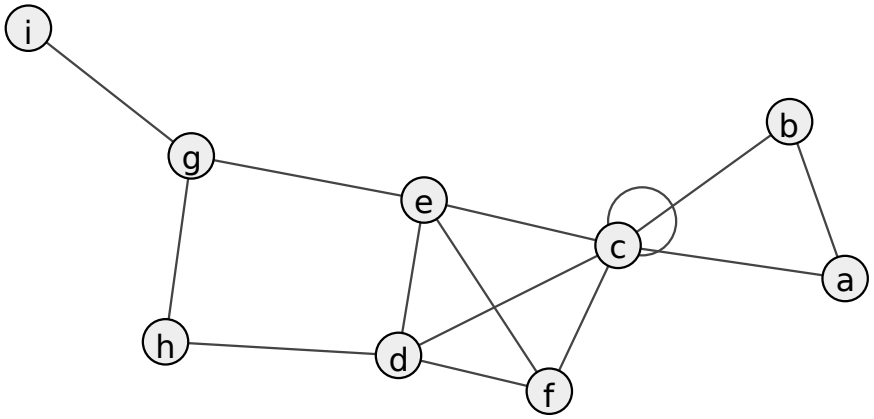


Figure 1.1 – Exemple de représentation d'un graphe $G = (V, E)$, comportant 9 sommets : $V = \{a, b, c, d, e, f, g, h, i\}$ et 14 arêtes : $E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, c\}, \{c, d\}, \{c, e\}, \{c, f\}, \{d, e\}, \{d, f\}, \{d, h\}, \{e, f\}, \{e, g\}, \{g, h\}, \{g, i\}\}$

1.2 Notions de théorie des graphes, notations

Cette section a pour objectif d'introduire les notions de base de la théorie des graphes, ainsi que les propriétés structurelles permettant de caractériser les graphes de terrain. Ces notions seront utiles dans le reste de ce chapitre, ainsi que dans les chapitres suivants. En particulier les propriétés présentées en sous-section 1.2.7 seront utilisées en section 1.3.

1.2.1 Graphes

Un *graphe* $G = (V, E)$ est une paire constituée d'un ensemble de *sommets* V et d'un ensemble d'*arêtes* E. L'ensemble des arêtes E est un sous-ensemble de $V^{(2)}$, l'ensemble des paires non ordonnées de V (chaque arête étant une paire non ordonnée de sommets). Les arêtes sont aussi parfois appelées *lien*. Dans la suite nous supposerons que V (et donc E) est toujours un ensemble fini. Notons que les notations V et E viennent des mots *vertex* et *edge* (sommet et arête en anglais).

La figure 1.1 donne un exemple simple de graphe avec 9 sommets et 14 arêtes. Les graphes sont habituellement dessinés, comme sur la figure, par un ensemble de points (les sommets) reliés par des lignes (les arêtes). Notons que le placement des sommets dans le plan (ou dans un espace à trois dimensions) n'est pas donné par le graphe.

On appelle l'*ordre* d'un graphe, noté $n = |V|$, le nombre de sommets du graphe. La *taille* d'un graphe, notée m, correspond au nombre d'arêtes : $m = |E|$. Deux sommets $u, v \in V$ sont dits *adjacents* (ou *voisins*) si une arête les relie, c'est-à-dire si il existe $e \in E$ telle que $e = \{u, v\}$. Par extension, on dit qu'une paire de

sommets est *adjacente* si les deux sommets qui la composent sont adjacents. On appelle *voisinage* d'un sommet v l'ensemble des sommets adjacents à V. On note $\Gamma(v)$ le voisinage de v :

$$\Gamma(v) = \{u \in V \mid \{v, u\} \in E\} \tag{1.1}$$

Le cardinal du voisinage d'un sommet est son *degré* (ou son *degré d'incidence*) on le note $d(v) = |\Gamma(v)|$. Par exemple, sur le graphe donné par la figure 1.1, le voisinage du sommet f est $\{c, d, e\}$ et son degré vaut $d(f) = 3$.

Une arête $e \in E$ telle que $e = \{v, v\}$ est appelée une *boucle* (*self-loop* en anglais). Par exemple l'arête $\{c, c\}$ est une boucle sur le graphe de la figure 1.1. Un graphe est *réflexif* s'il existe une boucle sur chacun de ses sommets : $\forall v \in V$, $\{v, v\} \in E$.

Il est possible de permettre que des paires de sommets soient reliées par plus d'une arête, on parle alors d'*arêtes multiples*, et de *multigraphe*. À l'inverse on parle de *graphe simple* si un graphe ne comporte aucune boucle et aucune arête multiple.

Le graphe $G' = (V', E')$ est un *sous-graphe* de $G = (V, E)$ si $V' \subseteq V$ et $E' \subseteq E$. Dans ce cas, on peut écrire $G' \subseteq G$. On dit qu'un sous-graphe $G' = (V', E')$ est un *sous-graphe induit* du graphe $G = (V, E)$ par l'ensemble de sommets V', si toutes les arêtes de E reliant les sommets de V' sont dans E', c'est-à-dire :

$$\forall u, v \in V', \quad \{u, v\} \in E \Rightarrow \{u, v\} \in E' \tag{1.2}$$

On note alors $G' = G[V']$.

1.2.2 Graphes dirigés, graphes pondérés et graphes bipartis

Cette sous section présente quelques extensions communes du concept de graphe, extensions que nous utiliserons dans ce rapport (en particulier les graphes bipartis). La définition de graphe présentée dans la sous-section précédente correspond au cas particulier de graphe uni-parti, non-pondéré et non-dirigé.

Graphes dirigés Un graphe $\overrightarrow{G} = (V, E)$ est *dirigé* si E est un ensemble de paires ordonnées, $E \subset V^2$. On parle alors d'*arcs* plutôt que d'arêtes, un arc est noté (u, v) plutôt que $\{u, v\}$. On parle parfois de *graphe orienté* plutôt que de graphe dirigé. Pour chaque sommet $v \in V$ on définit alors $d_{out}(v)$, le *degré sortant* de v comme étant le nombre d'arcs partant de v ; et $d_{in}(v)$, le *degré entrant* de v, le nombre d'arcs arrivant sur v. Dans la suite de ce rapport, nous considérons par défaut les graphes comme non-orientés, dans le cas contraire nous le préciserons explicitement.

Graphes pondérés Un graphe est dit *pondéré* si à chaque arête (ou arc dans le cas d'un graphe dirigé) est associé un poids. On note alors le graphe $G = (V, E, \omega)$, avec ω la fonction de pondération $\omega : E \to \mathbb{R}$ qui donne un poids à chaque arête (ou arc).

Graphes bipartis Un graphe est *biparti* si l'ensemble de ses sommets se découpe en deux classes $V_\top$ et $V_\bot$ telles que les arêtes ne relient que les sommets d'une classe à ceux de l'autre classe. C'est-à-dire, un graphe $G = (V, E)$ est biparti si et seulement si :

$$\text{il existe} \quad V_\top, V_\bot \subset V \quad \text{tels que} \quad \begin{cases} V_\top \cup V_\bot = V, \\ V_\top \cap V_\bot = \emptyset \\ \text{et} \\ E \subset (V_\top \times V_\bot) \cup (V_\bot \times V_\top) \end{cases} \tag{1.3}$$

Intuitivement, un graphe biparti comporte deux types de sommets (par exemple des auteurs $V_\top$ et des articles $V_\bot$) et les arêtes ne relient jamais deux sommets du même type (chaque article est relié à ses auteurs, mais n'est pas relié aux autres articles). Un graphe biparti est aussi appelé *bigraphe*, et peut être noté $G = (V = V_\top \cup V_\bot, E)$ ou directement $G = (V_\top, V_\bot, E)$.

Notons que l'on parle de *graphe uni-parti* lorsque l'on veut préciser qu'un graphe n'est pas (a priori) biparti.

1.2.3 Représentations matricielles usuelles

Un graphe peut être représenté par une matrice. Nous présentons ici les principales matrices pouvant être associées à un graphe. Ces descriptions matricielles des graphes seront intensément utilisées au chapitre 2 pour définir des similarités entre sommets.

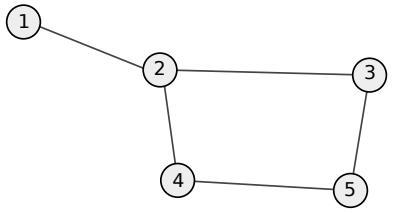
Matrice d'adjacence. Soit un graphe $G = (V, E)$ d'ordre $n = |V|$, la *matrice d'adjacence* de G est une matrice carrée de taille $n \times n$. On note A cette matrice, ses éléments se définissent de la manière suivante :

$$[A]_{ij} = \begin{cases} 1 & \text{si} \quad \{i, j\} \in E \\ 0 & \text{sinon.} \end{cases} \tag{1.4}$$

Chaque ligne et chaque colonne correspondent donc à un sommet, et il y a un 1 entre une ligne et une colonne si les sommets correspondants sont adjacents dans le graphe. La figure 1.2 donne un exemple de graphe avec la matrice d'adjacence associée.

La matrice d'adjacence d'un graphe non dirigé est symétrique ($a_{ij} = a_{ji}$). Les graphes non dirigés sont ainsi parfois nommés *graphes symétriques*. Si une pondération $\omega : V \to \mathbb{R}$ est définie sur les arêtes du graphe alors la matrice d'adjacence peut être définie ainsi :

$$[A]_{ij} = \begin{cases} \omega(i, j) & \text{si} \quad \{i, j\} \in E \\ 0 & \text{sinon.} \end{cases} \tag{1.5}$$



(a) Un graphe simple

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(b) Matrice d'adjacence associée

Figure 1.2 – Exemple de graphe avec sa matrice d'adjacence

Laplacien. La *matrice laplacienne* ou simplement le *Laplacien* est une autre matrice souvent utilisée sur des graphes. Cette matrice, notée L, se définit ainsi :

$$[L]_{ij} = \begin{cases} d(i) & \text{si} \quad i = j \\ -1 & \text{si} \quad i \neq j \text{ et } \{i,j\} \in E \\ 0 & \text{sinon.} \end{cases} \tag{1.6}$$

En notant D la matrice diagonale contenant sur chaque case de la diagonale le degré du sommet correspondant :

$$[D]_{i,j} = \begin{cases} d(i) & \text{si} \quad i = j, \\ 0 & \text{sinon,} \end{cases} \tag{1.7}$$

on a :

$$L = D - A \tag{1.8}$$

Par exemple, pour le graphe de la figure 1.2a on a :

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad \text{et} \quad L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

On utilise aussi parfois $\mathcal{L}$, le *Laplacien normalisé* :

$$\mathcal{L} = D^{-1/2} L D^{-1/2} \tag{1.9}$$

avec $D^{-1/2}$ la matrice diagonale dont chaque case (de la diagonale) est l'inverse de la racine carrée de la case correspondante dans D. Nous allons aussi utiliser $D^{1/2}$ et D^{-1}, qui sont construites similairement (racine carrée ou inverse des éléments diagonaux de D).

Matrice de transition. Enfin nous introduisons P, la *matrice de transition* associée au graphe :

$$P = D^{-1}A \tag{1.10}$$

Par exemple, la matrice de transition du graphe de la figure 1.2a vaut :

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 & 0 \end{bmatrix}$$

Nous reviendrons sur cette matrice au chapitre suivant, lorsque nous définirons des marches aléatoires en temps courts sur un graphe (section 2.3.3). En effet la valeur de $[P]_{ij}$ correspond à la probabilité qu'un marcheur aléatoire présent sur i avance jusqu'au sommet j.

1.2.4 Graphes complets, cliques et bicliques

Un graphe est dit *complet* si tous ses sommets sont adjacents, c'est-à-dire pour tout $u, v \in V$ si $u \neq v$ alors $\{u, v\} \in E$. En général, les boucles sont ignorées pour déterminer la complétude d'un graphe. On note K_n le graphe complet d'ordre n, alors sa taille vaut : $m = \frac{n(n-1)}{2}$.

Dans un graphe G quelconque, on nomme *clique* un sous-ensemble de sommets induisant un sous-graphe complet. Une *k-clique* est une clique de k sommets. Une *clique maximale* est un sous-ensemble maximal de sommets induisant un sous-graphe complet ; c'est-à-dire une clique telle qu'on ne puisse plus lui ajouter de sommet sans qu'elle perde la propriété de clique. Par exemple sur le graphe défini en figure 1.1, $\{c, d, e\}$ est une 3-clique non maximale, car $\{c, d, e, f\}$ est aussi une clique (qui, elle, est maximale).

Un graphe biparti $G = (V_\top, V_\bot, E)$ est *complet* si chaque sommet $u \in V_\top$ d'une classe est connecté à tous les sommets $v \in V_\bot$ de l'autre classe. Une *biclique* est un ensemble de sommets induisant un sous-graphe biparti complet. Notons qu'il peut exister des bicliques sur un graphe uni-parti, en effet un sous-graphe induit peut être biparti sans que le graphe complet ne le soit. Par exemple sur le graphe défini en figure 1.1, le sous-graphe induit par $\{d, e, g, h, i\}$ est biparti (les deux classes de sommets étant : $\{g, d\}$ et $\{e, h, i\}$), et le sous-ensemble de sommets $\{d, e, g, h\}$ forme une biclique. Les bicliques existent, bien sûr, aussi sur les graphes bipartis. De plus, il est facile de montrer que, sur un graphe biparti, un sous-graphe induit est biparti et que chaque sommet garde la même classe. Comme pour les cliques, une biclique est dite *maximale* si on ne peut y ajouter de sommet sans perdre la propriété de biclique. Toujours sur le même graphe en figure 1.1, $\{d, e, g, h\}$ est une

biclique maximale car si on ajoute un sommet, soit le sous-graphe induit n'est plus complet, soit il n'est plus biparti.

1.2.5 Chemins et composantes connexes

Un *chemin* entre un sommet u et un sommet v est une séquence de sommets démarrant par u et terminant par v et telle que chaque sommet est adjacent avec son suivant. La *longueur* (ou *taille*) d'un chemin est le nombre d'arêtes le composant, c'est-à-dire la longueur de la séquence moins un. On note $\langle u, x_1, \ldots, x_{k-1}, v\rangle$ un chemin de longueur k allant du sommet u au sommet v, les sommets intermédiaires étant notés x_1, …, x_{k-1}. On appelle parfois *chaîne* un chemin sur un graphe non-dirigé, en réservant le terme de chemin pour les graphes dirigés. Dans la suite nous utiliserons uniquement le terme de chemin, le contexte permettant de désambiguïser. Sur un graphe dirigé un chemin est orienté, ses arcs doivent être orientés dans le même sens que le chemin. Si le point de départ u et d'arrivée v d'un chemin (ou d'une chaîne) se confondent, alors on parle de *cycle*. Ainsi une *boucle* est un cycle de longueur 1.

Deux sommets sont dits *connectés* s'il existe un chemin de longueur quelconque entre eux. À l'inverse deux sommets sont *déconnectés* s'il n'existe aucun chemin les reliant. Deux sous-ensembles de sommets A, B sont *déconnectés* si chaque sommet de A est déconnecté de tous les sommets de B.

Dans un graphe $G = (V, E)$, un sous-ensemble de sommets $V' \subset V$ est une *composante connexe* s'il existe un chemin reliant chaque paire de sommets de V'. Un graphe $G = (V, E)$ est dit *connexe* si V est une composante connexe. Dans un graphe dirigé, on parle de *composante fortement connexe* si l'orientation des arcs est prise en compte. C'est-à-dire une composante fortement connexe est un ensemble de sommets reliés deux à deux et dans les deux sens par des chemins orientés. À l'inverse, on parle de *composante faiblement connexe* si l'orientation des arcs est ignorée ; il suffit qu'il existe un chemin de u à v (sans forcément qu'il existe de chemin « retour » de v à u) pour que u et v appartiennent à la même composante faiblement connexe. Une composante connexe est *maximale* si on ne peut lui ajouter de sommet sans qu'elle ne perdre sa connexité. Souvent dans la littérature on utilise simplement « composante connexe » pour parler des composantes connexes maximales. Nous ferons cette approximation lexicale dans la suite de ce rapport.

1.2.6 Graphes aléatoires et graphes réguliers

Le terme *graphe aléatoire* indique en général un graphe généré avec le modèle de graphe aléatoire de Erdős et Rényi [1959]. C'est un graphe dont les m arêtes sont choisies aléatoirement de façon équiprobable parmi les $\frac{n(n-1)}{2}$ arêtes possibles (n étant l'ordre du graphe). Un tel graphe est donc défini à partir du nombre de

sommets n et du nombre d'arêtes m. Un modèle de graphe aléatoire semblable (et aussi étudié par Erdős et Rényi) consiste en un graphe où chaque arête existe avec une probabilité p. L'espérance du nombre d'arêtes vaut donc $p.\frac{n(n-1)}{2}$.

Un *graphe régulier* est un graphe dont tous les sommets ont exactement le même degré.

1.2.7 Propriétés structurelles

Nous présentons ici les propriétés structurelles les plus couramment utilisées. Ces mesures permettent, en particulier, de caractériser les graphes de terrain par rapport aux graphes aléatoires. Nous présentons ces caractéristiques en section 1.3

1.2.7.1 Densité, degré moyen et connexité

Les propriétés structurelles les plus évidentes d'un graphe sont son ordre ($n = |V|$) et sa taille ($m = |E|$). Ces deux valeurs permettent de définir deux grandeurs classiques de l'étude des graphes : la densité et le degré moyen.

La *densité*, notée δ, d'un graphe est le rapport entre le nombre d'arêtes du graphe et le nombre d'arêtes pouvant exister, soit :

$$\delta(G) = \frac{2|E|}{|V|.(|V|-1)} \tag{1.11}$$

dans le cas d'un graphe simple non-dirigé.

Plutôt que la densité, la mesure fréquemment utilisée pour rendre compte de la proportion en arêtes du graphe est le *degré moyen* :

$$\langle k \rangle = \frac{1}{|V|} \sum_{v \in V} d(v) = \frac{2|E|}{|V|} \tag{1.12}$$

Nous verrons que souvent la taille des graphes réels croît linéairement par rapport à l'ordre. La densité tend donc vers zéro quand les graphes grandissent, alors que le degré moyen reste à peu près constant. Pour comparer deux graphes d'ordres différents, il est donc préférable de comparer leur degré moyen.

La *connexité* est une autre propriété notoire : à quel point un graphe est-il connexe ? Généralement il existe dans les graphes réels une composante connexe contenant une large majorité des sommets. On nomme *composante connexe principale* (ou *ccp*) cette composante. C'est la composante connexe d'ordre[7] maximal. Pour mesurer à quel point un graphe est connexe, on mesure simplement le rapport entre l'ordre de cette composante principale (noté n_{ccp}) et l'ordre du graphe.

7. L'ordre d'une composante connexe est son nombre de sommets.

1.2.7.2 Longueur moyenne des plus courts chemins

On appelle *distance géodésique* entre deux sommets la longueur d'un [8] plus court chemin les reliant. Elle est définie uniquement si les sommets sont connectés. Pour deux sommets u, v on note $l_{u,v}$ cette distance. Souvent, en théorie des graphes, « la distance entre deux sommets » réfère à la distance géodésique. Bien que ce soit la distance « classique » dans un graphe, ce n'est pas la seule qui soit pertinente. Au chapitre 2 nous présentons un état de l'art des mesures de similarité entre sommets ; beaucoup de ces mesures dérivent de distances (différentes de la distance géodésique). Nous renvoyons à la section 2.3 pour un état de l'art de ces mesures.

Une propriété importante des graphes est la *longueur moyenne des plus courts chemins*. On la note $\mathbf{l}$:

$$\mathbf{l} = \frac{1}{n(n-1)} \sum_{u,v \in V} l_{u,v} \tag{1.13}$$

On définit aussi le *diamètre* d'un graphe comme étant la longueur maximale des plus courts chemins, on le note $\mathbf{d}$:

$$\mathbf{d} = \max_{u,v \in V} l_{u,v} \tag{1.14}$$

Ces deux mesures ne sont bien définies que sur un graphe *connexe*. Or en général les graphes étudiés ne le sont pas. Toutefois ils comportent une *composante connexe principale*, $\mathbf{l}$ et $\mathbf{d}$ sont alors calculés sur cette composante connexe principale. On les note $\mathbf{l}_{ccp}$ et $\mathbf{d}_{ccp}$.

1.2.7.3 Coefficient de transitivité

La transitivité est une notion assez intuitive sur un réseau social : si une personne A est amie avec une personne B qui elle-même est amie avec une personne C, on dit qu'il y a transitivité si A et C aussi sont amies. Cela correspond à la maxime « les amis de mes amis sont mes amis ». Cette propriété des graphes est « un classique » de l'étude des réseaux sociaux [Wasserman et Faust, 1994]. On dit qu'un graphe est *transitif* si et seulement si toute paire de sommets $\{u, v\}$ connectés par un chemin de longueur deux est adjacente. Il en découle que dans un graphe transitif, toute paire $\{u, v\}$ de sommets connectés (i.e. il existe un chemin entre u et v) est une paire adjacente. Dit autrement, chaque composante connexe induit un sous-graphe complet. Attention le sens de *graphe transitif* tel qu'il est défini ici est différent du sens généralement entendu dans la littérature de la théorie des graphes [9]. Nous préférons utiliser ce terme dans ce sens de *relation transitive*, cette notion est en effet plus utile dans nos travaux que celle entendue classiquement en théorie des graphes.

8. Il peut exister plusieurs plus courts chemins.

9. Dans la littérature de la théorie des graphes classique, un graphe est dit transitif si pour toute paire de sommets $\{u, v\}$ il existe un automorphisme (bijection de l'ensemble des sommets vers lui-même qui préserve l'ensemble des arêtes) intervertissant u et v.

Les graphes étudiés ne sont quasiment jamais transitifs ; il est néanmoins intéressant de mesurer à quel point un graphe peut l'être localement. Étant donnés trois sommets $u, v, z \in V$ tels que v soit adjacent aux deux autres ($\{u, v\} \in E$ et $\{v, z\} \in E$), alors le *coefficient de transitivité* (ou *coefficient de clustering*) mesure la probabilité que u et z soient adjacents. On le note $\mathbf{c}$ et il est défini de la manière suivante :

$$\mathbf{c} = \frac{3 \times \text{nombre de triangles}}{\text{nombre de fourches}} \tag{1.15}$$

Les *triangles* sont des sous-graphes complets d'ordre 3 (c'est-à-dire des cycles de longueur 3), et les fourches des sous-graphes d'ordre 3 ne comportant que 2 arêtes. Par exemple sur le graphe donné en figure 1.1 : $\{d, e, f\}$ forme un triangle et$\{i, g, h\}$ une fourche. Le nombre de triangles est multiplié par trois, car chaque triangle compte trois fourches (toujours sur le même exemple, $\{d, e, f\}$ engendre bien trois fourches : *d-e-f*, *e-f-d* et *f-d-e*). Ainsi on assure que $0 \leq \mathbf{c} \leq 1$. Notons que les boucles sont ignorées pour compter triangles et fourches. Par exemple le graphe donné en figure 1.1 (lorsque l'on retire la boucle $\{c, c\}$) comporte 5 triangles et 31 fourches, on a donc $\mathbf{c} \simeq 0.48$.

Une variante locale de ce coefficient est parfois utilisée. On définit $\mathbf{c}_{loc}(v)$ pour chaque sommet $v \in V$:

$$\mathbf{c}_{loc}(v) = \frac{\text{nombre de triangles connectés à } v}{\text{nombre de fourches centrées sur } v} \tag{1.16}$$

Lorsque v ne comporte pas de voisin ou seulement un, alors par définition $\mathbf{c}_{loc}(v) = 0$. Le coefficient $\langle \mathbf{c}_{loc} \rangle$ pour le graphe entier est alors calculé comme la moyenne des $\mathbf{c}_{loc}(v)$ sur l'ensemble des sommets du graphe. Cette mesure est semblable à celle donnée par l'eq. (1.15) mais donne des valeurs différentes. Elle tend en particulier à donner plus de poids aux sommets de faibles degrés, car pour ceux-ci le nombre de fourches (dénominateur) est forcément petit. Dans la suite nous utiliserons uniquement $\mathbf{c}$ défini par l'eq. (1.15). Nous donnons toutefois cette définition locale du coefficient de transitivité pour éviter toute confusion, cette mesure étant assez régulièrement utilisée dans la littérature [10].

Notons enfin que le coefficient de transitivité peut être interprété comme une mesure de la *densité locale*. C'est-à-dire, la densité des sous-graphes induits sur des ensembles de sommets « proches ». De nombreuses mesures de densité locale ont été proposées dans la littérature. Nous renvoyons à [Boccaletti et al., 2006, section 2.1.1] pour un état de l'art.

Le coefficient de transitivité $\mathbf{c}$ peut être adapté aux graphes bipartis de la manière suivante :

$$\mathbf{c}_b = \frac{4 \times \text{nombre de carrés}}{\text{nombre de chemins de longueur trois}} \tag{1.17}$$

10. C'est en particulier cette variante locale qui est utilisée dans l'article fondateur [Watts et Strogatz, 1998], cf. section 1.3

Cette extension est naturelle si l'on considère que **c** est le nombre de cycles non triviaux de longueur minimale (notée l_{min}) divisé par le nombre de chemins de longueur $l_{min} - 1$. Sur un graphe uni-parti les plus petits cycles non triviaux sont bien les triangles (on considère comme trivial un cycle de deux sommets). Or sur un graphe biparti il n'existe pas de triangle, les plus petits cycles non triviaux sont donc bien les carrés (cycles de longueur 4). Cette mesure a été proposée par [Robins et Alexander, 2004] et reprise ensuite par [Latapy et al., 2008]. Notons que la mesure $\mathbf{c}_{loc}$ peut, elle aussi, être adaptée aux graphes bipartis. Mais Latapy et al. [2008] ont montré expérimentalement que les valeurs alors obtenues ne sont pas significativement différentes entre graphes réels et graphes aléatoires de même taille. Cela signifie que ces extensions ne capturent pas de propriété significative. À l'inverse, les mêmes expériences montrent que $\mathbf{c}_b$ capture bien une densité locale plus forte sur des graphes réels que sur des graphes aléatoires.

1.2.7.4 Distribution des degrés

On nomme *distribution des degrés* d'un graphe G, la fonction donnant pour chaque degré k la probabilité de tirer (uniformément) un sommet ayant ce degré. On note $\mathrm{P}(k)$ cette probabilité. Sur un graphe donné, on note $\mathrm{P}(k)$ la proportion (observée) de sommets de degré k. Une des caractéristiques souvent étudiée sur les graphes est la forme de cette distribution des degrés : est-ce que tous les sommets ont presque le même nombre de voisins ? ou au contraire, est-ce qu'il existe des sommets beaucoup plus connectés que les autres ? Nous revenons sur ces questions en section 1.3.2.

Sur un *graphe dirigé* on peut étudier la distribution des degrés entrants $\mathrm{P}_{in}(k)$ et la distribution des degrés sortants $\mathrm{P}_{out}(k)$. Sur un *graphe biparti*, on étudie la distribution des degrés pour chacun des groupes de sommets : $\mathrm{P}_{\top}(k)$ et $\mathrm{P}_{\bot}(k)$.

1.2.7.5 Assortativité et corrélation des degrés

On parle « *d'assortativité* » (*assortativity*, en anglais) lorsque les sommets d'un graphe tendent à être liés principalement à des sommets « similaires ». C'est une propriété particulièrement étudiée dans les réseaux sociaux, où elle correspond à la notion d'*homophilie* : les individus ont davantage tendance à se lier à des personnes semblables qu'à des personnes différentes (d'un point de vue de l'âge, de l'origine sociale, etc.) [Wasserman et Faust, 1994; Carrington et al., 2005].

Nous nous intéresserons spécialement à la *corrélation des degrés* qui est une assortativité particulière : est-ce que les sommets de forts degrés sont plutôt liés à d'autres sommets de forts degrés ? ou sont-ils, à l'inverse, liés à des « petits » sommets ? Cette caractéristique a l'intérêt de ne dépendre que de la topologie du graphe, contrairement aux autres mesures d'assortativités qui dépendent de données

externes (l'âge, l'origine ethnique ou sociale, etc.).

Nous présentons ici la mesure d'assortativité introduite par Newman [2003a] pour les données numériques discrètes, donc adaptée pour mesurer la corrélation de degrés. Notons que cette mesure s'adapte facilement à des données continues, et le même article propose une mesure pour des données discrètes non numériques. Pour un attribut numérique discret défini sur chacun des sommets et prenant valeur dans D, on pose e_{xy} comme la proportion d'arêtes (ou d'arcs) connectant un sommet de valeur x à un sommet de valeur y. Ainsi les e_{xy} vérifient les sommes suivantes :

$$\sum_{x,y\in D} e_{xy} = 1, \quad \sum_{y\in D} e_{xy} = a_x, \quad \sum_{x\in D} e_{xy} = b_y \tag{1.18}$$

où a_x (resp. b_y) est la proportion des arcs qui partent d'un sommet de valeur x (resp. qui aboutissent sur un sommet de valeur y). Si le graphe est non dirigé alors $a_x = b_x$ et a_x est simplement la proportion d'arêtes dont l'une des extrémités a pour valeur x. L'assortativité est mesurée par le *coefficient de Pearson* calculé pour cet attribut en extrémité de chaque arête (ou arc) :

$$\rho = \frac{\sum_{x,y\in D} x.y.(e_{xy} - a_x b_y)}{\sigma_a \sigma_b} \tag{1.19}$$

avec σ_a (resp. σ_b) l'écart-type de a_x (resp. b_x). On a $-1 \leq \rho \leq 1$. Une valeur de ρ proche de 0 indique qu'il n'y a pas d'assortativité dans le graphe. Une valeur positive témoigne de l'existence d'une assortativité. Et à l'inverse une valeur négative indique une « dis-assortativité » c'est-à-dire que les sommets tendent à être connectés avec des sommets différents (de par l'attribut considéré).

Sur un *graphe dirigé* $G = (V, E_d)$, lorsque l'attribut étudié est le degré des sommets, la formulation de ρ peut prendre la forme (facilement calculable) suivante :

$$\rho = \frac{\sum_{e\in E_d} j_e k_e - \frac{1}{m}(\sum_{e\in E_d} j_e)(\sum_{e'\in E_d} k_{e'})}{\sqrt{\left[\sum_{e\in E_d} j_e^2 - \frac{1}{m}(\sum_{e\in E_d} j_e)^2\right]\left[\sum_{e\in E_d} k_e^2 - \frac{1}{m}(\sum_{e\in E_d} k_e)^2\right]}} \tag{1.20}$$

où j_e (resp. k_e) est le degré « *en excès* » du sommet source (resp. cible) de l'arc e, et $m = |E_d|$ le nombre d'arcs du graphe. Le *degré en excès* est simplement le degré du sommet moins un. L'intérêt du degré en excès est de ne pas prendre en compte la contribution au degré de l'arc (ou l'arête) entre les sommets de la paire que l'on considère.

Sur un *graphe non dirigé* $G = (V, E)$, il suffit de transformer chaque arête $\{u, v\}$ en deux arcs (u, v) et (v, u). On note E_d cet ensemble d'arcs, on a alors $|E_d| = 2|E| = 2m$ arcs. Si pour chaque arc $e' \in E_d$ on note $j'_{e'}$ (resp. $k'_{e'}$) les degrés en excès du sommet source (resp. cible), et si pour chaque arête $e \in E$, on note j_e et k_e les degrés en excès de l'un et de l'autre des sommets extrémités, on a alors :

$$\sum_{e'\in E_d} j'_{e'} \;=\; \sum_{e'\in E_d} k'_{e'} \;=\; \frac{1}{2}\sum_{e\in E}(j_e + k_e) \tag{1.21}$$

La formulation de ρ se simplifie alors :

$$\rho = \frac{\sum_{e\in E} j_e k_e - \frac{1}{2m}\left(\sum_{e\in E}(j_e + k_e)\right)^2}{\frac{1}{2}\sum_{e\in E}(j_e^2 + k_e^2) - \frac{1}{2m}(\sum_{e\in E}(j_e + k_e))^2} \tag{1.22}$$

Sur un *graphe biparti* (non dirigé) il est intéressant de mesurer l'assortativité entre les sommets d'un type ($\top$) et les sommets de l'autre ($\bot$). On utilise donc l'équation (1.20), où cette fois j_e (resp. k_e) est le degré (en excès) du sommet de type $\top$ (resp. de type $\bot$) de l'arête e. Notons que cela revient à « diriger » chacune des arêtes du graphe dans le même sens : d'un type de sommet vers l'autre.

1.3 Propriétés non triviales communes

Comme nous l'avons vu en section 1.1, à partir de 1998 il a été découvert que des graphes issus de disciplines variées partagent des propriétés structurelles similaires distinctes de celles des graphes aléatoires [11] et des graphes artificiels en grille. Cela a été le déclencheur d'un important renouveau dans l'étude des graphes. L'objectif de cette section est de présenter plus en détail ces propriétés structurelles particulières.

Les deux caractéristiques principales des graphes de terrain sont présentées en sous-section 1.3.1 pour le phénomène de petit-monde, et en sous-section 1.3.2 pour le phénomène de distribution des degrés en loi de puissance.

1.3.1 Le phénomène de *petit monde*

L'une des caractéristiques en apparence la plus surprenante des graphes de terrain est le phénomène de *petit monde* : la longueur du plus court chemin existant entre deux sommets est toujours « faible », même dans de très grands réseaux.

Ce phénomène a été découvert d'abord sur les réseaux sociaux. Il fut expérimenté pour la première fois par le sociologue Stanley Milgram en 1967, [Milgram, 1967]. Milgram demanda à 60 personnes du Nebraska (États-Unis) de faire parvenir une lettre à une personne vivant dans le Massachusetts. Seulement, bien que connaissant l'adresse du destinataire, les participants ne pouvaient passer la lettre que, de main à main, à des personnes de leur connaissance. Les lettres qui arrivèrent à destination ne passèrent en moyenne que par six personnes [12].

Mais ce n'est pas cette caractéristique seule qui est remarquable. En effet, les graphes aléatoires de type Erdős et Rényi [1959] présentent eux aussi une faible longueur moyenne des plus courts chemins. C'est la combinaison d'une faible longueur moyenne des plus courts chemins (**l** faible) et d'un fort coefficient de transitivité (**c**

11. Graphes aléatoires de type Erdős et Rényi [1959], voir sous-section 1.2.6

12. Il faut préciser que plus de 90% des lettres n'arrivèrent pas à destination, Milgram fut vivement critiqué pour ne pas avoir signalé cela dans son article de 1967.

fort) qui est remarquable dans les graphes réels. C'est ce que Watts et Strogatz ont montré dans leur article de 1998. Les graphes réels se situent ainsi entre des graphes artificiels en « grilles » (fort **c** et fort **l**) et les graphes aléatoires (faible **c** et faible **l**). Watts et Strogatz [1998] montrent que ce phénomène se vérifie sur des réseaux réels de différentes origines (pas seulement sur les réseaux sociaux).

1.3.2 Distribution des degrés en loi de puissance

La seconde caractéristique commune aux graphes de terrain concerne la *distribution des degrés*. La distribution des degrés des graphes aléatoires (type Erdős-Rényi) suit une loi de Poisson, cela signifie que les sommets ont tous un degré relativement proche d'une valeur moyenne. À l'inverse la distribution des degrés des graphes de terrain tend à suivre une loi de puissance :

$$\mathrm{P}(k) \approx C.k^{-\alpha}, \tag{1.23}$$

où C est une constante, et α typiquement compris entre 2 et 3. Cela signifie que, bien que la majorité des sommets aient un faible degré, il existe un nombre non négligeable de sommets de forts degrés. La distribution des degrés présente une forte *hétérogénéité*. Ce phénomène a été mis en évidence par Barabási et Albert [1999].

Les graphes dont la distribution des degrés suit une loi de puissance sont souvent appelés *graphes sans échelle*, car la moyenne d'une variable suivant une loi de puissance n'est pas significative. En effet lorsque α est inférieur à 2 le degré moyen (théorique) diverge, et lorsque α est compris entre 2 et 3 c'est l'écart-type du degré moyen qui diverge.

Notons qu'il n'est pas aisé de vérifier si une distribution suit rigoureusement une loi de puissance. Une approche courante dans la littérature est d'effectuer une régression linéaire sur la distribution transformée dans un espace logarithmique. Cette méthode est une condition nécessaire, mais n'est pas suffisante. Aussi l'universalité du modèle de *graphe sans échelle* a souvent été remise en question (voir par exemple : [Pržulj et al., 2004; Khanin et Wit, 2006]). Récemment Clauset et al. [2009] ont proposé une méthode robuste pour évaluer la vraisemblance qu'une distribution donnée suive une loi de puissance. L'analyse de différents graphes réels amène alors à la conclusion que pour beaucoup de graphes il est raisonnable de décrire la distribution des degrés par une loi de puissance. C'est-à-dire que les données sont compatibles avec l'hypothèse qu'elles proviennent d'une loi de puissance, bien que d'autres lois de distribution puissent aussi être plausibles. À l'inverse l'hypothèse que les valeurs suivent une loi de puissance n'est pas vraisemblable pour certaines des distributions.

Comme le remarquent Clauset et al. [2009] dans leur conclusion, notons que pour de nombreux problèmes le fait important est de savoir que la distribution des

degrés suit une loi dite à « queue lourde » (*heavy tail* en anglais). C'est-à-dire que la distribution est très hétérogène (contrairement à une distribution homogène, comme sur un graphe aléatoire). Le fait que la distribution corresponde rigoureusement à une loi de puissance, une loi log-normale, une loi exponentielle, etc. n'est important que si l'on cherche une modélisation précise.

1.3.3 Plusieurs familles de graphes de terrains ?

Ces deux phénomènes (de « petit monde » et de distribution des degrés très hétérogène) montrent que les graphes de terrain partagent des propriétés non triviales communes. Cela signifie certainement qu'il existe des mécanismes sous-jacents communs à la construction de ces différents graphes ou systèmes qu'ils modélisent. Plus pragmatiquement, cela implique aussi que les outils et algorithmes utiles pour manipuler ces graphes peuvent (ou doivent) exploiter ces propriétés. Toutefois il convient d'être prudent, car ces phénomènes n'impliquent pas forcément que tous ces graphes soient facilement modélisables par un unique mécanisme.

Par exemple, dans la sous-section précédente, nous avons noté que la distribution des degrés ne correspond pas toujours rigoureusement à une loi de puissance (bien que toujours hétérogène) [Clauset et al., 2009]. Ce n'est pas la seule propriété qui ne se comporte pas de la même manière sur tous les graphes réels. En particulier la corrélation des degrés (voir sous-section 1.2.7.5) présente d'importantes différences entre graphes réels [Newman, 2003b]. Le fait que les sommets de forts degrés soient plutôt liés entre eux, ou à l'inverse liés de préférence aux sommets de petits degrés n'est pas universel. Aussi l'étude de la fréquence relative de motifs (sous-graphes de quelques sommets) permet de distinguer des différences de comportement entre graphes. En effet Milo et al. [2004] ont montré qu'il est possible de classer en plusieurs catégories des graphes réels d'origines variées en observant la fréquence d'apparition des différents sous-graphes d'ordre trois ou quatre.

1.4 Conclusion du chapitre

Ce premier chapitre d'introduction nous a permis de replacer l'étude des graphes de terrain dans un cadre historique et scientifique. Nous avons défini les concepts et notations qui seront utilisés dans la suite de ce rapport. Notamment les concepts de base de la théorie des graphes, les notations matricielles et les propriétés structurelles usuelles. Aussi nous avons présenté les différentes propriétés non-triviales communes aux graphes de terrain, à savoir : un fort coefficient de transitivité, une longueur moyenne des plus courts chemin faible et une distribution des degrés très hétérogène.

La suite de ce rapport se divise en deux parties. Tout d'abord nous nous intéressons à la définition de mesures proximité ou similarités entre sommets d'un

graphe. Nous utilisons ensuite ces mesures pour comparer des graphes, et dans une application en proposant un système d'assistance à la construction de ressources lexicales. Dans une seconde partie, nous nous concentrons sur le problème de clustering de graphe biparti. Ce problème est mis en parallèle avec les méthodes d'analyse formelle de concepts, puis différentes méthodes nouvelles sont proposées. Enfin une application de ces méthodes est proposée, elle consiste en un système de classification automatique des résultats d'une recherche d'information.

Première partie

Marches aléatoires courtes, application à différents problèmes de l'étude des graphes de terrain

Chapitre 2

Similarités entre sommets

Ce chapitre poursuit trois objectifs. Tout d'abord une nouvelle mesure de similarité entre sommets est introduite : la *confluence*[1]. Ensuite nous établissons un état de l'art des différentes similarités de la littérature. Enfin nous proposons, dans une dernière section, une comparaison expérimentale de ces mesures.

2.1 Introduction

Nous nous intéressons dans ce chapitre aux mesures de similarité, ou mesures de proximité, entre sommets d'un graphe. Notons que les deux notions de « proches » et de « similaires » (pour deux sommets) seront confondus dans la suite (et « similarité » sera préféré à « proximité »). En effet les méthodes présentées ici se comprennent naturellement comme des mesures de la proximité (entre sommets) mais peuvent aussi se comprendre (nous allons le voir tout de suite) comme des similarités.

L'approche classique pour définir une similarité entre objets est de mesurer la ressemblance de leurs attributs. Deux personnes sont (socialement) « similaires » si elles ont le même âge, le même sexe, le même travail, etc. Les attributs d'un sommet d'un graphe sont soit les voisins de ce sommet (c'est-à-dire la topologie du graphe), soit des données supplémentaires disponibles sur les sommets en dehors de la topologie du graphe. Dans ce chapitre, nous nous plaçons uniquement dans le premier cas. C'est-à-dire que nous considérons que les uniques attributs connus d'un sommet sont ses voisins. Deux sommets sont donc similaires si leurs voisinages, ou leur « positions » dans le graphe, sont similaires. Nous allons voir que les méthodes dites « locales » se comprennent naturellement dans ce cadre là (voir section 2.3.1).

Une autre manière d'imaginer et de comprendre les mesures de similarité entre sommets est de considérer les arêtes d'un graphe comme des indications binaires sur la similarité des sommets. Caricaturalement, deux sommets adjacents sont similaires,

1. Introduire une nouvelle mesure avant l'état de l'art permet de définir convenablement les notations et concepts utilisés ensuite pour décrire concisément les méthodes de l'état de l'art.

deux sommets non-adjacents ne le sont pas. Évidemment, utilisée telle quelle, cette information n'est pas suffisante[2]. Deux sommets non-adjacents peuvent être similaires, au moins jusqu'à un certain degré. Par exemple dans le graphe de synonymie *DicoSyn* aucun des trois mots « *épée* », « *couteau* » et « *poubelle* » ne sont adjacents, pourtant il est possible d'observer que « *épée* » et « *couteau* » sont plus proches que « *épée* » et « *poubelle* ». Pour cela, il faut considérer une certaine *transitivité* : « *épée* » et « *couteau* » partageant un certain nombre de voisins (*dague, poignard, lame*, etc.) ce qui n'est pas le cas de « *épée* » et « *poubelle* ». Dans cette optique, deux sommets sont similaires s'il existe de nombreux chemins courts entre eux. Aussi, il est possible d'observer des paires de sommets peu similaires et pourtant adjacents. Là encore la topologie du graphe peut aider à « corriger » l'indication de similarité atomique donnée par chaque arête. L'idée est donc de calculer une similarité entre les sommets en « lissant » la similarité binaire donnée par les arêtes du graphe. Ce point de vue permet de comprendre intuitivement la plupart des méthodes dites « globales » que nous allons présenter en section 2.3.2.

Ces deux approches de la similarité entre sommets dans un graphe, c'est-à-dire similarité du voisinage et lissage des « liens de similarité » donnés par le graphe, sont deux points de vue différents qui ne sont pas forcément antinomiques. Par exemple mesurer le nombre de voisins communs à deux sommets, méthode qui relève a priori du premier point de vue, est équivalent à la mesure du nombre de chemins de longueur deux existant entre deux sommets.

2.2 Proposition d'une mesure normalisée

Avant de présenter dans la section suivante un état de l'art des différentes mesures de similarité existant dans la littérature, nous introduisons dans cette section une nouvelle mesure de similarité entre sommets : la *confluence*. Cette mesure, basée sur des marches aléatoires en temps courts, a l'avantage d'être comprise entre 0 et 1 et d'être normalisée par rapport à un modèle de graphe aléatoire conservant le degré de chaque sommet. Ainsi montrons que la valeur de la *confluence* n'est pas sensible à la densité du graphe ni aux degrés des sommets comparés.

Avant de présenter la *confluence* (sous-section 2.2.2), nous présentons le mécanisme de marches aléatoires dans un graphe (sous-section 2.2.1). Nous verrons dans la section suivante qu'il existe d'autres mesures de similarité se basant sur des marches aléatoires semblables. Le principal avantage des marches aléatoires en temps courts est que leur calcul se fait avec une complexité temporelle raisonnable.

2. Notons que certaines applications peuvent considérer un graphe directement comme une similarité binaire. C'est par exemple le cas de méthodes de recherche d'information, qui pour lutter contre la faible densité des données, considèrent en plus des mots de la requête l'ensemble de leurs synonymes [Manning et al., 2008, section 9.2.2].

Aussi le résultat d'une marche en temps court peut être comparé à la limite obtenue pour une même marche mais après un temps infini. Nous allons exploiter cette limite en introduisant la mesure de *confluence*.

2.2.1 Marches aléatoires en temps courts

Nous définissons ici le mécanisme de marches aléatoires dans un graphe. L'idée est de considérer un marcheur passant aléatoirement, à chaque pas de temps, d'un sommet à l'autre du graphe en suivant les arêtes.

2.2.1.1 Matrice de transition

Imaginons donc un marcheur présent sur un sommet du graphe et qui se déplace, à chaque pas de temps, en suivant une arête choisie aléatoirement. Ce processus correspond à la *chaîne de Markov*, ou au processus de Markov, associé au graphe. Cette chaîne de Markov est caractérisée par la *matrice de transition* P suivante :

$$P = (p_{u,v})_{u,v \in V}, \quad \text{avec} \quad p_{u,v} = \begin{cases} \frac{1}{|\Gamma(u)|} & \text{si} \quad \{u,v\} \in E, \\ 0 & \text{sinon.} \end{cases} \tag{2.1}$$

avec $\Gamma(u)$ l'ensemble des voisins du sommet u. Notons que la définition matricielle suivante est équivalente :

$$P = D^{-1}A \tag{2.2}$$

où D est une matrice diagonale contenant sur chaque composante (de la diagonale) le degré du sommet correspondant. Le tableau 2.1 (page 44) récapitule les matrices usuelles associé à un graphe.

Remarquons qu'une ligne i de P décrit la probabilité de passer du sommet i vers chacun des autres sommets du graphe. Si le graphe est *non dirigé* et ne comporte pas de sommet isolé (c'est-à-dire chaque sommet possède au moins un voisin), alors P est une matrice stochastique. En effet il est facile de vérifier que la somme de chaque ligne vaut 1. La probabilité qu'un marcheur passe d'un sommet u à un sommet v en t pas, que l'on note $prx_t(u,v)$, est alors donnée par :

$$prx_t(u,v) = \left[P^t\right]_{u,v} = \delta_u P^t \delta_v^T \tag{2.3}$$

avec δ_i un vecteur ligne constitué de zéros, sauf pour le $i^{ème}$ élément qui vaut 1 et $(.)^T$ l'opérateur de transposition. Plus généralement, si P_0 est la distribution de probabilité initiale du marcheur sur l'ensemble des sommets (vecteur ligne) alors $P_0.P^t$ est la distribution de probabilité décrivant la position du marcheur après t pas. La matrice P^t donne l'ensemble des probabilités de passer d'un des sommets à un autre en t pas.

Il faut bien souligner que les notations utilisées ici ($prx_t(u,v)$, P, etc.) ne précisent pas le graphe concerné. Ce choix facilite la lecture, car la plupart du temps il n'y a pas d'ambiguïté quant au graphe utilisé. Aussi nous travaillons avec des vecteurs lignes ($prx_t(u,v)$, P_0, etc.) plutôt qu'avec des vecteurs colonnes. Ce n'est pas forcément le choix fait dans la littérature, nous le préférons pourtant car il évite de balader des opérateurs de transposition dans les écritures [3].

Proposition 1. *Si P est la matrice de transition associée à un graphe G non-dirigé et ne comportant pas de sommets isolés, on a :*

$$D.P^t = (D.P^t)^T \tag{2.4}$$

ce qui signifie, pour chaque case (u,v) de $D.P^t$:

$$|\Gamma(u)|.prx_t(u,v) = |\Gamma(v)|.prx_t(v,u) \tag{2.5}$$

Autrement dit la probabilité pour un marcheur de passer de u vers v multipliée par le degré de u est égale à la probabilité de passer de v vers u multipliée par le degré de v.

Démonstration. Comme D est symétrique, il suffit de remarquer que $DP^tD^{-1} = (P^t)^T$. En effet on a :

$$DP^tD^{-1} = D(D^{-1}A)^tD^{-1} = (AD^{-1})^t$$

De plus, comme le graphe est non-dirigé, A est symétrique, et D^{-1} l'est aussi car c'est une matrice diagonale. On a alors :

$$(AD^{-1})^t = (A^T{D^{-1}}^T)^t = (P^T)^t = (P^t)^T$$

□

Graphes pondérés. Il est possible de prendre en compte une fonction de pondération $w : E \to \mathbb{R}^+$ disponible sur les arêtes du graphe. La matrice de transition P se définit alors de la manière suivante :

$$P = (p_{u,v})_{u,v\in V}, \quad \text{avec} \quad p_{u,v} = \begin{cases} \frac{w(u,v)}{\sum_{w\in\Gamma(u)} w(u,w)} & \text{si} \quad \{u,v\} \in E, \\ 0 & \text{sinon.} \end{cases} \tag{2.6}$$

Le marcheur passe alors sur un voisin avec une probabilité proportionnelle au poids du lien entre les deux sommets.

3. En effet on trouve souvent dans la littérature exposé que la distribution d'un marcheur après t pas vaut : $(P^T)^tP_0$.

Complexité. Le calcul de P^t peut être fait récursivement avec $P^0 = I$ et $P^t = P^{t-1}.P$. Chaque étape consiste donc en un produit d'une matrice, a priori pleine, par une matrice creuse. En effet les graphes réels étant peu denses, leurs matrices d'adjacence (et donc de transition) sont creuses. Ce calcul peut être fait en $O(nm)$. Le calcul de P^t est donc effectué en $O(tnm)$.

Si l'on ne souhaite calculer qu'une seule ligne de P^t, alors il faut effectuer t multiplications d'un vecteur par une matrice creuse. La complexité temporelle est donc en $O(tm)$.

2.2.1.2 Convergence

Sur un graphe $G = (V, E)$ connexe, non-dirigé, non-périodique la probabilité qu'un marcheur aléatoire soit en un sommet u après une marche de longueur infinie vaut :

$$\pi(u) = \frac{|\Gamma(u)|}{\sum_{w \in V} |\Gamma(w)|} \tag{2.7}$$

Cette limite ne dépend donc pas du sommet de départ, et est simplement proportionnelle au degré du sommet d'arrivée. La démonstration se fait à l'aide du théorème de Perron-Frobenius, nous renvoyons pas exemple à [Pons, 2007, section 2.2, page 29] pour une démonstration complète. Lorsqu'une pondération est prise en compte, il est facile de montrer que cette limite $\pi(u)$ devient :

$$\pi(u) = \frac{\sum_{w \in \Gamma(u)} w(u, w)}{\sum_{v \in V} \sum_{w \in \Gamma(q)} w(v, w)} \tag{2.8}$$

Graphe connexe. Cette limite est définie sur un graphe connexe, cela peut poser problème en pratique. Une solution simple consiste à considérer chaque composante connexe indépendamment. Notons aussi que la plus grande composante connexe représente souvent une large majorité des sommets ; il est alors possible de se restreindre uniquement à cette plus grande composante connexe. Une autre solution est de toujours considérer le graphe comme connexe pour le calcul de la limite. Les sommets non-connectés ont alors une limite non nulle, et les sommets appartenant à la même composante connexe une limite un peu sous-estimée. En pratique, cela ne change pas grand chose pour des graphes dont la plus grande composante connexe contient la quasi totalité des arêtes.

Graphe périodique. Un graphe est dit *périodique* si les longueurs de tous ces cycles sont divisibles par un entier strictement supérieur à 1. Par exemple, les graphes bipartis sont des graphes périodiques de période 2 (ce sont les seuls graphes non-dirigés périodiques). On comprend aisément qu'une marche aléatoire ne peut converger vers une limite unique sur un graphe périodique. Il faut, pour assurer la

convergence, que le graphe soit *non-périodique* (où *apériodique*). C'est-à-dire que chaque sommet (de la composante connexe) puisse être atteint depuis n'importe quel autre en un nombre *quelconque* de pas (à condition que ce nombre soit suffisamment grand). Pour s'assurer de la non périodicité d'un graphe, une astuce simple consiste à ajouter une boucle sur chaque sommet. Le graphe est ainsi forcé réflexif. Souvent sur des données réelles aucune information n'est portée par ces liens de réflexivité, car soit aucun des sommets n'est relié à lui même, soit tous le sont. Aussi pour certaines des mesures basées sur les marches aléatoires courtes, nous transformons initialement le graphe pour ajouter une boucle sur chaque sommet.

2.2.1.3 Graphe bipartis

Les marches aléatoires fonctionnent de la même manière sur un graphe biparti, les définitions données jusque là sont donc toujours valables. Seulement un graphe biparti est un graphe de période 2 : en un temps pair seuls les sommets du même groupe que le sommet de départ sont accessibles, à l'inverse en un temps impair seuls ceux de l'autre type sont accessibles. Utiliser des marches aléatoires sur un graphe biparti va donc consister à effectuer *soit* des marches en temps pairs *soit* des marches en temps impairs. Un temps pair permet d'obtenir des probabilités d'atteindre les sommets du même type, alors qu'un temps impair donne des probabilités d'atteindre les sommets de l'autre type. Le fait important est que les marches en temps pairs, et impairs convergent chacune, et chacune vers deux limites différentes.

Proposition 2 (Projection)**.** *Une marche aléatoire en temps pair* $2t$ *sur un graphe biparti* $G = (V = V_\top \cup V_\bot, E)$ *est équivalente à une marche aléatoire en temps* t *sur le graphe projeté pondéré* $G_{proj} = (V = V_\top \cup V_\bot, E_{proj}, w_{proj})$*. Avec* $\{u, v\} \in E_{proj}$ *si et seulement si* u *et* v *ont (au moins) un voisin en commun dans* G*, l'arête les reliant est alors pondérée par :*

$$w_{proj}(u, v) = \sum_{w \in \Gamma_G(u) \cap \Gamma_G(v)} \frac{1}{d_G(w)} \tag{2.9}$$

Démonstration. Comme G est biparti, sa matrice d'adjacence A est anti-diagonale par bloc. Si l'on range les sommets de $V_\top$ avant ceux de $V_\bot$, on peut écrire A de la manière suivante :

$$A = \begin{bmatrix} 0 & A' \\ A'^T & 0 \end{bmatrix} \tag{2.10}$$

où A' est une matrice rectangulaire de taille $|V_\top| \times |V_\bot|$. La matrice de transition P associée est donc elle aussi anti-diagonale par bloc :

$$P = \begin{bmatrix} 0 & P_1 \\ P_2 & 0 \end{bmatrix} \tag{2.11}$$

P_1 donne la probabilité de passer d'un sommet de $V_\top$ vers un sommet de $V_\perp$, et P_2 la probabilité de passer d'un sommet de $V_\perp$ vers un sommet de $V_\top$. On remarque alors que la matrice P^2 est diagonale par bloc :

$$P^2 = \begin{bmatrix} P_1.P_2 & 0 \\ 0 & P_2.P_1 \end{bmatrix} \tag{2.12}$$

Alors pour tout $t \geq 0$, on a :

$$P^{2t} = (P^2)^t = \begin{bmatrix} (P_1.P_2)^t & 0 \\ 0 & (P_2.P_1)^t \end{bmatrix} \tag{2.13}$$

Il est alors trivial de voir qu'une marche en temps pair $2t$ sur G est équivalente à une marche de temps t sur le graphe dont la matrice de transition est P^2. Montrons que ce graphe est bien G_{proj}.

Considérons le graphe donné par la matrice d'adjacence pondérée suivante :

$$A_{proj} = A.D^{-1}.A = \begin{bmatrix} A'.D_\perp.A'^T & 0 \\ 0 & A'^T.D_\top.A' \end{bmatrix} \tag{2.14}$$

Il est facile de vérifier que :

$$[A_{proj}]_{u,v} = \sum_{w \in V_\top \cup V_\perp} \frac{[A]_{u,w}\,[A]_{w,v}}{\sum_{y \in V_\top \cup V_\perp} [A]_{w,y}} \tag{2.15}$$

ou simplement si A est binaire :

$$[A_{proj}]_{u,v} = \sum_{w \in \Gamma_G(u) \cap \Gamma_G(v)} \frac{1}{d_G(w)} \tag{2.16}$$

A_{proj} est donc la matrice d'adjacence du graphe G_{proj}. De plus on peut vérifier que :

$$\sum_{v \in V_\top \cup V_\perp} [A_{proj}]_{u,v} = \sum_{w \in V_\top \cup V_\perp} [A]_{u,w} \tag{2.17}$$

C'est-à-dire que le degré (ou ici le poids) d'un sommet est le même dans les graphes projetés (les deux composantes connexes de G_{proj}) que dans le graphe G de départ. Ainsi la matrice de transition de G_{proj} n'est autre que P^2.

□

Proposition 3 (Convergence)**.** *Sur un graphe G biparti et connexe, une marche aléatoire partant uniquement de sommets de $V_\top$ converge en temps pair vers* 0 *pour les sommets de $V_\perp$ et vers :*

$$\pi(u) = \frac{d(u)}{\sum_{v \in V_\top} d(v)} \tag{2.18}$$

pour les sommets de $V_\top$. *La même marche en temps impair converge vers* 0 *pour les sommets de* $V_\top$ *et vers :*

$$\pi(u) = \frac{d(u)}{\sum_{v \in V_\perp} d(v)} \tag{2.19}$$

pour les sommets de $V_\perp$.

Pour démontrer cette proposition il faut utiliser les propositions 1 et 2. En effet les deux composantes du graphe G_{proj} sont connexes si G est connexe, de plus G_{proj} comporte par construction des boucles sur chacun de ces sommets, il est donc non-périodique. Enfin il faut remarquer qu'une marche aléatoire en temps impair partant de sommets de $V_\top$ correspond à une marche aléatoire en temps pair partant des sommets de $V_\perp$.

2.2.2 Mesure de confluence

Nous introduisons ici une mesure que nous nommons la *confluence*. Cette mesure se base sur des marches aléatoires en temps courts. L'idée est de normaliser la probabilité de passer d'un sommet à un autre par la probabilité limite d'atteindre ce sommet. Deux sommets auront un score de confluence fort si la probabilité de passer de l'un à l'autre en t pas sur le graphe est forte et si la probabilité d'atteindre l'un ou l'autre des sommets « à l'infini » est faible. Nous montrons par ailleurs que cette probabilité limite correspond à la probabilité de passer de l'un à l'autre des sommets dans un graphe aléatoire équivalent où chaque sommet garde le même degré.

Nous proposons de normaliser le résultat d'une marche aléatoire en temps court de la manière suivante :

$$\mathrm{confl}_t(u, v) = \frac{[P^t]_{u,v}}{[P^t]_{u,v} + \pi(v)} \tag{2.20}$$

Par construction la confluence est comprise entre 0 et 1. Dans le cas où $[P^t]_{u,v} = \pi(v)$, on a $\mathrm{confl}_t(u, v) = 0.5$. Si la confluence est supérieure à 0.5 cela signifie que la probabilité d'aller de u vers v en un faible nombre de pas (t) est plus forte que la probabilité d'arriver en v après un nombre de pas infini. À l'inverse si la valeur est inférieure à 0.5 alors il est plus probable de se retrouver sur v après une marche infinie que après quelques pas.

On peut vérifier que cette mesure est *symétrique*. En effet d'après la proposition 1 on a : $[P^t]_{u,v} = {}^{d(v)}/_{d(u)}.\,[P^t]_{v,u}$ et on a $\pi(v) = {}^{d(v)}/_{d(u)}.\pi(u)$.

Interprétation par rapport à un *null model*. Il est possible grâce à la proposition suivante d'interpréter la limite π_v et donc la confluence par rapport au *null model*[4] utilisé en particulier par la *modularité* de Newman [2006].

4. Un *null model* est un graphe aléatoire possédant certaines propriétés d'un graphe donnée.

Proposition 4. *Sur un graphe $G_{nm} = (V, E_{nm})$ de même ordre que G et tel que chaque sommet a le même degré que dans G, alors la probabilité qu'un marcheur arrive en un sommet v après t pas ($t > 0$) depuis un sommet u vaut :*

$$\pi_v = \frac{d(v)}{\sum_{i \in V} d(i)} \tag{2.21}$$

G_{nm} pouvant posséder des arêtes multiples et des boucles.

Démonstration. Le graphe G_{nm} correspond au modèle de configuration [5] avec une séquence des degrés imposée par G. Un tel graphe est construit en donnant $d(i)$ « demi-arêtes » à chaque sommet, et en tirant aléatoirement un couplage entre toutes ces demi-arêtes. Si l'on note $d(i)$ le degré d'un sommet i et m le nombre total d'arêtes, on montre que l'espérance du nombre d'arêtes [6] $e_{u,v}$ entre deux sommets u et v vaut :

$$e_{uv} = \frac{d(u)d(v)}{2m} \tag{2.22}$$

Effectivement, il y a $d(u)$ arêtes partant de u et chacune a $2m$ points d'arrivée possibles, et seulement $d(v)$ de ces points d'arrivée appartiennent à v. Pour une démonstration différente et plus détaillée de cette équation nous renvoyons à [Newman, 2006, section III].

La probabilité p_{uv} qu'un marcheur sur ce graphe passe du sommet u au sommet v en 1 pas vaut donc :

$$p_{uv} = \frac{e_{uv}}{d(u)} = \frac{d(v)}{2m} = \pi_v \tag{2.23}$$

car on a bien $m = \frac{1}{2} \sum_{i \in V} d(i)$ sur un graphe non dirigé.

Il faut alors remarquer que la probabilité de passer d'un sommet u à un sommet v en 2 pas est bien la même que celle de passer de u à v en un seul pas :

$$\sum_{w \in V} p_{vw}.p_{wu} = \sum_{w \in V} \left(\frac{d(w)}{2m}.\frac{d(u)}{2m}\right) \tag{2.24}$$

$$= \frac{d(u)}{2m}.\frac{1}{2m} \sum_{w \in V} d(w) \tag{2.25}$$

$$= \frac{d(u)}{2m} = p_{vu} \tag{2.26}$$

On a alors par récurrence que la probabilité d'atteindre un sommet v en temps quelconque (> 0) vaut π_v. □

5. Voir par exemple [van der Hofstad, 2013, chap. 7].
6. Rappel : il peut y avoir plusieurs arêtes entre deux sommets.

Mesure entre sommets adjacents. Nous proposons une variante de la confluence où la similarité entre deux sommets adjacents est calculée en retirant du graphe l'arête les reliant. Nous nommons CFLtR cette variante. Pour chacune des m arêtes du graphe il faut t fois multiplier un vecteur par une matrice creuse (de m éléments non nuls) donc effectuer un calcul en $O(mt)$. Le surcoût est donc en $O(m^2t)$.

Graphe périodique et non-connexe. Notons que nous utilisons pour le calcul de la confluence, sur les graphes non bipartis, l'astuce présentée dans la sous-section suivante qui consiste à ajouter une boucle sur chacun des sommets. Cela permet d'éviter les phénomènes de rebond dus à une éventuelle périodicité dans le graphe. Aussi pour un graphe non-connexe, il est possible soit de se restreindre à la plus grande composante connexe, soit de calculer la limite comme si le graphe était connexe (voir sous-section précédente).

2.3 État de l'art des mesures de similarité entre sommets

Nous présentons ici un panorama aussi complet que possible des mesures de similarité entre sommets que l'on trouve dans la littérature. Tout d'abord, en sous-section 2.3.1, les méthodes dites « locales » sont introduites. Ce sont les méthodes ne prenant en considération que les voisinages directs des sommets comparés. Ensuite, en sous-section 2.3.2, nous présentons les méthodes « globales », c'est-à-dire prenant en considération l'ensemble du graphe. Enfin en sous-section 2.3.3 nous exposons les méthodes dites « semi-globales ». Ce sont des méthodes qui, comme la *confluence*, sont basées sur des marches aléatoires à temps courts. Elles ne dépendent donc que d'un voisinage limité de chaque sommet.

Nous présentons ici surtout des méthodes symétriques, et seules les méthodes symétriques seront évaluées. Notons qu'une mesure non-symétrique peut être intéressante dans certains cas. Par exemple, il peut être sensé que la similarité de *Cambodge* « vers » *Chine* soit plus importante que la similarité de *Chine* « vers » *Cambodge*.

Le présent travail est basé sur plusieurs états de l'art existants. Notamment sur [Lü et Zhou, 2011] qui est, à notre connaissance, la revue de la littérature la plus récente et complète. Toutefois cette revue est pensée dans le cadre du problème de prédiction de lien. Notons que nous avons essayé de reprendre autant que possible les notations introduites dans cet article. Concernant les méthodes à noyaux (voir plus bas) un très bon état de l'art est proposé par Fouss et al. [2012]. Enfin on peut citer [Liben-Nowell et Kleinberg, 2007] et [Hasan et Zaki, 2011] même si ces deux travaux sont un peu moins complets.

Rappel des notations. Nous utilisons ici les notations définies au chapitre précédent. En particulier rappelons que $\Gamma(v)$ est l'ensemble des voisins du sommet v, et $d(v) = |\Gamma(v)|$ est le degré de v. Nous utilisons aussi les différentes matrices définies en section 1.2.3. Le tableau 2.1 récapitule les définitions de ces matrices.

2.3.1 Mesures locales : comparaison des voisinages

Une approche simple pour mesurer une similarité entre sommets consiste à comparer leurs voisinages. Deux sommets sont similaires s'ils ont les mêmes voisins. Nous parlons pour ce type d'approche de mesures locales, car seule une connaissance du voisinage (donc locale) de chacun des sommets concernés est utilisée. Le tableau 2.2 résume les dix similarités locales que nous présentons dans cette sous-section.

Réflexivité. Notons que la quasi-totalité de ces mesures fait intervenir l'intersection des voisinages ($\Gamma(u) \cap \Gamma(v)$) et sont donc fortement dépendantes de la réflexivité ou non du graphe. En effet sur un graphe non-réflexif les sommets ne sont pas voisins d'eux mêmes ($u \notin \Gamma(u)$) et donc même si u et v sont adjacents :

$$\{u, v\} \not\subset \Gamma(u) \cap \Gamma(v) \tag{2.27}$$

Alors que sur un graphe réflexif, si u et v sont adjacents on a :

$$\{u, v\} \subset \Gamma(u) \cap \Gamma(v) \tag{2.28}$$

Souvent, sur des graphes réels la réflexivité ne code pas d'information primordiale. Il est possible de forcer le graphe comme réflexif (ou non-réflexif) sans perdre d'information. Dans la suite, les graphes sont considérés par défaut comme non-réflexifs. De plus, pour chacune de ces mesures, nous introduisons une variante forçant les graphes comme réflexifs. Pour une similarité notée « SIM », on note « SIM-I » cette variante réflexive (« I » signifiant *loops*).

Complexité temporelle. Toutes les mesures décrites ci-dessous ont une complexité temporelle moyenne de $O(d)$, où d est le degré moyen. Et donc le calcul de la matrice de similarité complète est effectué en $O(n^2 d)$. Seule la mesure d'attachement préférentiel (AP) a une complexité plus faible, à supposer que l'on connaisse le degré d'un sommet en $O(1)$ le calcul de la similarité est lui aussi en temps constant (pour une valeur, donc en $O(n^2)$ pour toute la matrice). Nous verrons que ces méthodes locales sont les plus efficaces en termes de complexité.

Généralisation. Plusieurs des mesures locales que nous définissons ci-dessous peuvent se généraliser avec l'écriture suivante :

$$sim(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{avg(|\Gamma(u)|, |\Gamma(v)|)} \tag{2.29}$$

Table 2.1 – Récapitulatif des matrices usuelles associées à un graphe

A	matrice d'adjacence	$[A]_{i,j} = \begin{cases} 1 & \text{si } \{i,j\} \in E, \\ 0 & \text{sinon.} \end{cases}$
D	matrice des degrés	$[D]_{i,j} = \begin{cases} d(i) & \text{si } i = j, \\ 0 & \text{sinon.} \end{cases}$
L	Laplacien	$L = D - A$
P	matrice de transition	$P = D^{-1}A$
$\mathcal{L}$	Laplacien normalisé	$\mathcal{L} = D^{-1/2}LD^{-1/2}$

Table 2.2 – Mesures de similarité locales

CN	Voisins communs	$\lvert\Gamma(u) \cap \Gamma(v)\rvert$
JD	Jaccard	$\frac{\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\lvert\Gamma(u)\cup\Gamma(v)\rvert}$
DS	Dice-Sørensen	$\frac{2\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\lvert\Gamma(u)\rvert+\lvert\Gamma(v)\rvert}$
CO	cosinus	$\frac{\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\sqrt{\lvert\Gamma(u)\rvert.\lvert\Gamma(v)\rvert}}$
AA	Adamic/Adar	$\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{\log\lvert\Gamma(w)\rvert}$
RA	allocation de Ressources	$\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{\lvert\Gamma(w)\rvert}$
HPI	*hub promoted index*	$\frac{\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\min(\lvert\Gamma(u)\rvert,\lvert\Gamma(v)\rvert)}$
HDI	*hub depressed index*	$\frac{\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\max(\lvert\Gamma(u)\rvert,\lvert\Gamma(v)\rvert)}$
LHN1	*Leich Holme Newman*	$\frac{\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\lvert\Gamma(u)\rvert.\lvert\Gamma(v)\rvert}$
PA	attachement préférentiel	$\lvert\Gamma(u)\rvert.\lvert\Gamma(v)\rvert$

où avg est une fonction de moyenne. En effet pour DS la moyenne arithmétique est utilisée, la moyenne géométrique pour CO, le minimum pour HPI, et le maximum pour HDI (minimum et maximum étant les deux moyennes extrêmes). De là, d'autres mesures sont imaginables, en utilisant par exemple la moyenne quadratique ($\sqrt{\frac{1}{2}(x^2+y^2)}$) ou la moyenne harmonique ($\frac{xy}{x+y}$).

2.3.1.1 Voisins communs (CN)

Une mesure de similarité simple consiste simplement à compter le nombre de voisins communs :

$$sim_{CN}(u,v) = |\Gamma(u) \cap \Gamma(v)| \tag{2.30}$$

Notons que sur un graphe non-dirigé, la matrice des similarités donnée par sim_{CN} s'écrit de la manière suivante :

$$S_{CN}(u,v) = A^2 \tag{2.31}$$

En effet le nombre de voisins communs à u et v correspond au nombre de chemins de longueur deux entre u et v.

Dans le cas d'un graphe dirigé on peut distinguer les voisins entrants et sortants. Si l'on considère qu'un lien dirigé $a \to b$ est un lien de citation (a cite b) alors ces mesures s'interprètent en termes de citations communes. la mesure de *bibliographic coupling* (AA^T) indique combien de voisins sont cités à la fois par u et v, alors que la *cocitation coupling* (A^TA) révèle combien de sommets citent u et v.

2.3.1.2 Jaccard (JD)

La classique mesure de *Jaccard* peut aussi être utilisée entre voisinages :

$$Jaccard(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{2.32}$$

Le nombre de voisins communs est normalisé par le nombre de voisins de l'un ou de l'autre des sommets. Cette mesure peut être interprétée comme la probabilité qu'un voisin de l'un ou l'autre soit voisin des deux sommets.

2.3.1.3 Index de Dice-Sørensen (DS)

La mesure de Dice [1945] ou de Sørensen [1948] [7] est une autre mesure classique de similarité entre vecteurs binaires. Elle peut s'appliquer entre les voisinages de deux sommets :

$$Dice(u,v) = \frac{2|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u)| + |\Gamma(v)|} \tag{2.33}$$

7. Cette mesure a été proposée indépendamment par ces deux botanistes, elle est donc notée *Dice*, *Sørensen* ou parfois *Dice-Sørensen* dans la littérature.

Notons que cette mesure s'obtient en ajoutant $|\Gamma(u) \cap \Gamma(v)|$ au numérateur et au dénominateur de la mesure de *Jaccard*.

2.3.1.4 Cosinus (CO)

Le cosinus est une mesure de similarité entre vecteurs maintenant classique. On l'appelle parfois *Salton index*, ou *Salton's cosine* car cette mesure a été introduite, dans le champ de la recherche d'information, comme mesure de similarité entre vecteurs par Salton et al. [1975a]. Cette mesure est définie en général pour des vecteurs réels, mais s'adapte facilement pour des vecteurs binaires. Le cosinus entre les vecteurs d'adjacence de deux sommets peut s'écrire de la manière suivante :

$$cosine(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{|\Gamma(u)|.|\Gamma(v)|}} \tag{2.34}$$

2.3.1.5 Adamic/Adar (AA)

La mesure proposée par Adamic et Adar [2003] prend une forme un peu différente des mesures proposées jusque là :

$$sim_{AA}(u, v) = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(z)|} \tag{2.35}$$

Mais elle peut aussi être comprise comme une pondération du nombre de voisins communs. L'idée étant que plus un voisin commun est « grand » moins il contribue à la similarité. Deux sommets vont être d'autant plus proches que leurs voisins communs sont de faibles degrés. Notons que cette mesure a été définie pour mesurer la similarité entre pages web.

2.3.1.6 Index d'allocation de ressources (RA)

Une mesure similaire a été introduite par Zhou et al. [2009] :

$$sim_{RA}(u, v) = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(z)|} \tag{2.36}$$

C'est une variante de Adamic/Adar sans l'amortissement du degré par le logarithme. Les voisins communs de forts degrés sont donc plus fortement pénalisés.

Notons que cette mesure est équivalente à une marche aléatoire de temps 2 multipliée par le degré de départ (notée PRX2_deg par la suite, voir sous-section 2.3.3.3).

2.3.1.7 *Hub Promoted Index* (HPI)

Les deux mesures HPI et HDI ont été introduites par [Ravasz et al., 2002] pour mesurer la similarité entre sommets d'un réseau métabolique.

$$sim_{HPI}(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\min(|\Gamma(u)|, |\Gamma(v)|)} \tag{2.37}$$

Cela correspond donc à une normalisation du nombre de voisins communs par le nombre de voisins du « plus petit » des sommets. Les *hub* ont donc tendance à avoir des scores élevés, en particulier avec des petits sommets.

2.3.1.8 *Hub Depressed Index* (HDI)

$$sim_{HDI}(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\max(|\Gamma(u)|, |\Gamma(v)|)} \tag{2.38}$$

C'est la mesure inverse de la précédente. Cette fois, le plus « gros » des sommets est utilisé pour la normalisation. Les *hubs* ont donc tendance à avoir des scores plus faibles.

2.3.1.9 *Leicht Holme Newman Index* (LHN1)

La mesure LHN1 [Leicht et al., 2006] propose une autre normalisation du nombre de voisins communs :

$$sim_{LHN1}(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u)|.|\Gamma(v)|} \tag{2.39}$$

L'idée est de comparer le nombre de voisins communs par rapport au nombre « attendu ». En effet, à une constante multiplicative près, le produit $|\Gamma(u)|.|\Gamma(v)|$ correspond au nombre espéré de voisins communs à u et à v, sur un graphe aléatoire dans lequel chaque sommet garde le même degré que sur G. Cette similarité est donc normalisée par rapport au même *null model* que la confluence (voir section 2.2.2).

2.3.1.10 Attachement préférentiel (PA)

L'idée de l'attachement préférentiel est que la probabilité qu'un lien se crée est proportionnelle aux degrés des extrémités. Les liens se forment préférentiellement sur les plus forts sommets. Cette idée est surtout utilisée dans le cadre de la modélisation de graphe mais il en découle une mesure de similarité triviale :

$$sim_{PA}(u,v) = |\Gamma(u)|.|\Gamma(v)| \tag{2.40}$$

Cette similarité n'utilise que le degré de chacun des sommets, elle sera donc peu informative (par contre sa complexité est très faible). Aussi cette mesure simple va nous permettre, par comparaison, de mieux comprendre le comportement d'autres similarités.

Table 2.3 – Mesures de similarité globales

GEO	plus court chemin (dist. géodésique)	$\frac{1}{1+l_{u,v}}$
SRK	SimRank	cf. Eq. (2.43)
BLD	mesure de Blondel et al.	cf. Eq. (2.45)
KTZ	index de Katz	$(I-\alpha A)^{-1}-I$
LP	*local path index*	$A^2+\alpha A^3$
EDK	*exponential diffusion kernel*	$\exp(\alpha A)$
LEDK	*Laplacian exp. diffusion kernel*	$\exp(-\alpha L)$
NLEDK	*normalized Laplacian exp. diffusion kernel*	$\exp(-\alpha \mathcal{L})$
CTRW	*continue time random walk*	$\exp(-\alpha(P-I))$
ACT	*average commute time*	cf. Eq. (2.58)
CTK	*commute time kernel*	L^+
RCTK	*regularized compute time kernel*	$(D-\alpha A)^{-1}$
RLK	*regularized Laplacian kernel*	$(I+\alpha L)^{-1}$
PPR	PageRank personnalisé	$(1-d)(I-dP)^{-1}$

2.3.2 Mesures globales : noyaux, chemins et marches aléatoires

Nous présentons ici des mesures dites « globales », c'est-à-dire basées plus seulement sur les voisinages directs des sommets mais sur le graphe complet. L'avantage évident de ces mesures globales est leur capacité à donner une similarité non systématiquement nulle à une paire de sommets n'ayant aucun voisin en commun. Ces paires, non adjacentes et non liées par un chemin de longueur 2, ne sont en effet pas forcément distantes de la même manière. Sur un graphe creux, elles peuvent être en nombre important, il est alors intéressant de pouvoir les discriminer.

On peut penser que ces mesures amélioreront aussi la qualité de la similarité entre sommets ayant des voisins en commun. Par exemple une paire de sommets reliés par 5 chemins de longueur 2 mais aucun de longueur 3 sera certainement, avec une mesure locale, plus proche qu'une seconde paire reliée par seulement 4 chemins de longueur 2 et 10 de longueur 3. Cela peut s'inverser avec une mesure globale, puisque les chemins de longueur 3 seront pris en compte.

Le tableau 2.3 résume les mesures présentées dans cette sous-section.

Noyaux entre les sommets. Beaucoup de ces similarités globales définissent des noyaux (*kernel*) entre les sommets du graphe [Fouss et al., 2006, 2012; Kunegis et Lommatzsch, 2009]. Un noyau $k : V \times V \to \mathbb{R}$ est une fonction telle que, pour chaque couple de sommets u, v on ait :

$$k(u, v) = \boldsymbol{u}^T.\boldsymbol{v} \tag{2.41}$$

avec $\boldsymbol{u}$ et $\boldsymbol{v}$ des représentations vectorielles implicites de u et v. La matrice d'un noyau est la matrice K telle que $[K]_{u,v} = k(u, v)$. Notons qu'une matrice de produits scalaires, telle que K, est parfois appelée matrice de Gram.

Les noyaux sont particulièrement utilisés en apprentissage automatique [Mohri et al., 2012, chap. 5]. Ils permettent en effet d'appliquer tout algorithme ne dépendant que du produit scalaire (SVM, kernel PCA, etc.) sur les données transformées de manière non-linéaire, et cela sans avoir à expliciter cette représentation non-linéaire des données.

2.3.2.1 Distance géodésique (GEO)

La mesure globale la plus classique sur un graphe est la distance géodésique. C'est-à-dire la longueur du plus court chemin existant entre deux sommets (voir section 1.2.7.2). Cette mesure n'est pas une similarité mais une distance, pour avoir des valeurs comparables avec les autres mesures nous prenons donc son inverse :

$$sim_{GEO}(u, v) = \frac{1}{1 + l_{u,v}} \tag{2.42}$$

Où $l_{u,v}$ est la longueur d'un plus court chemin entre u et v. Cette formule permet d'avoir une valeur de similarité comprise entre 0 et 1. De plus sim_{GEO} vaut 0.5 pour une paire de sommets adjacents, et 1 pour la similarité d'un sommet avec lui même. Pour une paire de sommets déconnectés [8], on peut considérer que $l_{u,v}$ est infini alors sim_{GEO} vaut 0.

Notons que la principale faiblesse de cette mesure est qu'elle est peu discriminante, en particulier sur les graphes réels. En effet la distance géodésique ne prend pas en compte le nombre de chemins (courts) existant entre deux sommets. Cela ne pose pas de problème sur un graphe artificiel quand ce nombre de chemins est à peu près constant. Mais sur les graphes réels, nous avons vu que le phénomène de « petit monde » (section 1.3.1) fait qu'il existe toujours un chemin court entre deux sommets. Ainsi énormément de paires de sommets ont une distance géodésique identique, alors qu'il est possible de les discriminer plus finement en considérant le nombre de chemins existants.

8. Notons que nous nous limitons dans la suite de ce chapitre à des graphes connexes.

2.3.2.2 SimRank (SRK)

La mesure *SimRank* [Jeh et Widom, 2002] se définit de manière récursive : la similarité d'un sommet avec lui même vaut 1, et celle de deux sommets distincts est la moyenne des similarités de leurs voisins :

$$simrank(u,v) = \begin{cases} 1 & \text{si} \quad u = v, \\ \frac{\gamma}{|\Gamma(u)||\Gamma(v)|} \sum_{a\in\Gamma(u)} \sum_{b\in\Gamma(v)} simrank(a,b) & \text{sinon.} \end{cases} \tag{2.43}$$

avec $\gamma \in [0,1]$. Il est possible de calculer itérativement *SimRank*. Si l'on note S_k la matrice des similarités après k itérations, et P la matrice de transition, on a :

$$\begin{aligned} S_0 &= I \\ S'_{k+1} &= \gamma P S_k P^T \\ \forall i, \forall j, \quad [S_{k+1}]_{ij} &= \begin{cases} 1 & \text{si} \quad i = j, \\ \left[S'_{k+1}\right]_{ij} & \text{sinon.} \end{cases} \end{aligned}$$

et donc :

$$S_{SRK} = \lim_{k\to\infty} S_k \tag{2.44}$$

La démonstration de la convergence est faite par [Jeh et Widom, 2002].

Notons que cette similarité peut être interprétée en termes de marches aléatoires : $simrank(u,v)$ est l'espérance de γ^l, où l est la variable aléatoire donnant le temps auquel deux marcheurs aléatoires partant de u et de v se retrouvent pour la première fois sur un même sommet.

2.3.2.3 Mesure de Blondel et al. (BLD)

La mesure de similarité de Blondel et al. [2004] permet de mesurer la similarité entre les sommets de deux graphes. Nous l'utilisons ici entre un graphe et lui-même. Cette mesure se calcule alors itérativement de la manière suivante [9] :

$$S_{k+1} = \frac{AS_kA^T + A^TS_kA}{||AS_kA^T + A^TS_kA||_F}, \quad S_0 = \mathbf{1} \tag{2.45}$$

On a :

$$S_{BLD} = \lim_{k\to\infty} S_k \tag{2.46}$$

Cette mesure, dans son écriture matricielle, est semblable à *SimRank*. Toutefois c'est la matrice d'adjacence A qui est utilisée ici et non la matrice de transition P. De plus *SimRank* force la similarité d'un sommet avec lui même à 1, ce qui n'est pas le cas ici.

9. $||.||_F$ est la norme de Frobenius : $||M||_F = \sqrt{\sum_i \sum_j m_{ij}^2}$.

2.3.2.4 Similarité de Katz (KTZ)

La similarité dite « de Katz », ou index de Katz, a été introduite en 1953 [Katz, 1953] afin de mesurer « l'importance » (i.e. la centralité) des sommets dans un graphe social et non directement pour mesurer une similarité entre sommets. La mesure de centralité définie par Katz est la somme de la similarité du sommet « cible » avec tous les autres. Nous nous intéressons ici uniquement à la mesure de similarité ainsi indirectement définie.

Cette mesure se base sur l'ensemble des chemins existant entre deux sommets, elle est définie ainsi :

$$ktz(u,v) = \sum_{k=1}^{\infty} \alpha^k . |paths^{<k>}(u,v)| \tag{2.47}$$

Avec $paths^{<k>}(u,v)$ l'ensemble des chemins de longueur k entre u et v et α un scalaire ($0 < \alpha < 1$). $ktz(u,v)$ est donc la somme pondérée du nombre de chemins existant entre u et v. Le nombre de chemins de longueur k étant pondéré par α^k.

Si S_{KTZ} est la matrice des similarités de Katz ($[S_{KTZ}]_{u,v} = ktz(u,v)$), et A est la matrice d'adjacence alors on a :

$$S_{KTZ} = \sum_{k=1}^{\infty} \alpha^k A^k \tag{2.48}$$

On voit que K_{katz} est, presque, la somme S d'une suite géométrique de raison αA (presque car la somme commence à 1, non à 0). On a donc :

$$S_{katz} = (I - \alpha A)^{-1} - I \tag{2.49}$$

à condition que α soit plus petit que λ^{-1}, λ étant la plus grande des valeurs propres de A (pour que $I - \alpha A$ soit inversible).

Notons que cette mesure est quasiment équivalente au noyau de *Neumann* (*Neumann kernel* ou *von Neumann diffusion kernel*). En effet ce noyau se définit de la manière suivante :

$$S_{NK} = \sum_{k=0}^{\infty} \alpha^k A^k = (I - \alpha A)^{-1} = S_{KTZ} - I \tag{2.50}$$

Pour plus de détails sur le noyau de *Neumann*, nous renvoyons à [Kandola et al., 2003] et à [Ito et al., 2005].

2.3.2.5 *Local path index* (LP)

Le *local path index* a été introduit par [Zhou et al., 2009]. Cette mesure se définit ainsi :

$$S_{LP} = A^2 + \beta A^3 \tag{2.51}$$

C'est une approximation de la similarité Katz ne considérant que les chemins de longueur 2 et 3. Cette approche a l'avantage d'être semi-globale [10] tout en ayant une complexité raisonnable. En effet si le graphe est peu dense, A est une matrice creuse et donc S_{LP} se calcule en $O(nm)$ (m le nombre d'arêtes, et n de sommets). En effet la multiplication de A par A se fait en $O(nm)$, celle de A^2 par A se fait aussi en $O(nm)$, et la somme de A^2 et de A^3 se fait au pire en $O(n^2)$.

2.3.2.6 *Exponential diffusion kernel* (EDK)

Le noyau de diffusion exponentiel (ou *exponential diffusion kernel*) a été introduit par [Kondor et Lafferty, 2002]. Il est défini de la manière suivante :

$$S_{EDK} = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!} A^k = \exp(\alpha A) \tag{2.52}$$

Cette mesure est donc semblable à celle de *Katz* (ou noyau de *Neumann*), à la différence que les chemins de longueurs k sont ici pondérés par $\frac{\alpha^k}{k!}$. L'atténuation de l'importance d'un chemin en fonction de sa longueur est donc plus forte.

2.3.2.7 *Laplacian exponential diffusion kernel* (LEDKα)

Le *Laplacian exponential diffusion kernel* [Kondor et Lafferty, 2002] est une alternative au *Exponential diffusion kernel*. La matrice d'adjacence est remplacée par l'opposé du Laplacien du graphe :

$$S_{LEDK\alpha} = \exp(-\alpha L) \tag{2.53}$$

Les valeurs de $S_{LEDK\alpha}$ peuvent être interprétées en termes de diffusion sur le graphe. On note $x_u(t)$ la valeur en u à l'instant t. On imagine alors le phénomène de diffusion suivant sur le graphe : pendant un intervalle de temps δt une quantité $[A]_{u,v}\, x_u(t)\delta t$ est transférée de u vers v. C'est-à-dire, si u et v sont voisins, une quantité proportionnelle à $x_u(t)$ et à δt est passée de u vers v. On peut alors montrer (voir [Fouss et al., 2012, section 4.2]) que $[S_{LEDK\alpha}]_{u,v}$ est la quantité présente en v, après un temps α, étant donnée une distribution initiale intégralement en u.

2.3.2.8 *Normalized Laplacian exponential diffusion kernel* (NLEDK)

Le *normalized Laplacian exponential diffusion kernel* est une variante du noyau LEDK. Simplement le Laplacien normalisé $\mathcal{L}$ est utilisé en lieu et place du Laplacien :

$$S_{NLEDK} = \exp(-\alpha \mathcal{L}) \tag{2.54}$$

10. Cette mesure est bien semi-globale, nous avons préféré l'introduire ici plutôt que dans la section suivante (c'est-à-dire la section 2.3.3, qui regroupe les méthodes semi-globales) car cette méthode n'est pas basée sur les marches aléatoires en temps courts, et est une approximation de KTZ.

2.3.2.9 Marche aléatoire en temps continu (CTRW)

Les marches aléatoires en temps continu décrivent le phénomène de diffusion suivant : pour chaque paire de sommets i, j, pendant un intervalle de temps δt une quantité $[P]_{i,j}\, x_i(t)\delta t$ est transférée de i vers j. Le résultat d'un tel phénomène de diffusion pour un temps α est donné par :

$$S_{CTRW} = \exp(-\alpha(P - I)) \tag{2.55}$$

Pour une démonstration de cette écriture, voir par exemple [Pons, 2007, section 2.5].

Ce phénomène de diffusion est semblable à celui à l'œuvre avec le *Laplacian exponential diffusion kernel*. La différence avec LEDK est qu'ici la quantité transférée de u à v dépend du degré de u. Pour comprendre cette différence, prenons deux sommets adjacents a et b. Imaginons qu'à un temps t donné la valeur en a et en b est la même. Sur un intervalle de temps δt, dans le premier cas (LEDK), la même quantité va transiter de a vers b et de b vers a. Alors que dans le second cas (CTRW), la quantité qui transite de a vers b est proportionnelle à $1/d(a)$ alors que la quantité qui transite de b vers a est proportionnelle à $1/d(b)$. Un « gros » sommet va donc « recevoir » plus qu'il ne « donne ». A l'équilibre ($t \to \infty$) la quantité présente sur les gros sommets sera plus importante que sur les petits sommets (on montre en réalité que l'état stable est le même que pour une marche aléatoire classique en temps discret). Ce qui n'est pas le cas avec le premier mécanisme, où à l'équilibre tous les sommets ont la même valeur.

Notons que LEDK peut aussi être compris en termes de marches aléatoires. Effectivement, cela correspond à une marche aléatoire où un marcheur sur un sommet a, va passer sur un sommet voisin b avec une probabilité $1/d_{max}$ et rester sur le même sommet avec une probabilité $d_{max} - d(a)$, où d_{max} est le plus fort des degrés des sommets du graphe. Ce mécanisme est parfois nommé *lazy random walk* [Kondor et Lafferty, 2002, section 3.3].

2.3.2.10 *Average commute time* (ACT)

La distance dite « *average commute time* » entre u et v correspond au temps moyen que met un marcheur aléatoire pour aller de u à v puis de v à u. On parle aussi de *resistance distance* car cette mesure est proportionnelle à la résistance électrique mesurable entre les sommets u et v, si chaque arête du graphe est une résistance (de valeur constante pour un graphe non-pondéré) [Chandra et al., 1996].

On peut montrer [Fouss et al., 2007] que cette distance est calculable à partir de L^+ le pseudo-inverse du Laplacien du graphe :

$$d_{ACT}(u, v) = \left[L^+\right]_{u,u} + \left[L^+\right]_{v,v} - 2\left[L^+\right]_{u,v} \tag{2.56}$$

L^+ pouvant être calculé de la manière suivante :

$$L^+ = (L - \mathbf{1}/n)^{-1} + \mathbf{1}/n \tag{2.57}$$

où L est le Laplacien du graphe ($L = D - A$).

Aussi comme d_{ACT} est une distance, et afin d'avoir une mesure comparable avec les autres, nous prenons son inverse :

$$sim_{ACT}(u, v) = \frac{1}{1 + d_{ACT}(u, v)} \tag{2.58}$$

Notons que von Luxburg et al. [2010] ont montré que cette distance a peu d'intérêt sur des graphes de grande taille. Sous certaines conditions, la *commute time distance* entre deux sommets u et v peut être approximée par : $\frac{1}{d(u)} + \frac{1}{d(v)}$. Nous allons voir que cela se confirme dans notre étude expérimentale (voir section 2.4) où ACT apparaît fortement corrélée à PA (pour rappel $sim_{PA}(u, v) = d(u).d(v)$).

2.3.2.11 *Commute time kernel* (CTK)

Il est aussi possible d'utiliser directement la matrice L^+ comme mesure de similarité. On nomme cette similarité *commute time kernel*. Il est possible de la calculer de la manière suivante [Fouss et al., 2007] :

$$S_{CTK} = L^+ = (L - \mathbf{1}/n)^{-1} + \mathbf{1}/n \tag{2.59}$$

On peut montrer que $[S_{CTK}]_{u,v}$ est le produit scalaire entre des représentations de u et v dans un espace où chaque paire de sommets est séparée d'une distance correspondant à la valeur de la distance « *average commute time* » (d_{ACT}), voir [Fouss et al., 2007].

2.3.2.12 *Regularized Laplacian kernel* (RLK)

Un alternative à l'utilisation du pseudo-inverse du Laplacien est d'utiliser un opérateur de régularisation sur le Laplacien :

$$S_{RLK} = \sum_{k=0}^{\infty} \alpha^k (-L)^k = (I + \alpha L)^{-1} \tag{2.60}$$

avec $\alpha > 0$. On parle de *regularized Laplacian Kernel* [Ito et al., 2005].

Cette mesure a la même forme que le noyau de Neumann (ou mesure de Katz), mais ici la matrice d'adjacence est remplacée par le Laplacien. Notons aussi que pour un α suffisamment grand, cette mesure est semblable au *commute time kernel* (CTK) à un facteur multiplicatif et à une constante près [Fouss et al., 2012, section 4.5].

On réfère parfois à cette mesure par le terme *matrix forest index* [Chebotarev et Shamis, 1997]. En effet on peut l'interpréter en termes de forêts recouvrantes. Une *forêt recouvrante* (*spanning forest*) de G est un sous-graphe acyclique de G, qui a les mêmes sommets que G, et comportant un sommet *racine* dans chacune de ses composantes connexes. C'est une union d'arbres recouvrant G. Il a été montré par

[Chebotarev et Shamis, 1997] que $(I - L)^{-1}$ est bien défini, et que $[(I - L)^{-1}]_{i,j}$ est le ratio entre le nombre de forêts recouvrantes où i et j sont dans le même arbre de racine i et le nombre total de forêts recouvrantes de G. $[(I - L)^{-1}]_{i,j}$ est donc la probabilité que i et j soient dans le même arbre dont i est racine, parmi l'ensemble des forêts recouvrantes de G. Lorsque le graphe est pondéré, cette interprétation doit tenir compte des poids, nous renvoyons pour plus de détails à [Chebotarev et Shamis, 1997]. Aussi cette interprétation se généralise avec $(I - \alpha L)^{-1}$, les forêts dont les arbres ont k arêtes sont alors pondérées avec un poids de α^k [Chebotarev et Shamis, 1998, théorème 2]. Ainsi la mesure est renforcée si i et j sont dans la même forêt de « petits arbres ».

2.3.2.13 *Regularized commute-time kernel* (RCTK)

Une autre variante au *commute time kernel* (CTK) ou au *regularized Laplacian kernel* (RLK) est le *regularized commute-time kernel* :

$$S_{RCTK} = (D - \alpha A)^{-1} \tag{2.61}$$

pour $0 < \alpha < 1$. Ce noyau a été proposé par Fouss et al. [2006]. Nous allons voir que cette mesure correspond à la mesure de PageRank personnalisé (PPR) divisée par le degré du sommet d'arrivée.

2.3.2.14 PageRank personnalisé (PPR et PPR-avg)

Le *PageRank personnalisé* (*personalized PageRank*) est une adaptation de l'algorithme classique *PageRank* [Brin et Page, 1998]. C'est le résultat d'une marche aléatoire où à chaque pas le marcheur passe sur l'un des sommets voisins avec une probabilité d et retourne sur un sommet initial avec une probabilité $1 - d$.

Le *PageRank personnalisé* entre u et v est donc la probabilité qu'un tel marcheur (revenant « un peu » à chaque pas sur u) soit sur v après un nombre infini de pas. Pour un sommet de départ u, si $p_u(t)$ est le vecteur de distribution de probabilité du marcheur au temps t on a :

$$p_u(t+1) = (1 - d)\delta_u + dp_u(t)P \tag{2.62}$$

où δ_u est la certitude d'être en u, c'est-à-dire un vecteur n'ayant qu'une valeur non nulle égale à 1 sur la $u^{\text{ème}}$ composante. La valeur de ce vecteur converge quand t tend vers l'infini, et donne une mesure de similarité entre u et les sommets du graphe.

Il est possible d'exprimer matriciellement cette mesure. Soit S_{PPR} une matrice carrée telle que sa $u^{\text{ème}}$ *ligne* vaille $\lim_{t\to\infty} p_u(t)$, on a :

$$S_{PPR} = (1 - d)(I - dP)^{-1} \tag{2.63}$$

$[S_{PPR}]_{u,v}$ est donc la probabilité d'être en v après un nombre infini de pas telss qu'à chaque pas le marcheur retourne en u avec une probabilité $1-d$.

Cette mesure peut aussi s'écrire comme une somme infinie :

$$S_{PPR} = (1-d)\sum_{k=0}^{\infty} d^k P^k \tag{2.64}$$

Cette écriture est intéressante car elle permet de comparer PPR aux mesures semi-globales basées sur des marches aléatoires (comme défini en section 2.2 et dans la sous-section 2.3.3). En effet on observe que S_{PPR} est la somme, sur le temps t, du résultat d'une marche en temps t pondéré par un facteur $(1-d)d^t$. Chaque marche aléatoire de longueur i ayant un poids $(1-d)d^t$, la longueur moyenne des marches utilisées vaut donc $\sum_{k=0}^{\infty}(1-d)d^k k = \frac{d}{1-d}$. Notons que la valeur de $d = 0.8$ souvent utilisée correspond donc à des marches de longueur moyenne $t = 4$.

Il faut noter que S_{PPR} n'est pas symétrique. Comme nous l'avons déjà remarqué cela n'est pas forcément un défaut, toutefois afin d'avoir une comparaison cohérente nous utiliserons la mesure moyenne suivante :

$$S_{PPR\_avg} = \frac{1}{2}(S_{PPR} + S_{PPR}^T) \tag{2.65}$$

On peut remarquer le lien entre le *PageRank personnalisé* et la mesure *regularized compute time kernel* (RCTK). En rappelant que $P = D^{-1}$ on a :

$$\begin{aligned} S_{PPR} &= (1-d)(D^{-1}(D-dA))^{-1} && (2.66)\\ &= (1-d)(D-dA)^{-1}D && (2.67)\\ &= (1-d)S_{RCTK}D && (2.68) \end{aligned}$$

On a donc $(1-d)S_{RCTK} = S_{PPR}D^{-1}$. La mesure $[S_{RCTK}]_{u,v}$ correspond donc, à une constante multiplicative près, au *PageRank personnalisé* de u vers v divisé par le degré de v.

2.3.3 Mesures « semi-globales »

On appelle mesures « semi-globales » les similarités qui utilisent un voisinage plus large que les mesures locales sans forcément faire intervenir tout le graphe. Ces approches se basent en général, comme la *confluence*, sur des marches aléatoires en temps courts. Cette sous-section présente d'autres méthodes que la confluence en s'appuyant sur les notations introduites en sous-section 2.2.1. Le tableau 2.4 résume ces méthodes.

Notons qu'elles peuvent être classées en deux familles : soit la similarité entre u et v est seulement liée à la probabilité d'aller de u vers v ou de v vers u, soit elle prend en compte la probabilité d'atteindre depuis u (et depuis v) chacun des

sommets du graphe. Soit u et v sont similaires car il est probable d'aller de l'un à l'autre en peu de pas, soit u et v sont similaires car on atteint les mêmes sommets depuis l'un ou l'autre. Les méthodes PRXt, PRXt-avg, PRXt-deg, PRXt-dege, SRWt ainsi que la confluence (CFLt) relèvent de la première famille, les autres de la seconde.

Pour les méthodes de la seconde famille, on considère que u et v sont similaires s'il est possible d'atteindre les mêmes sommets avec la même probabilité depuis u ou depuis v. Pour cela chaque sommet u est représenté par la distribution de probabilité d'un marcheur après t pas depuis u. C'est-à-dire la ligne u de la matrice P^t. La similarité entre u et v est alors donnée par une mesure entre les deux vecteurs lignes correspondants extraits de P^t. Toutes les méthodes de similarité entre vecteurs, et en particulier entre distributions de probabilité, sont utilisables ici. Pour un état de l'art complet des méthodes envisageables nous renvoyons à [Cha, 2007]. Nous présentons ici seulement les méthodes que nous avons évaluées.

Table 2.4 – Mesures de similarité semi-globales utilisant des marches aléatoires à temps courts

PRXt	mesure directe	P^t
PRXt_avg	moyenne	$\frac{1}{2}(P^t + P^{tT})$
PRXt_deg	$\times$ degré de départ	$D.P^t$
PRXt_dege	/ degré d'arrivée	$P^t.D^{-1}$
SRWt	*Superposed Random Walk*	$\sum_{l=1}^{t} D.P^l$
PRXt_dot	produit scalaire	$P^t.P^{tT}$
PRXt_cos	cosinus	cf. Eq. (2.81)
MDKt	*Markov diffusion kernel*	$Z_t Z_t^T$ avec $Z_t = \frac{1}{t}\sum_{l=0}^{t} P^l$
CFLt	confluence	cf. Eq. (2.20)

2.3.3.1 Mesure directe (PRXt)

La manière la plus simple pour construire une mesure de similarité à partir de marches aléatoires en temps courts est d'utiliser directement la probabilité d'aller de u à v :

$$sim_{PRXt}(u, v) = \left[P^t\right]_{u,v} \tag{2.69}$$

Il faut bien remarquer que cette mesure n'est pas symétrique. C'est pourquoi nous ne l'inclurons pas dans les évaluations.

2.3.3.2 Moyenne (PRXt_avg)

Une méthode simple pour rendre la mesure symétrique est d'utiliser la moyenne :

$$sim_{PRXt\_avg}(u,v) = \frac{1}{2}(\left[P^t\right]_{u,v} + \left[P^t\right]_{v,u}) \tag{2.70}$$

ce qui donne en écriture matricielle :

$$S_{PRXt\_avg} = \frac{1}{2}(P^t + P^{tT}) \tag{2.71}$$

Notons que la même symétrisation est utilisée avec le *average commute time* (ACT).

2.3.3.3 Multiplié par le degré de départ (PRXt_deg)

Nous avons remarqué que la probabilité d'aller de a à b multipliée par le degré de a est égale à la probabilité d'aller de b à a multipliée par le degré de b (voir proposition 1, Eq. (2.5)). Il suffit donc de multiplier $[P^t]_{u,v}$ par le degré de u pour avoir une mesure symétrique :

$$sim_{PRXt\_deg}(u,v) = |\Gamma(u)|.\left[P^t\right]_{u,v} \tag{2.72}$$

et donc matriciellement :

$$S_{PRXt\_deg} = D.P^t \tag{2.73}$$

Notons que Liu et Lü [2010] proposent la mesure suivante :

$$sim_{LRWt}(u,v) = \frac{|\Gamma(u)|}{2|E|}\left[P^t\right]_{u,v} + \frac{|\Gamma(v)|}{2|E|}\left[P^t\right]_{v,u} \tag{2.74}$$

Or étant donnée la remarque faite précédemment, cette mesure est équivalente à sim_{PRX_deg} (à une division par $|E|$ près). Bien entendu cela n'est plus vrai sur un graphe dirigé.

2.3.3.4 Divisé par le degré d'arrivée (PRXt_dege)

Une autre symétrisation similaire est possible en divisant cette fois la mesure par le degré d'arrivée :

$$sim_{PRXt\_dege}(u,v) = \frac{[P^t]_{u,v}}{|\Gamma(v)|} \tag{2.75}$$

et donc matriciellement :

$$S_{PRXt\_dege} = P^t.D^{-1} \tag{2.76}$$

Nous adoptons la notation discutable « PRXt_dege », le « e » signifiant *end*. Notons que cette transformation est la même que celle qui permet de passer du *personalized PageRank* (PPR) au *regularized compute time kernel* (RCTK).

2.3.3.5 Somme des temps courts (SRWt)

Liu et Lü [2010] proposent d'utiliser la somme des marches aléatoires de longueur variant entre 1 et t, multipliée par le degré de départ :

$$sim_{SRWt}(u,v) = \sum_{l=1}^{t} sim_{PRXl\_deg}(u,v) = |\Gamma(u)| \sum_{l=1}^{t} \left[P^l\right]_{u,v} \tag{2.77}$$

SRW signifie *Superposed Random Walk*. Matriciellement cela donne :

$$S_{SRWt}(u,v) = \sum_{l=1}^{t} D.P^l \tag{2.78}$$

2.3.3.6 Produit scalaire (PRXt_dot)

Les méthodes vues jusqu'ici utilisent seulement $[P^t]_{u,v}$ et $[P^t]_{v,u}$. Nous introduisons maintenant plusieurs méthodes adoptant un point de vue différent. Les sommets sont comparés en utilisant les lignes correspondantes dans P^t, on note $[P^t]_{u,*}$ la ligne correspondant au sommet u.

Une méthode simple pour comparer deux lignes de P^t consiste à effectuer un produit scalaire :

$$sim_{PRXt\_dot}(u,v) = \left[P^t\right]_{u,*} . \left[P^t\right]_{v,*}^{T} \tag{2.79}$$

$[P^t]_{i,*}$ étant la ligne i de la matrice P^t. Ce qui donne en écriture matricielle :

$$S_{PRXt\_dot} = P^t . P^{tT} \tag{2.80}$$

2.3.3.7 Cosinus (PRXt_cos)

Une autre comparaison classique repose sur le cosinus :

$$sim_{PRXt\_cos}(u,v) = \frac{[P^t]_{u,*} . [P^t]_{v,*}^{T}}{||\,[P^t]_{u,*}\,||.||\,[P^t]_{v,*}\,||} \tag{2.81}$$

où $||.||$ est la norme 2.

2.3.3.8 Markov diffusion kernel (MDK)

Enfin la mesure appelée *Markov diffusion kernel* a été introduite par Fouss et al. [2006, 2012] :

$$S_{MDK} = Z_t Z_t^T \quad \text{avec} \quad Z_t = \frac{1}{t} \sum_{l=0}^{t} P^l \tag{2.82}$$

Ici chaque sommet est donc représenté par la moyenne des résultats des marches aléatoires de longueurs variant de 0 à t. Et la similarité entre deux sommets est calculée par un simple produit scalaire entre ces représentations. MDK est presque le produit scalaire entre les mesures de SRW, à la différence que les marches ne sont pas ici multipliées par le degré de départ.

2.4 Comparaison expérimentale

Nous proposons dans cette section une comparaison expérimentale des mesures de similarité présentées dans les sections précédentes. L'idée est d'observer la corrélation entre les valeurs de ces mesures sur différentes paires de sommets provenant de plusieurs graphes. Nous proposons ainsi plusieurs regroupements hiérarchiques de ces similarités basés sur la corrélation entre les valeurs de ces mesures.

Souvent dans la littérature, les comparaisons entre méthodes consistent en une évaluation sur une tâche (prédiction de liens [Lü et Zhou, 2011; Fouss et al., 2012] ou encore imitation d'une similarité sémantique [Hughes et Ramage, 2007]). Ce type d'évaluation permet de savoir quelles sont les méthodes les plus efficaces pour tel problème sur telles données. Mais cela ne permet pas de comprendre les différences de comportement des méthodes. Par exemple, il est possible que deux mesures aient des bons scores mais ne les obtiennent pas du tout de la même façon. De telles différences ne sont pas visibles avec une évaluation par tâche. Nous proposons ici une évaluation complémentaire, qui permet de mieux cerner les différences (et ressemblances) entre les mesures, sans chercher directement à savoir quelle est la « meilleure » mesure pour telle ou telle tâche.

Certains auteurs ont déjà proposé des évaluations similaires à ce que nous exposons ici, mais pas à notre connaissance concernant les mesures de similarité entre sommets d'un graphe. Notamment Cha [2007] propose une comparaison des mesures de similarité entre distributions de probabilité et plus récemment Choi et al. [2010] proposent une comparaison des mesures de similarité entre vecteurs binaires. Ces deux travaux fournissent un remarquable travail d'état de l'art. Ils proposent en outre un regroupement hiérarchique des différentes similarités basé sur les corrélations entre ces mesures calculées sur un jeu de données aléatoires. Ces comparaisons présentent deux limites. Tout d'abord, on peut douter que le modèle de données aléatoires utilisé incorpore les caractéristiques fréquentes des données réelles. Par exemple, dans [Choi et al., 2010], tous les points semblent posséder à peu près le même nombre de valeurs non-nulles (c'est-à-dire non « fausse », les vecteurs étant binaires). Or sur des données réelles, il est tout à fait possible que certains points aient énormément de composantes non-nulles alors que la plupart des points en ont très peu. Certainement les méthodes ne se comportent pas de la même manière sur des données ainsi déséquilibrées. L'autre limite de ces études est qu'elles ne permettent pas d'interpréter et de comprendre pourquoi les méthodes sont regroupées de cette façon.

Nous essayons dans l'étude proposée ici de dépasser ces deux limitations. D'une part en travaillant sur des données aléatoires mais aussi sur des données réelles. Et d'autre part en observant les valeurs des méthodes sur différentes familles de paires de sommets.

2.4.1 Mise en place expérimentale

Nous présentons ici le protocole expérimental mis en place afin de comparer les différentes mesures présentées dans les sections précédentes.

Les graphes. Nous avons travaillé sur 8 graphes, à savoir :

- 3 graphes aléatoires de type Erdős-Rényi (ER01, ER02 et ER05),
- 2 graphes aléatoires formés de 5 clusters de 50 sommets (CGMpd et CGMtd),
- 3 graphes réels (RW_protein_lcc, RW_power et RW_dsV_lcc).

Les caractéristiques de ces graphes sont données dans le tableau 2.5. RW_power est un graphe modélisant le réseau électrique de l'ouest des États-Unis : les sommets sont les générateurs et les différents postes électriques : transformateurs, nœud d'interconnexion de lignes, etc. Les arêtes représentent les lignes électriques existant entre ces éléments. Ce graphe a été diffusé par Watts et Strogatz [1998]. Le graphe RW_protein_lcc est la plus grande composante connexe d'un réseau d'interaction de protéines [Jeong et al., 2001]. Chaque sommet correspond à une protéine présente dans les *Saccharomyces cerevisiae* (une levure), et une arête relie deux protéines si une interaction directe a été identifié entre elles. Enfin RW_dsV_lcc est la plus grande composante connexe d'un graphe de synonymie des verbes du français (DicoSyn). Nous utiliserons des graphes provenant de cette même ressource au chapitre suivant, en section 3.5.

Les graphes aléatoires comportant des clusters (CGMpd et CGMtd) sont construits grâce au modèle suivant.

***Cluster Graph Model*, CGM.** Nous proposons d'appeler CGM (pour *Cluster Graph Model*) le modèle consistant à construire un graphe à partir de k groupes de r sommets chacun, tel que les arêtes soient aléatoires mais avec différentes probabilités à l'intérieur ou entre les groupes. Une arête entre deux sommets du même groupe existe avec une probabilité μ_{intra}, et avec une probabilité μ_{inter} entre des sommets de groupes différents. Les deux graphes ont été construits avec $k = 5$, $r = 50$. Pour CGMpd on a $\mu_{intra} = 0.15$ et $\mu_{inter} = 0.01$, alors que pour CGMtd $\mu_{intra} = 0.50$ et $\mu_{inter} = 0.02$. CGMtd est donc plus dense, et avec des clusters plus lisibles que CGMpd.

Table 2.5 – Pedigree des graphes utilisés pour la comparaison. n et m sont l'ordre et la taille des graphes, $\langle k \rangle$ le degré moyen, ncc le nombre de composante connexe, « directed » indique si les graphes sont dirigés ou non, $\mathbf{c}$ le coefficient de clustering, $\mathbf{l}_{ccp}$ est la longueur moyenne des plus courts chemins sur cette plus grande composante connexe, et ρ est corrélation des degrés.

	n	m	$\langle k \rangle$	ncc	directed	$\mathbf{c}$	$\mathbf{l}_{ccp}$	ρ
ER01	1000	4942	9.88	1	*False*	0.01	3.27	−0.005
ER02	1000	9977	19.95	1	*False*	0.02	2.64	−0.005
ER05	1000	25029	50.06	1	*False*	0.05	2.03	0.002
CGMpd	250	1176	9.41	1	*False*	0.10	2.86	−0.007
CGMtd	250	3522	28.18	1	*False*	0.37	2.17	0.009
RW_protein_lcc	1458	1948	2.67	1	*False*	0.05	6.81	−0.209
RW_power	4941	6594	2.67	1	*False*	0.10	18.99	0.004
RW_dsV_lcc	8993	51333	11.42	1	*False*	0.14	4.20	0.062

Les mesures de similarité. Nous avons comparé toutes les mesures présentées jusque là excepté celles qui sont non-symétriques. Les noms utilisés pour repérer les méthodes correspondent à ceux indiqués précédemment. Il convient d'indiquer les précisions suivantes :

- CO-I, DS-I, JD-I, etc. correspondent aux méthodes locales calculées en considérant une boucle sur chaque sommet, les autres méthodes locales sont donc calculées sans boucle sur les sommets ;
- (SRK) SRK08 : $\gamma = 0.8$,
- (KTZ) KTZ001 : $\alpha = 0.01$, KTZm09 : $\alpha = 0.9\lambda_1$ avec λ_1 la plus forte des valeurs propres de A,
- (LP) LP05 : $\alpha = 0.5$, LP001 : $\alpha = 0.01$,
- (EDK) EDK01 : $\alpha = 0.1$, EDK10 : $\alpha = 1$,
- (LEDK) LEDK01 : $\alpha = 0.1$, LEDK05 : $\alpha = 0.5$, LEDK10 : $\alpha = 1$,
- (NLEDK) NLEDK05 : $\alpha = 0.5$, NLEDK50 : $\alpha = 5$, NLEDK100 : $\alpha = 10$,
- (CTRW) CTRW05 : $\alpha = 0.5$, CTRW50 : $\alpha = 5$, CTRW100 : $\alpha = 10$,
- (RCTK) RCTK07 : $\alpha = 0.7$, RCTK09 : $\alpha = 0.8$,
- (RLK) RLK08 : $\alpha = 0.8$,
- (PPR-avg) PPR08-avg : $d = 0.8$,

Notons aussi que les méthodes CFL2R, CFL3R, CFL5R correspondent à la confluence calculée, pour les paires adjacentes, en retirant l'arête correspondante du graphe (voir sous-section 2.2.2). Aussi toutes les méthodes semi-globales (hormis MDK et SRW) sont calculées en ajoutant des boucles sur chaque sommet (voir section 2.2.1).

Table 2.6 – Récapitulatif des familles de paires de sommets utilisées pour la comparaison. Nous considérons que les « hubs » sont les 10% des sommets ayant le plus de voisins, les « petits » sont les 90% restants.

graphe	*paire*	
ER	E	adjacents
	NE	non-adjacents
CGM	C-E	adjacents dans le même cluster
	C-NE	non-adjacents dans le même cluster
	NC-E	adjacents dans différents clusters
	NC-NE	non-adjacents dans différents clusters
RW	E-BB	deux « hubs » adjacents
	E-BS	un « hub » et un « petit » adjacents
	E-SS	deux « petits » adjacents
	NE-BB	deux « hubs » non-adjacents
	NE-BS	un « hub » et un « petit » non-adjacents
	NE-SS	deux « petits » non-adjacents

Paires échantillons. Les mesures de similarité sont évaluées sur des échantillons des paires de sommets de chacun de ces graphes. Pour chacun des types de graphe nous avons en effet déterminé différentes familles de paires de sommets. Par exemple sur les graphes type Erdős-Rényi, il y a deux familles : les paires adjacentes et les paires de sommets non-adjacents. Ces différentes familles de paires de sommets sont présentées dans le tableau 2.6. Pour chacun des graphes, 100 paires de sommets de chaque famille ont été sélectionnées aléatoirement. Les méthodes sont comparées sur ces paires. Ce choix de familles de paires de sommets permet d'éviter des biais de représentation. En effet ces familles séparent l'ensemble des paires de sommets d'un graphe en des sous-ensembles de tailles très différentes. Si les paires étaient simplement tirées au hasard, certaines familles seraient sur-représentées et d'autres sous-représentées. Ce découpage en famille va aussi nous permettre de mieux comprendre les résultats, en effet nous pourrons observer le comportement des méthodes sur chaque famille de paires.

Corrélation et clustering hiérarchique. L'évaluation consiste donc à mesurer la corrélation entre les valeurs des mesures de similarité sur ces différentes paires, éventuellement en se limitant à certains graphes ou/et à certaines familles de paires. Pour cela nous avons utilisé le coefficient de corrélation de *Pearson* (voir par exemple [Saporta, 2006, section 6.1.2]), et le *coefficient de Spearman* [Saporta, 2006, section 6.3.1]. Le *coefficient de Pearson* mesure la corrélation linéaire entre deux séries de valeurs, alors que le *coefficient de Spearman* mesure la corrélation des rangs. Deux

méthodes ont un *coefficient de Spearman* fort (c'est-à-dire proche de 1) si leurs valeurs induisent presque le même classement des paires de sommets, sans forcément qu'elles soient corrélées linéairement.

Nous avons construit différents regroupements hiérarchiques des mesures de similarité à partir des coefficients de corrélation : initialement chaque mesure est seule dans un groupe, puis successivement les deux groupes les « plus proches » sont rassemblés. Cela jusqu'à ce que toutes les mesures soient dans un seul groupe. Pour mesurer une distance entre deux mesures s_1 et s_2 nous utilisons la méthode suivante :

$$d(s_1, s_2) = 1 - \max(c(s_1, s_2), 0) \tag{2.83}$$

où $c(s_1, s_2)$ est le coefficient de corrélation entre s_1 et s_2. Notons que les corrélations négatives sont ramenées à zéro. Cela ne change pas fondamentalement les résultats, simplement pour une distance de 1 toutes les mesures seront regroupées. La distance entre deux groupes est évaluée en utilisant la méthode dite de *saut maximal* (ou *complete linkage* en anglais). Cela consiste à prendre comme distance entre deux groupes, la plus grande des distances entre les éléments des deux groupes. Ainsi la distance entre deux groupes (de similarités) S_1 et S_2 vaut :

$$\begin{aligned} d(S_1, S_2) &= \max_{s_1 \in S_1,\, s_2 \in S_2} d(s_1, s_2) && (2.84) \\ &= 1 - \min_{s_1 \in S_1,\, s_2 \in S_2} \max(c(s_1, s_2), 0) && (2.85) \end{aligned}$$

Pour plus de détails sur les méthodes de clustering hiérarchique nous renvoyons à [Gan et al., 2007, section 7.2].

2.4.2 Corrélation entre les méthodes

Nous présentons ici les résultats obtenus en étudiant la corrélation entre les mesures de similarité sur toutes les paires sélectionnées sur chacun des 8 graphes.

Élimination des méthodes fortement similaires. La figure 2.1 présente les corrélations de Pearson entre 83 mesures de similarité. Nous constatons tout d'abord des corrélations très fortes entre certaines mesures. Aussi dans la suite, afin de rendre plus lisibles les résultats, une seule mesure de chacun de ces groupes sera présentée :

- DS, CO, JD, HDI : nous ne gardons que JD ;
- DS-I, CO-I, JD-I, HDI-I : de la même manière, nous ne gardons que JD-I ;
- LP05, LP001 : nous ne gardons que LP05 ;
- PPR07-avg, PPR08-avg : nous ne gardons que PPR08-avg ;
- RLK05, RLK08, RLK1 : nous ne gardons que RLK08, Notons que LEDK05 est aussi fortement corrélée avec ces mesures ;
- RCTK07, RCTK08 : nous ne gardons que RCTK07 ;

- EDK10, EDK05 : nous ne gardons que EDK10 ;
- MDK2, MDK3 : nous ne gardons que MDK2 ;
- MDK5, MDK7 : nous ne gardons que MDK5 ;
- SRW3, SRW5 : nous ne gardons que SRW5 ;
- PRX2-deg, RA-I [11] : nous ne gardons que RA-I, ;
- CTRW01, CTRW05, CTRW10 : nous ne gardons que CTRW05 ;
- NLDK01, NLDK05, NLDK10 : nous ne gardons que NLDK05 ;
- CTRW01, CTRW05, CTRW10 : nous ne gardons que CTRW05 ;
- CLF5R, CLF7R : nous ne gardons que CLF5R ;

Nous avons vérifié que les corrélations de Spearman, un peu moins discriminantes, amènent aux mêmes conclusions. La plupart de ces fortes corrélations indiquent la faible influence d'un paramètre sur la valeur de la mesure. Notons tout de même la forte corrélation des méthodes locales DS, CO, JD, HDI. D'autres groupes de mesures sont fortement corrélées, mais leurs définitions étant mathématiquement différentes, nous préférons les garder séparément. C'est le cas en particulier de SRW2 et KTZ001.

Les figures 2.3 et 2.2 présentent les matrices des corrélations de Pearson et de Spearman entre les méthodes de cette liste limitée. Dans la suite les comparaisons seront limitées à ces 60 mesures.

Quatre groupes de méthodes. On observe en figure 2.2 que les méthodes peuvent être regroupées en quatre groupes :

(1) SRW2, KTZ001, CN-I, RA-I, HPI-I JD-I, AA-I, SRK08, EDK01, LP05, GEO, PRX*-cos ;
(2) CFL*R, LHN1, HPI, JD, RA, AA, CN ;
(3) toutes les autres méthodes ;
(4) ACT, PA, EDK10, BLD.

Notons que ces groupes sont particulièrement denses, c'est en effet une conséquence de la méthode de saut maximal. Seul le groupe (1) est un peu moins dense. Il peut être redécoupé en 5 sous-groupes (dont un contenant les méthodes locales). On remarque ensuite que les groupes (2), (3) et (4) sont clairement séparés. À l'inverse le groupe (1) est moins clairement défini, en particulier les méthodes des groupes (1) et (3) sont assez fortement corrélées, à quelques exceptions près. Il en ressort donc trois familles de méthodes, que l'on peut interpréter ainsi :

- groupe (2) : méthodes *non* sensibles aux chemins de longueur 1,
- groupe (4) : méthodes rapprochant les sommets de forts degrés (corrélés à PA).
- groupe (1) et (3) : les autres méthodes, plutôt sensibles aux chemins de longueur 1,

11. Ces mesures sont mathématiquement équivalentes, voir section 2.3.1.6.

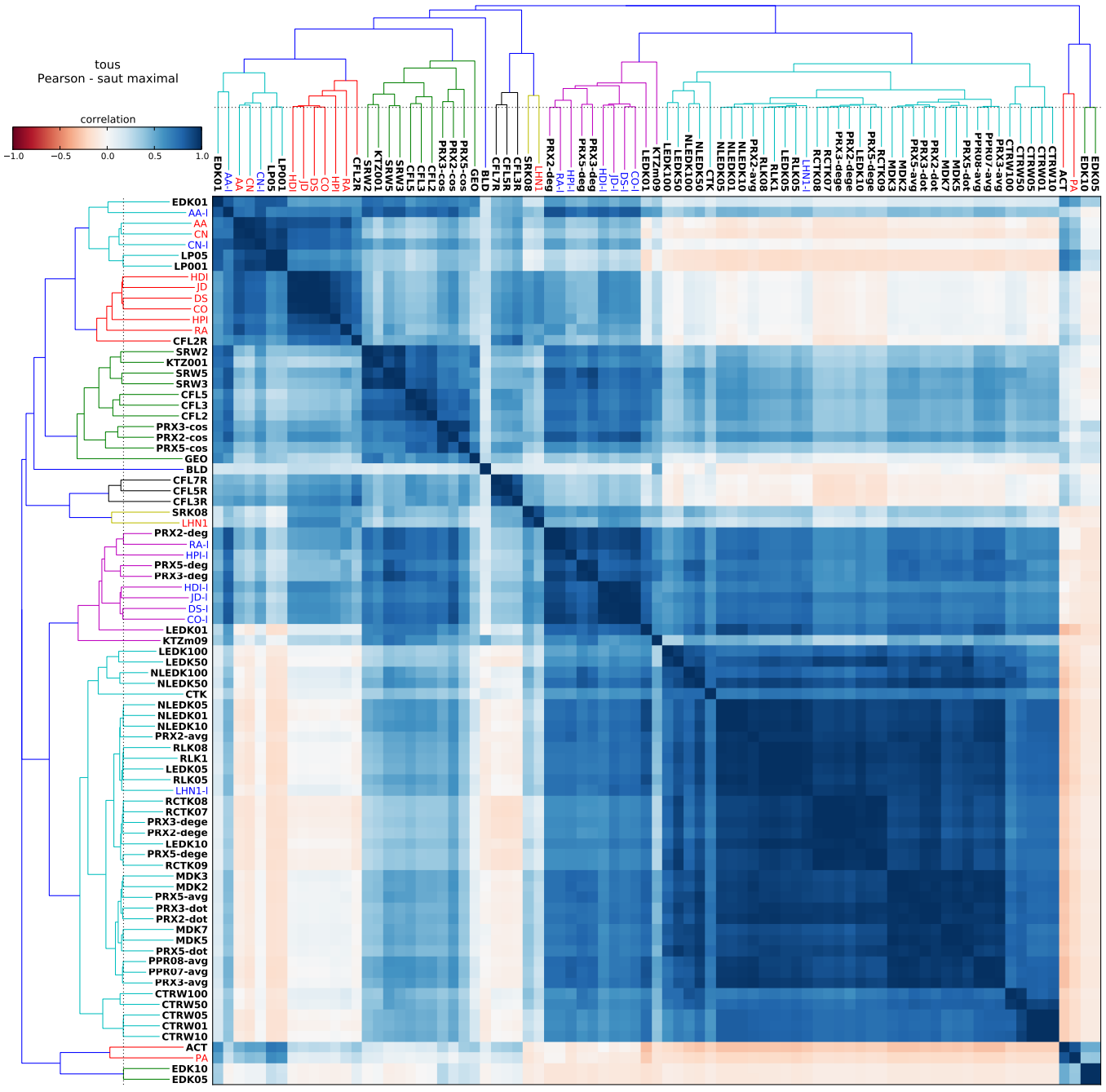


Figure 2.1 – Matrice des corrélations de *Pearson* entre les mesures de similarité calculées sur différentes paires de sommets de 8 graphes. Les deux dendrogrammes sont identiques, et sont calculés avec la méthode du saut maximal.

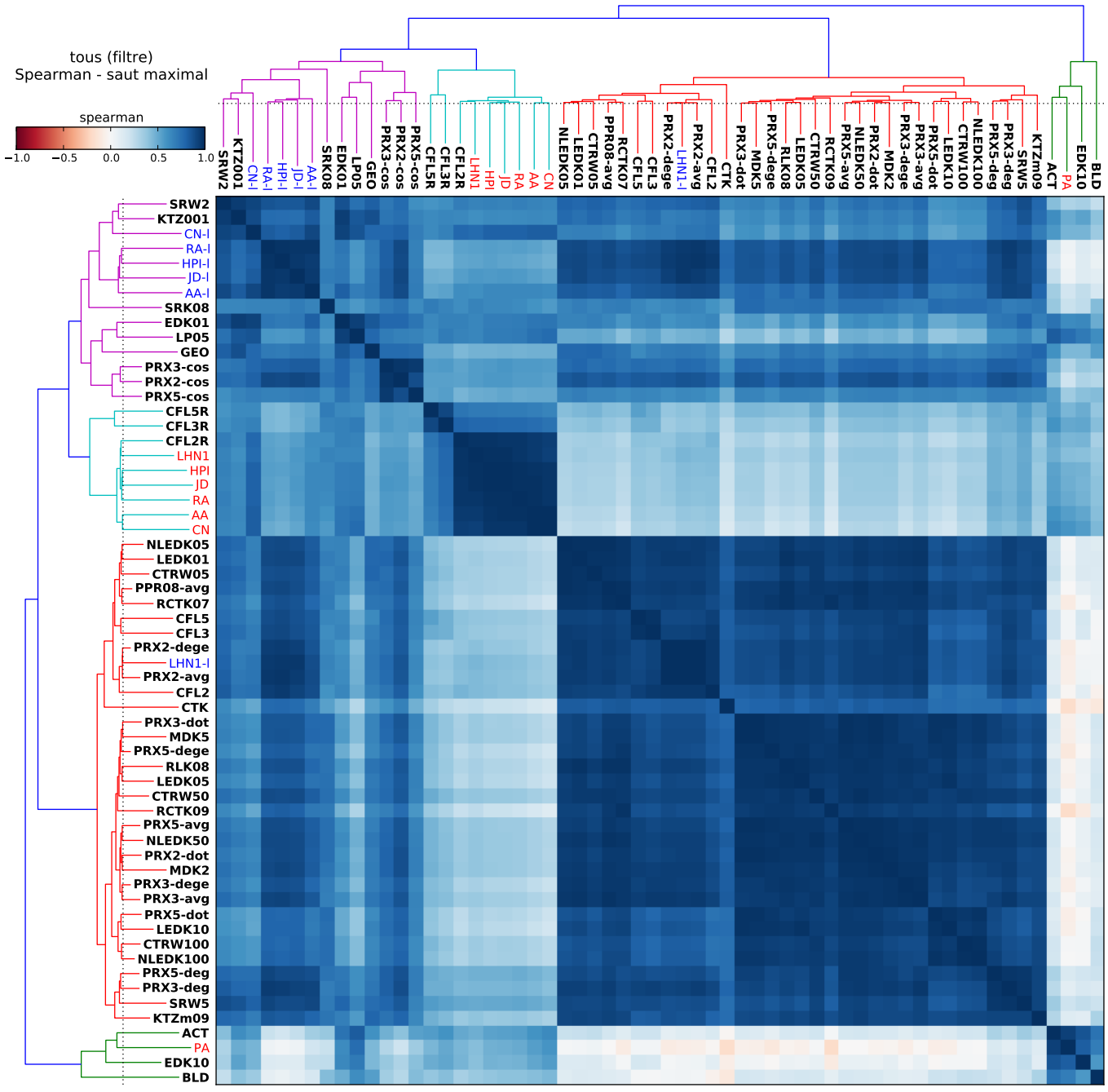


Figure 2.2 – Matrice des corrélations de Spearman entre une sélection des mesures de similarité, calculée sur différentes paires de sommets de 8 graphes. Les deux dendrogrammes sont identiques, et sont calculés avec la méthode du saut maximal.

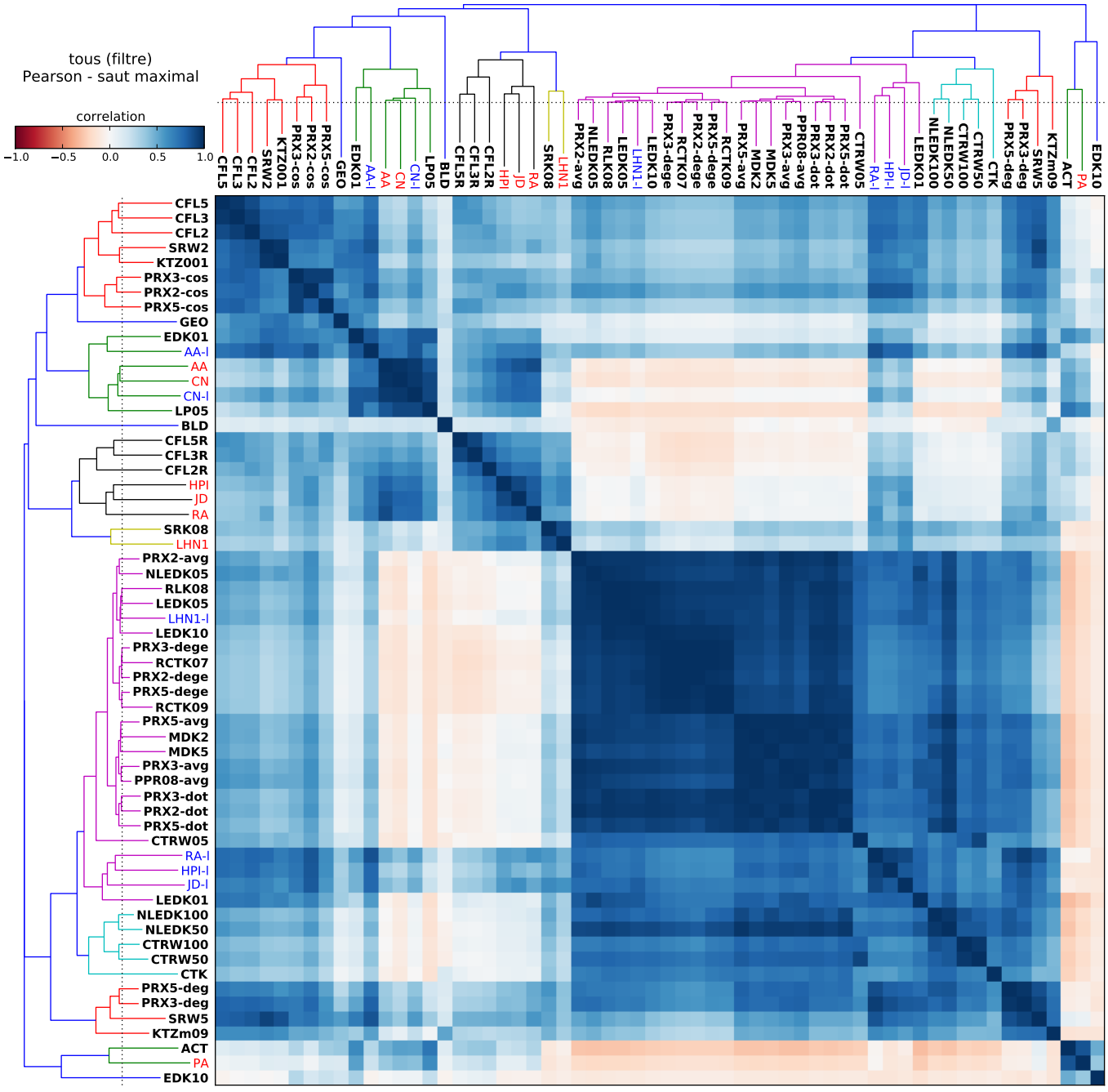


Figure 2.3 – Matrice des corrélations de Pearson entre une sélection des similarités, calculée sur différentes paires de sommets de 8 graphes. Cette figure est identique à la figure 2.1 simplement pour chaque groupe de méthodes très corrélées, une seule a été gardée.

Corrélations de Pearson. Le regroupement résultant des corrélations de Pearson (figure 2.3), est un peu plus fractionné. Globalement on retrouve les trois groupes de méthodes. On peut noter les différences suivantes :

- les méthodes des groupes (1) et (3) sont un peu moins homogènes,
- les méthodes CFL* sont ici regroupées avec les méthodes du groupe (1) (SRW2, KTZ001, etc.),
- BLD n'est plus corrélé au groupe (4),
- SRK08, fortement corrélé à LHN1, se retrouve ici proche des méthodes du groupe (2) (CFL*R, et les méthodes locales sans boucles),
- AA, CN, AA-I, CN-I, EDK01, LP05 se retrouvent ensemble, ce sont toutes des méthodes qui ne sont pas normées.

Mesures locales. On remarque aussi que pour les méthodes locales :

- l'ajout ou non des boucles (« -I ») est important,
- les méthodes se comportent semblablement, sauf bien sûr PA, mais aussi, et c'est plus remarquable, LHN1.

Ces observations sur l'ensemble des graphes et l'ensemble des familles de paires de sommets permettent donc de mettre en avant certains des comportements des méthodes. Seulement il n'est pas évident de comprendre pourquoi telle ou telle mesure se comporte similairement ou différemment. Pour cela nous allons continuer la comparaison en nous restreignant à certains graphes et/ou familles de paires des sommets.

2.4.3 Comparaison sur des graphes aléatoires Erdős-Rényi

Nous nous intéressons ici aux corrélations entre les mesures de similarité calculées uniquement sur les trois graphes aléatoires de type Erdős-Rényi. Il est intéressant d'observer les mesures de similarité sur des graphes aléatoires. En effet comme ces graphes ne présentent pas de structure particulière, il est facile d'observer les biais des méthodes. En particulier nous allons nous intéresser à la sensibilité des méthodes à la densité des graphes. Est-ce que les valeurs de similarité sont plus fortes sur des graphes denses que sur des graphes peu denses ? De plus, nous allons observer la sensibilité des méthodes aux chemins de longueur 1. Est-ce que deux sommets liés par une arête ont une similarité plus forte que deux sommets non adjacents ? A première vue, il peut sembler naturel que ce soit le cas. Pourtant la similarité entre deux sommets ne tient pas seulement au fait qu'ils soient adjacents ou non. Au contraire, il peut être intéressant grâce à une mesure de similarité de questionner la valeur d'une arête : est-ce que le « reste du graphe » confirme la proximité des deux sommets ? Comme dans un graphe aléatoire les arêtes sont indépendantes, a priori

Table 2.7 – Regroupement des mesures de similarité en fonction de leurs sensibilités à la densité, sur les arêtes et non-arêtes de graphes aléatoires. Sur chaque ligne, les méthodes sont classées par ordre alphabétique. Les * indiquent une valeur de paramètre quelconque (parmi celles testées). Les méthodes locales sont soulignées.

sensibilité à la densité sur les non-arêtes	sur les arêtes	méthodes
↗	↗	AA, ACT, CFL2R, CN, CN-l, EDK01, EDK10, HPI, JD, KTZ001, LP05, PA, PRX*-cos, RA, SRW2
↗	–	GEO
↗	↘	CFL2, CTK, HPI-l, JD-l, LEDK01, PRX3-deg, RA-l, RLK08
↗	↘↗	AA-l, PRX5-deg, SRW5
–	–	BLD, CFL3R, CFL5R, LHN1, SRK08
–	↘	CFL3, CFL5, CTRW05, CTRW50, CTRW100, KTZm09, LEDK05, LEDK10, LHN1-l, NLEDK05, NLEDK50, NLEDK100, MDK2, MDK5, PRX*-avg, PRX2-dot, PRX3-dot, PPR08-avg,
↘	↘	PRX*-dege, PRX5-dot, RCTK07, RCTK09

rien dans le « reste du graphe » ne rend deux sommets adjacents plus proches que deux sommets non-adjacents.

Les figures 2.5 et 2.4 présentent les valeurs des différentes mesures de similarité sur une sélection aléatoire d'arêtes et de paires de sommets non-adjacents dans chacun des graphes. Enfin la figure 2.6 donne ces valeurs à la fois sur les arêtes et sur les paires non-adjacentes. Sur chacune de ces figures un regroupement hiérarchique des méthodes est présenté. Ces dendrogrammes sont construits à partir des corrélations de Spearman calculées entre les lignes de valeurs affichées. Les regroupements sont donc différents dans chacune de ces figures.

Sensibilité à la densité. Il est intéressant d'observer à partir des figures 2.5 et 2.4 la sensibilité des différentes méthodes à la densité des graphes aléatoires. Sur chacune de ces figures, les valeurs de chaque méthode sont présentées sur les mêmes paires de sommets de trois graphes aléatoires : ER01, ER02 et ER05. La seule différence entre ces graphes est la densité, ER01 a une densité de 1%, ER02 de 2% et ER05 de 5% (voir tableau 2.5). Certaines méthodes ont tendance à avoir des valeurs plus fortes sur le graphe le plus dense, c'est l'inverse pour d'autres. De plus le comportement est parfois différent pour les paires adjacentes (figure 2.5) et les paires non-adjacentes (figure 2.4).

Le résultat de ces observations est résumé dans le tableau 2.7. On remarque que toutes les méthodes locales (sauf LHN1 et LHN1-l) sont positivement corrélées à la

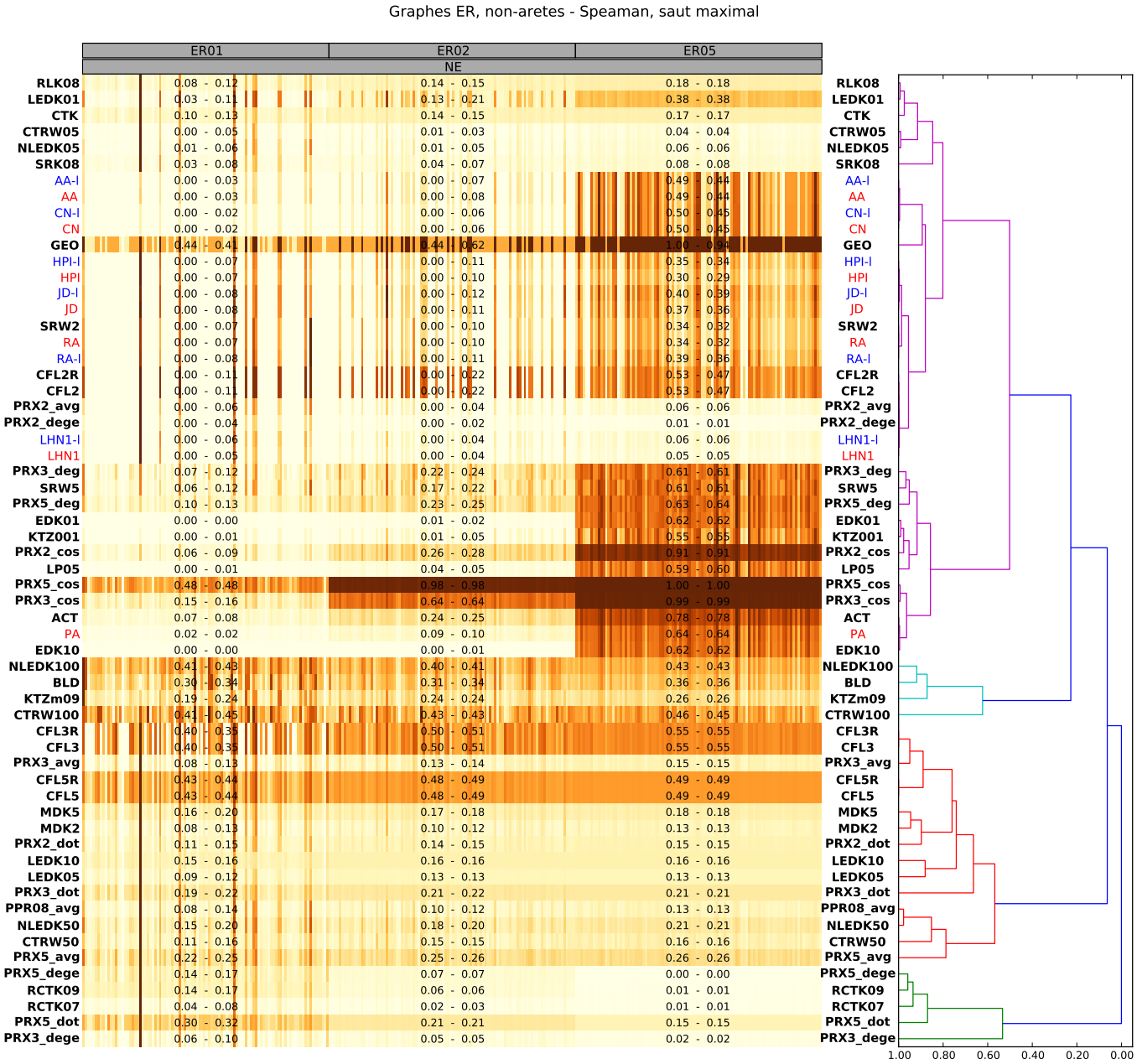


Figure 2.4 – Valeurs des similarités sur des paires de sommets non-adjacentes de graphes Erdős-Rényi (ER). Le regroupement hiérarchique est fait par la méthode de saut maximal à partir des corrélations de Spearman. Pour chaque groupe de paires de sommets il est indiqué la valeur médiane suivie de la valeur moyenne. Ces valeurs sont calculées en ramenant, pour chaque méthode, les scores de similarité entre 0 et 1 (normalisation par rapport à la plus forte valeur observée.).

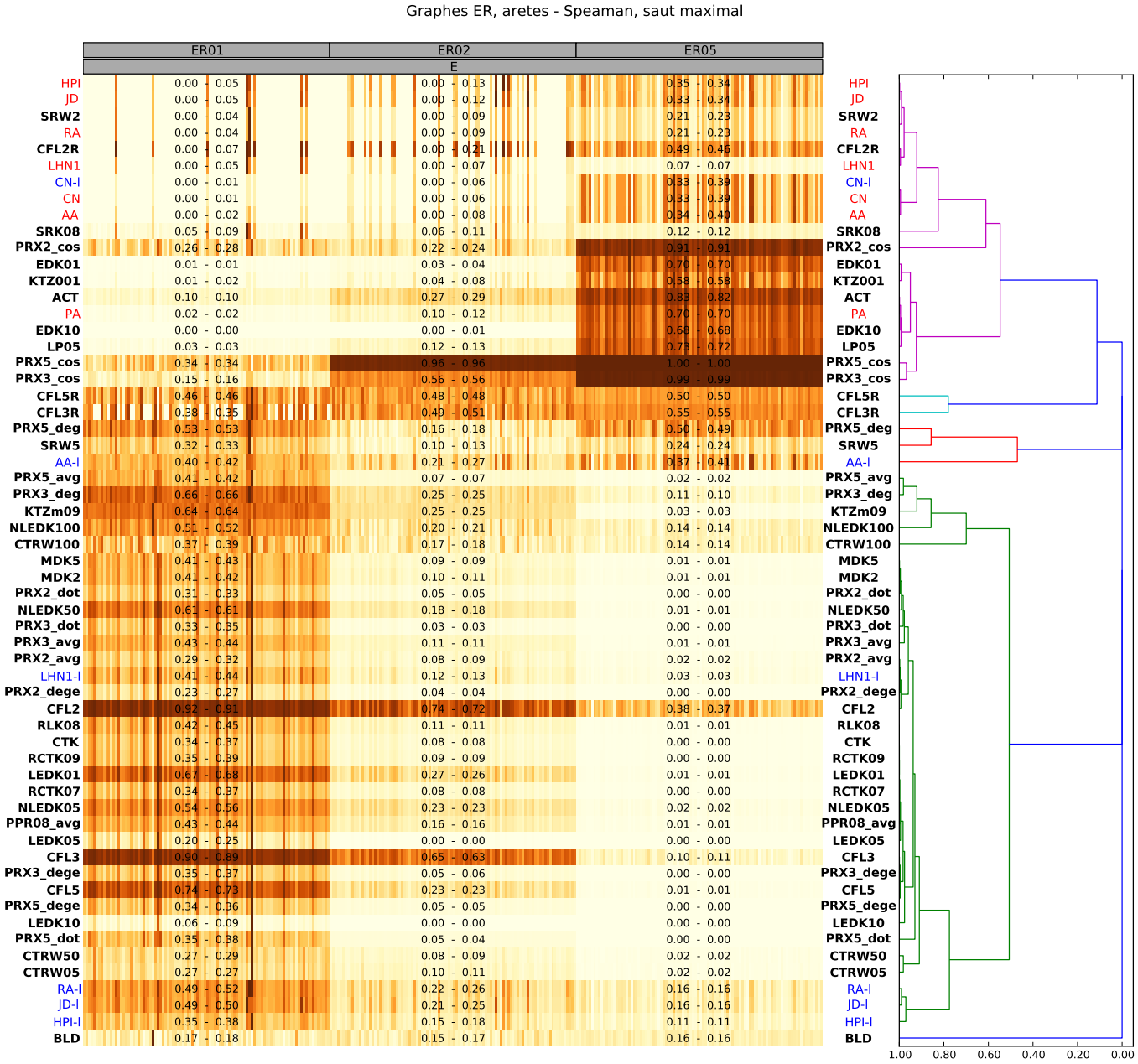


Figure 2.5 – Valeurs des similarités sur des arêtes de graphes Erdős-Rényi (ER). Le regroupement hiérarchique est fait par la méthode de saut maximal à partir des corrélations de Spearman. Pour chaque groupe de paires de sommets il est indiqué la valeur médiane suivie de la valeur moyenne. Ces valeurs sont calculées en ramenant, pour chaque méthode, les scores de similarité entre 0 et 1.

densité sur les paires non-adjacentes. L'ajout de boucles sur les sommets rend les mesures inversement corrélées à la densité sur les paires adjacentes (sauf pour CN-I et AA-I). Cela n'est pas surprenant au regard des définitions de ces méthodes. Plus le graphe est dense plus la probabilité d'avoir des voisins communs augmente. Pour les paires adjacentes, si le graphe comporte des boucles sur chaque sommet, le nombre de voisins communs vaut au moins 2 ; si le nombre de voisins de chaque sommet augmente (plus forte densité) alors toutes les méthodes qui normalisent le nombre de voisins communs en fonction du nombre total de voisins ont tendance à diminuer (jusqu'à une certaine densité à partir de laquelle cet avantage de 2 voisins communs ne suffit plus, les valeurs deviennent similaires à celle des paires non-adjacentes, c'est ce qui se passe pour AA-I).

Le comportement des méthodes globales (ou semi-globales) est moins évident. Sur les paires non-adjacentes, une bonne partie des méthodes n'est pas corrélée à la densité. Toutefois pour les paires non-adjacentes, il faut remarquer que ACT, CTK, EDK*, KTZ001, LP05, PRX*-cos, SRW* sont corrélées à la densité, et PRX*-dege, PRX5-dot, RCTK* sont inversement corrélées. Notons que plusieurs méthodes sont corrélées à la densité seulement quand leur paramètre est de faible valeur (c'est le cas de CFL et LEDK). Notons que KTZ001 est corrélé à la densité alors que KTZm09 ne l'est pas, cela est dû au fait que le choix du paramètre alpha pour KTZm09 dépend du graphe ($\alpha = 0.9\lambda_1$).

Toutes les méthodes non sensibles à la densité sur les paires non-adjacentes sont inversement corrélées à la densité sur les paires adjacentes, sauf BLD, CFL3R, CFL5R, LHN1 et SRK08. Le phénomène est semblable à ce qui se passe pour les méthodes locales avec boucles : plus le graphe devient dense moins l'avantage d'être directement lié a d'importance.

On observe donc que BLD, CFL3R, CFL5R et SRK08 sont les seules mesures globales à ne pas être corrélées à la densité des graphes. L'astuce consistant à ignorer l'arête liant deux sommets lors du calcul de la similarité de ces deux sommets fait que, sur les arêtes, CFL devient non sensible à densité. Certainement la même astuce appliquée aux autres méthodes non sensibles à la densité pour les paires non-adjacentes produirait le même résultat. Notons que les valeurs de CFL3R, CFL5R et de BLD ne sont pas corrélées.

Le fait que RCTK et PRX*-dege soient inversement corrélées à la densité se comprend en reprenant les définitions de ces deux méthodes : leurs valeurs sont le résultat d'une marche aléatoire divisée par le degré d'arrivée (marche aléatoire en temps courts pour PRX*-dege, et marche de type *PageRank* personnalisé pour RCTK). Donc plus le degré moyen devient important, plus les valeurs tendent à être faibles.

Chemin de longueur 1. Sur la figure 2.6 les colonnes marquées « E » correspondent aux paires adjacentes, celles marquées « NE » aux paires non-adjacentes.

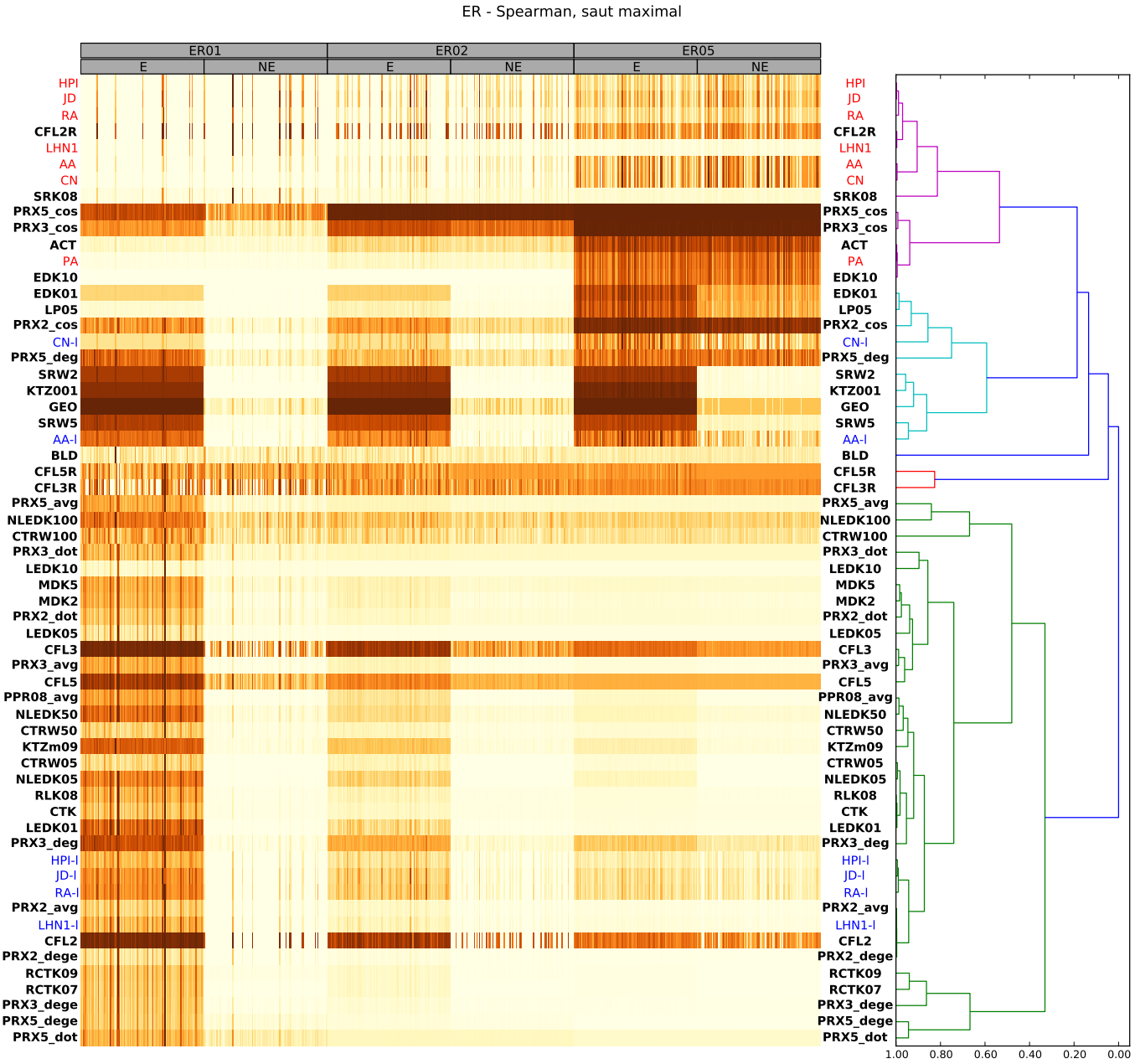


Figure 2.6 – Valeurs des mesures de similarité sur trois graphes Erdős-Rényi (ER). Le regroupement hiérarchique est fait par la méthode de saut maximal à partir des corrélations de Spearman.

Ainsi il est aisé d'observer la sensibilité des mesures aux chemins de longueur 1. On observe que les mesures, pour la plupart, tendent à donner des valeurs plus fortes aux paires adjacentes. Quatre groupes de mesures, à l'inverse, ne donnent pas des valeurs plus fortes aux paires adjacentes :

(1) HPI, JD, RA, CFL2R, LHN1, AA, CN, SRK08,
(2) ACT, PA, EDK10,
(3) BLD,
(4) CFL3R, CFL5R,
(5) NLEDK100, CTRW100.

Pour autant, ces groupes de mesures ne sont pas clairement corrélés entre eux.

La différence de valeurs entre paires adjacentes et non-adjacentes est en général plus marquée sur les graphes peu denses que sur les graphes denses. Cela rejoint les remarques faites précédemment quant à la sensibilité des méthodes à la densité du graphe. La plupart des méthodes donnent des valeurs aux paires adjacentes inversement corrélées à la densité. Plus le graphe est dense, moins le fait d'être adjacent a d'importance.

Nous avons vu ici que les valeurs absolues de beaucoup de mesures sont sensibles à la densité du graphe. Cela signifie, pour notre évaluation, que les corrélations que l'on peut observer lorsque l'on compare les mesures sur des graphes différents peuvent être dues à cette dépendance à la densité. Ainsi dans la suite, pour éviter de re-détecter cette corrélation à la densité, nous étudions les corrélations entre les mesures sur un seul graphe à la fois.

2.4.4 Comparaison sur des graphes simples comportant des clusters

Nous présentons ici le résultat de la comparaison des méthodes sur deux graphes aléatoires simples comportant des clusters. Ces deux graphes (CGMpd et CGMtd) sont construits comme indiqué en section 2.4.1 Dans cette section, nous nous intéressons uniquement au graphe CGMpd, les résultats sur CGMtd étant semblables. Sur ces graphes nous nous sommes intéressés à quatre familles de paires de sommets : les paires adjacentes ou non, et à l'intérieur d'un même groupe ou non (voir tableau 2.6).

La figure 2.7 présente les valeurs des similarités sur les paires non-adjacentes du graphe CGMtd. On constate que toutes les méthodes arrivent bien à différencier les paires intra-clusters (C-NE) des paires inter-clusters (NC-NE), mis à part le groupe de méthodes ACT, BLD, EDK10, KTZm09, PA. Les deux groupes qui apparaissent ensuite correspondent à la distinction entre méthodes locales et globales.

La figure 2.8 présente les valeurs des similarités sur un échantillon des paires adjacentes et non-adjacentes du graphe CGMtd. La figure 2.9 donne la matrice des

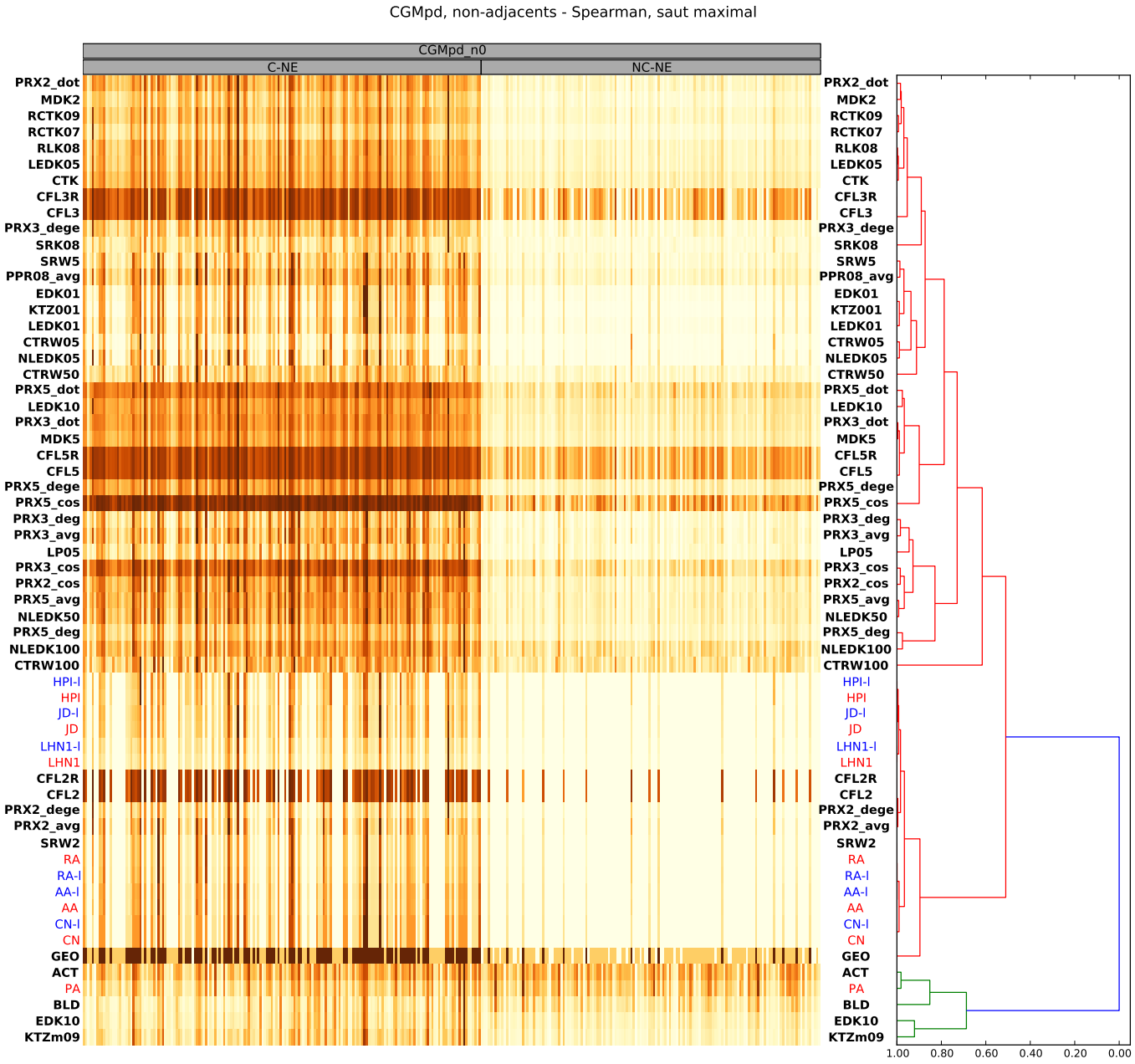


Figure 2.7 – Valeurs des similarités sur des paires de sommets *non-adjacents* du graphe CGMpd. Le dendrogramme représente le résultat d'une agrégation de ces similarités, en fonction de leurs *corrélations de Spearman* sur ces paires de sommets. Le regroupement est fait par la méthode dite du *saut maximal*.

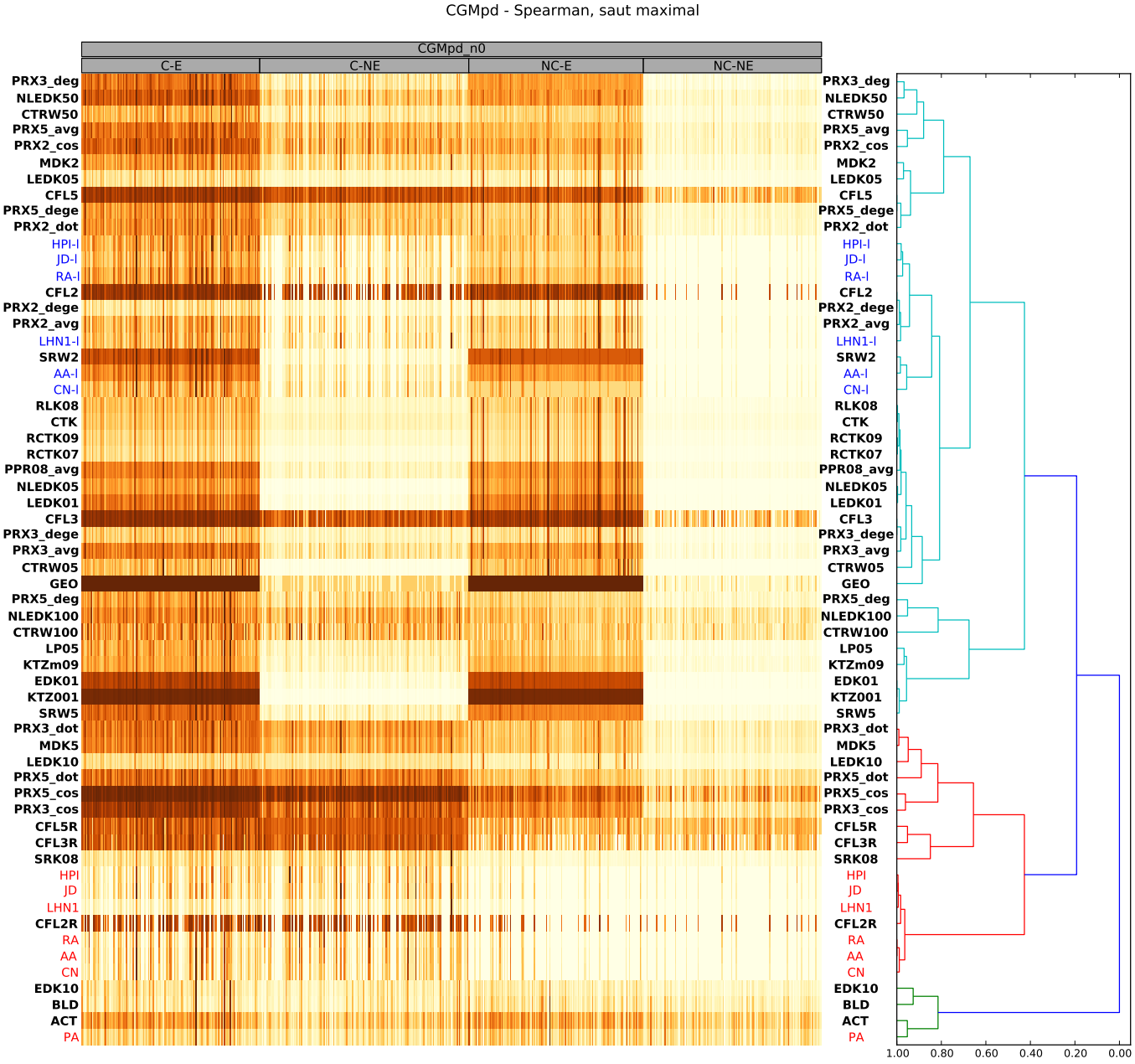


Figure 2.8 – Valeurs des similarités sur des paires de sommets *adjacents* et *non-adjacents* du graphe CGMpd. Le dendrogramme représente le résultat d'une agrégation de ces similarités, en fonction de leurs *corrélations de Spearman* sur ces paires de sommets. Le regroupement est fait par la méthode dite du *saut maximal*.

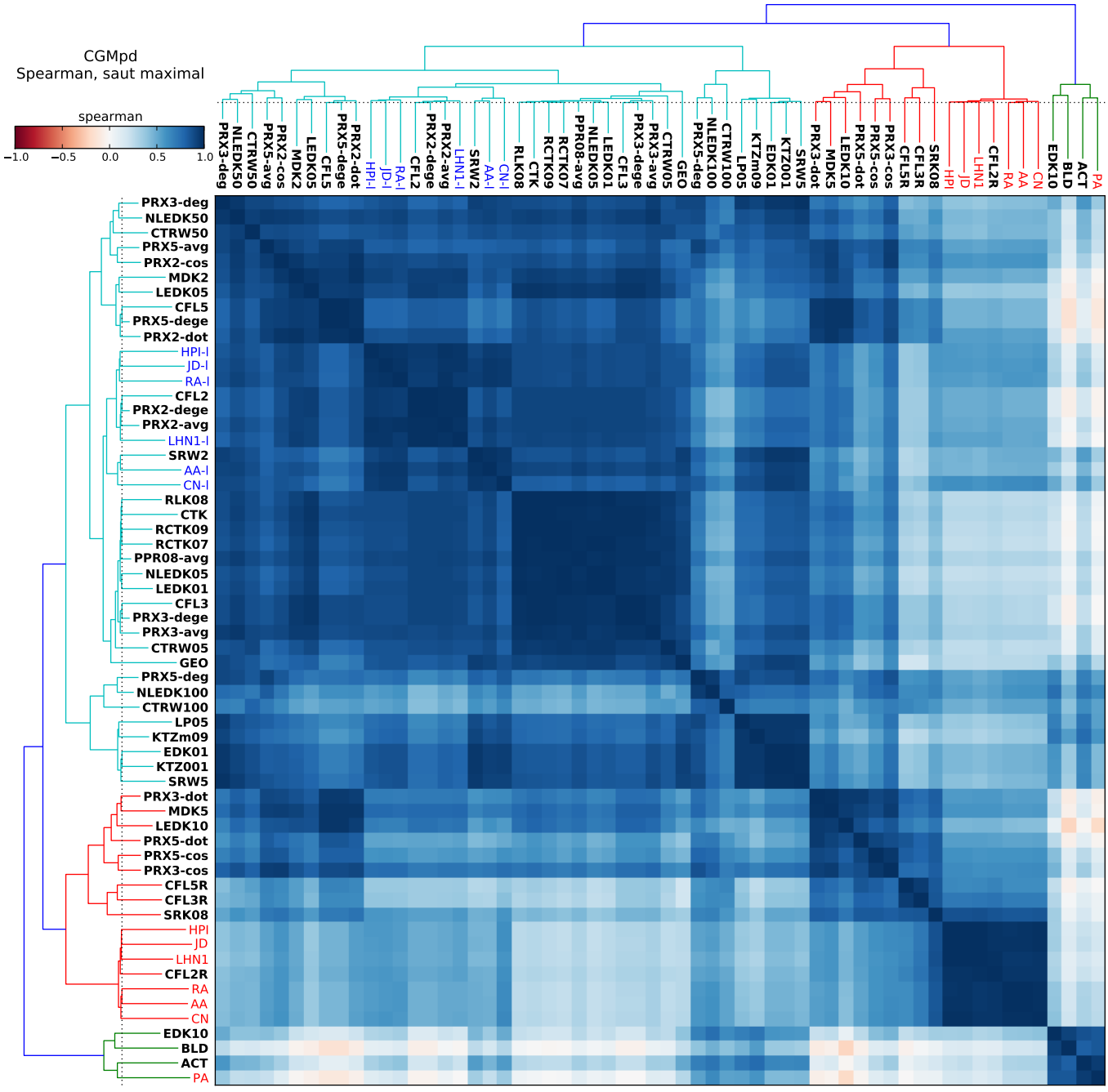


Figure 2.9 – Matrice des corrélations de Spearman entres les différentes mesures, sur des paires *adjacentes* et *non-adjacents* de CGMpd. Les deux dendrogrammes sont identiques à celui de la figure 2.8. Ils sont calculés par la méthode dite du *saut maximal*.

coefficients de corrélation de Spearman calculés à partir de ces valeurs. Il est plus intéressant d'observer ici les valeurs sur les paires adjacentes en même temps que les valeurs des paires non-adjacentes. En effet on observe assez clairement trois groupes de méthodes :

- celles donnant des valeurs *plus fortes* aux paires adjacentes inter-clusters (NC-E) qu'aux paires non-adjacentes mais *intra-clusters* (C-NE),
- celles donnant des valeurs *moins fortes* aux paires adjacentes inter-clusters (NC-E) qu'aux paires non-adjacentes mais *intra-clusters* (C-NE),
- les méthodes EDK10, BLD, ACT, PA qui semblent complètement ignorer la structure en clusters.

La différence entre les deux premiers groupes provient de la sensibilité des méthodes aux chemins de longueur 1. Il est intéressant de noter que la considération des boucles ou non sur les méthodes locales les fait passer de l'un à l'autre des deux premiers groupes. De même l'astuce de CFL*R permet bien d'éviter que les paires adjacentes entre deux clusters aient des valeurs plus fortes que des paires non-adjacentes mais dans le même cluster. Les méthodes globales donnant des valeurs moins fortes aux paires adjacentes inter-cluster qu'aux paires non-adjacentes intra-cluster (second groupe) sont : CFL*R, LEDK10, MDK5, PRX3-dot,PRX5-dot, PRX3-cos, PRX5-cos, SRK08.

2.4.5 Comparaison sur un graphe de terrain

Nous nous intéressons maintenant aux valeurs des méthodes sur un graphe réel. En effet pour éviter de capturer des comportements dus aux différences de densité des graphes, et pour éviter de multiplier les données à analyser, nous nous limitons à un seul graphe.

Les figures 2.10 et 2.12 donnent les valeurs des méthodes pour trois groupes de paires de sommets non-adjacents et adjacents, pour le graphe dsV_lcc. Les dendrogrammes présentés sur la droite des figures sont construits comme précédemment avec la méthode de saut maximal à partir des corrélations de Pearson [12]. Chaque colonne correspond à une paire de sommets : soit deux sommets de forts degrés (BB), soit un sommet de fort degré et un de faible degré (BS), soit deux sommets de faibles degrés (SS). Arbitrairement, nous considérons que les sommets de forts degrés sont les 10% les plus connectés, les 90% restant sont ceux de « petit » degré. Cette décomposition en trois groupes de paires va nous permettre de comprendre le comportement des méthodes face aux degrés des sommets. Est-ce que la similarité entre deux gros a tendance à être plus forte que celle entre deux petits ?

12. Nous utilisons les corrélations de Pearson plutôt que celles de Spearman, les regroupements étant ici plus discriminants et informatifs avec les coefficients Pearson.

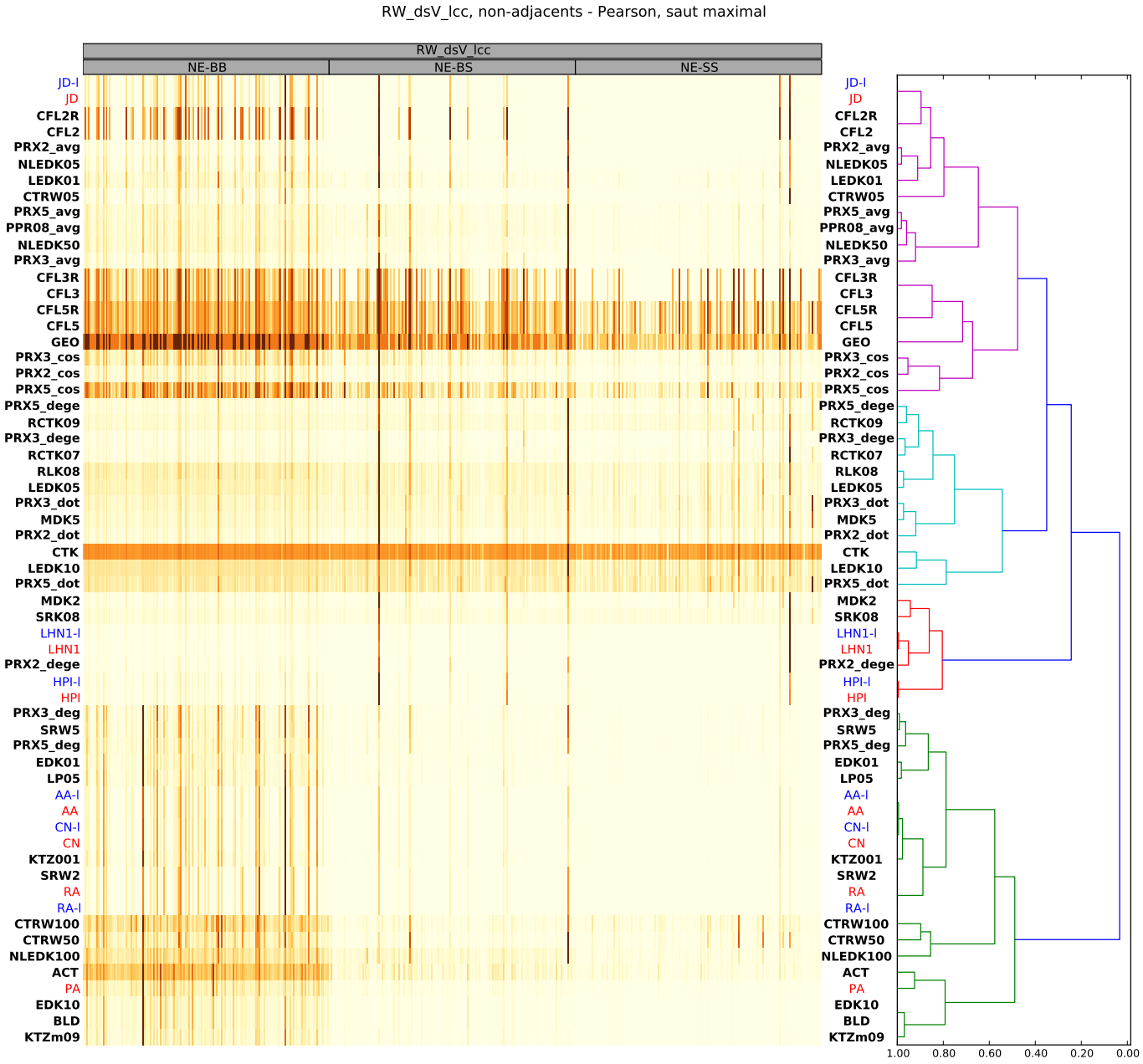


Figure 2.10 – Valeurs des similarités sur des paires de sommets *non-adjacents* du graphe dsV_lcc. Le dendrogramme représente le résultat d'une agrégation de ces similarités, en fonction de leurs *corrélations de Pearson* sur ces paires de sommets. Le regroupement est fait par la méthode dite du *saut maximal*.

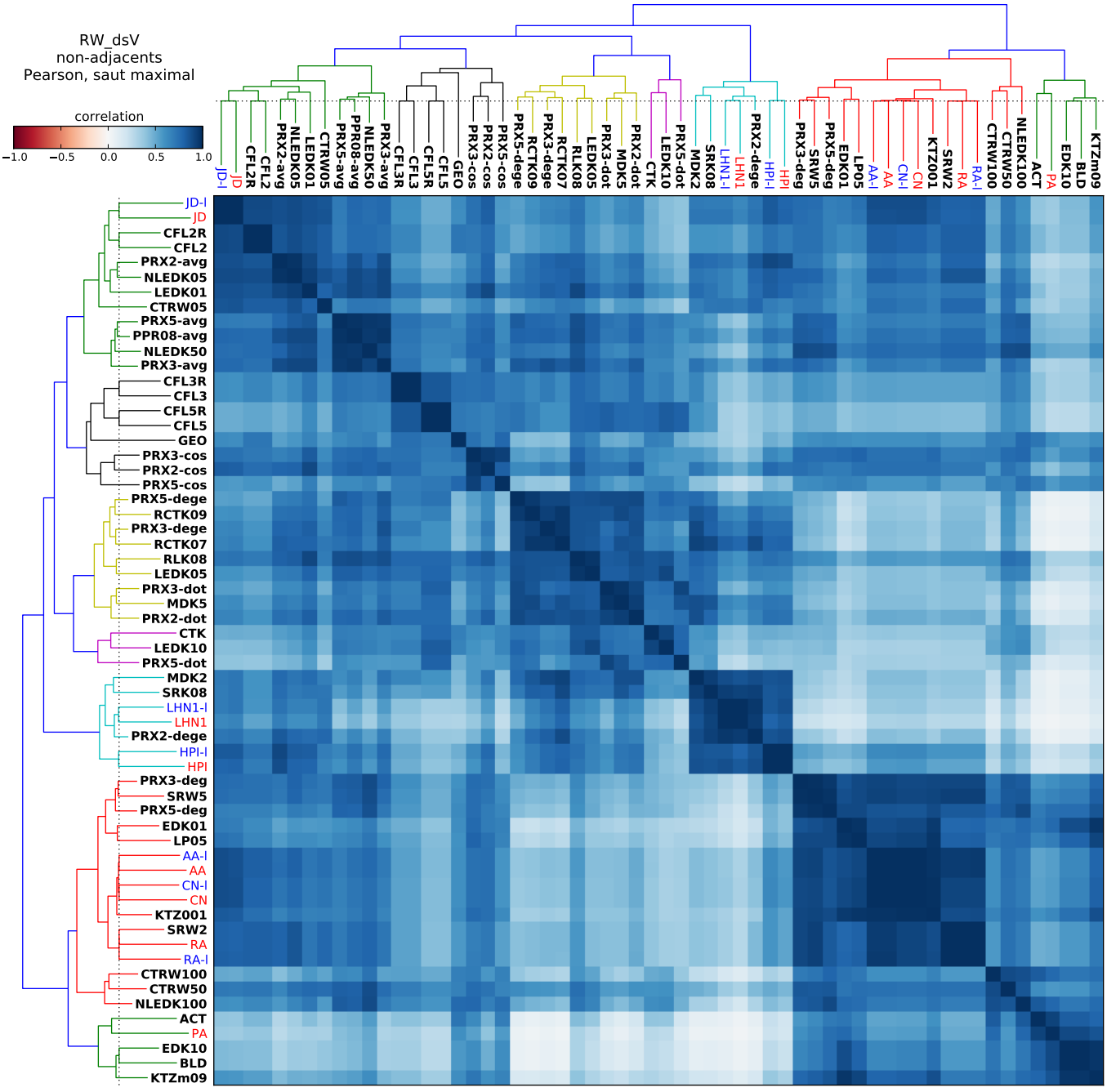


Figure 2.11 – Matrice des corrélations de Pearson entres les différentes mesures, sur des paires *non adjacentes* de dsV_lcc. Les deux dendrogrammes sont identiques, et identiques à celui de la figure 2.10. Ils sont calculés par la méthode dite du *saut maximal*.

Les figures 2.11 et 2.13 donnent les matrices des corrélations de Pearson obtenues à partir des valeurs présentées respectivement dans les figures 2.10 et 2.12.

Comportement sur les paires non-adjacentes. Les figures 2.10 et 2.11 présentent les résultats sur des paires non-adjacentes. Sur la figure 2.11, on remarque une structure en trois groupes de taille équivalente :

(NE.1) JD-l, JD, CFL*R, CFL*, PRX*-avg, NLEDK05, NLEDK50, LEDK01, CTRW05, PPR08-avg, GEO, PRX*-cos,

(NE.2) PRX*-dege, RCTK*, RLK08, LEDK05, LEDK10, PRX*-dot, MDK*, CTK, SRK08, LHN1-l, LHN1, HPI-l, HPI,

(NE.3) PRX*-deg, SRW5, EDK01, EDK10, LP05, AA-l, AA, CN-l, CN, KTZ001, SRW2, RA, RA-l, CRTW100, CRTW50, NLEDK100, ACT, PA, BLD, KTZm09,

En observant les valeurs des mesures sur la figure 2.10, il est possible de comprendre les différences de comportement entre ces trois groupes. Les mesures du groupe (NE.3) pénalisent les paires formées de petits sommets (SS et BS) et privilégient les paires de forts sommets (BB). Les méthodes du groupe (NE.2) semblent soit pénaliser les paires BB, soit être insensibles aux degrés des sommets. Enfin les méthodes du groupe (NE.1) ont un comportement intermédiaire : les paires BB ont des valeurs fortes sans que les autres paires ne soient pénalisées.

Il faut remarquer que la corrélation entre les groupes (NE.2) et (NE.3) est faible, alors que les mesures du groupe (NE.1) sont relativement corrélées avec les groupes (NE.2) et (NE.3). Aussi parmi les groupes (NE.2) et (NE.3) deux sous-groupes compacts sont remarquables :

- sous-groupe de (NE.2) : MKD2, SRK08, LHN, LHN-l, PRX2-dege, HPI-l, HPI, aucune valeur forte sur les paires BB ;
- sous-groupe de (NE.3) : ACT, PA, EDK10, BLD, KTZm09, aucune valeur forte sur les paires BS et SS ;

Les méthodes de ces deux sous-groupes ont nettement un comportement biaisé en fonction des degrés des paires observées. Cela est cohérent au regard des formules de certaines de ces méthodes (en particulier LHN, PRX2-dege ou HPI pour le premier groupe, et PA pour le second groupe), cela est un peu moins évident pour les autres méthodes.

Comportement sur les arêtes. Les figures 2.13 et 2.12 présentent les résultats sur des paires adjacentes. On constate en particulier sur la figure 2.13 que les différences de comportement des méthodes sont beaucoup plus marquées sur les arêtes que sur les paires non-adjacentes. On remarque la présence de deux groupes aux comportements opposés :

(E.1) NLEDK*, LHN1-I, PRX*-dot, MDK*, PRX*-dege, LEDK10, LEDK05, RLK08, PRX*-avg, PPR08-avg, RCTK*, CTK, CTRW100,

(E.2) ACT, BLD, EDK*, KTZm09, LP05, PA,

On comprend avec la figure 2.12 que les méthodes du groupe (E.1) privilégient les paires de petits sommets (SS) ; et à l'inverse les méthodes du groupe (E.2) favorisent les paires de gros sommets (BB).

Les autres méthodes ont des comportements plus ou moins semblables à l'un ou l'autre de ces groupes. Notons en particulier que les méthodes AA, AA-I, CN, CN-I, KTZ001 et dans une moindre mesure SRW* et RA ont un comportement analogue aux méthodes du groupe (E.2). Les méthodes restantes sont plutôt corrélées avec le groupe (E.1). Sauf les méthodes CFL*R qui se démarquent en n'étant corrélées ni à l'un ni à l'autre de ces deux groupes. On observe sur la figure 2.12 qu'un certain nombre de paires BS et SS, ont une valeur nulle (case blanche) avec CFL*R alors qu'elles ont une valeur plutôt forte avec CFL* et les méthodes du groupe (E.1). Ces paires de sommets tiennent donc leur forte similarité uniquement grâce à l'arête qui les relie.

Notons que les méthodes du groupe (E.1) correspondent principalement aux méthodes du groupe (NE.2), et celles du groupe (E.2) font toutes partie du groupe (NE.3). C'est-à-dire que, à quelques exceptions près, les méthodes ont un comportement semblable sur les paires adjacentes et non-adjacentes.

2.4.6 Conclusion

La première des conclusions de cette évaluation comparative est que les méthodes ACT, BLD, EDK10, PA ont plusieurs comportements qui rendent ces méthodes difficilement utilisables en pratique. Ce n'est pas surprenant pour PA. De plus le fait que ACT se comporte similairement à PA (ce qui n'est pas trivial) avait été démontré par von Luxburg et al. [2010]. Ces comportements sont plus surprenants pour EDK et BLD. Ces évaluations permettent ensuite de mettre en exergue certaines des propriétés des mesures. À savoir la sensibilité des méthodes à la densité des graphes, aux chemins de longueurs 1, et aux degrés des sommets observés.

Comme attendu, il n'est pas aisé d'observer une différence de comportement notable entre les différentes méthodes locales (sauf bien sûr pour PA). Toutefois la considération des boucles modifie clairement les méthodes pour les paires adjacentes. Il faut aussi remarquer que HPI et LHN1 se différencient des autres méthodes locales. Ces deux méthodes pénalisent fortement les similarités entre sommets de forts degrés (ce qui n'est pas surprenant à la vue de leurs formules). Aussi LHN1 se montre insensible à la densité sur des graphes aléatoires, ce qui là encore est cohérent par rapport à sa définition utilisant un *null model* de même densité.

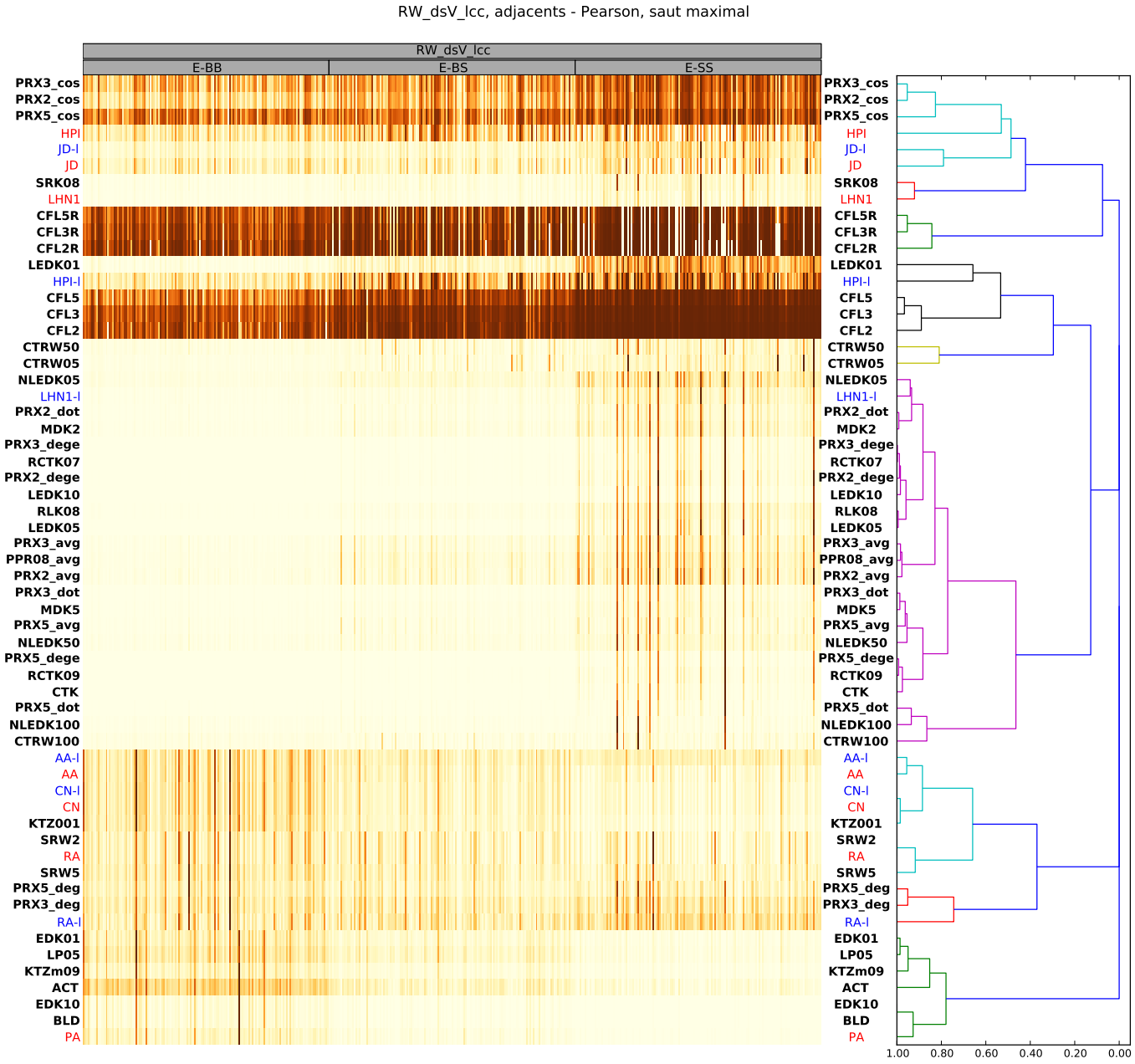


Figure 2.12 – Valeurs des similarités sur des paires de sommets *adjacents* du graphe dsV_lcc. Le dendrogramme représente le résultat d'une agrégation de ces similarités, en fonction de leurs *corrélations de Pearson* sur ces paires de sommets. Le regroupement est fait par la méthode dite du *saut maximal*.

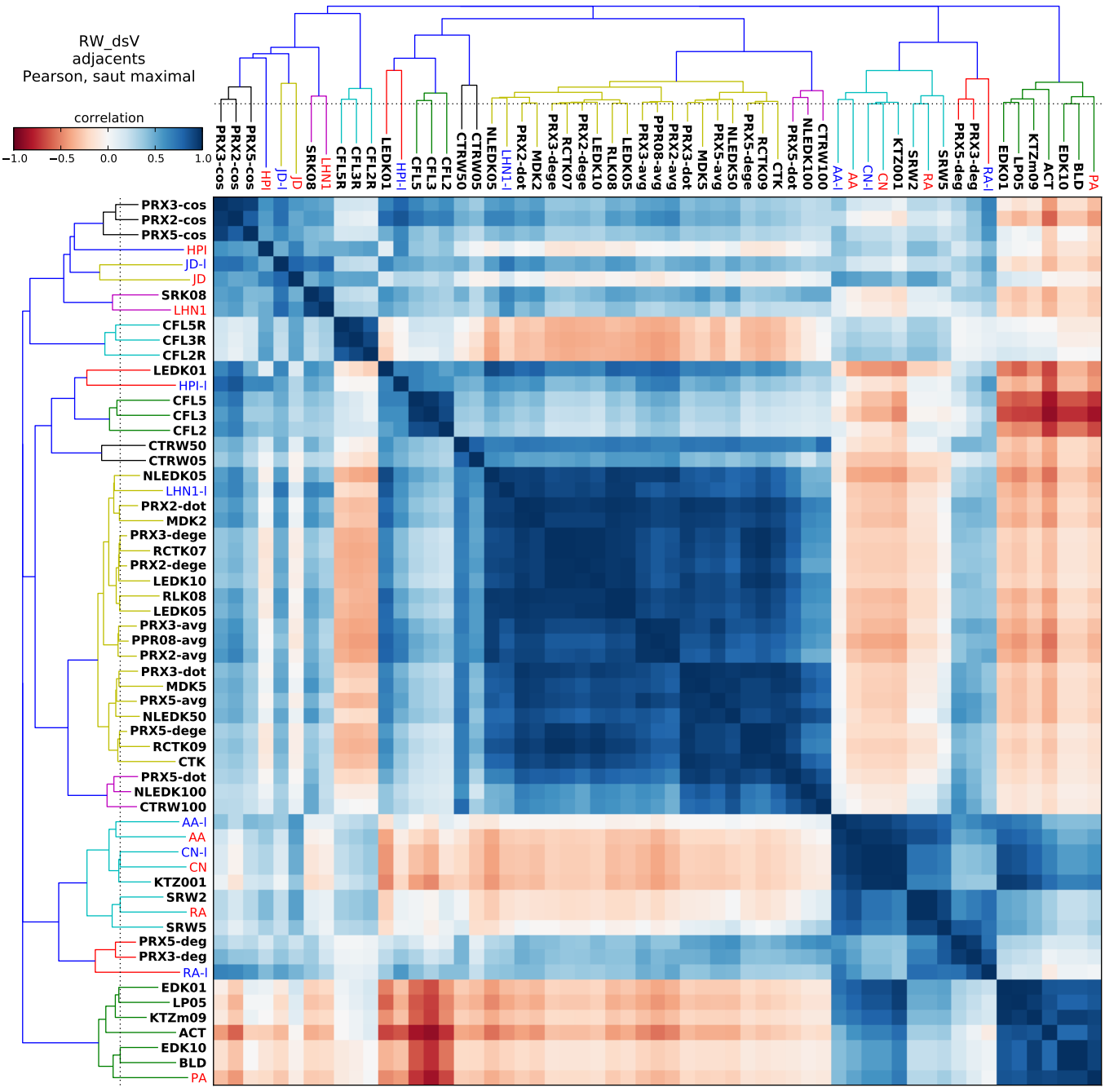


Figure 2.13 – Matrice des corrélations de Pearson entres les différentes mesures, sur des paires *adjacentes* de dsV_lcc. Les deux dendrogrammes sont identiques, et identiques à celui de la figure 2.12. Ils sont calculés par la méthode dite du *saut maximal*.

Nous avons observé que les valeurs de toutes les méthodes locales (sauf LHN1) ainsi qu'un certain nombre des méthodes globales (ou semi-globales) sont sensibles à la densité du graphe. Cela ne pose pas forcément de problème dans la pratique, mais il faut être prudent si l'on souhaite comparer les valeurs de similarités entre plusieurs graphes. Par exemple, si l'on cherche à mesurer, pour deux sommets présents dans plusieurs graphes, quel est le graphe dans lequel ils sont les plus proches. En particulier les méthodes (semi-) globales ACT, CTK, EDK*, KTZ001, LP05, PRX*-cos, SRW* sont à éviter si la sensibilité à la densité a de l'importance.

Aussi nous avons vu que les méthodes ne donnent pas toutes la même importance aux chemins de longueur 1. Là encore, cela ne pose pas forcément de problème. Notamment si l'on utilise une mesure uniquement sur des paires non-adjacentes. En revanche si l'on souhaite utiliser une mesure pour comparer des paires adjacentes et non-adjacentes, alors il faut être prudent. L'idée introduite avec la confluence, consistant à retirer du graphe l'arête reliant deux sommets u et v lorsque l'on calcule leur similarité, fonctionne pour éviter un biais dû à un chemin de longueur 1.

Sur l'exemple d'un graphe réel, nous avons pu observer que les méthodes ont des comportements remarquables par rapport aux degrés des sommets. Certaines méthodes ont un biais en faveur des paires de forts degrés, alors que d'autres ont un biais pénalisant les paires de forts degrés. Il faut faire attention, en particulier, aux méthodes AA, ACT, BLD, CN, CTRW*, EDK, KTZ*, NLEDK100, PRX*-deg, RA, SRW* qui privilégient les paires de sommets de forts degrés, et aux méthodes HPI, LHN1, RCTK*, PRX*-dege qui, elles, pénalisent fortement ces paires de forts degrés. Cette dernière propriété est certainement très importante dans nombre d'applications. Pour autant, mis à part quelques exceptions (en particulier [Leicht et al., 2006]), cette caractéristique est rarement étudiée dans la littérature. Par exemple, Liben-Nowell et Kleinberg [2007] montrent que la mesure de Katz (KTZ*) donne de bons résultats pour le problème de prédiction de liens, pour autant ils n'observent pas qu'elle tend à privilégier les paires de forts degrés (ce qui la rend inutilisable dans certaines applications).

La mesure de confluence définie précédemment se distingue de plusieurs manières. Elle est non corrélée à la densité et a un comportement « équilibré » par rapport aux degrés des sommets (elle n'avantage ni ne pénalise les sommets de forts degrés). Cela n'est pas surprenant étant donné la normalisation par rapport au *null model* (voir proposition 4). De plus l'astuce consistant à retirer l'arête avant de calculer la similarité d'une paire adjacente (CFL*R) permet de rendre la mesure insensible aux chemins de longueur 1. Les seules mesures étudiées présentant des caractéristiques similaires sont SRK et LHN1. Mais ces deux mesures tendent à pénaliser plus fortement les paires de forts degrés. De plus la confluence est une mesure semi-globale contrairement à LHN1, et a une complexité plus faible que SRK. Notons tout de même que le comportement de SRK est remarquable : LHN1 est définie par rapport

au même *null model* que la confluence par contre rien n'indique cela, a priori, pour SRK.

Il faut enfin remarquer qu'en dehors de ces différences les mesures sont fortement corrélées.

2.5 Conclusion du chapitre

Dans ce chapitre, nous avons introduit la *confluence*, une nouvelle mesure de similarité entre sommets d'un graphe, basée sur des marches aléatoires en temps courts. Cette mesure, définie entre 0 et 1, est normalisée par rapport à la similarité attendue entre ces deux sommets sur un graphe aléatoire conservant uniquement les degrés de chaque sommet. Aussi nous proposons d'ignorer volontairement l'arête liant deux sommets lorsque l'on veut calculer la similarité entre ces sommets.

Nous avons ensuite dressé un état de l'art des mesures de similarité entre sommets d'un graphe. Elles se classent en trois catégories : les méthodes locales, globales et semi-globales. Les méthodes locales n'utilisent que les voisinages immédiats des sommets alors que les méthodes globales et semi-globales utilisent potentiellement tout le graphe. Nous avons essayé autant que possible d'indiquer les relations mathématiques pouvant exister entre ces mesures.

Enfin nous avons proposé une étude comparative de plus de 80 mesures. Cette étude consiste à comparer les mesures sur différentes familles de paires de sommets sur ces différents graphes réels ou artificiels. Cette étude permet de mieux comprendre les comportements des méthodes, en particulier par rapport à la densité du graphe, aux chemins de longueur 1 et aux degrés des sommets évalués. Nous renvoyons à la conclusion faite en section 2.4.6 pour un résumé des résultats.

Nous avons essayé d'être aussi exhaustif que possible dans l'état de l'art proposé. Toutefois d'autres méthodes pourraient y être ajoutées. En particulier il manque la mesure globale introduite par Leicht et al. [2006]. Il aurait été d'autant plus intéressant d'étudier cette méthode, qu'elle est définie par rapport au même *null model* que la confluence (la confluence garde tout de même l'avantage d'être définie entre 0 et 1 et d'avoir une complexité plus faible).

Aussi cette comparaison expérimentale n'est qu'une première étape exploratoire. Cela nous a permis de détecter des propriétés intéressantes qui demanderais maintenant à être formalisées et étudiées plus en détail. Cela peut être fait par une étude analytique, mais aussi par des expérimentations plus ciblées. Par exemple il serait possible de tracer, pour une série de graphes aléatoires, le nuage de points indiquant la moyenne des similarités entre sommets non-adjacents et la densité du graphe. Ces mesures serait plus lisibles que celles que nous avons proposées, à condition de se limiter à un nombre restreint de mesures.

Nous nous sommes concentrés dans ce chapitre sur des méthodes définies pour

des graphes non dirigés, connexes et non pondérés. Une perspective évidente est donc d'étendre ce travail aux graphes non-connexes, pondérés, dirigés, etc.

Chapitre 3

Comparaison robuste de graphes

3.1 Introduction

Nous nous intéressons dans ce chapitre à la comparaison de graphes partageant les mêmes sommets. Comment mesurer à quel point deux graphes $G_1 = (V, E_1)$ et $G_2 = (V, E_2)$ ont une structure similaire ? Bien que limité au cas où les deux graphes partagent le même ensemble de sommets, ce problème de comparaison répond à des applications concrètes ; par exemple pour comparer des ressources lexicales définies sur le même ensemble de mots. Est-ce que ces deux dictionnaires de synonymie sont semblables ? Est-ce qu'une ressource construite automatiquement est similaire à ce dictionnaire étalon ?

Aussi nous allons voir que limiter le problème général de la comparaison de graphes à ce cas particulier amène à des questions auxquelles, semble-t-il, les méthodes de l'état de l'art ne se sont pas confrontées. Les mesures « classiques », ramenées à ce cas simple, consistent toutes plus ou moins à compter le nombre d'arêtes en accord et en désaccord. Ces méthodes ne prennent donc pas en compte la structure formée par les arêtes. En effet, deux sommets peuvent être « proches » dans deux graphes (c'est-à-dire, en substance, être lié par un nombre important de chemins courts [1]) sans pour autant être adjacents dans les deux graphes. De cette manière deux graphes peuvent comporter exactement, et très clairement, les mêmes clusters et n'avoir pourtant que peu d'arêtes en commun, et donc être peu similaires d'après une mesure classique.

Après avoir constaté ce problème nous proposons une solution consistant à généraliser une distance d'édition de graphes. L'idée est que le coût d'ajout (ou de suppression) d'une arête dépende de la structure du graphe entre les deux sommets concernés. Ce coût est calculé grâce à la mesure de *confluence* définie au chapitre 2. Nous présentons une évaluation de la méthode sur différents graphes artificiels.

1. Nous renvoyons au chapitre 2 pour une discussion plus en détail à propos des mesures de similarité entre sommets dans un graphe.

Nous introduisons aussi une méthode se basant sur la même idée pour fusionner deux graphes. Le résultat de cette méthode est, par construction, compris entre l'intersection et l'union des ensembles d'arêtes des deux graphes.

Enfin ces méthodes sont employées pour comparer des réseaux de synonymes. Ce travail, appliqué aux graphes de synonymie, tend à donner un éclairage nouveau sur la notion de synonymie. En effet nous montrons que plusieurs graphes issus de dictionnaires papiers différents, présentent un assez faible accord au niveau des arêtes alors qu'ils sont fortement en accord d'après notre méthode de comparaison. Il semble donc plus pertinent de considérer la synonymie pour la similarité qu'elle peut induire entre les mots, plutôt que comme un « sac » de relations binaires.

Ce travail vient à la suite de plusieurs publications. Dans [Gaillard et al., 2011] nous nous sommes d'abord interrogés sur ce problème à partir du cas pratique de comparaison de réseaux lexicaux. Puis dans [Navarro et al., 2012a] nous avons proposé une analyse plus abstraite du problème. Ce chapitre prolonge ces travaux, en particulier en introduisant une généralisation de la *distance d'édition entre graphes* et en proposant une évaluation plus solide.

3.2 Comparaison de graphes ayant les mêmes sommets

Nous nous intéressons donc à la comparaison de deux graphes partageant le même ensemble de sommets. Dans cette première section, nous définissons une formulation de la distance d'édition entre graphes, extrêmement simple dans ce cas. Nous montrons ensuite les problèmes que pose une telle comparaison arête par arête. Enfin, nous présentons un tour de la littérature qui semble indiquer que ce problème n'a encore jamais été abordé.

3.2.1 Distance d'édition entre graphes (*GED*)

Une méthode classique pour comparer des graphes est d'utiliser une *distance d'édition* [Gao et al., 2010]. Ces distances entre graphes sont inspirées des distances d'édition définies originellement entre chaînes de caractères [Levenshtein, 1966]. Une distance d'édition (« *graph edit distance* » en anglais) est définie entre deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ comme la série la moins coûteuse de modifications permettant de rendre G_1 et G_2 identiques. Les modifications sont généralement des insertions, suppressions ou substitutions de sommets et d'arêtes, appliquées à l'un des graphes. Le problème est complexe quand il nécessite de trouver un appariement des sommets (voir sous-section 3.2.3). Dans le cadre de notre travail, puisque $V_1 = V_2 = V$ la situation est beaucoup plus simple. En effet, les seules opérations possibles sont la suppression et l'ajout d'arêtes. Si le coût d'ajout ou de suppression d'une

arête est 1, alors la distance entre G_1 et G_2 vaut :

$$ED(G_1, G_2) = |E_1 \cap \overline{E_2}| + |E_2 \cap \overline{E_1}| \tag{3.1}$$

Remarquons que $ED(G_1, G_2) \in [0, |E_1| + |E_2|]$. Aussi on préfère généralement la mesure normalisée suivante :

$$GED(G_1, G_2) = \frac{|E_1 \cap \overline{E_2}| + |E_2 \cap \overline{E_1}|}{|E_1| + |E_2|} \tag{3.2}$$

Cette mesure est bien définie pour une paire de graphes partageant le même ensemble de sommets. Sa valeur est comprise entre 0 et 1 : 0 indique un accord total entre les graphes et 1 un désaccord complet. Cet accord ou désaccord est mesuré, comme nous allons le voir, au niveau des arêtes.

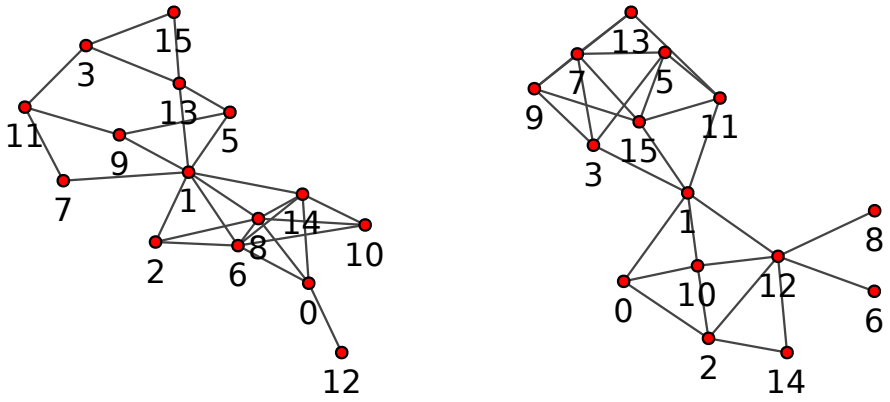
3.2.2 Structures identiques, pourtant sans « atome » commun

Nous montrons ici qu'une mesure de type GED est insuffisante : en effet, deux graphes peuvent présenter une structure similaire sans pour autant avoir d'arête en commun.

La figure 3.1 donne un exemple de deux graphes très similaires mais sans arête commune. En effet, on remarque que dans le graphe G_1 comme dans le graphe G_2 les sommets pairs et les sommets impairs forment deux groupes assez nets qui ne sont reliés entre eux que par le sommet 1. Les deux graphes sont complètement différents au niveau des arêtes. On a donc $GED(G_1, G_2) = 1$ la distance maximale, alors que l'on s'attendrait à une mesure indiquant une assez forte similarité entre ces deux graphes.

Le problème pointé du doigt ici est que les deux graphes comportent exactement les mêmes clusters mais construits par des arêtes différentes. Les graphes sont donc considérés à tort comme dissemblables. Le désaccord est créé par des paires de sommets *proches* dans les deux graphes, mais qui ne sont adjacentes que dans l'un des graphes. Chacune des paires de sommets en désaccord est considérée indépendamment des autres, indépendamment de la similarité qui peut exister entre ces sommets.

Il est possible d'imaginer le cas dual, c'est-à-dire un désaccord créé par des paires de sommets « *non-proches* » mais quand même adjacents dans l'un des deux graphes. Le problème se pose en particulier entre deux graphes en accord sur une certaine structure en cluster mais en conflit sur un ensemble d'arêtes aléatoires entre ces clusters. Ces deux graphes peuvent être considérés comme non similaires à cause de ces arêtes « entre les clusters », alors même que leurs structures en cluster sont les mêmes. Pour que le problème apparaisse, il suffit d'augmenter suffisamment



(a) graphe G_1 **(b)** graphe G_2

Figure 3.1 – Exemple jouet de deux graphes similaires pourtant sans arête commune.

le nombre de clusters. En effet, le nombre d'arêtes entre les clusters peut devenir (largement) plus grand que le nombre d'arêtes à l'intérieur, sans pour autant que la densité de ces arêtes aléatoires ne rende illisibles les clusters. Prenons par exemple deux graphes formés de 1 000 cliques identiques de 5 sommets. Cela fait donc 10 000 arêtes « intra-cliques » pour 5 000 sommets. On peut imaginer sur chacun de ces graphes avoir 100 000 arêtes aléatoires entre ces différentes cliques. La probabilité qu'il existe un lien entre deux sommets est donc inférieure à 0.01, alors que la probabilité d'avoir un lien entre deux sommets du même cluster est de 1. Les cliques sont donc toujours très nettes, bien que chaque sommet ait plus de voisins avec le « reste du monde » qu'avec les sommets de sa clique. Ces deux graphes comparés avec GED apparaîtront distants, car ils n'ont que 10% de leurs arêtes en commun, pourtant leurs structures, formées des cliques, sont les mêmes.

Dans un premier cas, un désaccord est considéré alors que les paires sont proches dans chacun des graphes d'après leurs structures. Dans le second cas un désaccord est compté alors que dans les deux graphes les paires sont distantes. Les deux situations problématiques présentées ici proviennent du fait qu'une mesure de type GED considère les arêtes comme indépendantes les unes des autres sans considérer la structure qu'elles dessinent. Nous présentons en section 3.3 une analyse plus fine de ce problème, avant de présenter une généralisation de GED qui y réponde.

3.2.3 De l'isomorphisme de graphes aux distances d'édition

Avant de proposer une méthode qui réponde aux problèmes exposés, nous revenons ici sur les méthodes existantes de comparaison de graphes. Il existe, en effet, une littérature importante sur la comparaison de graphes.

L'approche classique quand on cherche à comparer deux graphes est la recherche d'*isomorphisme*. Deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ sont *isomorphes* s'il existe une fonction bijective $f : V_1 \mapsto V_2$ telle que :

$$\forall u, v \in V, \quad \{u, v\} \in E_1 \Leftrightarrow \Big\{f(u), f(v)\Big\} \in E_2 \tag{3.3}$$

Dans la pratique il existe rarement un isomorphisme parfait entre deux graphes G_1 et G_2. On cherche alors à déterminer à quel point un isomorphisme approché existe entre G_1 et G_2 ou encore s'il existe un sous-graphe de G_1 isomorphe (ou quasi-isomorphe) à G_2.

Un nombre important de méthodes a été proposé pour répondre à ces problèmes. La plupart consistent à calculer une *distance d'édition* entre graphes. C'est-à-dire à chercher le coût minimal des éditions nécessaires pour passer de l'un des graphes à l'autre. Les opérations généralement disponibles sont l'insertion, la substitution et la suppression de sommets et d'arêtes. Les méthodes prennent ou non en considération des attributs sur les sommets et arêtes. Elles varient aussi dans la manière de définir les coûts des éditions et d'explorer l'espace des solutions. Pour plus de détails sur ces méthodes nous renvoyons à la revue proposée par Gao et al. [2010].

Il est important de remarquer que le cas typique d'utilisation des méthodes de comparaison de graphes de la littérature est très différent de ce que nous cherchons à faire ici. En effet, la comparaison de graphes est souvent employée pour répondre à un problème de *reconnaissance de forme*. Typiquement pour rechercher, dans une collection de graphes, ceux similaires à un graphe requête donné. Les graphes encodent par exemple des molécules [He et Singh, 2006; Jiang et al., 2007] ou des images [Sanfeliu et Fu, 1983; Robles-Kelly et Hancock, 2005]. Ils sont de taille modeste (quelques dizaines ou centaines de sommets) et leurs sommets sont « interchangeables » (c'est-à-dire ils n'ont pas d'étiquette, ou alors chaque étiquette est partagée par un ensemble de sommets). La principale difficulté est alors de chercher un appariement (qui peut être implicite) des sommets qui minimise la distance. Le fait d'avoir un appariement imposé est donc un cas particulier pour les approches de la littérature. De plus ce cas ne correspond pas aux situations d'applications usuellement visées par les méthodes. Aussi la plupart des méthodes sont soit inapplicables (car il est impossible de forcer un appariement des sommets) soit deviennent triviales et souffrent du même problème que la mesure GED définie précédemment.

Notons que les méthodes basées sur des noyaux (voir par exemple [Gärtner et al., 2003]), restreintes à notre cas particulier, répondent certainement aux problèmes que nous avons soulevés dans la sous-section précédente. Mais ni ces problèmes, ni

le fait que les méthodes puissent (ou non) y répondre ne sont mis en avant dans la littérature. Aussi l'un des avantages de la méthode que nous proposons, par rapport à ce que pourrait donner une méthode basée sur un noyau, est que la valeur obtenue est comparable avec la valeur de GED définie par l'équation (3.2). Ainsi il est possible de comparer la similarité « structurelle » des deux graphes à leur accord au niveau des arêtes (GED). Notons aussi que la complexité de nos méthodes est plus faible car elle ne nécessite un calcul que sur les paires en désaccord et non sur l'ensemble des paires de sommets. Nous y reviendrons en conclusion, mais il serait certainement pertinent d'explorer le parallèle entre les méthodes à noyaux et l'approche que nous développons ici.

3.3 Proposition d'une mesure de comparaison robuste

Nous proposons tout d'abord dans cette section une analyse plus fine des problèmes soulevés en sous-section 3.2.2. Cette analyse nous amène à proposer une version plus robuste de la distance d'édition permettant de comparer deux graphes partageant les mêmes sommets. L'idée suivie dans cette section consiste à ramener au niveau des paires de sommets une information sur la topologie plus large du graphe, cela en utilisant une mesure de similarité entre sommets. Nous évaluons ensuite cette méthode sur différents jeux de graphes artificiels.

3.3.1 Similarité binaire, différents cas de figure

Les deux cas problématiques soulevés en section 3.2.2 résultent du fait que les arêtes sont considérées indépendamment les unes des autres. En effet, GED suppose qu'un graphe n'est qu'un ensemble de jugements binaires indépendants : soit deux sommets sont en relation, soit ils ne le sont pas. Or, comme nous l'avons vu en sous-section 3.2.2, deux sommets peuvent être proches sans être adjacents. Intuitivement, s'il existe beaucoup de chemins courts entre deux sommets, la situation n'est pas la même que s'il n'existe que peu de chemins (longs) entre ces sommets. Pourtant ces deux situations sont considérées comme équivalentes avec une mesure de type GED.

Pour comprendre ces différents cas, supposons que l'on dispose d'une mesure binaire évaluant si deux sommets sont « proches » (i.e. similaires) d'après la topologie du graphe, et ce indépendamment du fait qu'ils soient adjacents ou non. Pour chaque paire de sommets, quatre cas sont possibles :

- adjacente et similaire (on note « 11 »),
- adjacente *mais* non-similaire (« 10 »),

- non-adjacente *mais* similaire (« 01 »),
- non-adjacente et non-similaire (« 00 »).

On comprend que les situations qui posaient problème jusque là correspondent au cas où une paire est adjacente (et a priori similaire) dans un graphe et non-adjacente *mais* similaire dans l'autre, et au cas où une paire est non-adjacente (et a priori non-similaire) dans un graphe et adjacente *mais* non-similaire dans l'autre.

Les tableaux 3.1 montrent le changement de perspective qui s'opère lorsque l'on considère une similarité binaire, en plus de la notion d'adjacence. Quand on ne considère que l'adjacence, le tableau 3.1a présente les quatre cas possibles pour une paire donnée. Le tableau 3.1b présente les seize cas existant lorsque l'on considère, en plus, une telle similarité binaire. Sur le tableau 3.1a les interprétations (ok = accord, ko = désaccord) des quatre cas sont peu discutables. Par contre, pour évaluer les seize cas du tableau 3.1b des choix doivent être faits. Les qualifications présentées correspondent aux choix suivants :

- priorité de l'adjacence sur la similarité,
- un conflit sur l'adjacence est « résolu » s'il n'y a pas de conflit sur la similarité.

Une façon de comprendre ces choix est de considérer que la similarité vient soit *confirmer* soit *mettre en doute* l'adjacence. Ainsi on a pour une paire :

- adjacente mais non-similaire (10) : doute sur l'adjacence (noté 1?),
- non-adjacente mais similaire (01) : doute sur la non-adjacence (0?),
- adjacente et similaire (11) : certitude sur l'adjacence (1!),
- non-adjacente et non-similaire (00) : certitude sur la non-adjacence (0!).

Ainsi, tant qu'une arête est présente (ou absente) dans les deux graphes, peu importe si il y a un doute ou non, il n'y a pas de conflit (ok^+ ou ok^-). Aussi lors d'un conflit

Table 3.1 – Différents cas possibles lors de la comparaison d'une paire de sommets entre deux graphes et proposition de qualification des conflits. En (a) on ne considère que l'adjacence (1 : la paire est adjacente, 0 : elle ne l'est pas). En (b) on considère l'adjacence et une similarité binaire entre sommets (11 : paire adjacente et similaire, 10 : adjacente mais non similaire, 10 : non adjacente mais similaire, 00 : non adjacente et non similaire).

(a) on ne considère que l'adjacence.

	1	0
1	ok^+	ko
0	ko	ok^-

(b) on considère l'adjacence et une similarité binaire.

		11 1!	10 1?	01 0?	00 0!
11	1!	ok^+	ok^+	$[ok^+]$	ko
10	1?	ok^+	ok^+	$ko^?$	$[ok^-]$
01	0?	$[ok^+]$	$ko^?$	ok^-	ok^-
00	0!	ko	$[ok^-]$	ok^-	ok^-

(paire adjacente d'un coté seulement) si la paire est en doute sur l'un des graphes mais confirmée sur l'autre alors le conflit est évité (cas $[ok^+]$ et $[ok^-]$). Il reste alors deux cas de conflit :

- ko : 11 face à 00 (i.e. 1! face à 0!) on parle de conflit fort,
- $ko^?$: 10 face à 01 (i.e. 1? face à 0?) on parle de conflit faible.

Notons qu'il peut être pertinent de ne pas considérer comme désaccord les conflits faibles. En effet, avec l'interprétation proposée, ce cas correspond au fait que les deux graphes présentent un doute. On ne sait donc pas s'il doit y avoir une arête ou non, mais les deux graphes sont prêts à « faire la concession ».

Il faut remarquer que l'interprétation inverse consistant à considérer la similarité prioritaire sur l'adjacence amène au même résultat. Deux paires similaires de la même manière dans les deux graphes ne seraient pas en conflit et l'adjacence viendrait résoudre les conflits entre paires similaires seulement dans un des graphes. Une dernière solution serait de ne considérer que la similarité en ignorant l'adjacence. Le problème qui se pose alors est que le désaccord de similarité entre les paires non-adjacentes peut « écraser » tout le reste. En effet le nombre de paires non-adjacentes est souvent de plusieurs ordres de grandeur plus grand que le nombre d'arêtes. Les conflits de structure qui peuvent apparaître sur l'ensemble de ces paires, même peu nombreux en proportion, dominent complètement la comparaison. Il faudrait sinon reprendre la mesure pour la normaliser par le nombre total de paires de sommets.

Ainsi en considérant une similarité binaire, qui vient confirmer ou remettre en cause l'adjacence, on observe que les deux exemples problématiques illustrés en section 3.2.2 peuvent être résolus. En effet le premier cas (figure 3.1) correspond à une comparaison où beaucoup des paires sont dans la situation 1! face à 0?. Alors que dans le second cas, la comparaison est faussée par un nombre important de paires 1! face à des paires 0?. Une similarité binaire peut donc être utilisée de la sorte pour proposer une comparaison robuste entre graphes. C'est le travail qui a été proposé dans [Navarro et al., 2012a]. Nous nous limitons ici à exploiter cette idée pour comprendre le problème, nous proposons en effet dans la suite une méthode utilisant une similarité continue plutôt que binaire.

Similarité seulement pour les paires non-adjacentes ? Il est possible de choisir de ne considérer la similarité que pour les paires non-adjacentes (ou symétriquement seulement sur les paires adjacentes). Cela revient à s'interdire de remettre en question les arêtes (ou les non-arêtes). Les tableaux 3.2 présentent ces deux situations. Dans certains cas spécifiques, il peut être intéressant de limiter la considération d'une similarité aux seules paires adjacentes, mais en pratique c'est surtout la mise en cause des paires non-adjacentes qui est intéressante. Il peut y avoir plusieurs raisons à cela. Tout d'abord on peut disposer d'une connaissance extérieure qui indique

Table 3.2 – Comparaison d'une paire de sommets entre deux graphes quand une similarité est considérée seulement sur les paires non-adjacentes ou seulement sur les paires adjacentes.

(a) Seules les paires *non-adjacentes* peuvent être mises en doute par la similarité.

		1 1!	01 0?	00 0!
1	1!	ok^+	$[ok^+]$	ko
01	0?	$[ok^+]$	ok^-	ok^-
00	0!	ko	ok^-	ok^-

(b) Seules les paires *adjacentes* peuvent être mises en doute par la similarité.

		11 1!	10 1?	0 0!
11	1!	ok^+	ok^+	ko
10	1?	ok^+	ok^+	$[ok^-]$
0	0!	ko	$[ok^-]$	ok^-

que les arêtes sont toutes correctes, et que donc les conflits ne peuvent être minimisés même lorsque c'est une arête entre deux sommets éloignés qui pose problème. Une autre raison, plus pragmatique, est que ces cas (paire 1? face à 0!) sont rares. En effet les graphes réels contiennent peu d'arêtes mises en doute par la topologie, et peu de celles-ci correspondent à des paires non-adjacentes dans l'autre graphe. Enfin pour résoudre le premier problème présenté en section 3.2.2, il suffit de prendre en compte une similarité seulement pour les paires non-adjacentes. Dans la pratique, ce problème est plus fréquent que le second. Notons qu'alors il n'existe plus de conflit faible. Dans la suite nous nous limiterons donc à calculer une similarité sur les paires non-adjacentes lors d'un conflit.

Une autre façon de comprendre ces approches « partielles » est de considérer que les graphes sont *corrects mais incomplets* ou *complets mais incorrects*. En effet si l'on considère qu'un graphe est un ensemble d'arêtes, alors dire qu'un graphe est *correct* signifie que toutes ses arêtes sont justes et ne peuvent être remises en cause, par contre il est possible qu'il en manque. À l'inverse dire qu'un graphe est *complet* signifie que toutes les arêtes sont bien présentes, par contre il peut y en avoir « en trop ». Quand on ne considère la similarité que pour les paires non-adjacentes, alors on autorise la topologie à remettre en cause seulement des paires non-adjacentes. Les arêtes sont considérées comme *correctes* dans tout les cas.

3.3.2 Similarité continue, généralisation de GED

Nous avons vu que les limites d'une distance d'édition simple peuvent être dépassées en ajoutant au niveau des paires de sommets une information sur la topologie plus large du graphe. Nous explorons dans cette sous-section une généralisation de la distance d'édition faisant intervenir une similarité prenant une valeur réelle entre 0 et 1. L'idée consiste simplement à calculer le coût d'une édition élémentaire grâce à la mesure de similarité. Ainsi les éditions en contradiction avec la structure du graphe coûteront cher, alors que celles en accord avec la structure seront peu coûteuses.

Nous supposons donc avoir une mesure de similarité Δ comprise entre 0 et 1. Une paire de sommets éloignés d'après la structure du graphe (adjacents ou non) aura une valeur proche de 0, à l'inverse une paire de sommets proches aura une valeur proche de 1. Nous présentons dans la sous-section 3.3.3 les qualités nécessaires de cette similarité. Cette mesure nous permet de définir une fonction de coût pour les deux opérations possibles : l'insertion d'une arête (*add*) ou la suppression d'une arête (*del*). Pour un graphe $G = (V, E)$, nous définissons ces deux fonctions ainsi :

$$\forall \{u, v\} \in E, \quad del_G(u, v) = \Delta_G(u, v) \tag{3.4}$$

$$\forall \{u, v\} \in \overline{E}, \quad add_G(u, v) = 1 - \Delta_G(u, v) \tag{3.5}$$

La distance d'édition consiste à mesurer le coût minimal des modifications à apporter à G_1 et à G_2 pour les rendre identiques. La formulation donnée par l'équation (3.2) se généralise donc de la manière suivante :

$$\frac{1}{|E_1| + |E_2|}\Bigg(\sum_{\{u,v\} \in E_1 \cap \overline{E_2}} \min\Big(del_{G_1}(u, v),\ add_{G_2}(u, v)\Big) + \sum_{\{u,v\} \in \overline{E_1} \cap E_2} \min\Big(add_{G_1}(u, v),\ del_{G_1}(u, v)\Big)\Bigg) \tag{3.6}$$

Chaque arête présente uniquement dans l'un des deux graphes doit être, soit ajoutée à l'un des graphes, soit supprimée de l'autre graphe. On peut vérifier que si les valeurs de *add* et *del* sont toujours égales à 1 on retombe sur l'équation (3.2).

Il est possible d'établir un parallèle avec l'approche présentée précédemment si l'on considère les correspondances suivantes :

- paire « 00 » (ou « 0! ») : $add(u, v) = 1$,
- paire « 01 » (ou « 0? ») : $add(u, v) = 0$,
- paire « 10 » (ou « 1? ») : $del(u, v) = 0$,
- paire « 11 » (ou « 1! ») : $del(u, v) = 1$.

Ainsi, dans le cas d'une similarité binaire, cette généralisation de GED revient à ne compter les conflits que dans les cas où une paire « 0! » est face à une paire « 1! ». Les conflits dit faibles (« 0? » face à « 1? ») ne sont pas comptés.

Seulement des ajouts. Comme nous l'avons vu précédemment il peut être souhaitable de ne considérer la similarité que sur les paires non-adjacentes. Cela permet en particulier de dépasser la principale des limites de GED (voir figure 3.1) sans avoir à calculer de similarité pour chacune des arêtes. Ici cela revient à interdire la suppression d'arête ou, ce qui est équivalent, à décider que le coût d'une suppression

est toujours supérieur au coût d'un ajout dans l'autre graphe. On arrive alors à la mesure suivante :

$$GUD(G_1, G_2) = \frac{1}{|E_1| + |E_2|}\left(\sum_{u,v \in E_1 \cap \overline{E_2}} add_{G_2}(u, v) + \sum_{u,v \in \overline{E_1} \cap E_2} add_{G_1}(u, v) \right) \quad (3.7)$$

Nous nommons GUD cette mesure, pour *graph union dissimilarity*. En effet cette mesure correspond au coût normalisé nécessaire pour transformer G_1 et G_2 en leur union.

Coût du désaccord. La mesure GUD donne donc une distance d'édition entre deux graphes qui est robuste aux désaccords quand ceux-ci sont résolus par l'ajout peu coûteux d'une arête. Toutefois il peut être utile, lorsque l'on compare deux graphes, de savoir si le désaccord provient d'un faible nombre d'arêtes en conflit ou alors de nombreux conflits peu coûteux à résoudre. Pour cela nous proposons la mesure suivante :

$$GUC(G_1, G_2) = \frac{1}{|E_1 \cap \overline{E_2}| + |\overline{E_1} \cap E_2|}\left(\sum_{u,v \in E_1 \cap \overline{E_2}} add_{G_2}(u, v) + \sum_{u,v \in \overline{E_1} \cap E_2} add_{G_1}(u, v) \right) \quad (3.8)$$

GUC signifie *Graph Union Cost*, c'est donc le coût moyen de résolution des conflits. On observe que :

$$GUD(G_1, G_2) = GED(G_1, G_2).GUC(G_1, G_2) \quad (3.9)$$

Enfin, notons que pour comprendre le désaccord entre deux graphes, il est possible de tracer la distribution des coûts d'ajout des arêtes en conflit. C'est-à-dire d'afficher combien d'arêtes peuvent être ajoutées pour chaque plage de « prix » (entre 0 et 1). Une telle courbe permet d'observer la réparation des paires en conflit, par rapport à leur similarité dans le graphe où elles ne sont pas adjacentes.

3.3.3 Choix de la similarité Δ

Le choix de la similarité utilisée est primordial. Il convient ici de contraindre un peu mieux cette mesure. Nous avons déjà supposé que cette similarité prend valeur entre 0 et 1. De plus, il faut que la mesure ne soit pas influencée par certaines propriétés globales du graphe. En particulier il est souhaitable que la similarité ne soit pas sensible à la densité. Nous avons vu au chapitre précédent que c'était le cas de certaines mesures. Par exemple pour le cosinus entre vecteurs d'adjacence : plus le graphe est dense plus cette similarité tend à être importante. Sur un graphe complet le cosinus entre n'importe quelle paire de sommets vaut 1.

Null model. Une manière efficace de contraindre la similarité utilisée pour *GUD* est de préciser qu'elle ne doit pas mesurer simplement à quel point deux sommets u et v sont proches, mais à quel point u et v sont *plus proches* qu'ils ne le seraient dans un graphe aléatoire équivalent. On appelle *null model*[2] ce graphe aléatoire équivalent. L'utilisation d'un tel *null model* permet de se « protéger » des propriétés du graphe aussi présentes dans le *null model*. En effet si deux sommets ont une forte similarité dans le graphe et dans le *null model* alors cette similarité n'est pas due aux propriétés particulières du graphe. Par exemple si le *null model* a la même densité que le graphe, une similarité forte causée par la densité sera forte aussi dans le *null model*. Un *null model* classique dans l'étude des grands graphes de terrain est celui utilisé par la *modularité* [Newman, 2003b]. Il consiste en un graphe aléatoire où chaque sommet possède le même degré que dans le graphe étudié. Ce modèle correspond au *modèle de configuration*[3] avec une séquence de degrés imposée par le graphe étudié. Ainsi la densité du graphe et la distribution des degrés sont conservées, en revanche la structure qui nous intéresse ici, c'est-à-dire les clusters ou « zones denses », est perdue.

Confluence. Afin de respecter ces contraintes, nous proposons d'utiliser la mesure de *confluence* introduite au chapitre précédent (voir section 2.2.2). Voici un rappel de sa formulation :

$$\mathrm{confl}_t(u,v) = \frac{[P^t]_{u,v}}{[P^t]_{u,v} + \pi_v} \tag{3.10}$$

où $[P^t]_{u,v}$ est la probabilité d'atteindre un sommet v en partant de u en t pas aléatoires sur le graphe, et π_v la probabilité d'atteindre v en un temps infini. En effet nous avons vu (cf. proposition 4) que π_v est aussi l'espérance de passer de u à v en un nombre quelconque de pas dans le *null model* que nous souhaitons utiliser.

Nous utilisons l'astuce consistant à ajouter une boucle sur chaque sommet du graphe avant d'effectuer le calcul de la confluence. Aussi pour comparer deux graphes non-connexes nous calculons la limite π_v comme si les graphes étaient connexes. (voir section 2.2.2 pour plus de détails).

3.3.4 Évaluation

Nous présentons ici une évaluation de *GUD* sur plusieurs graphes artificiels. Nous utilisons pour cela des graphes artificiels construits à partir du modèle de graphes CGM présenté au chapitre précédent (sous-section 2.4.1). Pour rappel ces graphes sont formés de k groupes de r sommets, avec une probabilité μ_{intra} qu'une arête

2. On pourrait traduire *null model* par « modèle de référence » en français.

3. Ce modèle consiste en un graphe aléatoire où les degrés de chaque sommet sont fixés, nous renvoyons à [van der Hofstad, 2013, chap. 7] pour plus de détails sur le modèle de configuration.

existe entre deux sommets du même groupe, et μ_{inter} entre des sommets de groupes différents.

L'objectif de cette première évaluation est de vérifier le comportement de GUD dans les quatre cas suivants :

(i) graphe CGM face à un autre graphe CGM avec les mêmes clusters,
(ii) graphe CGM face à un autre graphe CGM mais avec des clusters différents,
(iii) graphe CGM face à un graphe aléatoire de type Erdős-Rényi,
(iv) graphe aléatoire de type Erdős-Rényi face à un autre graphe aléatoire de type Erdős-Rényi.

Dans le cas (i) la valeur de GUD doit a priori être faible alors que celle de GED ne l'est pas forcément. Dans les trois autres cas, GUD et GED doivent détecter la non-similarité des deux graphes, c'est-à-dire présenter des valeurs fortes.

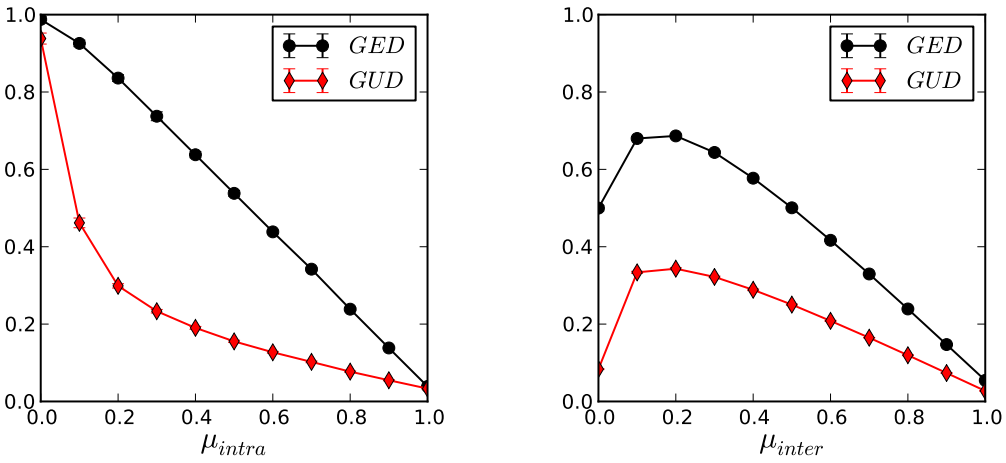
3.3.4.1 Cas (i) : Graphes issus du même modèle

Les courbes de la figure 3.2 présentent les résultats de comparaison entre graphes CGM construits avec les mêmes paramètres (et les mêmes groupes de sommets). Les graphes sont tous construits avec $r = 50$ et $k = 5$. Pour la figure 3.2a $\mu_{inter} = 0.01$ et μ_{intra} varie entre 0 et 1 ; à l'inverse sur la figure 3.2b $\mu_{intra} = 0.5$ et μ_{inter} varie entre 0 et 1. Nous sommes donc dans une situation où les sommets appartiennent aux mêmes clusters dans chaque graphe, mais par contre les liens constituant ces clusters, étant aléatoires, ne sont a priori pas les mêmes.

Analyse des résultats des figures 3.2. On observe tout d'abord, sur la figure 3.2a, que lorsque μ_{intra} vaut 0 alors $GED = 1$ et GUD est très proche de 1. En effet dans ce cas les clusters ne sont pas du tout définis, les seules arêtes du graphe sont complètement aléatoires et la probabilité qu'elles soient en accord est faible. Ensuite lorsque μ_{intra} augmente, GUD diminue nettement. On a donc bien le comportement souhaité : GUD détecte que les graphes sont proches même si la proportion d'arêtes en commun en faible. Pour $\mu_{intra} = 1$, chaque cluster est une clique, donc parfaitement en accord entre les deux graphes.

Sur la figure 3.2b, le nombre de liens aléatoires entre les clusters augmente jusqu'à ce que les deux graphes deviennent presque [4] complets. Au départ GUD détecte que les deux graphes sont très similaires (alors que GED indique que seulement la moitié environ des arêtes sont communes). Ensuite GED et GUD augmentent à cause du désaccord créé par les arêtes inter-cluster, mais arrivées à un certain seuil (environ 0.2) les deux mesures décroissent. En effet le nombre important des arêtes inter-clusters fait que ces arêtes commencent à être en accord. Les deux graphes convergent vers un même graphe quasiment complet.

4. En effet, comme $\mu_{intra} = 0.5$, les graphes ne sont jamais complets même quand $\mu_{inter} = 1$.



(a) μ_{intra} varie et $\mu_{inter} = 0.01$. **(b)** $\mu_{intra} = 0.5$ et μ_{inter} varie.

Figure 3.2 – Évaluation de GUD sur différentes paires de graphes CGM. Sur toutes les courbes, les deux graphes comparés sont issus du même modèle de graphe CGM avec $m = 50$ et $k = 5$. GUD est calculée avec $t = 4$. Chaque point donne la moyenne pour 20 réalisations, l'écart-type (faible) est donné par les barres verticales associés à chaque point (presque invisibles).

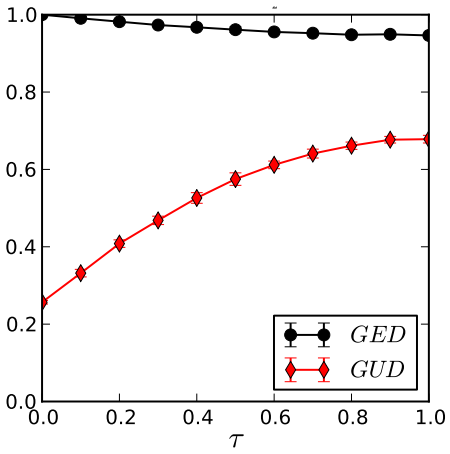
3.3.4.2 Cas (ii), (iii) et (iv) : graphes non similaires

Pour les différentes courbes de la figure 3.3 nous nous plaçons initialement dans le cas limite de la section 3.2.2. En effet les deux graphes comparés sont construits à partir d'un graphe CGM « coupé en deux ». C'est-à-dire la moitié de ses arêtes est sélectionnée au hasard pour former le premier graphe G_a ; les arêtes restantes forment le second graphe G_b. Ainsi ces deux graphes n'ont aucune arête en commun, pourtant les groupes de sommets (les clusters) sont les mêmes. Le graphe de départ est construit avec les paramètres suivants : $k = 5$, $m = 50$, $\mu_{intra} = 0.5$ et $\mu_{inter} = 0.01$. Ces deux graphes correspondent à des graphes CGM construits avec des probabilités μ_{intra} et μ_{inter} divisées par deux. Mais nous nous plaçons ainsi artificiellement dans le pire des cas, puisque les graphes n'ont aucune arête en commun. L'un de ces graphes (ou les deux) est alors transformé soit par un re-câblage aléatoire des arêtes, soit par une permutation des sommets.

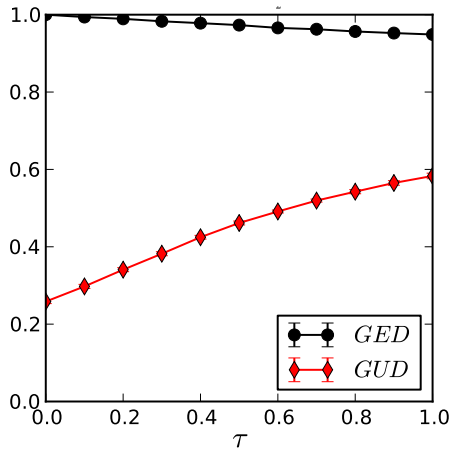
Permutation des sommets. Soit $X \subseteq V$ un ensemble de $(1-\tau).|V|$ sommets choisis aléatoirement et $\sigma : V \setminus X \mapsto V \setminus X$ une permutation aléatoire des sommets de V non présents dans X. On définit alors $f : V \mapsto V$ une bijection de V sur V telle que : si $u \in X$ alors $f(u) = u$ et si $u \notin X$ alors $f(u) = \sigma(u)$. On peut ainsi définir le graphe $\overset{\tau}{\bullet}G = (V, \overset{\tau}{\bullet}E)$ tel que :

$$\overset{\tau}{\bullet}E = \Big\{\{f(u), f(v)\},\ \{u, v\} \in E\Big\} \tag{3.11}$$

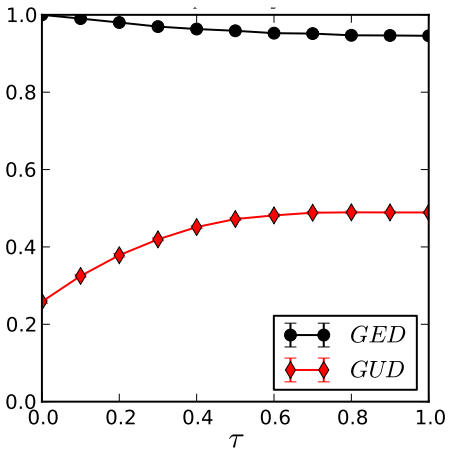
Pour tout $\tau \in [0, 1]$, $\overset{\tau}{\bullet}G$ est donc isomorphe à G par f. Si $\tau = 0$ alors $\overset{\tau}{\bullet}G = G$.



(a) G_a face à $\overset{\tau\bullet}{G}_b$. Une proportion τ des sommets de G_b est renommée aléatoirement.



(b) G_a face à $\overset{\tau\hookrightarrow}{G}_b$. Une proportion τ des arêtes de G_b est re-câblée aléatoirement.



(c) $\overset{\tau\hookrightarrow}{G}_a$ face à $\overset{\tau\hookrightarrow}{G}_b$. Une proportion τ des arêtes de chaque graphe est re-câblée aléatoirement.

Figure 3.3 – Évaluation de GUD sur différentes paires de graphes aléatoires. Sur toutes les courbes, pour $\tau = 0$ les deux graphes sont issus d'un graphe CGM « coupé en deux ». Ensuite l'un des graphes ou les deux sont transformés. Le graphe CGM de départ est construit avec les paramètres suivants : $k = 5$, $m = 50$, $\mu_{intra} = 0.5$ et $\mu_{inter} = 0.01$. GUD est calculée avec $t = 4$. Chaque point donne la moyenne pour 20 réalisations, l'écart-type (faible) est donné par les barres verticales associés à chaque point.

Re-câblage des arêtes. Soit $A \subseteq E$ un ensemble de $(1-\tau).|E|$ arêtes choisies aléatoirement, et B un second ensemble de $\tau.|E|$ paires de sommets choisies aléatoirement **et** non présentes dans A. C'est-à-dire $B \subseteq \mathbf{P}_2^V \setminus A$, avec $\mathbf{P}_2^V$ l'ensemble des parties de taille deux de V. On construit alors le graphe $\overset{\tau\hookrightarrow}{G} = (V, \overset{\tau\hookrightarrow}{E})$ tel que $\overset{\tau\hookrightarrow}{E} = A \cup B$. Alors $\overset{\tau\hookrightarrow}{G}$ a autant d'arêtes que G, mais $\tau.|E|$ de ses arêtes sont aléatoires, les $(1-\tau).|E|$ restantes sont communes. Quand $\tau = 1$ alors $\overset{1\hookrightarrow}{G}$ est un graphe aléatoire de type Erdős-Rényi ayant la même densité que G.

En figure 3.3a, l'un des graphes n'est pas transformé, l'autre subit une permutation des sommets. Sur la figure 3.3b, les arêtes de l'un des deux graphes sont re-câblées aléatoirement, l'autre graphe reste inchangé. Enfin pour la figure 3.3c, les arêtes des deux graphes sont re-câblées aléatoirement. Dans les trois cas l'axe des abscisses représente τ, c'est-à-dire soit la proportion de sommets inter-changés, soit la proportion d'arêtes re-câblées.

Analyse des résultats de la figure 3.3. Pour $\tau = 0$, c'est-à-dire quand on compare deux graphes CGM ayant les mêmes clusters mais aucune arête en commun ($GED = 1$), on observe que GUD vaut un peu plus de 0.2. La similarité de ces graphes initiaux est donc bien repérée. Ensuite on observe que sur les trois courbes la valeur de GUD augmente quand τ croît. C'est un comportement souhaitable car dans les trois cas la dose d'aléatoire introduite rend petit à petit les deux graphes « étrangers ». Notons qu'à l'inverse la valeur de GED diminue. Cela n'est pas surprenant, car au fil des transformations certaines arêtes deviennent *par hasard* en accord. C'est sur la figure 3.3a que la valeur de GUD devient la plus forte (supérieure à 0.6). Sur la figure 3.3b elle vaut environ 0.6 pour $\tau = 1$ et elle vaut environ 0.5 pour $\tau = 1$ sur la figure 3.3c. Ce comportement est souhaitable, en effet, pour $\tau = 1$ sur la figure 3.3a, les deux graphes ont des structures en cluster mais ces structures sont indépendantes l'une de l'autre, le désaccord coûte donc cher. Sur la figure 3.3b le désaccord est un peu plus faible, car sur le graphe devenu aléatoire, les arêtes manquantes sont moins chères à ajouter (confluence proche de 0.5). Lorsque les deux graphes sont aléatoires (figure 3.3c pour $\tau = 1$) les arêtes à ajouter ont toutes une confluence proche de 0.5, GUD est donc proche de 0.5.

3.3.4.3 Conclusion de l'évaluation

Nous avons montré que la mesure GUD se comporte bien comme attendu sur une série d'expériences avec des graphes artificiels. En particulier deux graphes comportant les mêmes clusters apparaissent bien comme similaires même si ces clusters ne sont pas formés des mêmes arêtes. Nous avons aussi pu vérifier que GUD indique correctement la dissimilarité de deux graphes soit aléatoires, soit ayant des clusters différents.

Cette évaluation pourrait être poursuivie sur des graphes aléatoires ayant des structures en clusters plus réalistes. En particulier en utilisant le modèle de graphes LFR proposé par Lancichinetti et al. [2008] pour comparer des méthodes de clustering de graphe. Ces graphes permettent en effet d'imposer une distribution des degrés en loi de puissance, ainsi qu'une répartition de la taille des clusters en loi de puissance.

3.4 Fusion de graphes de terrain

En présence de deux graphes sur les mêmes sommets, il peut être utile des les regrouper en un seul. En se basant sur cette idée consistant à ramener au niveau des arêtes (en conflit) une information sur la topologie du graphe, nous proposons ici une méthode permettant de fusionner deux graphes.

Lorsque l'on veut fusionner deux graphes ayant les mêmes sommets, deux opérations évidentes sont envisageables : calculer l'union ou l'intersection des arêtes. L'union correspond à une démarche optimiste : une arête est « acceptable » du moment qu'elle est présente dans l'un des deux graphes. À l'inverse, l'intersection est une approche pessimiste : une arête n'est « acceptable » que si elle est présente dans les deux graphes. Nous proposons une méthode intermédiaire qui consiste à accepter les arêtes présentes dans les deux graphes ainsi que les arêtes présentes seulement dans un des graphes si la similarité de la paire correspondante dans l'autre graphe est supérieure à un certain seuil. Ce seuil peut être différent dans chacun des graphes. On note E^{τ_1,τ_2} l'ensemble d'arêtes ainsi fusionnées :

$$E^{\tau_1,\tau_2} = (E_1 \cap E_2) \cup F_1^{\tau_1} \cup F_2^{\tau_2} \tag{3.12}$$

avec :

$$F_1^{\tau_1} = \left\{ \{u,v\} \in E_2 \cap \overline{E_1} |\ \text{confl}_{t,G_1}(u,v) > \tau_1 \right\} \tag{3.13}$$

$$F_2^{\tau_2} = \left\{ \{u,v\} \in E_1 \cap \overline{E_2} |\ \text{confl}_{t,G_2}(u,v) > \tau_2 \right\} \tag{3.14}$$

Il est facile de vérifier que par construction, pour tout $\tau_1, \tau_2 \in [0,1]$, on a :

$$E_1 \cap E_2 \subset E^{\tau_1,\tau_2} \subset E_1 \cup E_2 \tag{3.15}$$

Le graphe ainsi construit contient toutes les arêtes présentes dans G_1 et G_2, et une partie de celles présentes uniquement dans un seul des graphes.

Notons qu'il est possible d'imaginer un mécanisme de fusion analogue consistant à retirer de l'union des deux graphes les paires présentes seulement dans un graphe et ayant une similarité en dessous d'un certain seuil (dans ce graphe).

3.5 Application à la comparaison de ressources lexicales

Nous présentons dans cette section comment la mesure GUD peut être appliquée pour comparer des ressources lexicales, en particulier des réseaux de synonymie. Nous montrons que des différences « apparentes » entre des ressources, c'est-à-dire un GED fort, peuvent être rattrapées lorsque ces ressources sont comparées avec GUD. Ce qui signifie que ces ressources même si elles n'utilisent pas les mêmes arêtes définissent la synonymie de manière semblable.

3.5.1 Introduction des ressources comparées

Pour appliquer notre méthode nous comparons six paires de graphes de synonymie. Ces graphes sont issus de deux ressources pour le français et de deux ressources pour l'anglais. Un graphe est extrait de chacune de ces ressources pour chaque *partie du discours* (noté POS par la suite pour *Part Of Speech* en anglais). À savoir un graphe pour les noms (noté N par la suite), un pour les verbes (V) et un pour les adjectifs (A). Nous présentons ici ces ressources et le tableau 3.3 résume les pedigrees de ces graphes.

ROB et LAR. *Robert* et *Larousse*, ce sont deux dictionnaires de synonymie du français. Ils font partie des sept dictionnaires de synonymes papier numérisés pour créer DicoSyn. Ces numérisations ont été réalisées par l'INALF (aujourd'hui ATILF [5]). Chaque sommet correspond à un lemme (et à une POS), il n'y a pas de sous-sens indiqué. Deux sommets sont adjacents si au moins l'un des deux apparaît dans la liste des synonymes de l'autre. Les graphes sont donc non-dirigés.

PWN. WordNet [Fellbaum, 1998] de l'Université de Princeton, est la ressource lexicale de référence pour l'anglais. Elle est organisée en *synsets* liés par diverses relations. Un *synset* étant un ensemble mots synonymes partageant un sens ou un usage particulier. Les graphes construits à partir de WordNet sont tels que, pour chaque POS, deux lemmes sont liés s'ils apparaissent ensemble dans au moins un *synset*.

ROG. Pour *Roget thesaurus* [6], c'est une ressource créée au XIX^{e} siècle par le lexicographe Peter Mark Roget. Les mots sont organisés en une hiérarchie de classes. Nous avons construit un graphe par POS en liant les mots apparaissant ensemble dans les feuilles de cette arborescence.

5. http://www.atilf.fr/
6. http://www.gutenberg.org/ebooks/10681

Table 3.3 – Pedigree des 12 graphes de synonymie. n et m sont l'ordre et la taille des graphes, $\langle k \rangle$ le degré moyen, **c** le coefficient de clustering, λ est le coefficient de la loi de puissance calculée par régression linéaire sur la distribution des degrés, r^2 est le coefficient de corrélation de cette régression. n_{ccp} et m_{ccp} sont l'ordre et la taille de la plus grande composante connexe et $\mathbf{l}_{ccp}$ est la longueur moyenne des plus courts chemins sur cette plus grande composante connexe.

Graphes		n	m	<k>	**c**	λ (r^2)	n_{ccp}	m_{ccp}	$\mathbf{l}_{ccp}$
LAR	*A*	5510	21147	7.68	0.2099	-2.06 (0.88)	5011	20826	4.92
	N	12159	31601	5.20	0.2023	-2.39 (0.88)	9802	30091	6.10
	V	5377	22042	8.20	0.1728	-1.94 (0.88)	5193	21926	4.61
ROB	*A*	7693	20011	5.20	0.1446	-2.05 (0.94)	6306	19103	5.26
	N	24570	55418	4.51	0.1142	-2.34 (0.94)	20785	53006	6.08
	V	7357	26567	7.22	0.1192	-2.01 (0.93)	7056	26401	4.59
PWN	*A*	21479	22164	2.06	0.6859	-2.82 (0.92)	4406	11276	9.08
	N	117798	104929	1.78	0.6894	-3.27 (0.95)	12617	28608	9.89
	V	11529	23019	3.99	0.4722	-2.46 (0.90)	6534	20806	5.93
ROG	*A*	13193	30078	4.56	0.5644	-2.39 (0.91)	6325	23466	6.35
	N	29793	83581	5.61	0.6551	-2.45 (0.87)	15331	68476	6.51
	V	15258	40591	5.32	0.6066	-2.39 (0.87)	7996	33465	6.09

3.5.2 Analyse des comparaisons

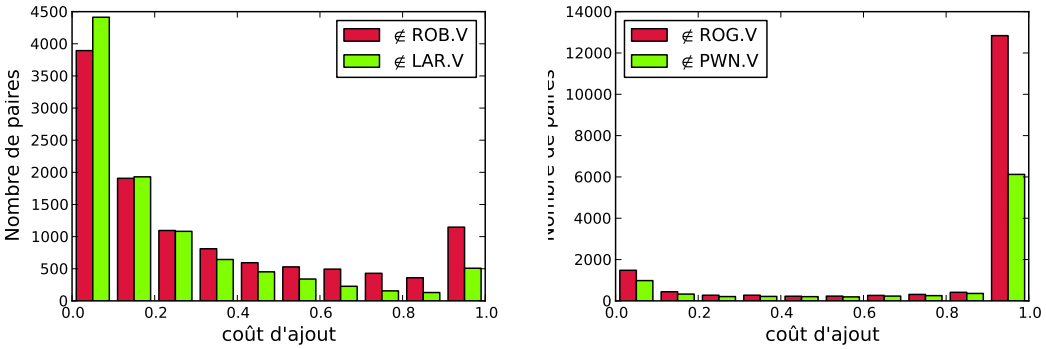
Pour comparer ces réseaux, la première étape est de les réduire à leur ensemble de sommets communs. En effet les couvertures lexicales des ressources ne sont pas identiques. Un certain nombre de mots ne sont présents que dans l'un des deux graphes. Le tableau 3.4 donne les valeurs de GED, GUD et GUC pour les six paires de graphes comparées. Le cardinal de l'ensemble des sommets communs est aussi donné.

On observe que ROB et LAR sont relativement distants comparés avec GED, ils n'ont de fait qu'un peu plus de la moitié de leurs arêtes en commun. Cela est surprenant pour ces deux ressources d'origine similaire, mais le point intéressant est que le score GUD est très faible. Effectivement, comme l'indique GUC, le coût des conflits entre les deux graphes est faible. C'est-à-dire les arêtes présentes dans l'un et non dans l'autre des graphes, sont formées de sommets proches dans les deux graphes.

Les valeurs observées entre ROG et PWN sont différentes. L'accord mesuré avec GED est en effet très important, et il ne varie pas énormément avec GUD. Ces deux ressources ont donc peu d'arêtes en commun, et ces arêtes sont coûteuses à ajouter. Le contexte et la méthode de création de ces deux ressources expliquent certainement ces différences.

Table 3.4 – Valeurs de GED et de GUC entre différentes paires de réseaux de synonymie. GUD et GUC sont calculés avec des marches aléatoires de temps $t = 4$.

Graphes		$\lvert V_a \cap V_b \rvert$	GED	GUD	GUC
ROB / LAR	A	4809	0.45	0.11	0.24
	N	10881	0.48	0.15	0.32
	V	4973	0.48	0.13	0.28
ROG / PWN	A	6853	0.87	0.79	0.91
	N	16576	0.89	0.84	0.95
	V	5459	0.85	0.70	0.82



(a) ROB face à LAR pour les verbes **(b)** ROG face à PWN pour les verbes

Figure 3.4 – Distributions des coûts d'ajout (calculés avec la confluence) des paires de sommets en conflit. Les coûts sont calculés avec des marches aléatoires de temps $t = 4$.

Pour comprendre plus en détail ces comparaisons la figure 3.4 présente l'histogramme de répartition du coût des arêtes manquantes dans l'un ou l'autre des graphes. Pour dix intervalles de coût, les barres verticales indiquent le nombre d'arêtes ayant un coût d'ajout dans cet intervalle. Chaque histogramme (d'une couleur donnée) indique donc la répartition des paires de sommets manquantes dans l'un ou l'autre des graphes en fonction de leur coût d'ajout. Les résultats sont présentés pour les graphes entre verbes, les histogrammes sont très similaires pour les autres parties du discours.

On vérifie bien qu'une large part des paires en désaccord entre ROB et LAR ont un coût d'ajout faible (< 0.3). Dans les deux cas un certain nombre de paires ont un coût entre 0.95 et 1. Ce sont les paires de sommets réellement en désaccord. Entre ROG et PWN, la majorité des paires sont dans ce cas. Mais on observe aussi qu'entre ROB et LAR, il existe un nombre significatif de paires dans ce cas. Une large part de ces fortes dissimilarités viennent de paires de sommets non-connectés dans l'un des deux graphes.

3.6 Conclusion du chapitre

Nous avons dans ce chapitre introduit une méthode de comparaison de graphes partageant le même ensemble de sommets. Cette méthode est robuste dans le sens où deux graphes n'ayant aucune arête en commun mais tels que les sommets sont proches de la même manière dans chacun d'eux seront considérés comme similaires alors qu'une mesure basée simplement sur le nombre d'arêtes communes les indiquerait comme complètement dissemblables. Deux graphes ayant exactement les mêmes clusters mais construits par des arêtes en partie différentes sont typiquement dans ce cas. Notre méthode généralise une distance d'édition entre graphes en calculant les coûts d'édition à partir de la mesure de confluence définie au chapitre précédent.

Le comportement de la méthode est vérifié sur différentes configurations de graphes aléatoires. Et une application pour comparer des réseaux de synonymie est présentée. Cette application permet en particulier de montrer que certains graphes de synonymie ayant environ seulement la moitié de leurs arêtes en commun, sont pourtant très proches.

Différentes pistes de travaux futurs sont envisageables. Tout d'abord, l'évaluation de la méthode peut être étendue à des graphes artificiels plus réalistes. En particulier il serait intéressant d'étudier le comportement de GUD sur des graphes artificiels présentant un recouvrement entre les clusters ou une distribution des degrés plus hétérogène. Par exemple en utilisant le modèle LFR [Lancichinetti et al., 2008].

Aussi, d'autres mesures de similarité peuvent être utilisées pour calculer les coûts d'édition. La confluence a l'avantage d'avoir une complexité raisonnable et d'être normalisée par rapport à un *null model*. Mais l'utilisation de certaines méthodes globales (le *PageRank personnalisé* par exemple) permettrait d'être plus robuste sur des graphes peu denses. En effet deux sommets peuvent avoir un score de confluence nul sur un graphe peu dense, uniquement parce qu'ils ne sont connectés que par des chemins « trop longs » par rapport à la longueur des marches aléatoires utilisées.

Aussi il est envisageable d'utiliser une méthode de clustering pour évaluer les coûts d'ajout. Mais plusieurs problèmes se posent alors. Tout d'abord si le graphe est « trop » dense alors les méthodes de clustering ont tendance à ne faire qu'un seul cluster, les sommets deviennent alors tous similaires ce qui n'est pas souhaitable. Enfin un second problème est que la plupart des méthodes de clustering ne gèrent pas de recouvrement. Et donc un sommet présent dans deux clusters ne sera similaire qu'avec les sommets d'un seul de ces clusters.

Enfin l'application sur les réseaux lexicaux peut être poursuivie de plusieurs manières. Il est possible d'utiliser notre méthode pour évaluer des ressources lexicales construites automatiquement. En effet il est possible que deux ressources ayant une même valeur de GED par rapport à un étalon aient des valeurs de GUD très différentes. La méthode peut aussi être employée pour détecter les « zones » réellement différentes entre deux ressources, permettant ainsi de guider une analyse d'erreur.

Aussi la méthode de fusion que nous avons présentée pourrait être employée. Il est possible notamment d'imaginer évaluer cette méthode de fusion sur une tâche donnée. Il s'agirait de mesurer si un système de traitement automatique du langage (TAL) donne de meilleurs résultats en utilisant directement l'union ou l'intersection de deux graphes ou en utilisant le graphe fusionné avec notre méthode.

Chapitre 4

Enrichissement semi-automatique de réseaux lexicaux

Nous proposons dans ce chapitre un système endogène et semi-automatique d'enrichissement de ressources lexicales. Ce système repose sur une mesure de similarité entre sommets d'un graphe, semblable à celles introduites au chapitre 2.

Les ressources lexicales sémantiques sont nécessaires pour beaucoup de systèmes de traitement automatique du langage naturel (TAL). Pour autant alors que le traitement de l'anglais est doté de WordNet [Fellbaum, 1998], de l'Université de Princeton (ci-après noté PWN), ressource éprouvée depuis de nombreuses années, plusieurs langues telles que le français ne bénéficient encore d'aucune ressource de qualité satisfaisante. « *We desperately need linguistic resources !* » écrit Sekine [2010], soulignant qu'il n'est pas réaliste de penser qu'une seule institution pourra développer des ressources à large échelle, qu'une collaboration est donc nécessaire et que partager les ressources est crucial. La première difficulté s'opposant à la construction de telles ressources découle des modalités de développement et du compromis coût/qualité qui en résulte. Recourir à des experts pour construire manuellement des ressources coûte cher. Par ailleurs, on ne peut préjuger de la qualité des ressources construites automatiquement (donc bruitées), qui devraient être validées par des experts, ce qui ramène au problème initial. Enfin, dans le cas où le recours à la validation par des experts est envisageable, la mise en place d'une mesure d'accord entre ces experts est problématique. C'est à partir de ce constat que nous développons un système semi-automatique d'enrichissement de réseaux lexicaux. Ce système permet de lister des « candidats synonymes » pour chaque mot d'une ressource en construction. Le contributeur peut alors valider ou invalider ces candidats. L'ajout de liens de synonymie à la ressource est ainsi accéléré, car elle se résume maintenant en une simple validation par l'utilisateur. Ce système peut soit permettre d'accélérer une construction collaborative, soit assister des lexicographes afin de réduire le coût d'élaboration d'une ressource. Nous explorons ici la première piste en présentant

une mise en œuvre du système sur le dictionnaire collaboratif *Wiktionary*.

Après avoir décrit, en section 4.1, les tentatives antérieures de construction de ressources et dressé un inventaire des différentes méthodes d'extraction de relations lexico-sémantiques, nous nous intéresserons tout particulièrement aux difficultés liées à l'évaluation des ressources. Nous présentons, en section 4.2, de nouvelles tendances fondées sur l'édition collaborative qui constitue une piste intéressante pour la construction de ressource. Dans ce cadre, *Wiktionary*, un dictionnaire libre disponible en ligne, nous paraît être une clé pour régler simultanément le problème du coût de développement et, dans une certaine mesure, celui de l'évaluation. Nous présentons, en section 4.3, un processus d'enrichissement semi-automatique visant à densifier les réseaux de synonymie extraits de cette ressource. Nous mesurons l'impact de l'utilisation de différentes sources de données sur ce processus en section 4.3.3. Enfin en section 4.4, nous présentons *Wisigoth* une implémentation de notre système.

Notons que ce travail a été présenté dans les publications suivantes : [Sajous et al., 2011a,b], et deux étapes intermédiaires ont auparavant été publiées dans : [Navarro et al., 2009; Sajous et al., 2010].

4.1 Ressources lexicales, construction et évaluation

PWN est probablement le seul projet de construction d'une ressource lexicale sémantique à connaître un tel succès et à être aussi largement utilisé. D'autres projets suscités par cette réussite, tels qu'EuroWordNet [Vossen, 1998] et BalkaNet [Tufis, 2000], prévoyaient une couverture moins ambitieuse. Malheureusement, ces ressources se sont figées dès la fin de leur développement initial (alors que PWN continue d'évoluer). Jacquin et al. [2007] ont pointé les faiblesses de la partie française d'EuroWordNet et ont proposé des méthodes automatiques pour ajouter des relations manquantes. Ces méthodes, comme celles que nous énumérons en section 4.1.1, bien qu'intéressantes, nécessiteraient une validation manuelle par des experts (donc coûteuse) pour produire une ressource fiable.

Les problèmes de temps et coût de développement, ainsi que de disponibilité des ressources, sont de plus en plus pris en compte : en linguistique de corpus, par exemple, une méthode « AGILE », empruntée à la gestion de projets, a été proposée par Voormann et Gut [2008] pour permettre simultanément de maximiser la taille d'un corpus et de ses annotations tout en réduisant le temps et le coût de son développement. Brunello [2009] s'est intéressé au problème de disponibilité et propose une méthode pour construire des corpus libres en recourant au web et aux méta-données qui identifient des pages web sous licence libre.

4.1.1 Méthodes de construction de ressources lexicales

Proposées initialement par Hearst [1992] pour repérer en corpus les relations d'hyperonymie, les approches par patrons lexico-syntaxiques ont été affinées par Pantel et Pennacchiotti [2006] pour réduire l'intervention manuelle nécessaire. De tels patrons ne sont pas toujours aisés à mettre au point et sont par ailleurs spécifiques à la langue étudiée.

Des méthodes d'enrichissement mutuel de ressources par liens interlangues consistent à projeter des relations sémantiques en suivant des liens de traduction. Sagot et Fišer [2008] ont construit WOLF, un WordNet *libre* du français, en utilisant plusieurs ressources existantes pour amorcer d'une part une base de concepts (basés sur la synonymie) français et anglais et pour construire un index interlangue à partir duquel les projections ont été opérées. Un travail similaire mais utilisant Wiktionary a été proposé par Hanoka et Sagot [2012]. De même, Soria et al. [2009] ont utilisé des liens de traduction pour effectuer une « fertilisation mutuelle automatique » de deux lexiques, italien et chinois, ayant une structure similaire à celle de PWN. Ces expériences, comparables à ce que nous proposons dans la suite de ce chapitre, ont le mérite de prouver la faisabilité de la méthode, mais ne permettent pas de produire des ressources fiables en l'absence de validation manuelle.

De nombreux travaux ont enfin porté sur la mise au point de calculs de *similarité sémantique* : différentes propriétés sont associées aux mots et la propension à partager des propriétés communes est interprétée comme le signe d'une proximité sémantique. La plupart de ces méthodes reposent sur des modèles vectoriels : les mots sont représentés par des vecteurs dont chaque coordonnée représente une propriété du mot et la similarité sémantique entre mots est donnée par un calcul de similarité entre ces vecteurs. Les méthodes diffèrent ensuite par les propriétés qui constituent les vecteurs et les algorithmes de calcul de similarité. Un mot peut être représenté, par exemple, par un vecteur contenant ses cooccurrents en corpus ou ses contextes syntaxiques. Heylen et al. [2008] utilisent la similarité entre vecteurs pour extraire des relations de synonymie et comparent les résultats obtenus en utilisant des sacs de mots et des contextes syntaxiques. Ils étudient également l'impact de certaines propriétés linguistiques (fréquence, spécificité et classes sémantiques) sur la similarité ainsi calculée : les contextes syntaxiques s'avèrent plus efficaces que les sacs de mots, les meilleurs résultats sont obtenus avec les termes de haute fréquence et appartenant à des classes sémantiques abstraites. L'effet de la spécificité sémantique n'est pas probant. Ils montrent également que les vecteurs rapprochés par leur méthode n'ayant pas trait à la synonymie reflètent souvent d'autres relations sémantiques (cohyponymie et hyperonymie/hyponymie). Des comparaisons de différentes mesures et pondérations de contextes syntaxiques sont données dans Curran et Moens [2002]. Van der Plas et Bouma [2005] étudient quels contextes syntaxiques particuliers mènent aux meilleurs résultats, observant par exemple qu'une meilleure

précision est obtenue avec la relation *objet* qu'avec la relation *modifieur*.

Ces méthodes peuvent également être utilisées avec des propriétés non-linguistiques. Ollivier et Senellart [2007] et Weale et al. [2009] par exemple utilisent la structure hypertextuelle de Wikipédia et de Wiktionary comme objet de leur calcul de similarité. Ces méthodes peuvent donner de bons résultats sur certaines tâches (78 à 93% sur des tests de type TOEFL) mais ne sont pas reproductibles en dehors de ces architectures spécifiques (e.g. réseaux classiques de type WordNet en cours de construction).

L'approche que nous présentons par la suite (section 4.3) est assez proche de ces dernières méthodes. La première des différences est que nous utilisons un modèle de graphe biparti pondéré plutôt qu'un modèle vectoriel (ce n'est qu'une différence de point de vue, mais qui permet d'imaginer utiliser les similarités « globale » et « semi-globale » vues au chapitre 2). Une seconde différence est que nous proposons une méthode *endogène*[1], c'est-à-dire que la ressource elle-même est utilisée pour calculer les similarités permettant de l'enrichir. Aussi, comme nous allons le voir, nous proposons un processus de construction semi-automatique.

4.1.2 Évaluation des ressources lexicales

Bien qu'ayant donné lieu à de nombreux travaux, le problème de l'évaluation des ressources construites automatiquement reste ouvert. Une approche traditionnelle est la comparaison à une ressource étalon. En supposant l'existence et la disponibilité d'une telle ressource, cette dernière a ses limites et donc l'évaluation qui en découle également. L'étalon choisi peut en effet avoir été conçu dans une optique particulière (comme c'est le cas pour les dictionnaires) et ne refléter qu'un aspect de la réalité. Il n'est donc nécessairement ni générique, ni approprié à toute évaluation. Un étalon doit être lui-même évalué ou caractérisé avant d'être choisi pour une évaluation donnée. Nous avons montré au chapitre précédent (section 3.5.1) qu'il n'y a pas nécessairement de consensus entre les ressources de référence, en particulier si on les considère comme des « sacs de liens ». Ainsi, le choix de la ressource étalon influence largement une évaluation. Les résultats obtenus ne permettent donc pas de tirer des conclusions définitives : si, par exemple, un système propose deux mots comme synonymes et qu'ils n'apparaissent pas comme tels dans l'étalon, le système est-il en défaut ou est-ce dû au caractère non-exhaustif de l'étalon ?

Une évaluation manuelle peut également consister à extraire un échantillon de relations et à les étiqueter manuellement comme correctes ou incorrectes. Cependant, un tel jugement est entâché de la subjectivité liée notamment à une représentation du monde différente pour chaque annotateur et mène souvent à un accord inter-juges faible. En cas d'accord satisfaisant, celui-ci est régulièrement utilisé comme

1. Mis à part lorsque pour les contextes syntaxiques issus de Wikipédia.

seul critère de qualité alors que rien ne garantit la pertinence des jugements rendus. Murray et Green [2004] ont analysé les facteurs corrélés à des taux d'accords faibles sur une tâche de désambiguïsation lexicale et ont montré que les accords élevés sont corrélés à des compétences lexicales *comparables* entre les annotateurs et non à des compétences *élevées*. Deux juges « naïfs » sont susceptibles d'obtenir le même taux d'accord que deux experts ; ajouter un expert à un groupe de naïfs tend à diminuer ce taux. Ils concluent que la mesure de l'accord inter-annotateurs seule ne constitue pas un critère de qualité d'une annotation et qu'elle doit être couplée à une mesure de compétence par rapport à l'annotation envisagée.

Enfin, on peut comparer plusieurs ressources via une évaluation par la tâche en mesurant les performances d'un système utilisant les différentes ressources pour réaliser une tâche donnée. Des ressources lexicales sémantiques peuvent être évaluées par exemple, dans des tâches de recherche d'information, de traduction automatique, de WSD [2], etc. Pour estimer les performances de ce système, le processus d'évaluation doit pouvoir déterminer, pour une entrée donnée, quelle sortie doit être produite. Ce problème nécessite alors la construction d'un étalon et soulève les mêmes problèmes que ceux mentionnés plus haut. Par exemple, Kilgarriff [1998] a montré les difficultés rencontrées dans la préparation d'un étalon pour la campagne SENSEVAL.

Enfin il faut noter le rôle central de la granularité notamment dans la modélisation de la synonymie. Dans une tâche de WSD, Palmer et al. [2007] ont montré qu'un regroupement des sens d'un dictionnaire électronique permettait de régler des désaccords « à la marge » entre annotateurs. De manière générale, l'accord augmentait de 10 à 20% lorsqu'il était mesuré sur une annotation du système utilisant une granularité plus grosse. Cependant, ils soulignent qu'atteindre des taux d'accords très élevés avec des mots fortement polysémiques est un objectif non réaliste. Ils notent également qu'accroître cet accord n'est pertinent que si cela n'affecte pas ou peu la qualité résultante des applications de TAL.

4.2 Édition collaborative par les foules

Depuis la création de Wikipédia, l'exactitude des ressources construites collaborativement « par les foules » a été mise en doute. Wikipédia était initialement la seule ressource de ce type et la question de son bien-fondé a mené à une polémique : Giles [2005] a prétendu que l'encyclopédie en ligne était aussi précise que l'encyclopédie Britanica, qui a pour sa part réfuté les critères d'évaluation utilisés. Depuis, les wikis et les ressources collaboratives se sont multipliés. Plus modérés que Giles, Zesch et Gurevych [2010] ont montré, à travers une tâche de mesure de similarité sémantique, que les ressources fondées sur « la sagesse des foules » ne sont pas

2. WSD signifie *Word-sense disambiguation*, voir par exemple [Manning et Schütze, 1999, chap. 7]

meilleures que celles fondées sur « la sagesse des linguistes », mais sont sérieusement compétitives. Elles dépassent même les ressources construites par les experts dans certains cas, notamment en termes de couverture.

Dans cette section, nous présentons les différents modèles de construction « par les foules » de ressources lexicales. Nous présentons ensuite une description détaillée de Wiktionary.

4.2.1 Plusieurs modèles

Au problème de la qualité du contenu s'ajoute celui de la participation « des foules ». En effet, les approches collaboratives n'impliquent pas la seule participation de collègues et d'étudiants mais doivent mobiliser une plus large population ne partageant pas nécessairement les motivations des chercheurs. Plusieurs tendances récentes refaçonnent le domaine de la création de ressources. Parmi elles, quelques chercheurs ont mis au point des *jeux en ligne* destinés à attirer des internautes « ordinaires » qui jouent pour le plaisir indépendamment de la finalité première du jeu. C'est le cas de JeuDeMots, développé par Lafourcade [2007], qui a réussi à collecter un nombre important de relations sémantiques (essentiellement non typées, mais également des relations telles qu'hyperonymie et méronymie). Cependant, réaliser un jeu suffisamment distrayant est une tâche difficile en soi et il est probable que certaines tentatives échouent si elles ne produisent pas des jeux suffisamment attractifs.

Le système « *Mechanical Turk* », créé par Amazon (AMT), consiste à définir des micro-tâches que des travailleurs (*turkers*) accomplissent contre une rétribution minimale. Ces tâches, ou HITs (*human intelligence tasks*), permettent d'effectuer ce dont un système informatique est incapable : étiqueter une image, exprimer sa préférence pour une couleur, etc. Utilisé généralement dans le commerce électronique, ce système a également servi à juger la qualité des articles de Wikipédia en recueillant les appréciations d'un groupe d'utilisateurs. Des chercheurs ont également eu recours à ce modèle pour mettre en place des systèmes d'évaluation. Snow et al. [2008], par exemple, ont évalué les performances d'annotateurs naïfs utilisés comme turkers dans des tâches telles que désambiguïsation lexicale, jugement de similarité sémantique, annotation de sentiments dans les documents textuels, etc. Ils ont obtenu, dans 5 tâches sur 7, le résultat contre-intuitif qu'un système entraîné sur les annotations des turkers surpasse celui entraîné par un expert unique. Ils expliquent ce résultat par le fait que le jugement de plusieurs naïfs permet de corriger le biais introduit par l'unicité d'un annotateur, fût-il expert. Le modèle d'Amazon a montré qu'il pouvait être approprié à des tâches d'annotation. Il faut néanmoins accueillir ce constat avec mesure et garder à l'esprit qu'AMT a été conçu pour accomplir des tâches courtes et ne doit être utilisé que pour ces tâches-là. Dans le cas contraire, les turkers peu scrupuleux peuvent être tentés d'abuser le système en passant un minimum de temps sur chaque tâche et en donnant des réponses au hasard. Par

ailleurs, même avec des « turkers honnêtes », vérifier la compétence d'un annotateur pour une tâche donnée peut être nécessaire (*cf.* section 4.1.2). Notons enfin que, comme le soulignent Adda et al. [2011], ces systèmes de *Mechanical Turk* posent des questions sociales et éthiques légitimes.

La mise en place d'infrastructures sophistiquées et coûteuses dans la perspective d'une édition « par les foules » présente le risque d'aboutir à des coquilles vides attendant indéfiniment d'être remplies. En effet, dans le contexte actuel de compétition pour attirer les internautes, les plateformes dépourvues de contenu, aussi prometteuses soient-elles, n'attirent personne. Toute solution envisagée qui sous-estime la contrainte d'attractivité est vouée à l'échec. Seules quelques initiatives collaboratives et quelques réseaux sociaux ont réussi à subsister et concentrent la majorité des internautes. Le simple fait de s'adosser à l'une de ces « *success stories* » du web laisse espérer une multitude de visiteurs. Wikipédia et Wiktionary sont certainement parmi les meilleurs exemples de réussite et la communauté du TAL peut apporter ses compétences et outils à leurs utilisateurs, bénéficiant en retour d'un nombre important de contributeurs. C'est dans cette optique de « *piggybacking* » que s'inscrivent les travaux présentés ici. En s'appuyant sur l'architecture de Wiktionary et son contenu déjà existant, nous proposons aux contributeurs des synonymes calculés automatiquement afin qu'ils les valident. Nous espérons ainsi aider ces contributeurs à accélérer l'enrichissement du dictionnaire tout en profitant de son infrastructure pour établir des méthodes également applicables à d'autres types de ressources.

4.2.2 Le cas de *Wiktionary*

Wiktionary, le *« compagnon lexical de Wikipédia »*, est un dictionnaire multilingue libre et accessible en ligne. Comme les autres satellites de la Wikimedia Foundation, c'est un projet collaboratif : n'importe quel internaute peut contribuer et ses modifications sont publiées immédiatement. Chaque article peut mentionner des informations sur l'étymologie, contenir des gloses, exemples, traductions et des relations sémantiques. Du point de vue du TAL, Wiktionary peut sembler un terrain de jeu idéal et la couverture lexicale apparente renforce cette impression. Nous présentons ci-après un examen plus approfondi qui nuance des propos souvent (trop ?) enthousiastes.

4.2.2.1 Encodage et structuration des données

Précisons tout d'abord qu'il existe, comme pour Wikipédia, un Wiktionary par langue. Et ces différents Wiktionary sont, a priori, indépendants et chacun contient des informations concernant le lexique de potentiellement n'importe quelle langue (ces informations étant données dans la langue du Wiktionary). Ainsi, par exemple, le Wiktionary anglais contient des mots français définis en anglais. Notons que le

Wiktionary français se nomme « Wiktionnaire ».

Comme tous les projets de la Wikimedia Foundation, les Wiktionary utilisent un système de gestion de contenu appelé *MediaWiki*[3]. Le contenu des articles est enregistré dans un langage nommé *code wiki*. Malheureusement, ce langage ne vient pas avec une spécification formelle, et des déviances par rapport au langage standard (supposé) sont fréquentes. En analysant qualitativement les modifications des contributeurs, nous avons remarqué un nombre significatif d'erreurs dans les articles dûes à une incompréhension ou à un non-respect de la syntaxe. Nous pressentons également l'effet, peut-être plus dommageable, que nombre d'utilisateurs ne deviendront jamais contributeurs seulement parce qu'ils resteront rebutés par cette syntaxe.

Un article contient potentiellement plusieurs sections de langues, la première étant celle de l'édition du Wiktionary consultée. Une section de langue peut contenir plusieurs sections liées à une catégorie syntaxique donnée. Dans une telle section, on peut trouver des gloses et des exemples, parfois répartis selon plusieurs sens. Ensuite viennent les traductions et les relations sémantiques.

Une grande variation existe cependant par rapport à ce cas prototypique. Chaque Wiktionary a ses propres conventions et au sein d'une langue donnée (c'est-à-dire pour un Wiktionary), les conventions écrites ne sont pas toujours respectées. La notion de *flexibilité* étant revendiquée comme une propriété intrinsèque des projets de la Wikimedia Foundation, l'analyse automatique du contenu de Wiktionary n'est pas tâche aisée. Elle l'est encore moins lors de l'analyse des fichiers historiques retraçant la chronologie des articles (*i.e.* contenant toutes leurs versions datées), lorsqu'à la fois syntaxe et conventions évoluent avec le temps.

4.2.2.2 Nomenclature et couverture lexicale

Le Wiktionary anglais fait état[4] de *« 2 038 436 entries with English definitions from over 400 languages »* et la version française de *« 1 821 097 articles [qui] décrivent en français les mots de plus de 700 langues »*. Ces chiffres impressionnants sont néanmoins à tempérer : les méta-articles (pages d'aide, de discussion, etc.) sont comptés comme entrées et nombre de mots étrangers (à la langue du Wiktionary considéré) sont curieusement inclus dans la nomenclature (e.g. *lexicon* apparaît comme une entrée anglaise dans le Wiktionary français). Les formes fléchies font l'objet d'articles quand on pourrait les attendre dans les articles consacrés à leur forme citationnelle. Alors que dans l'édition française, les locutions sont catégorisées comme telles, l'édition anglaise les classe sous la catégorie de leur tête (voire une autre catégorie), ce qui gonfle artificiellement le nombre de mots annoncé (e.g. *« caught on the hop »* apparaît comme un verbe standard).

3. http://www.mediawiki.org
4. Ces chiffres datent de 2010.

Table 4.1 – Couverture lexicale du Wiktionnaire (2011) et de Morphalou (1.0) par rapport aux lexiques extraits des corpus Frantext, Le Monde et Wikipédia

	Taille de la nomenclature			Couverture des vocabulaires des corpus (%)								
				Frantext			Le Monde			Wikipédia		
	Morph.	Wikt.	∩	Morph.	Wikt.	M∪W	Morph.	Wikt.	M∪W	Morph.	Wikt.	M∪W
N.	41 005	134 203	29 604	76,4	80,6	84.4	47,3	54,1	58,1	23,5	26,7	31,6
V.	7 384	18 830	6 964	84,2	86,5	87.1	75,1	80,0	80,8	66,3	71,5	72,2
Adj.	15 208	42 263	10 014	88,9	84,6	94.0	78,9	76,8	88,1	73,9	72,4	84,7

Malgré des catégorisations surprenantes et des imports automatiques massifs (la moitié des noms sont des gentilés, *cf.* section 4.2.2.3), le nombre de mots contenus dans les éditions française et anglaise de Wiktionary reste élevé. Nous avons évalué sa couverture lexicale effective par rapport à celle du Trésor de la Langue Française informatisé (TLFi). Nous avons pour cela examiné le recouvrement entre les noms, verbes et adjectifs (formes canoniques) du dictionnaire collaboratif par rapport à ceux présents dans Morphalou [5]. Nous voyons dans le tableau 4.1 que le Wiktionnaire comprend trois-quart des noms du TLFi, quasiment tous ses verbes et deux tiers de ses adjectifs.

Nous avons également évalué la proportion couverte par chaque dictionnaire du vocabulaire des trois corpus suivants : 30 millions de mots issus de 515 romans du XX^{e} siècle de la base Frantext, 200 millions de mots issus des articles du quotidien Le Monde (période 1991 à 2000) et 260 millions de mots extraits de l'encyclopédie Wikipédia (version 06/2008). Nous avons lemmatisé et catégorisé ces corpus avec *TreeTagger* [6], puis gardé les lemmes catégorisés dont le nombre d'occurrences est supérieure ou égale à 5. Nous observons que les deux lexiques suivent la même tendance : le vocabulaire de Frantext est mieux couvert que celui du Monde, lui-même mieux couvert que celui de Wikipédia. La moindre couverture pour Wikipédia peut s'expliquer notamment par la diversité des domaines couverts et des rédacteurs, ainsi que par la présence des mots étrangers. Le faible pourcentage de noms couverts peut s'expliquer par le fait que les articles de l'encyclopédie comportent de nombreux mots isolés inconnus de *TreeTagger* qu'il étiquette souvent comme des noms communs. On observe sur les trois corpus que le Wiktionnaire couvre plus de noms et de verbes (2 à 7%) et que Morphalou couvre mieux les adjectifs (1 à 4%). Si l'on construit l'union des deux nomenclatures (noté M∪W), on augmente clairement la couverture pour les noms (5%) et les adjectifs (10%).

Ces observations nous amènent à formuler le constat que, bien que bruité et souffrant d'un manque de formalisme, Wiktionary, par sa couverture et sa dispo-

5. Lexique distribué par l'ATILF issu de la nomenclature du TLFi, disponible à l'adresse : http://www.cnrtl.fr/lexiques/morphalou/

6. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

Table 4.2 – Wiktionary : croissance des éditions française et anglaise de 2007 à 2010.

		2007			2010		
		Noms	**Verbes**	**Adjectifs**	**Noms**	**Verbes**	**Adjectifs**
FR	Entrées	38 973	6 968	11 787	106 068 **(x2.7)**	17 782 **(x2.6)**	41 725 **(x3.5)**
	Syn.	9 670	1 793	2 522	17 054 **(x1.8)**	3 158 **(x1.8)**	4 111 **(x1.6)**
	Trad.	106 061	43 319	25 066	153 060 **(x1.4)**	49 859 **(x1.2)**	32 949 **(x1.3)**
ANG	Entrées	65 078	10 453	17 340	196 790 **(x3.0)**	67 649 **(x6.5)**	48 930 **(x2.8)**
	Syn.	12 271	3 621	4 483	28 193 **(x2.3)**	8 602 **(x2.4)**	9 574 **(x2.1)**
	Trad.	172 158	37 405	34 338	277 453 **(x1.6)**	70 271 **(x1.9)**	54 789 **(x1.6)**

nibilité, mérite d'être envisagé comme une ressource lexicale intéressante pour le TAL. De plus, les résultats présentés dans le tableau 4.1 confirment les observations de Zesch [2010] : les ressources créées par des experts et celles construites collaborativement par les foules ne se confondent pas mais peuvent contenir des connaissances complémentaires. En effet, s'il est vrai que l'on trouve dans les entrées du Wiktionnaire absentes du TLFi, des néologismes liés notamment au domaine d'internet tels que *googler* et *wikifier*, ou des variations régionales (*dracher*) et francophones (*diplomation*, *hommerie*), le Wiktionnaire compte également des termes spécialisés comme *clitique* et *métier* (comme adjectif, « une application métier ») et d'autres, passés dans l'usage courant comme *sinogramme, homophobie, sociétal, fractal, ergonomique, médicaliser, étanchéifier, désactiver, décélérer, paramétrer*...

4.2.2.3 Entrées et relations sémantiques : une croissance à deux vitesses

Afin d'étudier comment évolue une ressource collaborative telle que Wiktionary, nous avons analysé son « *dump* historique » [7]. Ces fichiers contiennent les versions intégrales de tous les articles après chaque édition. Comme on le voit dans la figure 4.1, l'édition anglaise a connu une croissance constante en termes d'entrées (nombre de mots renseignés) alors que l'édition française a connu deux sauts. Le premier, début 2006, est dû à un import automatique d'articles du Dictionnaire de l'Académie Française (DAF). Des imports d'articles du Littré ont également été effectués, de manière plus étalée. Le nombre d'imports automatiques pour l'anglais est non significatif. On observe pour le français un second saut en 2008, plus important, qui concerne les noms et les adjectifs. Ce saut est dû à un import automatique de 76 347 gentilés (il n'affecte donc pas les verbes) extraits d'un site spécialisé.

Les contributeurs étant plus enclins à ajouter de nouvelles entrées, la croissance du nombre de relations est plus lente que celle des entrées. Parmi elles figurent es-

7. Ces dumps sont disponibles à l'adresse : http://download.wikipedia.org/

sentiellement des synonymes et quelques antonymes, les autres relations restant très rares. La figure 4.2 montre l'évolution du nombre de relations sémantiques et de liens de traduction dans les éditions anglaise et française. Aucun import automatisé concernant les traductions n'est explicitement mentionné dans Wiktionary. Cependant, nous avons remarqué dans l'édition française un ajout massif de traductions effectué début 2006 par un automate, sans que cet import ne soit documenté. Après examen, nous avons trouvé une note succincte dans la page de discussion de l'auteur de cet automate mentionnant l'import depuis un site en ligne dont ni l'origine du contenu ni la licence n'étaient clarifiés. Finalement, malgré une augmentation importante du nombre de relations sémantiques et de liens de traduction, l'écart par rapport à la croissance du nombre de mots continue de se creuser (*cf.* tableau 4.2). Or ces relations sémantiques sont des données essentielles pour l'exploitation par les applications de TAL.

4.3 Enrichissement semi-automatique

Pour combler l'écart entre le nombre d'entrées et celui des relations sémantiques, nous proposons de développer un système d'enrichissement de réseaux sémantiques, à commencer par les relations de synonymie. Ainsi nous décrivons ci-après une méthode d'enrichissement de ressource par calcul de liens de synonymie candidats dans une perspective de validation « par les foules ». L'idée est d'utiliser un graphe biparti extrait de la ressource et/ou de données externes (sous-section 4.3.1), pour calculer, avec des mesures de similarité (sous-section 4.3.2), des candidats synonymes qui pourront être validés (ou rejetés) par un contributeur. Nous proposons une validation de la méthode en sous-section 4.3.3. Cette évaluation permet aussi de mesurer l'impact des différentes similarités et sources de données utilisées.

4.3.1 Modèle de graphes bipartis pondérés

Nous nous appuyons dans les expériences qui suivent sur des *graphes bipartis pondérés*. Afin d'homogénéiser la présentation, toutes les sources de données que nous utiliserons seront modélisées par un graphe $G = (V, V', E, w)$ constitué de deux sous-ensembles de sommets :

- V, l'ensemble des mots présents dans Wiktionary pour la langue et la catégorie syntaxique considérée,
- V', un second ensemble de sommets provenant des différentes sources de données considérées.

E est l'ensemble des arêtes ($E \subseteq (V \times V')$) modélisant les relations qu'entretiennent les sommets de V avec ceux de V', un poids est donné à chaque arête par la fonction $w : E \to \mathbb{R}^+$.

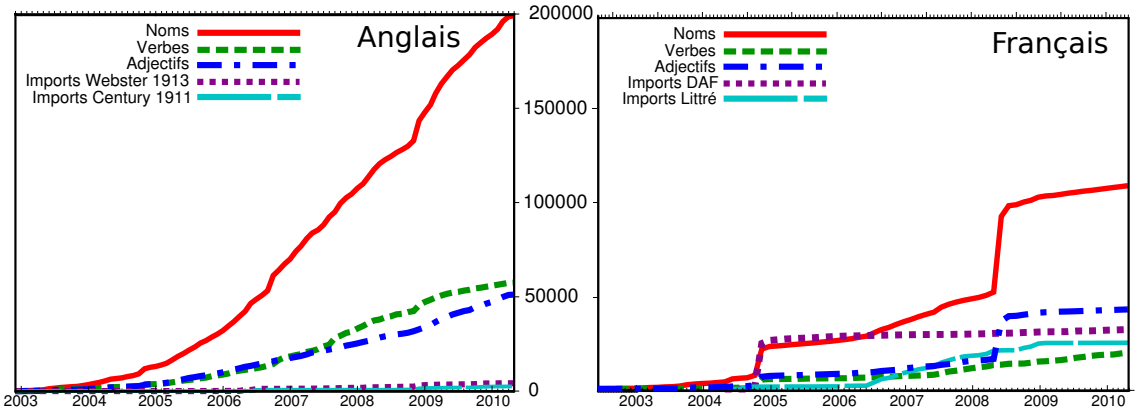


Figure 4.1 – Wiktionary : évolution du nombre de mots et imports automatisés.

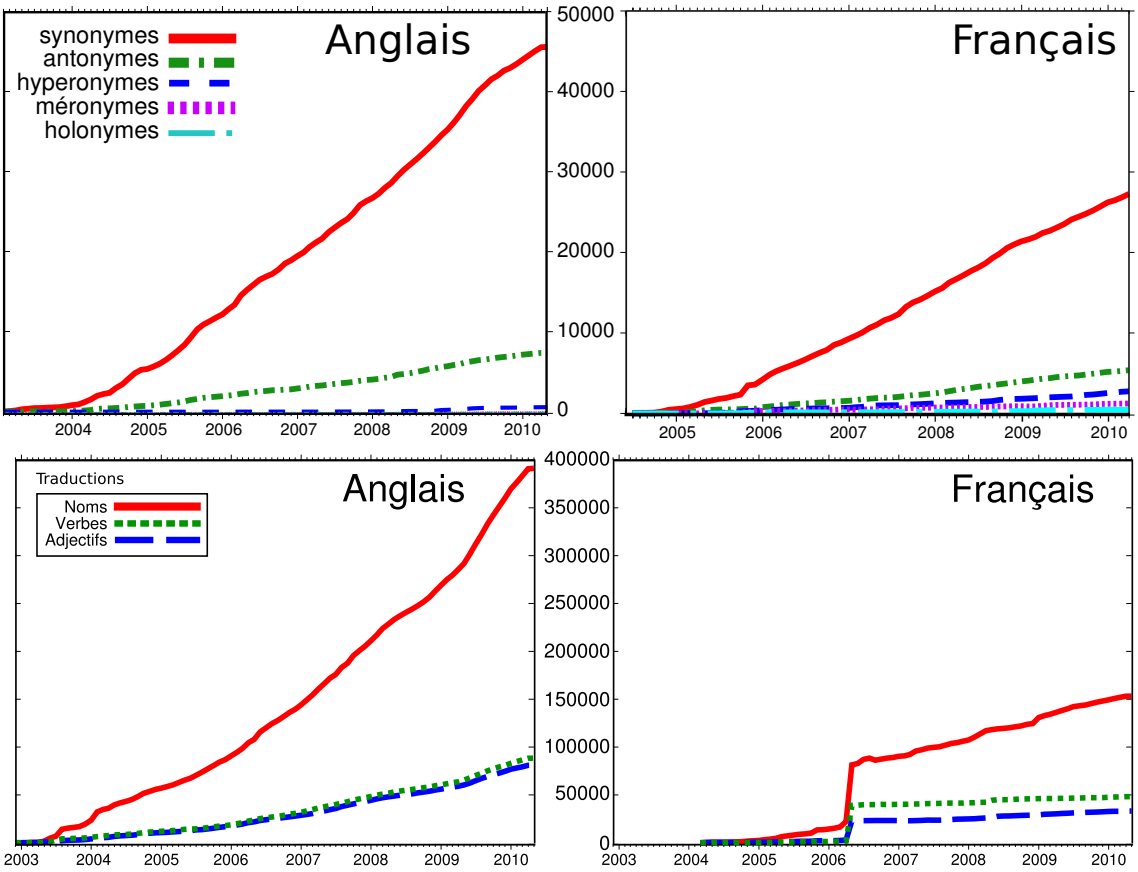


Figure 4.2 – Wiktionary : évolution des relations sémantiques et des liens de traduction.

Nous considérons les graphes suivants :

• **Graphe de traduction** $G_{Wt} = (V, V_{Wt}, E_{Wt}, w_{Wt})$
$V' = V_{Wt}$ est ici constitué des traductions des mots du Wiktionnaire étudié. E_{Wt} représente les liens de traduction : une arête relie $v \in V$ à $t \in V_{Wt}$ si t est une traduction de v dans Wiktionary. Tous les liens ont la même pondération : $\forall e \in E$, $w_{Wt}(e) = 1$.

• **Graphe de synonymie** $G_{Ws} = (V, V_{Ws}, E_{Ws}, w_{Ws})$
$V' = V_{Ws}$ est ici une duplication de V. Une arête relie $v \in V$ à $u \in V_{Ws}$ si $v = u$ ou si u (resp. v) est mentionné comme synonyme dans l'entrée de v (resp. u). Les arêtes ont ici aussi toutes la même pondération : $\forall e \in E$, $w_{Ws}(e) = 1$.

• **Graphe des gloses** $G_{Wg} = (V, V_{Wg}, E_{Wg}, w_{Wg})$
$V' = V_{Wg}$ correspond ici à l'ensemble des mots lemmatisés trouvés dans les gloses de toutes les entrées. Une arête relie $v \in V$ à $g \in V_{Wg}$ si g apparaît dans une des gloses de v. Pour chaque mot, ses gloses ont été concaténées, lemmatisées et étiquetées avec *Treetagger*, puis les mots vides ont été supprimés. Parmi plusieurs pondérations envisageables, nous avons utilisé le nombre d'occurrences : le poids de l'arête liant $u \in V$ et $g \in V_{Wg}$ est le nombre d'occurrences de g dans la glose de u. Un score plus complexe de tf-idf est envisageable, ou encore il est possible d'utiliser la position du terme dans la glose.

• **Graphe des contextes syntaxiques** $G_{Wpc} = (V, V_{Wpc}, E_{Wpc}, w_{Wpc})$
Nous avons extrait de la version française de Wikipédia un corpus de 260 millions de mots que nous avons analysé avec *Syntex* [Bourigault, 2007]. Cet analyseur produit des relations de dépendance syntaxique que nous avons utilisées pour construire une liste de paires `<mot, contexte>`, un contexte est ici un couple formé d'un (autre) mot et d'une relation syntaxique. V_{Wpc} désigne l'ensemble de ces contextes et une arête $e = \{v, c\} \in E_{Wpc}$ signifie qu'un mot v apparaît dans le contexte c. Nous utilisons l'information mutuelle ponctuelle comme pondération de ces arêtes :

$$\forall e = \{v, c\} \in E,\ w_{Wpc}(e = \{v, c\}) = \log\left(\frac{f(v, c) f(*, *)}{f(v, *) f(*, c)}\right) \tag{4.1}$$

où $f(v, c)$ est le nombre d'occurrences du mot v dans le contexte c ; $f(v, *)$, $f(*, x)$ et $f(*, *)$ sont respectivement le nombre d'occurrences totale de v (tous contextes confondus), le nombre d'occurrences totale de c et le nombre total de couples.

Les tailles des graphes ainsi construits et des graphes combinant les différentes sources de données sont récapitulées dans le tableau 4.3. Par exemple, $s + t + g$ est le graphe résultant de l'agglomération des sommets et des arêtes des graphes de synonymie, de traduction et de gloses :

$$G = \Big(V, \quad V' = V_{Ws} \cup V_{Wt} \cup V_{Wg}, \quad E = E_{Wt} \cup E_{Wt} \cup E_{Wg}, \quad w\Big)$$

Deux sommets provenant d'ensembles différents (de différents V', e.g. un de V_{Wt} et un de V_{Wg}) restent dupliqués même s'ils correspondent au même mot. Nous pondé-

Table 4.3 – Ordre et taille des graphes bipartis utilisés pour le calcul de similarité. n et n' : nombre de sommets de V et V' ayant au moins un voisin. m : nombre d'arêtes. s, t, g et c : graphes de synonymie, traductions, gloses et contextes syntaxiques.

		Anglais			Français		
		n	n'	m	n	n'	m
Adjectifs	trad.	8 178	43 976	54 840	5 335	23 976	32 944
	syn.	8 723	8 723	27 257	4 482	4 482	127 54
	gloses	45 703	39 409	218 993	41 620	42 455	263 281
	contextes	–	–	–	6 262	129 199	934 969
	s + t	13 650	52 699	82 097	7 849	28 458	45 698
	s + t + g	47 280	92 108	301 090	42 507	70 913	308 979
	s + t + g + c	–	–	–	42 517	200 761	1 248 779
Verbes	trad.	7 473	52 862	70 432	3 174	30 162	49 866
	syn.	7 341	7 341	23 927	3 190	3 190	9 510
	gloses	42 901	36 051	222 004	17 743	16 942	101 458
	contextes	–	–	–	4 273	2 312 096	5 499 611
	s + t	11 423	60 203	94 359	5 054	33 352	59 376
	s + t + g	44 295	96 254	316 363	18 226	50 294	160 834
	s + t + g + c	–	–	–	18 229	2 374 679	5 700 602
Noms	trad.	29 489	235 233	277 897	18 468	129 426	153 033
	syn.	31 227	31 227	86 195	19 407	19 407	53 869
	gloses	194 694	127 198	1 218 414	105 760	69 994	844 805
	contextes	–	–	–	22 711	1 671 655	8 719 464
	s + t	50 305	266 460	364 092	30 810	148 833	206 902
	s + t + g	202 920	393 658	1 582 506	111 228	218 827	1 051 707
	s + t + g + c	–	–	–	111 290	1 898 564	9 818 553

rons les arêtes du graphe combiné en multipliant le poids des arêtes initiales par un coefficient dépendant du type d'arête : un graphe noté $\alpha_s.s + \alpha_t.t + \alpha_g.g + \alpha_c.c$ aura la fonction de pondération suivante :

$$w(e) = \begin{cases} \alpha_s.w_{Ws}(e) & \text{si} \quad e \in E_{Ws}, \\ \alpha_t.w_{Wt}(e) & \text{si} \quad e \in E_{Wt}, \\ \alpha_g.w_{Wg}(e) & \text{si} \quad e \in E_{Wg}, \\ \alpha_c.w_{Wc}(e) & \text{si} \quad e \in E_{Wc}. \end{cases} \tag{4.2}$$

D'autres manières d'agréger les données et de pondérer les arêtes peuvent évidemment être envisagées. À ce stade, les configurations que nous proposons permettent déjà d'augmenter sensiblement le nombre de candidats pertinents proposés (voir sous-section 4.3.3).

4.3.2 Calcul de similarités, calcul de candidats

Le calcul des candidats synonymes se base sur un calcul de similarité entre les sommets de V dans les graphes bipartis que nous venons de présenter. Les candi-

dats synonymes pour un mot $s_i \in V$, sont simplement les sommets de V les plus « proches » et n'étant pas déjà synonymes de v.

Nous avons expérimenté plusieurs mesures de similarité basées sur des *marches aléatoires* en temps courts. Le fonctionnement de ces marches est défini plus en détail en section 2.2.1. L'idée est de considérer un marcheur parcourant aléatoirement le graphe : à chaque pas ce marcheur passe du sommet courant à l'un des voisins choisi aléatoirement mais proportionnellement au poids des arêtes correspondantes. Nous notons $prx_t(u, v)$ la probabilité qu'un marcheur soit sur un sommet v apres t pas depuis un sommet u. En se basant sur ces probabilités nous avons testé les mesures de similarité entre sommets suivantes :

$$\text{simple}(u, v) = prx_t(u, v) \tag{4.3}$$

$$\text{avg}(u, v) = \frac{prx_t(u, v) + prx_t(v, u)}{2} \tag{4.4}$$

$$\cos(u, v) = \frac{\sum_{w \in V} .prx_t(u, w) prx_t(v, w)}{\sqrt{\sum_{w \in V} prx_t(u, w)^2} \sqrt{\sum_{w \in V} prx_t(v, w)^2}} \tag{4.5}$$

$$\text{dot}(u, v) = \sum_{w \in V} prx_t(u, w).prx_t(v, w) \tag{4.6}$$

$$\text{ZKL}_\Omega(u, v) = \sum_{w \in V} prx_t(u, w). \begin{cases} \log(\frac{prx_t(u,w)}{prx_t(v,w)}) & \text{si } prx_t(v, w) \neq 0, \\ \Omega & \text{sinon.} \end{cases} \tag{4.7}$$

La mesure « simple » est directement la probabilité d'atteindre u en partant de v, « avg » est une version naïvement symétrisée de « simple ». Les autres similarités sont des mesures classiques calculées sur les vecteurs de probabilité donnés par $prx_t(v, *)$ et $prx_t(u, *)$. « cos » et « dot » sont respectivement le cosinus et le produit scalaire. « ZKL_Ω » est une variante de la mesure de divergence Kullback-Leibler introduite par Hughes et Ramage [2007].

Nous avons vu en section 2.2.1 que sur un graphe biparti, le marcheur est exclusivement sur l'un ou l'autre des ensembles de sommets en fonction du sommet de départ et de la parité du nombre t de pas effectués. Pour pouvoir mesurer une similarité entre sommets d'un même groupe V nous utilisons donc des marches en temps pairs. Les expériences présentées ensuite sont réalisées avec $t = 2$ (marches de longueur 2). Nous avons effectué des expériences préliminaires avec $t = 4$, mais les résultats sont similaires (avec un coût de calcul plus important).

4.3.3 Évaluation

4.3.3.1 Mesure de la pertinence des candidats proposés

En vue d'un enrichissement semi-automatique dans lequel les contributeurs valident ou invalident les candidats proposés, nous considérons qu'une courte liste de suggestions est *acceptable* si elle contient au moins un candidat pertinent. Ainsi,

notre évaluation consiste essentiellement à compter pour combien de mots le système produit une liste *acceptable*. Nous examinons également pour combien de mots sont proposées des listes contenant 2, 3 ou plusieurs candidats pertinents.

Soit $G_{GS} = (V_{GS}, E_{GS})$ un réseau de synonymie étalon, V_{GS} ses mots, et $E_{GS} \subseteq V_{GS} \times V_{GS}$ ses relations de synonymie. Nous évaluons ci-après la pertinence des relations proposées pour enrichir la ressource déficiente par rapport aux relations de l'étalon. Nous effectuons cette évaluation pour les mots communs à la ressource à enrichir et l'étalon. Nous supprimons donc de la liste des candidats ceux qui ne figurent pas dans l'étalon [8] et limitons cette liste à $k = 5$ candidats. Pour chaque mot $v \in V \cap V_{GS}$, on note $\Omega_k(v)$ la liste des meilleurs candidats *« évaluables »* :

$$\Omega_k(v) = [c_1, c_2, \ldots, c_{k'}] \quad \text{avec} \quad \begin{cases} k' \leq k \\ \forall i,\ c_i \in V \cap V_{GS} \\ \forall i, \mathrm{sim}(v, c_i) \geq \mathrm{sim}(v, c_{i+1}) \end{cases} \tag{4.8}$$

où *sim* est l'une des mesures présentées en section 4.3.2.

$\Omega_k(v)$ contient un maximum de k candidats (mais peut en contenir moins ou être vide). De plus, $\Omega_k(v)$ dépend de la ressource étalon, car les candidats non présents dans l'étalon sont éliminés. On note $\Omega_k^+(v)$ l'ensemble des candidats corrects de $\Omega_k(v)$:

$$\Omega_k^+(v) = \left\{ c^+ \in \Omega_k(v) \middle/ \{v, c^+\} \in E_{GS} \right\} \tag{4.9}$$

On note N_k l'ensemble des mots pour lesquels sont proposés k candidats et N_k^{+p} le sous-ensemble des mots pour lesquels *au moins p candidats pertinents* sont proposés :

$$N_k = \left\{ v \in V \cap V_{GS} \middle/ |\Omega_k(v)| = k \right\} \tag{4.10}$$

$$N_k^{+p} = \left\{ v \in N_k \middle/ |\Omega_k^+(v)| \geq p \right\} \tag{4.11}$$

Pour mesurer l'impact des sources de données utilisées sur le calcul des candidats, nous mesurons le ratio R_k entre le nombre de listes *suggérées* [9] et le nombre de mots *évaluables*, et P_k, le ratio entre le nombre de listes *acceptables* et le nombre de listes *suggérées* :

$$R_k = \frac{|N_k|}{|V_{GS} \cap V|}, \qquad P_k = \frac{|N_k^{+1}|}{|N_k|} \tag{4.12}$$

Nous utilisons PWN et DicoSyn comme ressources étalons respectivement pour l'anglais et le français. Nous renvoyons à la section 3.5 pour une description plus en détail de ces ressources.

8. Il peut s'agir d'un néologisme ou d'un terme de spécialité, plus rarement d'une forme mal orthographiée de Wiktionary, ou être dû à une couverture non exhaustive de l'étalon.

9. *i.e.* le nombre de mots pour lesquels au moins un candidat est proposé.

Table 4.4 – Comparaison de différentes mesures de similarité sur la mesure P_5 des listes de candidats.

	Synonymie				Traductions				Syn. + Trad.			
	EN		FR		EN		FR		EN		FR	
	V	N	V	N	V	N	V	N	V	N	V	N
simple	41.4	32.4	58.6	47.3	51.4	37.8	**78.7**	58.3	51.9	39.0	**74.6**	**55.3**
avg	42.5	33.5	58.2	46.8	50.5	38.0	**78.7**	58.3	51.1	39.3	74.0	55.1
cos	**43.4**	**34.6**	**60.2**	47.9	51.8	38.5	78.3	58.6	51.3	39.4	73.1	54.2
dot	42.0	34.0	59.7	46.7	**52.3**	**38.7**	78.2	58.7	**52.4**	39.7	73.6	54.8
ZKL_{10}	43.2	34.0	60.1	**48.2**	51.8	38.6	**78.7**	**58.8**	51.9	**39.8**	74.0	54.5

4.3.3.2 Résultats de l'évaluation

Les tableaux 4.4 et 4.5 donnent les résultats de la méthode. Le tableau 4.4 présente les valeurs de la mesure P_5, pour les cinq similarités et pour trois sources de données. Le tableau 4.5 se limite à la mesure « simple », mais présente les résultats pour toutes les sources de données et donne les valeurs des différentes mesures d'évaluation.

Notons que pour une même langue, une même partie du discours, et la même similarité, les résultats sont légèrement meilleurs dans le tableau 4.5 que dans le tableau 4.4. Cela s'explique par le fait que le tableau 4.4 a été calculé sur des données extraites de Wiktionary en 2007 [Sajous et al., 2010], alors que le tableau 4.5 a été calculé à partir de données de 2010 [Sajous et al., 2011a]. Cela montre un autre intérêt de cette méthode endogène : plus la ressource est complète [10] plus les propositions sont de qualité. Nous pouvons imaginer que cela restera vrai jusqu'à un point où la plupart des candidats de fortes similarités seront déjà indiqués comme synonymes.

On observe tout d'abord dans le tableau 4.4 qu'il n'y a pas de différence de performance flagrante entre les différentes mesures. C'est pourquoi nous nous limitons par la suite à la mesure « simple » qui présente la complexité la plus faible.

On remarque que les graphes de traduction donnent de meilleures précisions que ceux de synonymie. Ce résultat était prévisible car les entrées ont plus de liens de traduction que de synonymie dans Wiktionary. De plus, ces liens sont répartis sur un nombre important de langues, ce qui rend l'information relativement fiable. Pour l'anglais, agglomérer les graphes de synonymie et de traduction améliore le rappel sans détriment notable pour la précision. Pour le français, cela conduit à une baisse de précision par rapport à l'utilisation du seul graphe de traduction.

Le meilleur rappel est obtenu avec les gloses et les contextes syntaxiques (tableau 4.5). Ce résultat était attendu également, l'information véhiculée par ces don-

10. voir section 4.2.2.3 pour une étude de l'évolution de Wiktionary.

Table 4.5 – Impact des sources de données sur le calcul des candidats (similarité « simple »).

			R_5	P_5	$\lvert N_5\rvert$	$\lvert N_5^{+1}\rvert$	$\lvert N_5^{+2}\rvert$	$\lvert N_5^{+3}\rvert$	$\lvert N_5^{+4}\rvert$	$\lvert N_5^{+5}\rvert$
ANG	**A.**	syns	17.4	49.1	2456	1207	439	**165**	**57**	**22**
		trads	9.2	**65.7**	1299	853	406	144	27	3
		gloses non pondérées	**93.5**	25.9	**13205**	3421	774	154	34	2
		gloses pondérées	**93.5**	26.6	**13205**	**3510**	**794**	158	30	1
		$s+t$	21.9	**58.1**	3096	1800	805	283	91	29
		$\mathbf{10.s+10.t+g}$	**95.0**	35.9	**13417**	**4819**	**1567**	**455**	**125**	**32**
		$\mathbf{10^2.s+10^2.t+g}$	**95.0**	35.9	**13417**	**4818**	**1567**	**455**	**125**	**32**
	N.	syns	8.7	34.6	3862	1335	483	**200**	**95**	**54**
		trads	8.5	**51.0**	3759	1916	655	178	41	2
		gloses non pondérées	**95.6**	14.8	**42337**	6252	926	106	6	0
		gloses pondérées	**95.6**	15.3	**42337**	**6467**	**933**	114	5	1
		$s+t$	14.5	**47.6**	6440	3063	1061	348	110	45
		$\mathbf{10.s+10.t+g}$	**96.4**	23.3	**42688**	**9944**	2344	**561**	**142**	**43**
		$\mathbf{10^2.s+10^2.t+g}$	**96.4**	23.3	**42688**	9942	**2345**	**561**	**143**	**43**
	V.	syns	23.9	44.5	2153	959	431	**216**	**115**	**59**
		trads	24.7	**60.4**	2223	1342	**609**	187	43	1
		gloses non pondérées	**98.5**	27.0	**8852**	2389	518	98	10	2
		gloses pondérées	**98.5**	28.1	**8852**	**2490**	548	100	13	2
		$s+t$	37.6	**58.0**	3380	1962	918	358	119	**43**
		$\mathbf{10.s+10.t+g}$	**99.2**	41.0	8916	**3655**	**1352**	**448**	**136**	34
		$10^2.s+10^2.t+g$	**99.2**	40.9	**8917**	3644	1351	**448**	**136**	34
FR	**A.**	syns	11.9	75.2	480	361	224	**139**	55	**16**
		trads	6.0	**91.4**	243	222	184	117	**56**	11
		gloses non pondérées	**90.2**	32.2	**3627**	1167	309	91	12	1
		gloses pondérées	**90.2**	33.6	**3627**	**1220**	**337**	100	17	0
		contextes	86.2	20.7	3468	719	157	40	11	1
		$s+t$	15.7	**81.3**	631	513	375	243	105	28
		$10.s+10.t+g$	89.5	44.3	3602	1594	728	371	154	38
		$10^2.s+10^2.t+g$	91.2	43.6	3668	1600	729	370	155	38
		$10^2.s+10^2.t+10.g+c$	**97.3**	41.9	3913	1640	680	347	143	32
		$\mathbf{10^3.s+10^3.t+10^2.g+c}$	**97.3**	45.3	**3915**	**1774**	**791**	**408**	**172**	**43**
	N.	syns	10.4	54.4	1722	936	478	194	68	15
		trads	5.5	**79.7**	916	730	472	**245**	**94**	**20**
		gloses non pondérées	**95.8**	20.6	**15828**	3268	607	116	16	2
		gloses pondérées	**95.8**	22.5	**15828**	**3560**	693	127	21	3
		contextes	84.0	20.9	13882	2898	**721**	181	34	5
		$s+t$	15.2	**66.9**	2511	1681	983	480	166	33
		$10.s+10.t+g$	96.5	33.3	15948	5303	1956	735	219	50
		$10^2.s+10^2.t+g$	96.5	33.2	15948	5298	1952	736	218	52
		$10^2.s+10^2.t+10.g+c$	**98.5**	33.1	**16274**	5394	1908	649	196	38
		$\mathbf{10^3.s+10^3.t+10^2.g+c}$	98.4	36.7	16273	**5980**	**2240**	**825**	**260**	**56**
	V.	syns	10.0	68.0	412	280	172	86	30	5
		trads	19.0	**90.4**	785	710	544	**352**	**146**	**38**
		gloses non pondérées	**95.6**	41.2	**3947**	1628	530	149	38	3
		gloses pondérées	**95.6**	44.9	**3947**	**1773**	**638**	198	45	8
		contextes	81.8	35.3	3378	1192	426	126	28	3
		$s+t$	25.7	**85.6**	1062	909	669	418	165	48
		$10.s+10.t+g$	96.6	55.9	3989	2229	1161	580	216	**58**
		$10^2.s+10^2.t+g$	96.6	55.8	3989	2226	1160	580	214	**58**
		$10^2.s+10^2.t+10.g+c$	**98.1**	53.2	**4053**	2158	1004	433	146	43
		$\mathbf{10^3.s+10^3.t+10^2.g+c}$	**98.1**	58.4	**4053**	**2368**	**1243**	**604**	**223**	53

nées étant moins spécifique. Elle permet cependant de proposer des candidats pour la quasi-totalité des mots. La pondération des gloses par les nombres d'occurrences améliore légèrement les résultats et l'on peut penser qu'une pondération plus fine (e.g. en favorisant les termes situés à l'initiale des gloses) produirait une nouvelle amélioration.

Plus étonnamment, les contextes syntaxiques donnent une précision inférieure, ce qui ne va pas dans le sens des résultats de Van der Plas et Bouma [2005]. Filtrer les contextes de basse fréquence permettrait probablement d'améliorer ce score : en effet, un mot apparaissant avec un unique contexte syntaxique tend à avoir avec lui une information mutuelle élevée sans que le rapprochement avec un autre mot apparaissant avec ce même contexte ne soit réellement significatif.

Dès lors que l'on introduit les gloses, des candidats sont proposés pour quasiment tous les mots. Cela est intéressant car l'objectif, en vue de l'application, est d'effectuer des suggestions certes les plus pertinentes possible mais surtout pour un nombre maximal de mots. Il serait en effet dommage de ne proposer des candidats que pour un nombre restreint de mots, même si cela permettrait d'avoir un score de confiance élevé (nous atteignons dans ce cas 80 à 90% de listes pertinentes en ne recourant qu'aux graphes de traduction pour le français, mais alors des propositions ne sont faites que pour seulement 6 à 20% des mots). Notons que l'utilisation des gloses et contextes syntaxiques dans les graphes combinés ne fait chuter la précision qu'au niveau global : au niveau local, les mots pour lesquels les liens de synonymie et de traduction conduisent à proposer des candidats pertinents ne sont pas affectés par cette baisse (la proportion P_5 baisse mais pas le cardinal N_k^{+i}).

Les meilleurs résultats sont obtenus en combinant les graphes de synonymie, de traduction et des gloses pour l'anglais (graphe $10.s + 10.t + g$) et celui de synonymie, traductions et des contextes syntaxiques pour le français (graphe $10^3.s + 10^3.t + 10^2.g + c$). L'utilisation de ces graphes nous permet de proposer 5 candidats pour presque tous les mots et 35 à 60% des listes proposées comportent au moins un candidat validé par un étalon. Ce résultat nous permet de considérer la méthode comme valable en vue de l'application envisagée.

Notons enfin que les résultats sont meilleurs pour le français que pour l'anglais. Cela pourrait s'expliquer par une différence de densité des réseaux initiaux, mais est surtout dû aux étalons utilisés : DicoSyn, étant une compilation de 7 dictionnaires, est plus dense que PWN.

4.3.3.3 Qualification des relations proposées

Afin de mieux cerner le type de relations capturées par notre mesure de similarité, nous avons étudié la nature des synonymes candidats rejetés par l'étalon choisi. En utilisant PWN, nous avons réparti les candidats anglais des listes $\Omega_k(v)$ présentées plus haut ($k \leq 5$) suivant les relations *synonymie, hyperonymie, hyponymie, cohypo-*

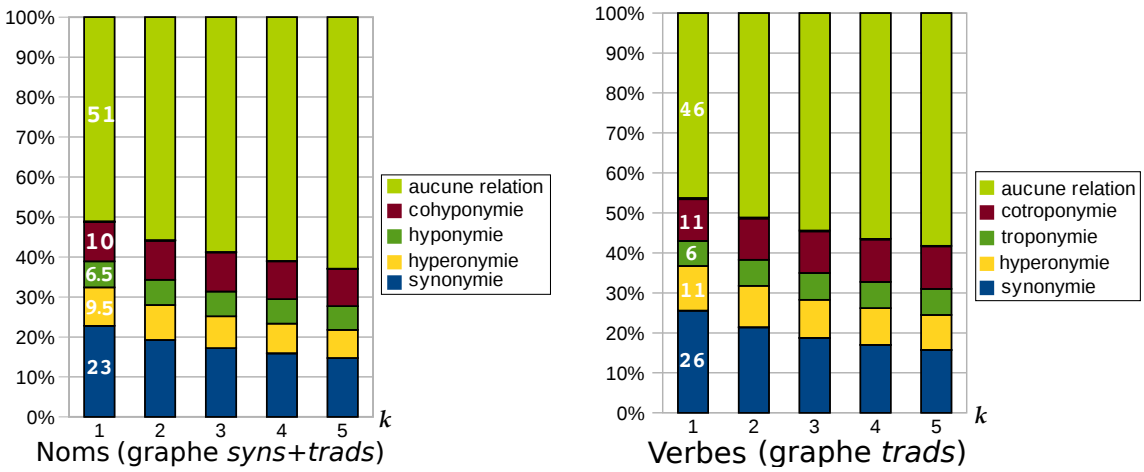


Figure 4.3 – Proportion des candidats par type de relation sémantique.

Table 4.6 – Exemples de candidats appartenant à une relation sémantique autre que la synonymie.

Catégories	Mots	Candidats	Relations
Noms	*hound*	*greyhound*	hyponymie
	law	*rule*	hyponymie
	fool	*idiot, dummy*	cohyponymie
			(synset hyperonyme : *{simpleton, simple}*)
Verbes	*to represent*	*to depict*	troponymie
	to negociate	*to bargain*	troponymie
	to blame	*to incriminiate*	cotroponymie
			(synset troponymie : *{to charge, to accuse}*)

nymie pour les noms et *synonymie, hyperonymie, troponymie, cotroponymie* pour les verbes [11]. Nous avons gardé dans cette évaluation la finesse de granularité de PWN en respectant ses définitions strictes : deux mots sont synonymes s'ils apparaissent dans un même synset, et nous considérons seulement les relations strictes d'hyperonymie/hyponymie/cohyponymie (e.g. w_1 est en relation d'hyperonymie avec w_2 si w_2 apparaît dans un synset qui est un fils direct d'un synset dans lequel apparaît w_1).

La figure 4.3 montre la répartition, en termes de type de relation, des candidats calculés. Pour tous les noms et verbes communs à Wiktionary et PWN, nous avons considéré des listes de 1 à 5 candidats, puis supprimé ceux absents de PWN. Nous pouvons voir qu'une grande partie des candidats non pertinents pour la synonymie recouvre une autre réalité sémantique. Le tableau 4.6 donne quelques exemples de candidats appartenant à une relation sémantique autre que la synonymie. Sans que

11. Nous avons également considéré, puis écarté les relations d'antonymie, holonymie/méronymie et causalité, qui concernaient entre 1‰ et 0.1‰ de cas.

les résultats soient directement comparables (enrichissement endogène vs. extraction en corpus, PWN vs. EuroWordNet néerlandais), nous pouvons noter que la similarité de Heylen et al. [2008] produit autant de cohyponymes que de synonymes alors que nos mesures de similarité sur les sources de données utilisées produisent deux fois plus de synonymes que de cohyponymes. Ce dernier point indique qu'il est intéressant que l'application ne permette pas seulement aux utilisateurs de valider ou d'invalider un candidat synonyme, mais permette aussi de « réorienter » chaque candidat vers une autre relation sémantique.

4.4 Une implémentation fonctionnelle : le système Wisigoth

Pour mettre en œuvre le système d'enrichissement avec validation décrit plus haut, nous avons implémenté le système *Wisigoth* (pour WIktionarieS Improvement by Graph-Oriented meTHods), constitué en trois parties :

- une **chaîne de traitement** hors-ligne,
- un **serveur de candidats** en ligne,
- une **extension Firefox**.

La *chaîne de traitement* (hors-ligne) extrait, à partir des dumps de Wiktionary, les relations de synonymie, de traduction, etc. et calcule les relations candidates comme présenté dans les sections précédentes. Ces listes de candidats sont données au *serveur de candidats*, qui les propose quand il reçoit une requête pour un mot ciblé donné. Ce serveur est interrogé par l'*extension Firefox* qui est la seule partie « visible » du système. On peut en voir une capture d'écran en figure 4.4.

Lorsqu'un internaute installe cette extension et visite une page française ou anglaise de Wiktionary, le serveur de candidats est interrogé et renvoie une liste ordonnée de candidats pour le(s) mot(s) en cours de consultation. Ces candidats sont alors présentés à l'internaute (*cf.* figure 4.4). S'il les valide (bouton « + »), l'extension prend en charge l'ajout du synonyme dans Wiktionary et l'édition du code wiki correspondant. Un champ texte libre permet également de proposer un synonyme non-suggéré. Indépendamment de notre méthode d'enrichissement, *Wisigoth* permet ainsi aux internautes non familiers avec le code wiki de contribuer. Pour rester proche du principe wiki, nous n'avons pas mis en place de système de validation croisée et les ajouts sont publiés immédiatement. Néanmoins, facilitant l'ajout de synonymes, nous devons également en faciliter leur suppression : un bouton « − » est ajouté à chaque synonyme présent, qui déclenche sa suppression. Enfin, un système de liste noire permet à un contributeur de signaler une suggestion jugée non pertinente (bouton « × »). Cette suggestion ne lui sera plus faite et permettra aux autres candidats de remonter dans la liste. Au-delà d'un seuil (réglé pour l'instant

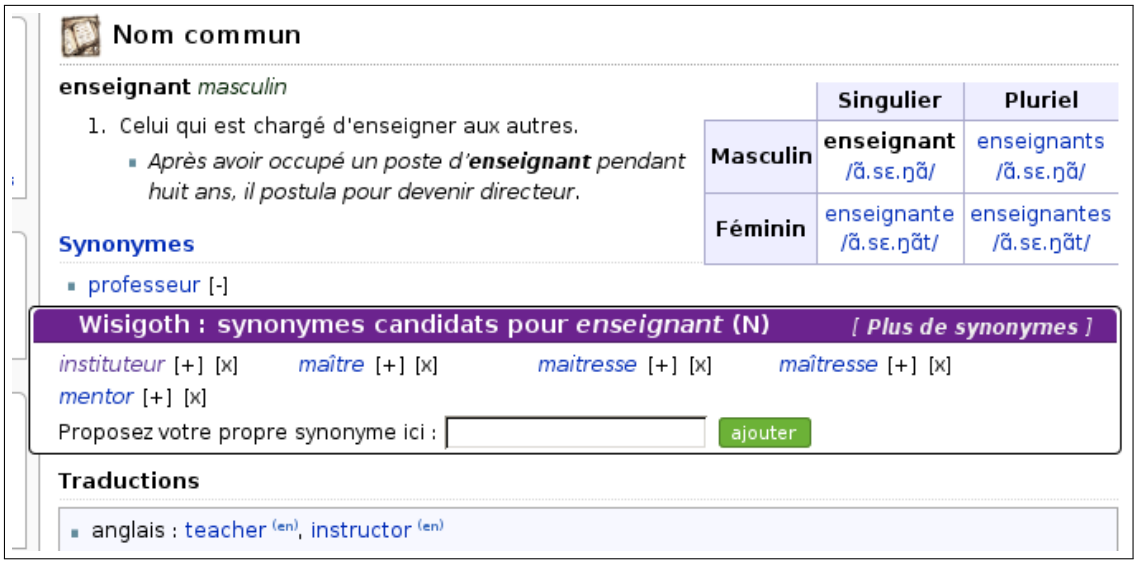


Figure 4.4 – Exemple de propositions faites par *Wisigoth* sur le Wikitionary français pour « enseignant ».

arbitrairement à 3) de mises sur listes noires individuelles, la proposition est insérée dans une liste noire globale de manière à ne plus être proposée à aucun utilisateur.

4.5 Conclusion du chapitre

Nous avons dans ce chapitre proposé un système (en grande partie) endogène d'enrichissement semi-automatique de ressources lexicales. Ce système se base sur un calcul de similarités, par des marches aléatoires courtes, entre sommets d'un graphe biparti construit à partir de la ressource.

Nous appliquons ce système sur Wiktionary. En effet nous avons présenté une étude qui montre que la couverture lexicale de cette ressource construite « par les foules » croit plus vite que son nombre de relations. Un système d'aide à l'ajout de relations lexicales est donc justifié. Nous avons validé le système proposé grâce à des ressources étalons. Ce système est capable de proposer des courtes listes de candidats pour la quasi totalité des entrées de Wiktionary et de 30 à 60% de ces listes permettent d'ajouter au moins une relation validée par une ressource étalon (les résultats dépendent de la partie du discours et de la langue considérée).

Enfin nous avons présenté une implémentation sous forme d'une extension *Firefox*. Celle-ci permet de proposer des candidats synonymes à l'utilisateur lorsqu'il navigue sur Wiktionary. Ainsi il peut les marquer comme incorrects ou les ajouter dans la ressource collaborative.

Cette extension *Firefox* est une preuve de concept qui a été développé en 2010 et 2011. Elle était alors pleinement fonctionnelle, malheureusement nous n'avons pu la

maintenir et des modifications tant dans *Firefox* que dans les formats et structures de données de Wiktionary font qu'elle n'est plus utilisable aujourd'hui. Nous sommes néanmoins convaincus que ce type de solution d'aide à la construction de ressource est nécessaire pour développer des ressources de qualité à moindre coût, que ce soit « par les foules » ou par des lexicographes experts. Notamment le développement d'outils (extension *Firefox* par exemple) d'assistance à la contribution sur Wiktionary est, il nous semble, quasiment indispensable pour réellement voir la qualité et le nombre de relations lexicales de Wiktionary atteindre un niveau comparable à celui de Wikipedia. Il serait intéressant de développer un outil d'assistance à la contribution, similaire à l'extension que nous avons proposé, mais qui puisse se connecter à différents serveurs de candidats. Ainsi chacun (particulier, entreprise ou équipe de recherche) pourrait facilement proposer un serveur de candidats. Un contributeur disposerait alors des différentes propositions de candidats. Il serait possible d'évaluer a posteriori les serveurs ayant amené le plus de contributions.

Concernant la méthode de calcul des candidats, différentes améliorations sont envisageables. Nous avons vu que la construction du graphe est primordiale. Il serait possible d'explorer différentes méthodes de pondération, en particulier pour les gloses et les contextes syntaxiques. Par exemple en pondérant les termes des gloses de manière inversement proportionnelle à leur position dans la définition (les premiers termes d'une définition étant souvent plus semblables au mot défini que les derniers).

Il serait possible d'évaluer les différentes mesures de similarité introduites au chapitre 2. Mais cette perspective n'apparaît pas comme prioritaire à la vue des faibles différences observées ici entre les quelques méthodes utilisées. Ces faibles différences sont par ailleurs confirmées par les fortes corrélations constatées au chapitre 2 entre beaucoup de mesures. De surcroît, la taille importante des graphes utilisés ici limite l'utilisation de méthodes globales. Il faudrait aussi vérifier que ces méthodes peuvent s'appliquer sur un graphe biparti. Il serait tout de même intéressant [12] d'évaluer la *confluence* (introduite en sous-section 2.2.2). Cela étant dit, le fait que la confluence soit définie entre 0 et 1 n'a aucun intérêt ici, seule la prise en compte du degré d'arrivée peut être intéressante. Cela aurait comme effet de pénaliser les candidats de forts degrés par rapport à ceux de faibles degrés. Il n'est donc pas sûr que cela améliore les résultats, car dans une optique de construction de ressource, le fait de privilégier les candidats de forts degrés peut être souhaitable : un peu à la manière de l'attachement préférentiel, les liens se forment certainement plus facilement avec des sommets forts.

Notons aussi que l'application présentée ici peut être mise en parallèle avec le problème dit « de recommandation ». Ce problème consiste typiquement à proposer des produits à un client étant donné l'ensemble des achats passés [Ricci et al., 2011]. Aussi, la littérature s'intéressant à la prédiction de liens dans un graphe biparti

12. Au moins pour la cohérence de ce rapport.

[Allali et al., 2013] peut proposer des méthodes (spécifiques aux graphes bipartis) qui méritent d'être évaluées sur cette application.

Enfin l'évaluation pourrait profiter des propositions faites au chapitre précédent. Notamment il pourrait être intéressant de mesurer la *confluence* dans le graphe étalon des candidats qui sont, pour le moment, rejetés par l'étalon.

Deuxième partie

Clustering de graphe biparti, théories et applications

Chapitre 5

Clustering de graphe biparti

On s'intéresse dans ce chapitre au problème consistant « à regrouper intelligemment les sommets d'un graphe biparti ». Cette formulation est volontairement vague pour nous permettre de recouper les approches généralement proposées sous diverses étiquettes : « détection de communautés », « conceptualisation » ou encore « clustering ».

Rappelons qu'un graphe biparti est un graphe comportant deux types de sommets et des liens seulement entre des sommets de types différents. C'est une relation (binaire) entre deux ensembles de sommets. Nous considérons dans ce chapitre que l'un de ces ensembles correspond à des objets, alors que le second correspond à des propriétés.

Différentes motivations peuvent amener à définir plus clairement ce problème de « clustering », par exemple : comment diviser un graphe en différents groupes de façon à minimiser le nombre de liens entre ces groupes ? comment, pour simplifier la représentation des objets, peut-on les classer, d'après leurs propriétés, dans un nombre raisonnable de catégories « naturelles » ? quels sont les objets formant des groupes indissociables et quelle peut être leur organisation hiérarchique ? Ces différentes motivations ont amené plusieurs communautés de chercheurs à proposer des approches différentes. Pour autant il est intéressant de mettre en parallèle ces approches. C'est un travail auquel nous contribuons dans ce chapitre, en établissant notamment une traduction de l'analyse formelle de concepts en termes de graphe.

Ce parallèle permet d'apporter un éclairage intéressant sur les méthodes de la littérature, et nous amène en particulier vers une problématique du clustering de graphe biparti souvent ignorée : Comment partitionner les objets sans imposer de partition correspondante sur les propriétés ? Nous proposons une méthode simple qui répond à cette question. Cette méthode est basée à la fois sur l'analyse formelle de concepts et sur un algorithme de détection de communautés.

Enfin, en utilisant des mesures de similarité définies dans la première partie de ce rapport, nous proposons une méthode de prétraitement d'un graphe biparti

éventuellement pondéré. Cette méthode permet de filtrer un graphe biparti et de le binariser s'il est pondéré, pour simplifier les résultats obtenus par l'analyse formelle de concepts.

Ces contributions sont organisées en quatre sections. En section 5.1 nous dressons un panorama des différentes familles d'approches. En section 5.2 nous dressons un parallèle entre l'analyse formelle de concepts et l'analyse de graphe biparti. Ce parallèle nous permet d'apporter deux contributions dans les sections suivantes. En section 5.3 nous proposons et évaluons une méthode pour catégoriser les objets, et ce même si les propriétés sont fortement partagées entre les catégories. Enfin en section 5.4, après avoir décrit les différentes extensions envisageables ou existantes de l'analyse formelle de concepts pour des données graduelles ou empreintes d'incertitude, nous proposons une méthode basée sur des marches aléatoires permettant de simplifier et de binariser un graphe biparti. Nous proposons ensuite une première évaluation de cette méthode sur des graphes artificiels et sur des graphes réels, avant de conclure en section 5.5.

5.1 Différentes familles d'approches

L'objectif de cette section est de dresser un panorama rapide des différents problèmes et différentes familles d'approches s'intéressant au « regroupement intelligent » des sommets d'un graphe biparti. Nous partons volontairement de cette définition vague pour essayer d'apporter une formulation plus claire à ce problème, en explorant les approches existant dans la littérature.

Nous détaillons les méthodes citées seulement quand cela nous semble nécessaire pour la suite du chapitre, ou pour comprendre les similarités ou différences existant avec d'autres approches. Aussi nous nous efforçons de citer les travaux ayant établi ou utilisé des parallèles entre différentes approches ou familles d'approches.

Notons que beaucoup de méthodes d'apprentissage automatique supervisé ont été appliquées sur ce type de données avec un objectif semblable. Ces méthodes nécessitent un jeu d'entraînement sur lequel on connaît le résultat escompté. Nous nous intéressons ici uniquement aux méthodes dites « non supervisées », ne nécessitant pas un tel jeu d'entraînement.

5.1.1 Partitionnement de graphe biparti

Le *partitionnement de graphe* est un problème classique de la théorie des graphes [Bichot et Siarry, 2010]. Il consiste, sous sa forme première (c'est-à-dire le partitionnement dit « contraint »), à découper un graphe en un certain nombre de parties de taille semblable, en minimisant le « *cut* » , c'est-à-dire le nombre d'arêtes entre ces parties. L'objectif n'est pas (a priori) de révéler une structure sous-jacente existante

dans les données, mais plutôt d'optimiser un découpage en un nombre imposé de parties de taille fixée.

Bien que pensé sur des graphes uni-partis ce problème a aussi été posé pour des graphes bipartis. Et certains algorithmes ont pu être adaptés. On peut citer en particulier [Dhillon, 2001] et [Hu et al., 2006].

Le principal inconvénient de ce type d'approche est que le nombre de clusters est un paramètre fixé. Aussi ces méthodes cherchent à construire autant que possible des clusters de taille semblable, ce qui ne correspond pas forcément à la réalité des données. Notons qu'une comparaison de ce type d'approches (sur des graphes uni-partis) avec des méthodes de détection de communautés (voir section 5.1.3) est faite dans [Newman, 2006]. Dans ce papier le problème de la taille fixée des clusters est soulevé, mais aussi la pertinence des approches optimisant la mesure de *cut* est remise en question, quand on s'intéresse à faire émerger des groupes « existant » dans les données. L'idée défendue par Newman est que les groupes pertinents ne sont pas ceux ayant le moins de liens possible entre eux, mais ceux ayant moins de liens entre eux « qu'attendu ». Il faut chercher un découpage qui optimise le nombre de liens entre les groupes, *par rapport* au nombre de liens qui existerait entre ces mêmes groupes sur un graphe « neutre » partageant certaines des caractéristiques du graphe étudié (*null model*). La mesure de *modularité* permet de mesurer ce rapport (voir section 5.1.3).

L'approche proposée par Hu et al. [2006] consiste à appliquer une méthode de partitionnement de graphe biparti sur un graphe reliant les objets et des *hyperclique patterns* qui sont des bicliques maximales approchées. Cette méthode est très similaire à celle que nous proposons en section 5.3. Elle souffre néanmoins des faiblesses que nous venons d'évoquer concernant le partitionnement de graphe.

5.1.2 Bi-clustering

Le terme « bi-clustering » fait référence, en général, aux méthodes cherchant à regrouper les objets similaires *avec* les propriétés qui les rapprochent. Les méthodes en elles-même ne sont pas détaillées dans cette section, pour un état de l'art plus complet nous renvoyons à [Busygin et al., 2008] ou à [Eren et al., 2012]. Notons que le principal champ d'application des méthodes de bi-clustering concerne l'analyse de données génétiques, en particulier l'exploitation des puces à ADN (ou *microarray*), voir [Babu, 2004].

À l'inverse du partitionnement de graphe, ici le résultat ne doit pas simplement être optimal pour une fonction de coût donnée, mais doit « faire sens » vis à vis des données. On dit parfois que le regroupement ou le découpage doit être « naturel » [1].

1. Le terme « naturel » peut sembler bien vague, et c'est certainement là toute la difficulté du clustering : il est très difficile de définir précisément ce qui est recherché. Précisons tout de même que l'idée derrière ce terme « naturel » est que les méthodes doivent rendre visible un découpage

Cela amène une plus grande liberté dans la formalisation de ce qu'est un bon découpage, mais aussi une difficulté accrue à évaluer les performances des différentes méthodes.

L'autre différence évidente avec le partitionnement de graphe est que les données ne sont plus représentées par un graphe. Souvent ces algorithmes travaillent sur une matrice de nombres réels, où chaque ligne correspond à un objet et chaque colonne correspond à une propriété. Mais cette différence n'est, a priori, qu'une question de point de vue : une matrice de réels n'est rien d'autre qu'un graphe biparti pondéré. Est-ce que ces différents points de vue sous-entendent des hypothèses différentes sur la structure des données ? sur leur densité ? ou sur la signification des valeurs réelles ?

Nous allons voir dans la section suivante que les méthodes que l'on nomme « détection de communautés » sont en quelque sorte le produit du partitionnement de graphe et du clustering : c'est-à-dire la recherche d'un regroupement « naturel » des sommets d'un graphe. Malheureusement il n'existe pas encore, à notre connaissance, de comparaison théorique ou expérimentale entre ces méthodes (en particulier de celles adaptées aux graphes bipartis) et les méthodes de bi-clustering. Une telle comparaison pourrait pourtant se montrer fertile. D'autant plus que certaines approches de bi-clustering se basent sur une binarisation des données, citons par exemple [Serin et Vingron, 2011], ce qui rend le parallèle avec le clustering de graphe encore plus évident.

5.1.3 Détection de communautés sur un graphe biparti

Comme nous l'avons vu au chapitre d'introduction, le problème de détection de communautés est l'un des problèmes maintenant classique de l'analyse des graphes de terrain. Il consiste à rechercher les regroupements de sommets formant des « communautés » intelligibles. Le terme « communauté » provient de l'application de prédilection de ces méthodes aux réseaux sociaux. Nous préférons plus généralement parler de clustering de graphe, en effet la détection de communautés se différencie du clustering principalement par la forme des données initiales. De la même façon que pour le clustering, il n'existe pas une définition formelle universelle de ce que doit être une « communauté ». Presque chaque algorithme propose, plus ou moins implicitement, sa propre définition.

La plupart des méthodes de la littérature s'intéressent aux graphes uni-partis. Quelques méthodes ont tout de même été proposées pour les graphes bipartis. En particulier plusieurs adaptations de la modularité de Newman [Clauset et al., 2004] ont été publiées. La modularité de Newman est une mesure de qualité d'un partitionnement des sommets d'un graphe, plus complexe que la mesure de *cut* utilisée en partitionnement de graphe (voir section 5.1.1). Une première adaptation aux graphes

qui existerait, de manière peu lisible, dans les données.

bipartis a été proposée par Barber [2007]. Une autre adaptation a été développée par Liu et Murata [2010], cette dernière autorise un partitionnement des objets et un partitionnement des propriétés tels que ces deux partitionnements ne soient pas en correspondance directe (à un cluster d'objets ne correspond pas directement un cluster de propriétés).

D'autres approches utilisent les cliques comme noyaux des communautés. Citons en particulier la méthode *CFinder* de Palla et al. [2005] qui consiste, sur un graphe uni-parti, à définir les clusters comme des chaînes de k-cliques adjacentes (une k-clique est une clique de taille k, et deux k-cliques sont adjacentes si et seulement si elles ont, au moins, $k-1$ sommets en commun). Cette méthode a été adaptée aux graphes bipartis par Lehmann et al. [2008] : les clusters deviennent des chaînes de $K_{a,b}$-bicliques adjacentes. Une $K_{a,b}$-biclique étant une biclique comportant a sommets-objets et b sommets-propriétés, et deux $K_{a,b}$-bicliques sont adjacentes si et seulement si elles partagent $a-1$ sommets-objets et $b-1$ sommets-propriétés.

Dans [Navarro et al., 2011], nous avons proposé une adaptation simple de la méthode *Walktrap* de Pons et Latapy [2006]. Cette méthode se base sur un algorithme de clustering hiérarchique utilisant une distance entre groupes de sommets calculée par des marches aléatoires en temps courts semblablement à celles que nous présentons au chapitre 2.

Notons aussi que certaines méthodes prévues pour les graphes uni-partis peuvent être appliquées directement sur des graphes bipartis. Le caractère biparti du graphe est alors ignoré, les sommets étant regroupés indépendamment de leur type.

En particulier, nous utiliserons dans la suite de ce chapitre la méthode *Infomap* [Rosvall et Bergstrom, 2008], qui est reconnue comme l'une des méthodes de clustering de graphe les plus performantes [Lancichinetti et Fortunato, 2009]. La méthode consiste à rechercher les clusters qui compressent au mieux la description de la trajectoire d'un marcheur aléatoire sur le graphe. Cette trajectoire étant décrite sur deux niveaux en fonction des clusters : quand le marcheur entre dans un cluster, le nom du cluster est noté, ensuite seuls les noms des sommets sont inscrits (avec une notation locale au cluster). De cette manière, un même nom court peut être utilisé pour nommer deux sommets s'ils sont dans des clusters différents. Une description concise de la trajectoire est donc possible si les clusters sont tels que le marcheur tend à rester à l'intérieur de ceux-ci. Cela correspond à l'idée que les marcheurs aléatoires sont « capturés » quand ils entrent dans un cluster, puisqu'un cluster n'est que faiblement connecté aux autres. Cette idée a été utilisée de différentes manières dans la littérature sur le clustering de graphe [Pons et Latapy, 2006; Schaeffer, 2007; Delvenne et al., 2010].

5.1.4 Analyse formelle de concepts

L'analyse formelle de concepts (AFC dans la suite) est une méthode d'analyse de données prenant la forme d'une relation binaire entre objets et propriétés. Une telle relation peut être représentée comme un tableau où chaque ligne (resp. colonne) correspond à un objet (resp. une propriété), et une croix entre une ligne et une colonne indique que l'objet correspondant vérifie la propriété correspondante. Un exemple est donné dans le tableau de gauche de la figure 5.1 (page 147).

L'AFC telle que nous la présentons ici a été introduite par Wille [1982]. Notons que les constructions qui forment la base de l'AFC ont été introduites auparavant par Barbut et Monjardet [1970, chap. V]. Nous présentons ici les concepts et définitions de base utiles à l'AFC ainsi que certaines méthodes utilisées pour rendre plus robuste l'AFC. Contrairement aux méthodes abordées jusque là, nous introduisons donc assez précisément l'AFC. Cela nous sera utile pour la suite de ce chapitre. Nous revenons en effet sur l'AFC dans les sections suivantes, en particulier en section 5.2.1. Pour une documentation de référence sur l'AFC nous renvoyons à [Ganter et Wille, 1999].

5.1.4.1 Cadre théorique de l'AFC

Soit R une *relation binaire* entre un ensemble d'objets $\mathbf{O}$ et un ensemble de propriétés $\mathbf{P}$. On note $\mathcal{R} = (\mathbf{O}, \mathbf{P}, R)$ le tuple formé par l'ensemble d'objets, l'ensemble de propriétés et la relation binaire. On nomme *contexte formel* ce tuple $\mathcal{R}$. La notation $(x, y) \in R$ signifie que l'objet x vérifie la propriété y. Soit $R(x) = \{y \in \mathbf{P} | (x, y) \in R\}$ l'ensemble des propriétés vérifiées par l'objet x et $R^{-1}(y) = \{x \in \mathbf{O} | (x, y) \in R\}$ l'ensemble des objets vérifiant la propriété y.

L'AFC définit deux opérateurs ensemblistes, que nous noterons ici $(.)^{\Delta}$ et $(.)^{-1\Delta}$, respectivement appelés *intension* et *extension*, tels que $\forall Y \subseteq \mathbf{P}$ et $\forall X \subseteq \mathbf{O}$:

$$X^{\Delta} = \{y \in \mathbf{P} | \forall x \in X, (x, y) \in R\} \tag{5.1}$$

$$Y^{-1\Delta} = \{x \in \mathbf{O} | \forall y \in Y, (x, y) \in R\} \tag{5.2}$$

L'*intension* de X, X^{Δ} est l'ensemble des propriétés possédées par *tous* les objets de X et l'*extension* de Y, $Y^{-1\Delta}$ est l'ensemble des objets vérifiant *toutes* les propriétés de Y. Ces deux opérateurs induisent une *connexion de Galois* antitone entre $2^{\mathbf{O}}$ et $2^{\mathbf{P}}$. C'est-à-dire que la propriété suivante est vérifiée :

$$X \subseteq Y^{-1\Delta} \Leftrightarrow Y \subseteq X^{\Delta}. \tag{5.3}$$

Une paire telle que $X^{\Delta} = Y$ et $Y^{-1\Delta} = X$ est appelée *concept formel*. X est appelé l'*extension* et Y l'*intension*. Un concept formel est donc une paire (X, Y) telle que X est l'ensemble d'objets vérifiant toutes les propriétés de Y et Y est l'ensemble

de propriétés partagées par tous les objets de X. On montre qu'un concept formel correspond à une *paire* (X, Y) *maximale* telle que :

$$X \times Y \subseteq R. \tag{5.4}$$

Il est possible de définir une relation d'ordre partiel sur l'ensemble des concepts formels. Si (X_1, Y_1) et (X_2, Y_2) sont deux concepts, on a :

$$(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2 \Leftrightarrow Y_2 \subseteq Y_1 \tag{5.5}$$

Avec cet ordre partiel, pour chaque paire de concepts $((X_1, Y_1), (X_2, Y_2))$ il existe un unique concept étant le plus grand des concepts plus petits. C'est le concept défini par les objets $X_1 \cap X_2$ et toutes les propriétés vérifiées par tous ces objets (on parle d'opération *meet*). Symétriquement il existe, pour chaque paire de concepts, un unique concept étant le plus petit des concepts plus grands. C'est le concept défini par les propriétés $Y_1 \cap Y_2$ et tous les objets vérifiant toutes ces propriétés (on parle d'opération *join*). Ces deux opérations *meet* et *join* forment un treillis sur l'ensemble des concepts, que l'on nomme *treillis des concepts*.

Il convient de noter que les notations utilisées ici ne sont pas celles utilisées classiquement dans la littérature de l'AFC. Ces notations proviennent de la théorie des possibilités, et sont introduites ici par souci de cohérence avec la suite de ce chapitre. En effet une extension possibiliste de l'AFC est présentée en section 5.2.1.

5.1.4.2 Approximation de l'AFC

Le principal problème de l'AFC est sa sensibilité au bruit. Le nombre de concepts explose très vite lorsque les données sont bruitées, en particulier si les données sont de grande taille (nous allons le voir dans l'évaluation en section 5.4.3). Cela rend l'AFC quasiment inutilisable telle quelle dans la pratique. Différentes méthodes d'approximation, de post-traitement ou de pré-traitement ont ainsi été proposées. Notons qu'en section 5.4, nous proposons une méthode de pré-traitement permettant de simplifier un contexte pour réduire le nombre de concepts formels produits. Aussi la méthode proposée en section 5.3 peut être vue comme une technique de post-traitement pour simplifier le résultat de l'AFC.

La principale approche permettant de simplifier l'ensemble des concepts formels consiste à sélectionner les concepts pertinents en utilisant différentes mesures [Klimushkin et al., 2010]. La mesure de *stabilité* [Kuznetsov et al., 2007; Jay et al., 2008] est la plus couramment utilisée. Plus un concept (X, Y) a de sous-ensembles X' de X tels que $(X')^{\Delta} = Y$ plus sa *stabilité* est grande. Cela signifie en pratique que ces objets n'ont en général pas de propriété à l'extérieur de Y, ou en d'autres termes que le concept a de grandes chances de rester « stable » si on lui retire quelques objets.

Une telle approche ne fait que sélectionner des concepts formels particuliers parmi les concepts existants. Cela ne permet donc pas de regrouper plusieurs contextes formels séparés uniquement parce que quelques objets ne vérifient pas quelques propriétés. Une autre famille d'approches consiste à regrouper les concepts formels sur la base d'une mesure de similarité entre concepts. On parle de factorisation du treillis des concepts [Belohlavek, 2000; Belohlavek et al., 2007]. Souvent la similarité est basée sur les contextes eux-mêmes mais il est aussi envisageable de définir une similarité utilisant une ressource externe. Formica [2008], par exemple, utilise une ressource lexicale (*WordNet*) pour mesurer une similarité entre concepts.

Il est aussi possible de définir des concepts formels approchés. En effet l'opérateur X^{Δ} peut être assoupli en considérant qu'il recouvre les propriétés partagées par « la plupart » des objets de X plutôt que par tous [Dubois et Prade, 2012]. Cela revient à définir un opérateur $X^{\Delta,k}$ qui autorise au plus k exceptions parmi les objets (en supposant que X comporte plus de k éléments). De la même manière, on peut définir un opérateur $Y^{\Delta,j}$ qui autorise au plus j exceptions parmi les propriétés. Alors un concept formel approché est une paire (X, Y) telle que $Y = X^{\Delta,k}$ et $X = Y^{\Delta,j}$. Un tel concept a au plus k « trous » par colonne et au plus j par ligne. Nous allons voir dans la sous-section suivante que cette idée est utilisée par des méthodes cherchant des *itemset* tolérants aux erreurs.

5.1.5 Recherche d'*itemsets* fréquents

Une autre approche qui doit être citée est la recherche d'*itemsets* fréquents. Ces méthodes sont particulièrement utilisées pour analyser des bases de transactions : l'ensemble des achats effectués dans un magasin. Les méthodes d'extraction de règles d'association cherchent des ensembles d'articles (*itemsets*) fréquemment présents ensemble dans des bases de transactions [Agrawal et al., 1993; Tan et al., 2005]. C'est-à-dire, plus simplement des ensembles d'articles souvent achetés ensemble. La recherche de tels *itemsets* peut être mise en relation avec l'AFC [Pasquier et al., 1999]. La base de transaction est vue comme un contexte formel où chaque transaction correspond à un objet et chaque article dans cette transaction correspond à une propriété vérifiée par l'objet. Alors l'intension de chaque concept formel correspond à un *itemset* fermé. Ensuite les *itemsets* fréquents peuvent être trouvés en filtrant le treillis des concepts. En particulier la règle : $Y_0 \Rightarrow Y \setminus Y_0$ n'a pas d'exception si et seulement si $((Y_0)^{-1\Delta})^{\Delta} = Y$ avec $Y_0 \subset Y$.

La présence de bruit dans les données a motivé le développement de différentes méthodes pour rechercher des *itemsets* tolérants aux erreurs [Gupta et al., 2008]. Brièvement, l'idée est qu'il n'est plus nécessaire que chaque article d'un *itemset* apparaisse absolument dans chaque transaction support. Cette idée est très proche de l'idée de tolérer des « trous » dans les concepts formels (voir sous-section précédente). Une remarque intéressante qui ressort de la littérature à propos des *itemsets* appro-

chés est qu'il est dangereux de considérer uniquement la proportion de « trous » dans les concepts, il faut aussi considérer leurs « positions relatives », c'est-à-dire est-ce que ces « trous » sont tous sur les mêmes objets ou propriétés. On distingue en effet les *itemsets* approchés faibles (*weak error-tolerant itemsets*) et les *itemsets* approchés forts (*strong error-tolerant itemsets*) [Yang et al., 2001]. Dans le premier cas la tolérance est globale, alors que dans le second cas le nombre d'articles manquants est limité sur chaque transaction support, et le nombre de transactions manquantes à un article particulier est, lui aussi, limité. De plus un *itemset* approché peut être plus fortement contraint en imposant qu'une certaine proportion des transactions supports inclue tous les articles de l'*itemset* [Cheng et al., 2006].

Par ailleurs, il est envisageable de regrouper les *itemsets* en se basant sur une mesure de similarité. Cette similarité peut par exemple consister à mesurer à quel point l'*itemset* approché résultant de l'union est dense [Cerf et al., 2009]. Ce type d'approche est à mettre en lien avec la méthode que nous présentons en section 5.3 et avec les techniques de factorisation du treillis des concepts présentées dans la sous-section précédente.

Notons enfin que les méthodes d'extraction d'*itemsets* fréquents ont inspiré des méthodes de bi-clustering (voir section 5.1.2) appliquées en particulier à des données issues de la biologie [Serin et Vingron, 2011].

5.2 De l'analyse formelle de concepts au clustering de graphe biparti

Cette section présente un parallèle formel entre une extension de l'AFC et l'analyse de graphe biparti. Nous présentons, en section 5.2.1, cette extension possibiliste de l'AFC, avant de présenter, en section 5.2.2, comment cette extension de l'AFC peut être comprise en termes de graphes bipartis.

Notons que ce travail a fait l'objet de trois publications : [Gaume et al., 2010], [Gaume et al., 2013] et [Navarro et al., 2012b].

5.2.1 AFC, extension possibiliste

Nous présentons dans cette sous-section une extension de l'AFC qui a été introduite dans [Dubois et al., 2007] et développée dans [Djouadi et al., 2010b; Dubois et Prade, 2012].

Soit $\mathcal{R} = (\mathbf{O}, \mathbf{P}, R)$ un contexte formel comme défini en section 5.1.4.1. Nous avons vu dans cette même section 5.1.4.1 que l'AFC définit deux opérateurs ensem-

blistes, l'*intension* et l'*extension*, tels que $\forall Y \subseteq \mathbf{P}$ et $\forall X \subseteq \mathbf{O}$:

$$X^{\Delta} = \{y \in \mathbf{P} | \forall x \in X, (x,y) \in R\} \quad (5.6)$$
$$Y^{-1\Delta} = \{x \in \mathbf{O} | \forall y \in Y, (x,y) \in R\} \quad (5.7)$$

Un parallèle entre l'AFC et la théorie des possibilités a mis en avant trois autres opérateurs ensemblistes remarquables : $(.)^{\Pi}$, $(.)^{N}$ et $(.)^{\nabla}$. Ces trois opérateurs (et l'opérateur d'intension déjà présenté) peuvent être définis de la manière suivante, $\forall X \subset \mathbf{O}$:

$$X^{\Pi} = \{y \in \mathbf{P} | R^{-1}(y) \cap X \neq \emptyset\} \quad (5.8)$$
$$X^{N} = \{y \in \mathbf{P} | R^{-1}(y) \subseteq X\} \quad (5.9)$$
$$X^{\Delta} = \{y \in \mathbf{P} | R^{-1}(y) \supseteq X\} \quad (5.10)$$
$$X^{\nabla} = \{y \in \mathbf{P} | R^{-1}(y) \cup X \neq \mathbf{O}\} \quad (5.11)$$

Notons que la définition donnée par l'eq. (5.10) est équivalente à celle donné par l'eq. (5.6). Ces opérateurs peuvent être compris de la manière suivante :

- X^{Π} est l'ensemble des propriétés possédées par *au moins un* objet de X,
- X^{N} est l'ensemble des propriétés telles que chaque objet vérifiant l'une d'elles est nécessairement dans X ; ou, dit autrement, X^{N} est l'ensemble des propriétés vérifiées seulement par les objets de X (mais pas forcément par tous),
- X^{Δ} est, comme nous l'avons déjà vu, l'ensemble de propriétés partagées par *tous* les objets de X,
- Et enfin, X^{∇} est l'ensemble des propriétés qui ne sont pas vérifiées par certains objets en dehors de X.

Les opérateurs $(.)^{-1\Pi}$, $(.)^{-1N}$, $(.)^{-1\Delta}$ et $(.)^{-1\nabla}$ sont définis similairement sur un ensemble de propriétés Y en substituant R^{-1} à R et en remplaçant $\mathbf{P}$ par $\mathbf{O}$ et inversement $\mathbf{O}$ par $\mathbf{P}$.

Il est courant de considérer que la relation R est telle que pour toute propriété $y \in \mathbf{P}$: $R^{-1}(y) \neq \emptyset$ et $R^{-1}(y) \neq \mathbf{O}$, ce qui signifie qu'il n'y a pas de propriété vérifiée par aucun objet, ni de propriété vérifiée par tous les objets. On parle de *bi-normalisation*. Avec cette bi-normalisation on a :

$$X^{N} \subseteq X^{\Pi} \quad \text{et} \quad X^{\Delta} \subseteq X^{\nabla} \quad (5.12)$$

Si cette même bi-normalisation est valable sur l'ensemble des objets ($\forall x \in \mathbf{O}$ on a $R(x) \neq \emptyset$ et $R(x) \neq \mathbf{P}$) alors cette propriété tient pour les opérateurs inverses :

$$X^{-1N} \subseteq X^{-1\Pi} \quad \text{et} \quad X^{-1\Delta} \subseteq X^{-1\nabla} \quad (5.13)$$

	1	2	3	4	5	6	7	8
a					×	×	×	×
b					×	×		
c					×	×	×	×
d					×	×	×	×
e							×	
f					×	×		×
g	×	×	×	×				
h	×	×	×	×				
i				×				

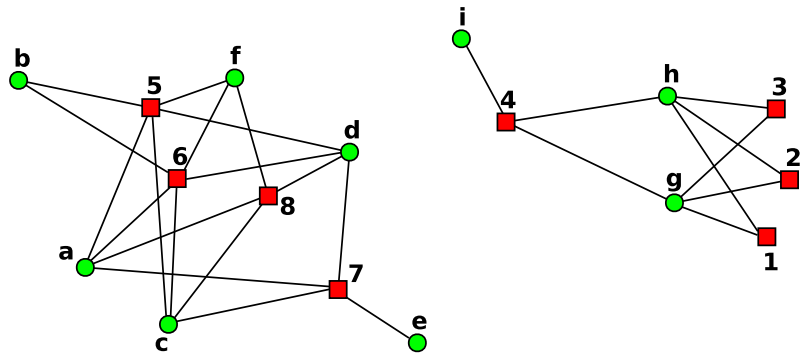


Figure 5.1 – Premier exemple de contexte formel R_1 avec le graphe biparti correspondant

Ces nouveaux opérateurs nous amènent à considérer une nouvelle connexion qui correspond aux paires (X, Y) telles que $X^{\Pi} = Y$ et $Y^{-1\Pi} = X$ ou, de manière équivalente, telles que $X^{N} = Y$ et $Y^{-1N} = X$. Notons que $(.)^{\nabla}$ et $(.)^{\Delta}$ induisent les mêmes [2] paires remarquables correspondant aux concepts formels.

Les paires (X, Y) telles que $X^{\Pi} = Y$ et $Y^{-1\Pi} = X$ forment ce que nous pouvons appeler des *sous-contextes indépendants*. Ces paires sont différentes des contextes formels. En effet, il a été montré [Djouadi et al., 2010b; Dubois et Prade, 2012] que ces paires sont telles que :

$$(X \times Y) \cup (\overline{X} \times \overline{Y}) \supseteq R, \tag{5.14}$$

de la même manière que les concepts formels sont des paires (X, Y) maximales telles que :

$$X \times Y \subseteq R. \tag{5.15}$$

Par exemple, les paires $(\{1, 2, 3, 4\}, \{g, h, i\})$ et $(\{5, 6, 7, 8\}, \{a, b, c, d, e, f\})$ dans la figure 5.1 sont deux *sous-contextes indépendants*, alors que, par exemple, les paires $(\{1, 2, 3, 4\}, \{g, h\})$, $(\{5, 6\}, \{a, b, c, d, f\})$ ou encore $(\{5, 6, 7, 8\}, \{a, c, d\})$ sont des concepts formels. Notons que, en général, il est possible qu'un sous-contexte indépendant puisse être à nouveau découpé en plusieurs sous-contextes indépendants plus petits.

5.2.2 Traduction en théorie des graphes

Nous présentons ici une lecture des différentes notions introduites dans la section précédente en termes de théorie des graphes. Les notions de la théorie des graphes que nous utilisons ici sont définies au premier chapitre, en section 1.2. En plus des

2. Si $X^{\nabla} = Y$ et $Y^{-1\nabla} = X$ alors $(\overline{X}, \overline{Y})$ est un concept formel.

	1	2	3	4	5	6	7	8	9	10	11
a1	×	×	×				×	×			
a2		×	×				×	×		×	
a3	×	×					×	×			
a4	×	×	×				×				
b1		×	×	×	×		×				
b2		×	×		×						
b3			×	×	×						
b4		×	×	×	×						
c1				×	×	×		×			
c2				×	×	×	×	×			
d1									×	×	×
d2									×	×	

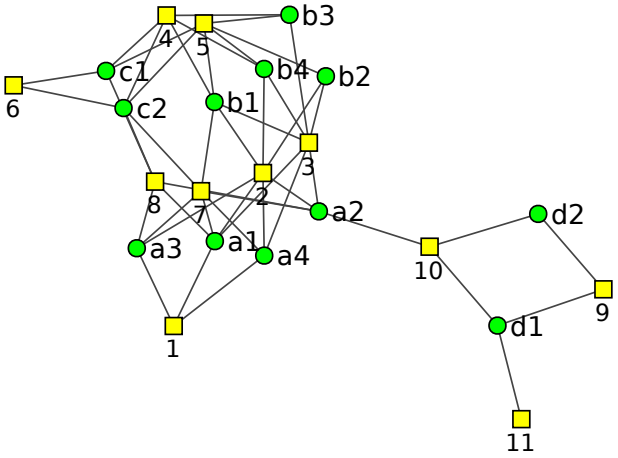


Figure 5.2 – Second exemple de contexte formel R_2 avec le graphe biparti correspondant

notions introduites dans ce premier chapitre, nous utilisons les notations suivantes : dans un graphe $G = (V, E)$ (éventuellement biparti), soit un sous-ensemble de sommets $S \subseteq V$, nous notons alors S^k l'ensemble des sommets connectés à au moins un sommet de S par un chemin de longueur plus petite ou égale à k. Par définition $S^0 = S$. On peut alors observer que $\forall k$, $S^k \subseteq S^{k+1}$. On note S^* l'ensemble des sommets connectés à au moins un des sommets de S (par un chemin de longueur quelconque). On a donc $S^* = \bigcup_{k \geq 0} S^k$. Enfin, il nous semble utile de rappeler que pour un sommet x, $\Gamma(x)$ représente l'ensemble des voisins de x.

5.2.2.1 Du contexte formel au graphe biparti

Pour chaque contexte formel $\mathcal{R} = (\mathbf{O}, \mathbf{P}, R)$, on peut construire un graphe biparti non-dirigé $\mathcal{G} = (V_o, V_p, E)$ tel qu'il y ait une correspondance directe entre : l'ensemble des objets $\mathbf{O}$ et le premier ensemble de sommets V_o, l'ensemble des propriétés $\mathbf{P}$ et le second ensemble de sommets V_p, et entre la relation binaire R et l'ensemble d'arêtes E. On a donc un sommet de V_o pour chaque objet, un sommet de V_p pour chaque propriété, et une arête entre un sommet-objet et un sommet-propriété si et seulement si l'objet correspondant possède la propriété correspondante (d'après R).

Les quatre opérateurs $(.)^\Pi$, $(.)^N$, $(.)^\Delta$ et $(.)^\nabla$ peuvent alors être traduits sur des ensembles de sommets en remplaçant, dans les équations (5.8) à (5.11), $\mathbf{O}$ par V_o, $\mathbf{P}$ par V_p et $R^{-1}(y)$ par $\Gamma(y)$. Les opérateurs $(.)^\Pi$ et $(.)^\Delta$ peuvent aussi être réécrits de la manière suivante :

$$X^\Pi = \cup_{x \in X} \Gamma(x) \tag{5.16}$$

$$X^\Delta = \cap_{x \in X} \Gamma(x) \tag{5.17}$$

Ces dernières notations sont intéressantes car seul le voisinage des sommets de X est utilisé. Cela permet de comprendre plus facilement ces opérateurs en termes de voisinage dans le graphe biparti : X^{Π} est l'union des voisins des sommets de X alors que X^{Δ} en est l'intersection. Notons qu'avec cette écriture (et cette interprétation) il n'y a pas de différence entre $(.)^{\Pi}$ et $(.)^{-1\Pi}$ ni entre $(.)^{\Delta}$ et $(.)^{-1\Delta}$.

Interpréter $(.)^{N}$ et $(.)^{\nabla}$ en termes de voisinage est moins évident. Néanmoins X^{N} peut être compris comme l'union des voisins de X qui n'ont pas de voisin en dehors de X. C'est l'ensemble des sommets connectés exclusivement avec les sommets de X (mais pas nécessairement avec tous). X^{∇} est l'ensemble des sommets-propriétés qui ne sont pas connectés à tous les sommets-objets de $\overline{X}$.

5.2.2.2 Deux connections, deux regards sur les « clusters » dans un graphe

Les connections induites par $(.)^{\Delta}$ et $(.)^{\Pi}$ peuvent aussi se traduire en termes de graphes. Sur le graphe biparti $\mathcal{G} = (V_o, V_p, E)$, avec $X \subseteq V_o$ et $Y \subseteq V_p$ on a :

Proposition 5. *$X = Y^{-1\Delta}$ et $Y = X^{\Delta}$, si et seulement si $X \cup Y$ est une biclique maximale.*

Démonstration. Soit (X, Y) une paire telle que $X = Y^{-1\Delta}$ et $Y = X^{\Delta}$. Pour tous $x \in X$ et $y \in Y$, tels que $Y = \cap_{x \in X} \Gamma(x)$ on a $y \in \Gamma(x)$ donc $\{x, y\} \in E$. Cela signifie que le sous-graphe induit par $X \cup Y$ est complet. De plus il n'existe pas de sommet adjacent à tous les sommets de X (resp. Y) qui ne soit pas dans X^{Δ} (resp. $Y^{-1\Delta}$), donc $X \cup Y$ est une biclique maximale.

Si $X \cup Y$ est une biclique maximale, chaque sommet de X (resp. Y) est adjacent à chaque sommet de Y (resp. X) et il n'existe pas de sommet adjacent à tous les les sommets de X (resp. Y) qui ne soit pas dans Y (resp. X), il est alors évident que $Y = X^{\Delta}$ (resp. $X = Y^{-1\Delta}$). □

Proposition 6. *Pour une paire (X, Y) les deux propositions suivantes sont équivalentes :*

1. $X = Y^{-1\Pi}$ *et* $Y = X^{\Pi}$.
2. *(a)* $(X \cup Y)^{*} = (X \cup Y)$ *et*
   (b) $\forall v \in (X \cup Y), \Gamma(v) \neq \emptyset$.

Démonstration. 1. ⇒ 2.. Par définition $(X \cup Y) \subseteq (X \cup Y)^{*}$. On montre par récurrence que $(X \cup Y)^{*} \subseteq (X \cup Y)$. $(X \cup Y)^{0} \subseteq (X \cup Y)$ est donné par définition. On suppose ensuite qu'il existe k tel que $(X \cup Y)^{k} \subseteq (X \cup Y)$. On peut noter que $(X \cup Y)^{k+1} \subseteq ((X \cup Y)^{k})^{1}$, en considérant qu'un chemin de longueur $k+1$ est un chemin de longueur k suivi par un chemin de longueur 1. Alors $(X \cup Y)^{k+1} \subseteq (X \cup Y)^{1}$.

De plus comme $X = Y^{-1\Pi}$ et $Y = X^{\Pi}$ tous les sommets connectés à $X \cup Y$ par un chemin de longueur 1 sont dans $X \cup Y$. Donc $(X \cup Y)^{k+1} \subseteq (X \cup Y)$. Cela implique par récurrence que $\forall k \geq 0, (X \cup Y)^k \subseteq (X \cup Y)$. Donc $(X \cup Y)^* = \bigcup_{k \geq 0} (X \cup Y)^k \subseteq (X \cup Y)$. Nous avons encore à montrer que chaque sommet v de $X \cup Y$ a au moins un voisin, ce qui est immédiat si on considère que soit $v \in X^{\Pi}$ soit $v \in Y^{-1\Pi}$.

2. $\Rightarrow$ *1.*. Nous montrons que $X = Y^{-1\Pi}$, la preuve est exactement la même pour $Y = X^{\Pi}$. $Y^{-1\Pi}$ est l'ensemble des sommets adjacents à (un moins) un sommet de Y, donc $Y^{-1\Pi} \subset Y^*$ et donc $Y^{-1\Pi} \subset (X \cup Y)^*$. Cela signifie, d'après 2.a, que $Y^{-1\Pi} \subset (X \cup Y)$, or le graphe est biparti, donc $Y^{-1\Pi} \subset X$. Soit x un sommet de X, d'après 2.b[3] x a au moins un voisin v, v est dans X^* et donc dans $(X \cup Y)^*$, donc d'après 2.a : $v \in X \cup Y$, mais le graphe est biparti, donc $v \in Y$. Il est alors évident que $X \subset Y^{-1\Pi}$ et donc $X = Y^{-1\Pi}$. □

Un ensemble de sommets S tel que $S^* = S$ n'est pas exactement une composante connexe maximale mais est un ensemble de sommets déconnectés du reste du graphe. C'est une union de composantes connexes maximales. S'il n'existe pas de sous-ensemble strict S' de S satisfaisant $S'^* = S'$ cela signifie qu'il n'y a pas de sous-ensemble de S déconnecté des autres sommets de S. C'est-à-dire, S est connecté et donc S est une composante connexe maximale. Alors la propriété suivante est évidente :

Proposition 7. *Pour une paire (X, Y) les deux propositions suivantes sont équivalentes :*

1. *$X = Y^{-1\Pi}$, $Y = X^{\Pi}$ et il n'existe pas de sous-ensembles strict $X' \subset X$ et $Y' \subset Y$ tels que $X' = Y'^{-1\Pi}, Y' = X'^{\Pi}$.*

2. *$X \cup Y$ est une composante connexe maximale (qui contient au moins deux sommets).*

D'après les propositions 5 et 7, il est intéressant de noter que les deux connections correspondent aux deux définitions extrêmes de ce que peut être un cluster dans un graphe :

1. un groupe (maximal) de sommets n'ayant **aucun lien manquant à l'intérieur**,

2. un groupe de sommets n'ayant **aucun lien avec l'extérieur**.

D'un coté une biclique maximale est un sous-ensemble de sommets de densité maximale : en un sens, ses sommets ne peuvent être plus proches, et donc on ne

3. Notons que la condition 2.b n'est pas requise si l'on suppose que la relation est bi-normalisée.

peut construire un cluster plus « solide ». De l'autre coté, un ensemble de sommets déconnectés du reste du graphe ne peut être plus clairement séparé des autres sommets.

Ces deux définitions triviales et extrêmes ont déjà été mises en avant par Schaeffer [2007], dans le cadre du clustering de graphe uni-parti.

5.2.2.3 Parallèle avec l'opérateur de diffusion des marches aléatoires

Les marches aléatoires sont un mécanisme fréquemment utilisé tant pour définir des similarités entre sommets (voir chapitre 2) que pour construire des algorithmes de clustering de graphe (voir section 5.1.3). Nous proposons ici un parallèle entre l'opérateur de base utilisé par les marches aléatoires et des généralisations des opérateurs de l'AFC. Ce parallèle n'est malheureusement pas exploité plus en profondeur dans la suite de ce rapport. Cependant, il constitue le début d'une piste certainement fertile à explorer, notamment pour définir un cadre plus général au clustering de graphe.

Une généralisation de l'opérateur de base de l'AFC pour des contextes formels flous a été proposée par Belohlavek [2002] :

$$X^{\Delta}(y) \quad = \quad \min_{x \in \mathbf{O}} \Big(X(x) \to R(x,y) \Big) \tag{5.18}$$

où R est maintenant une relation floue, c'est-à-dire qu'à chaque paire est associée une valeur dans $[0,1]$, et X et X^{Δ} sont des ensembles flous d'objets et de propriétés. L'opérateur $(.)^{\Pi}$ (induisant les sous-contextes indépendants) a été généralisé de la même manière par Dubois et Prade [2009] :

$$X^{\Pi}(y) \quad = \quad \max_{x \in \mathbf{O}} \Big(X(x) * R(x,y) \Big) \tag{5.19}$$

Un choix usuel pour $*$ est l'opérateur min, et l'implication résiduée de Gödel pour $\to$ ($a \to b = 1$ si $a \leq b$ et $a \to b = b$ sinon). Nous revenons sur ces généralisations en sous-section 5.4.1.

D'un autre côté, étant donné la position initiale d'un marcheur aléatoire (fonctionnant comme défini en section 2.2.1) sa position après un pas peut s'écrire :

$$Y^{P}(x) = \sum_{y \in \mathbf{P}} Y(y).\,[P]_{y,x} \tag{5.20}$$

où Y un vecteur de probabilité donnant la position initiale (sur les propriétés) du marcheur, et Y^P sa position après un pas. $[P]_{y,x}$ est le coefficient y, x de la matrice de transition P associée au graphe, c'est-à-dire la probabilité de passer de y à x.

Dans les deux cas, il est intéressant de remarquer que l'on utilise un opérateur de projection d'un ensemble flou de propriétés vers un ensemble flou d'objets (ou d'un vecteur de probabilité sur les propriétés vers un vecteur de probabilité sur les

objets). Les opérateurs inverses (des objets vers les propriétés) n'ont pas été exposés ici, car ils sont semblables. Aussi les ensembles remarquables (clusters, concepts formels, sous-contextes indépendants) sont définis formellement ou approximativement comme des points fixes pour ces opérateurs. En effet les marches aléatoires sont souvent exploitées avec l'idée que les clusters doivent être tels qu'ils « capturent » les marcheurs aléatoires. Un marcheur à l'intérieur d'un cluster doit avoir une faible probabilité d'en sortir [Pons, 2007; Rosvall et Bergstrom, 2008; Delvenne et al., 2010].

5.2.3 Conclusion : vers deux types de clustering ?

Nous avons établi dans cette section une traduction en termes de graphes, des éléments de base de l'AFC étendue. Ce parallèle n'apporte pas d'application algorithmique directe, en revanche il permet, comme nous venons de le voir, de mieux comprendre ce qu'est un cluster dans un graphe biparti. En particulier si l'on considère les deux connections comme des modèles (trop) rigides qui demandent, en pratique, à être assouplis ; alors deux familles d'approches apparaissent :

1. les méthodes recherchant des sous-ensembles d'objets et de propriétés les plus larges *et* cohérents [4] possibles,

2. les méthodes recherchant des sous-ensembles d'objets et de propriétés les plus « indépendants » possible les uns des autres.

Bien entendu, ces deux familles ne forment pas une dichotomie absolue, beaucoup de méthodes se positionnant quelque part entre ces deux points de vue, en évaluant les clusters à la fois sur leur densité et sur leur indépendance. Il est cependant intéressant de relire l'état de l'art établi en section 5.1 pour confronter à cette dichotomie les principales approches de la littérature. Assez clairement l'AFC, ses différentes variantes et la recherche d'*itemset* se classent dans la première famille ; alors que les méthodes de partitionnement de graphe appartiennent à la seconde. Par contre la classification des méthodes regroupées sous les étiquettes « bi-clustering » ou « détection de communautés » ne peut être automatique. Par exemple les méthodes proposant un partitionnement des sommets en se basant sur la modularité ou une métrique similaire appartiennent plutôt à la seconde famille alors que celles proposant de définir les clusters à partir de noyaux formés de bicliques sont plutôt de la première famille.

Il faut bien remarquer que ces deux classes d'approches produisent des structures de clustering différentes. Tout d'abord concernant le nombre de clusters. La recherche de groupes *indépendants* (la seconde famille) amène un nombre de groupes a priori

4. « cohérent » signifiant ici « densément connecté ».

beaucoup plus faible que le nombre de sommets alors que ce n'est pas forcément le cas lorsqu'on recherche des groupes *cohérents* (première famille).

Notons ici qu'une « conceptualisation » amenant autant, ou plus, de groupes qu'il y a de sommets n'est pas forcément inutile ou absurde. Bien sûr une telle méthode ne simplifie pas la représentation d'un jeu de données, mais le résultat peut tout de même avoir du sens et se révéler utile. Sur un graphe lexical par exemple, où les sommets sont des mots/formes, si un cluster (ou, justement, un « concept ») définit un sens non ambigu, il est naturel d'avoir un nombre comparable de clusters et de sommets (c'est le cas dans *Wordnet*[5] qui comporte par exemple, pour les noms : 117 798 formes et 82 115 synsets, ou pour les verbes : 11 529 formes et 13 767 synsets, un synset étant une clique de formes synonymes.).

Une seconde différence concerne le recouvrement entre les clusters. La seconde famille d'approches ne tolère pas ou peu le recouvrement. En effet comment un groupe de sommets peut-il paraître indépendant si chacun de ses membres appartient à d'autres groupes ? À l'inverse les approches de la première famille peuvent engendrer un recouvrement fort, où chaque sommet appartient à plusieurs groupes (comme avec l'AFC par exemple).

Nous tenons toutes ces discussions pour le clustering de graphe *biparti*, pourtant les mêmes commentaires pourraient être faits concernant le clustering de graphe uni-parti.

On peut alors remarquer que sur les graphes bipartis une approche intermédiaire est envisageable. En effet la seconde famille de méthodes, cherchant les groupes de sommets les plus « indépendants » possible, amène à un partitionnement des objets en correspondance directe avec un partitionnement des propriétés. On cherche des groupes d'objets et de propriétés liés « le moins possible » aux autres propriétés ou autres objets. Il est pourtant envisageable de chercher un partitionnement des objets (resp. des propriétés) sans imposer un partitionnement correspondant des propriétés (resp. des objets). Il peut exister un découpage sans recouvrement des objets, sans que celui-ci ne puisse être mis en correspondance directe avec un partitionnement des propriétés. C'est ce cas intermédiaire que nous explorons dans la section suivante.

5.3 Partitionnement des objets, en passant par les concepts

Nous nous intéressons dans cette section au cas particulier où l'on souhaite obtenir un clustering sans recouvrement des objets, sans imposer un partitionnement correspondant sur les propriétés. Nous proposons de passer par une phase d'analyse formelle de concepts avant d'utiliser une méthode de clustering de graphe (pro-

5. Voir http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html

duisant un partitionnement). Nous montrons ainsi qu'un partitionnement pertinent peut être obtenu sur les objets, alors même qu'il n'existe pas de partitionnement correspondant sur les propriétés. Ce travail a fait l'objet de la publication [Navarro et al., 2012b].

Comme nous l'avons vu précédemment, beaucoup de méthodes de clustering de graphe biparti recherchent un partitionnement des objets qui soit en correspondance avec un partitionnement des propriétés. Or ces méthodes échouent généralement lorsque les propriétés sont fortement partagées entre plusieurs groupes d'objets, car elles regroupent tous les objets dans un seul cluster.

Cela est illustré par le graphe de la figure 5.3a qui correspond au contexte formel du tableau 5.1a. Sur cet exemple la méthode de clustering regroupe tous les sommets dans un seul cluster. Les trois groupes d'animaux (mammifères, oiseaux et poissons) ne sont pas retrouvés. Cet exemple est un petit extrait du jeu de données *Zoo* que nous utiliserons en sous-section 5.3.3. On observe, sur cet exemple, que le passage au graphe objets-concepts (figure 5.3b et tableau 5.1b) permet de résoudre le problème convenablement.

5.3.1 Clustering du graphe objets-concepts

Nous décrivons ici en détail la méthode proposée. Tout d'abord, une étape préliminaire consiste à construire le contexte formel associé au graphe biparti entre objets et propriétés, et à utiliser une méthode d'extraction des concepts formels, par exemple [Fu et Nguifo, 2004] ou [Krajca et al., 2008] [6].

Ensuite, un graphe biparti reliant objets et concepts est construit. Chaque objet $o \in \mathbf{O}$ est connecté à un concept (X, Y) si et seulement si $o \in X$. Chacune des arêtes de ce graphe est pondérée par $W(o, (X, Y)) = |Y|$, c'est-à-dire le nombre de propriétés du concept (X, Y) extrémité de l'arête. Ce poids est introduit pour favoriser les concepts les plus grands, que nous espérons plus pertinents. En effet un concept comportant peu de propriétés a plus facilement un nombre (trop) important d'objets. Les concepts extrémités du treillis (c'est-à-dire le concept contenant tous les objets et celui contenant toutes les propriétés) sont ignorés. Cette transformation est illustrée par la table 5.1 et la figure 5.3.

Si le graphe de départ est pondéré par une fonction $w : E \to \mathbb{R}^+$, alors le graphe objets-concepts peut être pondéré de différentes manières en agrégeant les poids du

6. Nous avons utilisé la méthode présenté dans [Krajca et al., 2008] et dont une implémentation est disponible à l'adresse suivante : http://fcalgs.sourceforge.net/

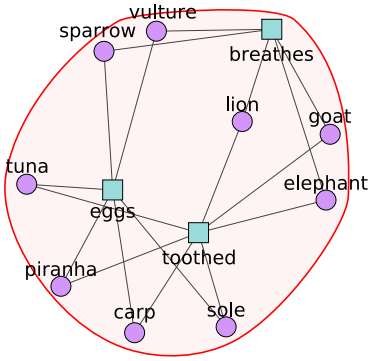
Table 5.1 – Contextes formels entre 9 animaux et 3 propriétés, et entre les 9 animaux et les 6 concepts formels issus du premier contexte. Ces contextes correspondent aux graphes de la figure 5.3. Les couleurs indiquent le regroupement attendu.

(a) Contexte formel objets-propriétés.

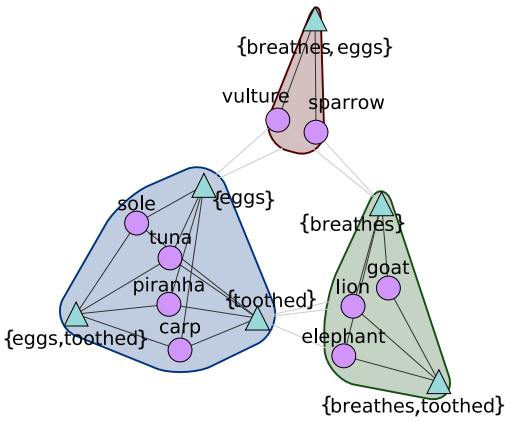
	breathes	eggs	toothed
elephant	×		×
goat	×		×
lion	×		×
sparrow	×	×	
vulture	×	×	
carp		×	×
piranha		×	×
sole		×	×
tuna		×	×

(b) Contexte formel objets-concepts.

	{breathes}	{eggs}	{toothed}	{breathes & eggs}	{breathes & toothed}	{eggs & toothed}
elephant	×		×		×	
goat	×		×		×	
lion	×		×		×	
sparrow	×	×		×		
vulture	×	×		×		
carp		×	×			×
piranha		×	×			×
sole		×	×			×
tuna		×	×			×



(a) graphe objets-propriétés, tous les sommets sont regroupés dans le même cluster.



(b) graphe objets-concepts, les 3 groupes d'animaux sont bien retrouvés.

Figure 5.3 – Résultat d'*Infomap* [Rosvall et Bergstrom, 2008] sur un petit jeu de données représentant 9 animaux avec 3 propriétés. Ces graphes correspondent aux contextes formels de la table 5.1. La méthode de clustering est appliquée soit directement sur le graphe objets-propriétés (a) soit sur le graphe objets-concepts (b).

graphe de départ. Par exemple les pondérations suivantes sont imaginables :

$$W_{sum}(o,(X,Y)) = \sum_{y\in Y} w(o,y) \tag{5.21}$$

$$W_{min}(o,(X,Y)) = \min_{y\in Y} w(o,y) \tag{5.22}$$

$$W_{max}(o,(X,Y)) = \max_{y\in Y} w(o,y) \tag{5.23}$$

$$W_{prod}(o,(X,Y)) = \prod_{y\in Y} w(o,y) \tag{5.24}$$

$$W_{avg}(o,(X,Y)) = \frac{1}{|Y|}\sum_{y\in Y} w(o,y) \tag{5.25}$$

Notons que $W_{sum}(o,(X,Y))$ peut être compris comme une généralisation du poids que nous proposons dans le cas où le graphe de départ n'est pas pondéré. Dans le chapitre suivant (section 6.4.3) nous évaluons ces différentes pondérations.

Finalement, l'ensemble des sommets du graphe objets-concepts est partitionné avec la méthode de clustering de graphe *Infomap* [Rosvall et Bergstrom, 2008]. Nous décrivons cette méthode en section 5.1.3. Notons, à nouveau, que *Infomap* n'a pas spécialement été conçu pour les graphes bipartis. La méthode ne prend simplement pas en compte le fait que le graphe soit biparti.

Comme on peut le voir sur la figure 5.3b, *Infomap* appliqué sur le graphe objets-concepts donne maintenant, si on considère les objets (les animaux), les trois clusters attendus.

5.3.2 Étiquetage des clusters

Pour donner à chaque cluster d'objets un label fait d'un sous-ensemble de propriétés, nous proposons la méthode suivante.

Soit X un cluster d'objets et l'on note T l'ensemble des concepts formels. Pour chaque cluster d'objets on cherche deux ensembles de concepts particuliers : l'ensemble T^*, formé des concepts (X^*,Y^*) composés des plus petits sur-ensembles d'objets contenant le cluster, c'est-à-dire :

$$(X^*,Y^*)\in T \quad \text{t.q.} \quad \begin{cases} X\subseteq X^* \\ \nexists (X',Y')\in T \quad \text{tel que} \quad X\subset X'\subseteq X^* \end{cases} \tag{5.26}$$

et l'ensemble T_*, formé des concepts (X_*,Y_*) composés des plus grands des sous-ensembles d'objets contenus dans le cluster, c'est-à-dire :

$$(X_*,Y_*)\in T \quad \text{tel que} \quad \begin{cases} X_*\subseteq X \\ \nexists (X',Y')\in T \quad \text{t.q.} \quad X_*\subset X'\subseteq X \end{cases} \tag{5.27}$$

On peut vérifier que $\forall (X^*,Y^*)\in T^*$ et $\forall (X_*,Y_*)\in T_*$, on a :

$$X_*\subseteq X\subseteq X^* \quad et \quad Y^*\subseteq Y_* \tag{5.28}$$

Il est alors possible d'utiliser les propriétés des concepts de T^* et de T_* pour décrire le cluster. Notons que toutes les propriétés des ensembles Y^* des concepts de T^* sont vérifiées par *tous* les objets du cluster.

5.3.3 Évaluation

Nous proposons d'évaluer cette procédure de clustering sur deux jeux de tests : sur des graphes artificiels, puis sur un graphe « réel » issu de la littérature.

5.3.3.1 Évaluation sur des graphes artificiels

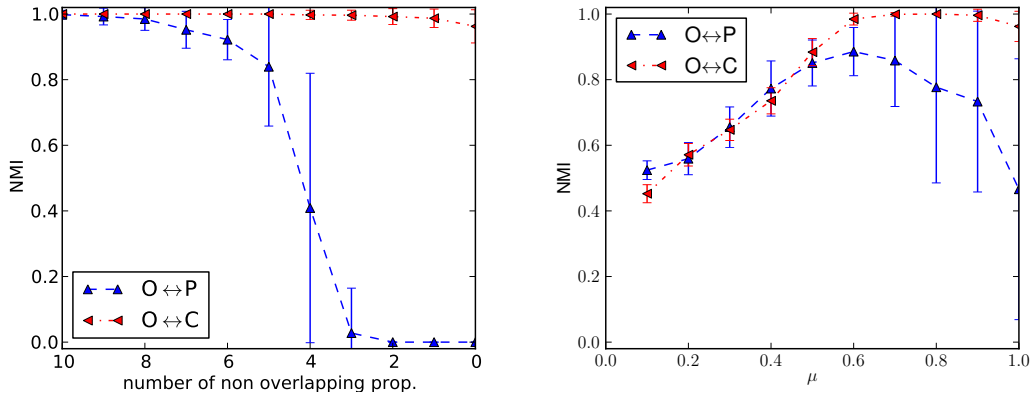
Pour évaluer notre procédure nous construisons des contextes formels artificiels de la manière suivante :

- On pose n groupes de k objets,
- chaque groupe est associé à :
  - m_{own} propriétés spécifiques (c'est-à-dire seuls les objets de ce groupe peuvent les vérifier),
  - m_{shared} propriétés partagées par s autres groupes (c'est-à-dire que les objets de $s+1$ groupes peuvent les vérifier) ;
- chaque objet du groupe satisfait chaque propriété du groupe (spécifique ou partagée) avec une probabilité μ.

Un exemple d'un tel contexte formel est donné par le tableau 5.2.

Table 5.2 – Exemple de contexte formel artificiel construit pour l'évaluation par la procédure décrite en sous-section 5.3.3, avec $n = 3$, $k = 3$, $m_{own} = 2$, $m_{shared} = 4$, $s = 1$, $\mu = 0.8$. Chaque ligne correspond à un objet (a*, ..., c*) et chaque colonne correspond à une propriété (A*, ..., C* pour celles spécifiques et AB*, ..., CA* pour celles partagées).

	A0	A1	B0	B1	C0	C1	AB0	AB1	BC0	BC1	CA0	CA1
a0	×						×	×			×	×
a1		×					×	×				×
a2	×	×					×	×			×	
b0				×				×	×	×		
b1			×	×			×	×	×	×		
b2			×				×		×	×		
c0						×			×	×	×	×
c1					×	×			×	×	×	×
c2					×	×			×	×		



(a) Résultats en fonction de n_{own}, avec $\mu = 0.8$ **(b)** Résultats en fonction de μ, avec $m_{own} = 5$

Figure 5.4 – Valeur de l'information mutuelle normalisée (NMI) [Danon et al., 2005] de la méthode de clustering *Infomap* appliquée directement sur les graphes objets-propriétés (courbe O ↔ P, triangles vers le haut en bleu) et sur les graphes objets-concepts (courbe O ↔ C, triangle vers la gauche en rouge). Les contextes (i.e. graphes objets-propriétés) sont tous construits avec les paramètres suivants : $n = 5, k = 10$, $m_{shared} = 6$, $s = 2$, les autres paramètres sont indiqués sur les courbes. Chaque point indique la valeur moyenne obtenue sur 50 réalisations, l'écart type est indiqué par les barres verticales sur chaque point.

Pour évaluer la qualité du résultat d'une méthode par rapport au partitionnement correct (ici donné par construction) nous utilisons l'information mutuelle normalisée (NMI, pour *normalised mutual information* en anglais). Une valeur de 0 indique que les deux partitions sont complètement dissimilaires, alors qu'une valeur de 1 indique que les deux partitions sont identiques. Cette mesure est très populaire dans la littérature du clustering de graphe [Danon et al., 2005].

Les figures 5.4a et 5.4b présentent les résultats du clustering sur le graphe objets-propriétés (la courbe O ↔ P) et sur le graphe objets-concepts (la courbe O ↔ C). Comme le montre la figure 5.4a, les résultats restent stables avec notre approche quand le nombre de propriétés partagées entre les différents groupes d'objets augmente. On voit à l'inverse que ce n'est pas le cas quand on travaille directement sur le graphe objets-propriétés.

5.3.3.2 Jeu de données UCI Zoo

Le jeu de données *Zoo* décrit 101 animaux sur 16 propriétés booléennes et un attribut numérique (le nombre de pattes). Nous avons transformé cet attribut numérique en 7 propriétés booléennes (pas de patte, une patte, deux pattes, etc.). Pour chaque animal une classe est indiquée, il y a 7 classes d'animaux : mammifères (*mammal*), oiseaux (*bird*), reptiles (*reptile*), poissons (*fishes*), amphibiens (*amphibians*),

insectes (*insects*), invertébrés (*invertebrates*). Ce jeu de données est disponible sur le dépôt « UCI Machine Learning Repository [7] ».

La Table 5.4 montre les résultats du clustering sur le graphe objets-propriétés, alors que la Table 5.3 montre les résultats sur le graphe objets-concepts. On peut voir que le clustering échoue sur le graphe objets-propriétés (NMI proche de zero), alors que la partition calculée sur le graphe objets-concepts correspond presque exactement aux classes des animaux ($NMI = 0.81$).

5.3.4 Conclusions et perspectives

Nous avons dans cette section présenté une méthode permettant de partitionner un ensemble d'objets dans un graphe biparti sans imposer de partitionnement des propriétés. Cette méthode repose sur l'utilisation de l'AFC pour remplacer dans le graphe l'ensemble des propriétés par l'ensemble des concepts. Ce nouveau graphe objets-concepts est ensuite partitionné avec une méthode de clustering de graphe récente. Nous avons montré sur deux évaluations l'intérêt de la méthode, en particulier quand il existe un fort recouvrement des propriétés entre différents groupes d'objets.

Il serait intéressant de poursuivre cette évaluation en particulier sur des graphes réels plus grands que celui utilisé ici. Notons que nous présentons au chapitre suivant une utilisation de cette méthode sur des graphes bipartis documents/termes.

Aussi, il serait souhaitable de comparer expérimentalement notre approche avec les quelques méthodes de l'état de l'art proposant de partitionner un groupe de sommets sans imposer de partitionnement correspondant sur l'autre groupe. En particulier les méthodes basées sur l'adaptation de la modularité de Liu et Murata [2010], ou la méthode utilisant les *hypercliques* de Hu et al. [2006]. Malheureusement, aujourd'hui, aucune de ces méthodes ne semble avoir d'implémentation disponible.

Nous présentons dans la section suivante une méthode de prétraitement d'un graphe biparti (ou d'un contexte formel) qui peut être utilisée en amont de la méthode que nous venons de présenter. C'est ce que nous ferons sur une application en recherche d'information au chapitre suivant (section 6.4).

7. http://archive.ics.uci.edu/ml/datasets/Zoo

Table 5.3 – Résultats du clustering avec *Infomap* du graphe objets-concepts pour le jeu de données *Zoo* (on a 10 clusters et un score $NMI = 0.81$). Les animaux sont regroupés, dans chaque cluster, par leur catégorie dans le jeu de données.

$Y^* =$	$\{backbone, breathes, hair, milk, toothed\}$
$Y_* =$	$\{backbone, breathes, hair, milk, tail, toothed\}$
Mammals :	aardvark, lynx, leopard, bear, boar, puma, lion, cheetah, raccoon, mink, pussycat, mongoose, wolf, polecat, antelope, calf, elephant, oryx, goat, deer, reindeer, buffalo, pony, giraffe, vole, mole, hare, cavy, hamster, opossum, sealion, girl, wallaby, gorilla, fruitbat, squirrel, vampire
$Y^* =$	$\{0legs\}$
$Y_* =$	$\{0legs, aquatic, eggs\}$
Fishes :	stingray, pike, piranha, catfish, herring, dogfish, tuna, chub, bass, sole, seahorse, carp, haddock
Invertebrates :	clam, seawasp
Reptiles :	seasnake
$Y^* =$	$\{2legs, backbone, breathes, eggs, feathers, tail\}$
$Y_* =$	$\{2legs, backbone, breathes, eggs, feathers, predator, tail\}$
Birds :	flamingo, gull, skimmer, sparrow, wren, skua, hawk, crow, duck, vulture, lark, swan, pheasant, kiwi, rhea, ostrich, penguin
$Y^* = Y_* =$	$\{4legs, eggs\}$
Amphibians :	newt, frog2, frog1, toad
Reptiles :	tortoise, tuatara
Mammals :	platypus
Invertebrates :	crab
$Y^* = Y_* =$	$\{0legs, aquatic, backbone, breathes, catsize, fins, milk, predator, toothed\}$
Mammals :	porpoise, dolphin, seal
$Y^* = Y_* =$	$\{6legs, breathes, eggs\}$
Insects :	flea, ladybird, moth, gnat, wasp, honeybee, housefly, termite
$Y^* = Y_* =$	$\{0legs, breathes, eggs\}$
Reptiles :	slowworm, pitviper
Invertebrates :	worm, slug
$Y^* = Y_* =$	$\{2legs, airborne, backbone, breathes, domestic, eggs, feathers, tail\}$
Birds :	chicken, parakeet, dove
$Y^* =$	$\{aquatic, eggs, predator\}$
$Y_* =$	$\{6legs, aquatic, eggs, predator\}$
Invertebrates :	crayfish, starfish, lobster
$Y^* = Y_* =$	$\{8legs, breathes, predator, tail, venomous\}$
Invertebrates :	scorpion
$Y^* = Y_* =$	$\{8legs, aquatic, catsize, eggs, predator\}$
Invertebrates :	octopus

Table 5.4 – Résultats du clustering du graphe objets-propriétés pour le jeu de données *Zoo* (seulement 2 clusters et $NMI = 0.02$). Seuls les objets des clusters (les animaux) sont présentés. Les animaux sont regroupés, dans chaque cluster, par leur catégorie dans le jeu de données.

Mammals :	aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby, wolf
Birds :	chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren
Fishes :	bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna
Invertebrates :	clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, worm
Insects :	flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp
Reptiles :	pitviper, seasnake, slowworm, tortoise, tuatara
Amphibians :	frog1, frog2, newt, toad
Invertebrates :	starfish

5.4 Pré-traitement par marches aléatoires

Nous présentons dans cette section une méthode basée sur une mesure de similarité pour simplifier un graphe biparti en vue d'une AFC. Avant de présenter cette méthode en sous-section 5.4.2, nous proposons dans la sous-section 5.4.1 un rapide état de l'art des extensions de l'AFC aux données pondérées. Enfin la sous-section 5.4.3 montre l'intérêt de la méthode avec une évaluation sur des graphes artificiels et sur des graphes réels.

5.4.1 Généralisations de l'AFC sur données pondérées ou incomplètes

Il existe différentes motivations pour gérer des contextes formels pondérés, c'est-à-dire un contexte formel où les liens entre objets et propriétés sont pondérés. En effet ces poids peuvent être compris de différentes manières. Tout d'abord ils peuvent représenter l'idée que les propriétés sont graduelles. Un autre type de pondération peut correspondre à la situation où les propriétés sont toujours binaires mais connues avec incertitude. Dans les deux cas, on suppose que cette information graduelle ou incertaine est disponible.

La première approche, qui a été la plus explorée jusqu'à présent, consiste à considérer que les objets vérifient les propriétés jusqu'à un certain degré. Dans cette optique, des généralisations de l'AFC ont été proposées par Belohlavek [2002], elles sont basées sur l'opérateur :

$$X^{\Delta}(y) = \bigwedge_{x \in \mathbf{O}} \Big(X(x) \to R(x,y) \Big) \tag{5.29}$$

où R est maintenant une relation floue, et X et X^{Δ} sont des ensembles flous d'objets et de propriétés. $\bigwedge$ est l'opérateur de conjonction min et $\to$ un opérateur d'implication. Un choix possible de connecteur (l'implication résiduée de Gödel : $a \to b = 1$ si $a \leq b$, et $a \to b = b$ si $a > b$) permet toujours de voir un concept formel flou à travers ses différents niveaux de coupe X_α, Y_α tels que $(X_\alpha \times Y_\alpha) \subseteq R_\alpha$ avec $X_\alpha \times Y_\alpha$ maximal et $R_\alpha = \{(x,y) | R(x,y) \geq \alpha\}$, $X_\alpha = \{x \in \mathbf{O} | X(x) \geq \alpha\}$, $Y_\alpha = \{y \in \mathbf{P} | Y(y) \geq \alpha\}$.

La seconde approche correspond à l'idée d'incertitude [Djouadi et al., 2010a], [Djouadi et al., 2010b]. La représentation possibiliste de l'incertitude consiste à associer à chaque lien (x,y) une paire de nombres (α, β) tels que $\alpha, \beta \in [0,1]$ et tels que $\min(\alpha, \beta) = 0$. Ces nombres indiquent respectivement à quel point il est certain que ce lien existe (α) ou n'existe pas (β). Un lien dans l'analyse formelle de concepts classique correspond à une paire $(1,0)$, l'absence de lien à une paire $(0,1)$, et une paire $(0,0)$ modélise l'ignorance complète à propos de l'existence ou non d'un lien. Sur cette base, plus α (resp. β) est grand, plus un lien peut être facilement ajouté (resp. supprimé).

Table 5.5 – Exemple de pré-traitement en utilisant la confluence. Le contexte (b) est le résultat du filtrage du contexte (a) par la méthode décrite par l'eq. (5.32) avec $t = 3$, $\tau^+ = 0.5$ et $\tau^- = 0.4$.

(a) Contexte formel original

	1	2	3	4	5	6	7	8	9	10	11
a1	×	×	×	×	×		×				
a2	×	×		×	×						
a3	×		×	×	×						×
a4		×	×	×	×						
b1				×	×	×		×			
b2				×	×	×	×	×			
b3				×	×	×	×				
c1							×		×	×	×
c2									×	×	
c3		×							×	×	×

(b) Après le pré-traitement

	1	2	3	4	5	6	7	8	9	10	11
a1	×	×	×	×	×						
a2	×	×	×	×	×						
a3	×	×	×	×	×						
a4	×	×	×	×	×						
b1				×	×	×	×	×			
b2				×	×	×	×	×			
b3				×	×	×	×	×			
c1									×	×	×
c2									×	×	×
c3									×	×	×

On peut aussi considérer que certaines propriétés sont moins importantes, ou que certains objets sont plus typiques [Djouadi et al., 2010a]. Les poids ne sont plus sur les liens mais sur les sommets. Oublier une propriété non primordiale (par exemple la capacité de voler pour un oiseau) peut alors aider à construire un concept plus large (par exemple la classe des oiseaux, bien que les oiseaux typiques volent).

Ces trois vues nécessitent différents types de données additionnelles qui ne sont pas toujours disponibles. De plus les deux dernières vues ont été introduites récemment et n'ont pas été explorées en détail.

5.4.2 Filtrage et binarisation d'un graphe biparti éventuellement pondéré

Nous proposons dans cette section une méthode qui consiste à calculer une similarité pour chaque paire (objet, propriété) dans le graphe biparti (i.e. contexte formel) éventuellement pondéré. Nous utilisons pour cela une mesure de similarité entre sommets comme celles définies au chapitre 2, mais adaptée aux graphes bipartis. Nous proposons de supprimer les arêtes dont la similarité (entre les sommets extrémités) est inférieure à un certain seuil et d'ajouter les paires non-adjacentes ayant une similarité supérieure à un second seuil. Un exemple est donné par la table 5.5 : la méthode transforme le contexte formel 5.5a en un contexte formel plus simple présenté dans le tableau 5.5b.

L'objectif est donc de transformer le graphe (ou contexte formel) initial pour réduire le nombre de concepts formels associé. En effet, un lien manquant dans une zone dense (a priori de forte similarité) amène à diviser ce qui pourrait être un concept clair en plusieurs petits concepts se recouvrant fortement. À l'inverse un lien dans une zone peu dense (faible similarité) crée plusieurs concepts non pertinents.

Notons que ce pré-traitement peut être vu de deux manières : soit comme une tentative de correction d'une donnée bruitée, soit comme une modification de données pourtant exactes en vue d'en simplifier la représentation. Lors d'une première exploration, il peut par exemple être pertinent de considérer que le *kiwi* vole pour simplifier la représentation de la classe des oiseaux, même si cela est faux.

La méthode permet aussi de prendre en considération une pondération disponible sur les liens (pondération indiquant à quel degré un objet vérifie une propriété). Il est possible d'appliquer un seuil directement sur le graphe pondéré, mais notre approche permet de normaliser et de lisser ces poids et de rendre ainsi la coupe cohérente [8].

5.4.2.1 Calcul des confluences

Soit un graphe biparti $G = (V_o \cup V_p, E)$, avec V_o un ensemble d'objets et V_p un ensemble de propriétés. Ce graphe peut être pondéré par une fonction $w : E \to \mathbb{R}^+$.

Pour calculer une similarité entre les sommets de V_o et de V_p nous utilisons une adaptation aux graphes bipartis de la *confluence* présentée au chapitre 2 (sous-section 2.2.2). Comme nous souhaitons calculer une similarité entre objets et propriétés, nous utilisons des marches en temps impairs $t = 2l + 1$ (nous renvoyons à la sous-section 2.2.1.3 pour plus de détails concernant les particularités des marches aléatoires sur des graphes bipartis.). Aussi, pour éviter que les valeurs des paires adjacentes soient biaisées, nous utilisons l'astuce consistant à supprimer du graphe l'arête reliant deux sommets adjacents avant de calculer leur similarité. La confluence entre un objet o et une propriété p *non adjacents* vaut donc :

$$\{o,p\} \notin E \ : \quad \mathrm{confl}_{2l+1}(o,p) = \frac{\left[P^{2l+1}\right]_{o,p}}{\left[P^{2l+1}\right]_{o,p} + \pi(p)} \tag{5.30}$$

P étant la matrice d'adjacence (éventuellement pondérée) du graphe, et $\pi(p)$ la probabilité limite d'atteindre le sommet p en un nombre impair de pas avec une probabilité de départ quelconque mais limitée à l'ensemble des objets. Cette limite vaut, si le graphe est pondéré :

$$\pi(p) = \frac{\sum_{i \in \Gamma(p)} w(p,i)}{\sum_{q \in V_p} \sum_{a \in \Gamma(q)} w(q,a)} \tag{5.31}$$

Pour un objet o et une propriété p *adjacents*, la confluence vaut :

$$\{o,p\} \in E \ : \quad \mathrm{confl}_{2l+1}(o,p) = \frac{\left[P^{2l+1}_{\{o,p\}}\right]_{o,p}}{\left[P^{2l+1}_{\{o,p\}}\right]_{o,p} + \pi_{\{o,p\}}(p)} \tag{5.32}$$

8. La normalisation que nous proposons n'est cependant pas la seule imaginable et n'est certainement pas celle à adopter dans tous les cas.

avec $P_{\{o,p\}}$ la matrice d'adjacence de $G' = (V_o \cup V_p, E \setminus \{o,p\})$. La limite $\pi_{\{o,p\}}(p)$ correspond à $\pi(p)$ mais calculée sur le graphe G'. Pour le calcul des limites, lorsque le graphe n'est pas pondéré, il suffit de considérer que w est une fonction constante égale à 1. Remarquons que les équations de cette section utilisent « $2l+1$ » ; or pour présenter les résultats nous indiquons t plutôt que l. On a toujours $t = 2l + 1$.

D'autres similarités peuvent être utilisées (voir chapitre 2) mais cette mesure a l'avantage d'être symétrique, comprise entre 0 et 1 et d'avoir un seuil naturel à 0.5. En effet une valeur au-dessus de 0.5 indique que la probabilité qu'un marcheur passe de l'un des sommets à l'autre en $t = 2l + 1$ pas est plus forte que la probabilité qu'a le marcheur d'atteindre l'un de ces deux sommets après une marche de temps infini. Une valeur de confluence supérieure à 0.5 indique donc que les deux sommets se trouvent dans une même zone dense en arêtes. À l'inverse une valeur inférieure à 0.5 indique qu'il est moins probable de passer de l'un des sommets à l'autre en temps court qu'en temps long, et donc que ces deux sommets ne se trouvent pas dans une même zone de sur-densité. Cela indépendamment du fait que les deux sommets soient reliés ou non par une arête.

5.4.2.2 Ajout et suppression d'arêtes

Plusieurs options sont possibles pour obtenir un nouveau graphe biparti à partir de ces valeurs de confluence. Une première solution consiste à ne garder comme arête que les paires ayant une confluence supérieure à un certain seuil :

$$E_{cut-} = \{(o,p) \in E \mid \text{confl}_{2l+1}(o,p) > \tau^-\} \tag{5.33}$$

Cette méthode ne fait que supprimer des arêtes.

Il est aussi possible d'ajouter toutes les paires ayant une confluence au-dessus d'un certain seuil :

$$E_{cut+} = E \cup \{\, (o,p) \in V_o \times V_p \mid \text{confl}_{2l+1}(o,p) > \tau^+\} \tag{5.34}$$

On ne fait alors qu'ajouter des arêtes. Enfin, il est aussi possible d'ajouter et de supprimer des arêtes :

$$\begin{aligned} E_{cut+-} = &\{(o,p) \in E \mid \text{confl}_{2l+1}(o,p) > \tau^-\} \\ &\cup \{\, (o,p) \in V_o \times V_p \setminus E \mid \text{confl}_{2l+1}(o,p) > \tau^+\} \end{aligned} \tag{5.35}$$

Et ce éventuellement avec un seuil différent pour l'ajout et la suppression.

5.4.3 Évaluation

Nous présentons ici une première évaluation de la méthode, sur des graphes artificiels et sur des graphes réels. Nous étudions d'abord, sur deux séries de graphes

artificiels, l'effet d'un ajout d'arêtes aléatoires (figure 5.5 et 5.6) puis d'une suppression aléatoire des arêtes (figure 5.7 et 5.8). Avant d'être ainsi bruités, les graphes sont construits par la méthode présentée en sous-section 5.3.3.1 : une série « a » de graphes présentent 5 groupes de 10 sommets (figures « a ») et une série « b » présentent 15 groupes de 10 sommets (figures « b »). Les autres paramètres sont les mêmes dans les deux cas et sont donnés en légende de la figure 5.5. Notons que dans les deux séries de graphes les groupes de sommets ont la même taille et sont reliés au même nombre de propriétés. Aussi le degré moyen des sommets est le même. La seule différence est que les graphes « b » sont plus grands (et donc moins denses) que les graphes « a ».

Sur chacune des figures présentées (figure 5.5 à 5.8), le premier tracé représente le score de Jaccard de l'ensemble des arêtes du graphe de départ calculé par rapport aux arêtes du graphe bruité (courbe en triangles rouges) et calculé par rapport à l'ensemble d'arêtes du graphe filtré (courbe en ronds verts). Le second tracé représente le nombre d'arêtes du graphe de départ (courbe de carrés bleus), du graphe bruité (courbe en triangles rouges) et du graphe filtré (courbe en ronds verts). Enfin le dernier tracé indique le nombre de concepts formels ($|T|$) issus du graphe original, du graphe bruité et du graphe filtré. Chacun des points indique la moyenne pour 20 réalisations, les barres verticales autour des points indiquant l'écart type.

Dans un second temps, nous utilisons des graphes mots-documents issus de l'application présentée au chapitre suivant. Sur 25 graphes construits à partir de différentes requêtes, nous comparons le nombre de concepts obtenus sans pré-traitement, ou avec un pré-traitement consistant uniquement en une suppression des arêtes, uniquement en un ajout, ou en un ajout et une suppression.

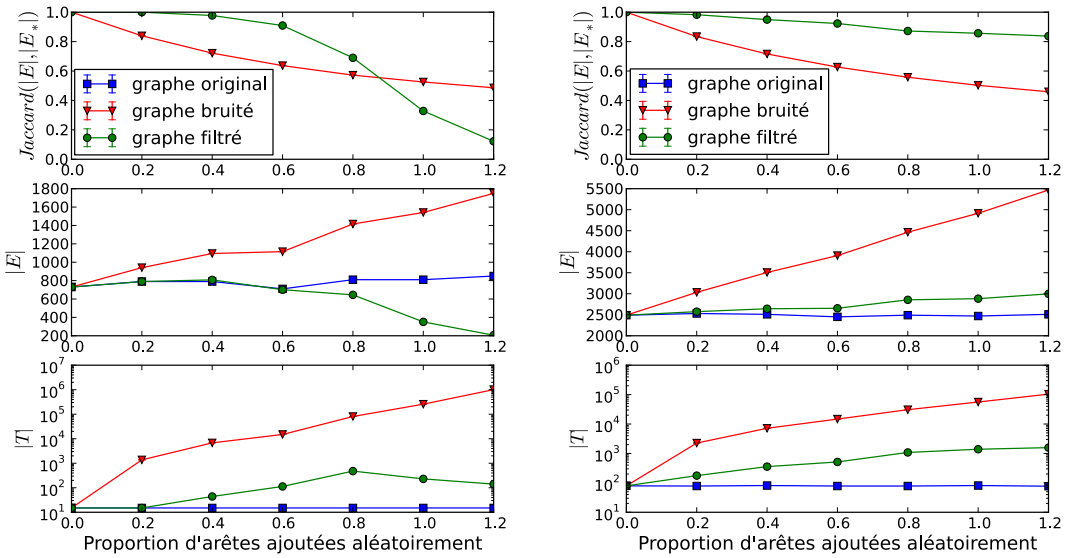
5.4.3.1 Supprimer des arêtes bruits

Nous évaluons dans un premier temps la méthode de filtrage face à des graphes bruités par un ajout d'arêtes aléatoires. Les graphes sont bruités par un ajout d'arêtes aléatoires, avant d'être filtrés par notre méthode. Le filtrage est configuré pour supprimer les arêtes ayant un score de confluence (calculé avec $t = 3$) inférieur à un certain seuil. Ici, aucune arête ne peut être ajoutée par la procédure de filtrage. Dans une première expérience le seuil est fixe ($\tau^- = 0.5$) et le bruit varie (figure 5.5), dans une seconde expérience le bruit est fixe et c'est le seuil qui varie (figure 5.6).

Sur la figure 5.5, on observe tout d'abord (sur le dernier tracé) que le nombre de concepts explose [9] dès lors qu'un peu de bruit aléatoire est présent.

On remarque ensuite que le Jaccard entre l'ensemble des arêtes au départ et après filtrage reste assez haut et cela même quand ont été ajoutées autant d'arêtes « bruit » qu'il y a d'arêtes au départ. La méthode de filtrage fonctionne donc bien sur ces exemples. On note toutefois que le filtrage ne supprime pas toutes les arêtes

9. Notons que les échelles des ordonnées de ces derniers tracés sont logarithmiques.



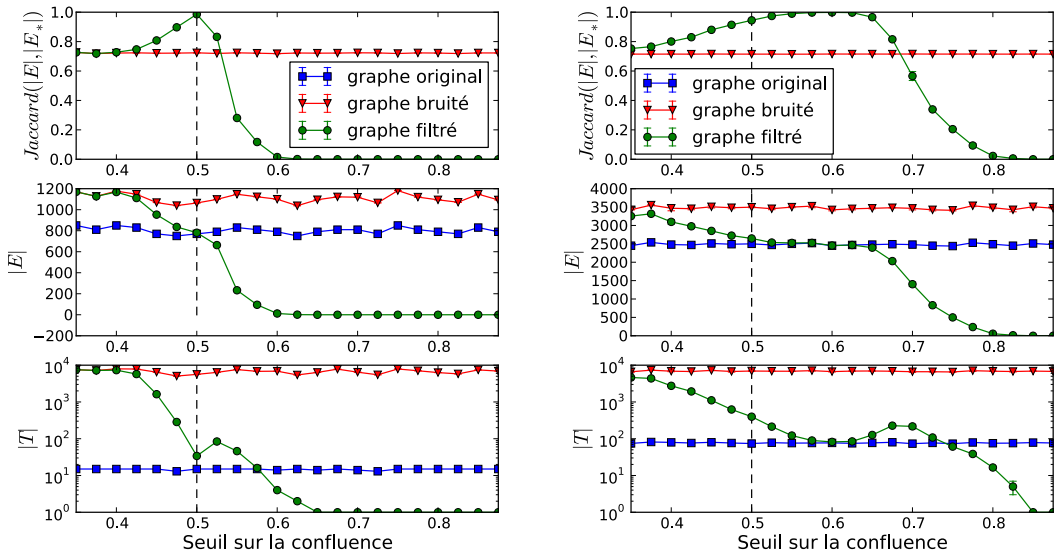
(a) 5 groupes de 10 sommets ($n = 5$) **(b)** 15 groupes de 10 sommets ($n = 15$)

Figure 5.5 – Résultats de la méthode de filtrage sur des graphes artificiels bruités par un ajout d'arêtes aléatoires. Le filtrage consiste à supprimer les arêtes ayant une confluence (calculée pour $t = 3$) strictement inférieure à $\tau^- = 0.5$. Le premier tracé indique le score de Jaccard entre les arêtes originales et les arêtes bruitées (triangles rouges), et les arêtes filtrées (ronds verts). Les tracés suivants indiquent respectivement le nombre d'arêtes et le nombre de concepts, sur le graphe original (carrés bleus), le graphe bruité (triangles rouges), le graphe filtré (ronds verts). L'axe des abscisses indique la proportion d'arêtes aléatoirement ajoutées au graphe de départ. Chaque point indique la moyenne pour 20 réalisations, l'écart type étant indiqué par les barres verticales. Les graphes sont construits avec la méthode présentée en section 5.3.3.1 avec les paramètres suivants : $k = 10$, $m_{own} = 6$, $m_{shared} = 6$, $s = 2$, $\mu = 1.0$, avant un ajout d'arêtes aléatoires.

« bruit » : il y a presque toujours plus d'arêtes sur le graphe après filtrage que sur le graphe de départ. Cela fait que le nombre de concepts reste plus important même après le filtrage, en étant néanmoins beaucoup plus faible que sur le graphe bruité.

Sur les petits graphes (figure 5.5a), lorsque le nombre d'arêtes ajoutées est important, on observe que les graphes filtrés comportent moins d'arêtes que les graphes de départ. Le phénomène se produit lorsque le nombre d'arêtes a doublé (bruit= 1), la densité dépasse alors 0.6. Le même phénomène se produit quand on ajoute suffisamment d'arêtes aux graphes de la figure 5.5b pour que leurs densités dépassent 0.5 (cela n'est pas visible sur la courbe car il faut ajouter 5 fois le nombre d'arêtes présentes au départ). L'ajout d'arêtes fait que le score de confluence de certaines arêtes du graphe original se retrouve inférieur à 0.5.

Choisir un seuil plus élevé pourrait permettre de mieux nettoyer le graphe. La figure 5.6 présente les mêmes mesures que celles de la figure 5.5 mais cette fois avec



(a) 5 groupes de 10 sommets ($n = 5$) **(b)** 15 groupes de 10 sommets ($n = 15$)

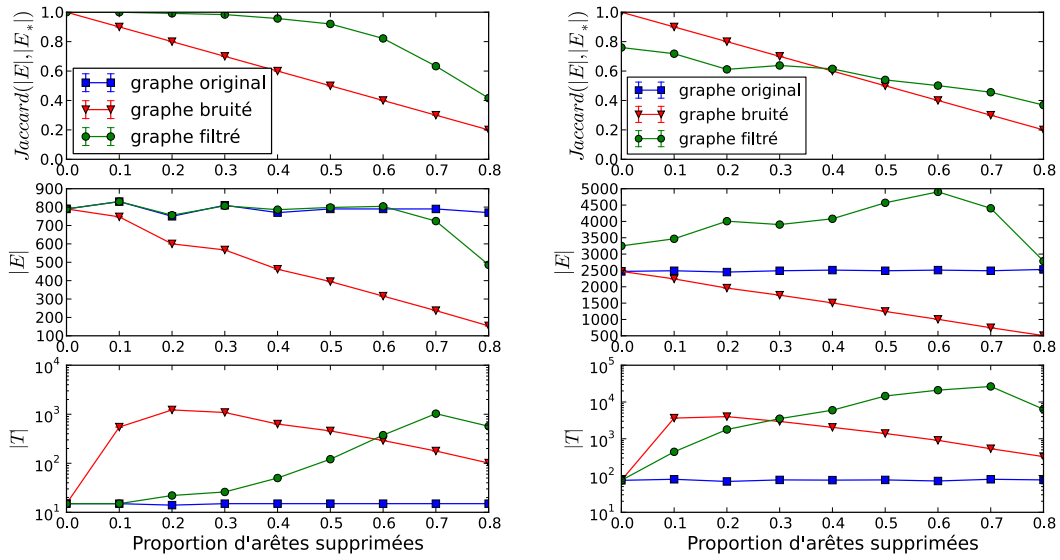
Figure 5.6 – Résultats de la méthode de filtrage en fonction du seuil sur des graphes artificiels bruités par un ajout d'arêtes aléatoires. La confluence est calculée avec $t = 3$. Le seuil sur la confluence est indiqué sur l'axe des abscisses, les arêtes ayant un score inférieur sont filtrées. Le bruit ajouté sur les graphes est ici constant : 40% du nombre d'arêtes de départ est ajouté aléatoirement. Pour une description en détail de la signification de ces courbes voir la légende de la figure 5.5.

un seuil τ^- variant et un bruit fixe de 0.4. Là encore, aucune arête n'est ajoutée par la méthode de filtrage : seules les arêtes ayant un score strictement inférieur au seuil sont supprimées.

On observe en figure 5.6a qu'un seuil supérieur à 0.5 amène à retirer trop d'arêtes. En effet, on remarque que les scores de confluence des arêtes (non issues du bruit) sont proches de 0.5 sur les graphes « a ». À l'inverse en figure 5.6b, il apparaît qu'un seuil de 0.6 permet de revenir presque exactement au graphe de départ ($Jaccard \approx 1$). La confluence des arêtes des graphes de départ étant supérieure à 0.5, on peut supprimer les arêtes ayant un score supérieur à 0.5 sans supprimer les arêtes originales.

5.4.3.2 Ajouter des arêtes manquantes

Voici une seconde évaluation sur les mêmes séries de graphes artificiels, mais cette fois face à un bruit consistant en une suppression aléatoire des arêtes. La méthode de filtrage est configurée ici uniquement pour ajouter comme arêtes les paires de sommets ayant un score de confluence supérieur à un certain seuil τ^+. Aucune arête ne peut être supprimée par le filtrage.



(a) 5 groupes de 10 sommets ($n = 5$) **(b)** 15 groupes de 10 sommets ($n = 15$)

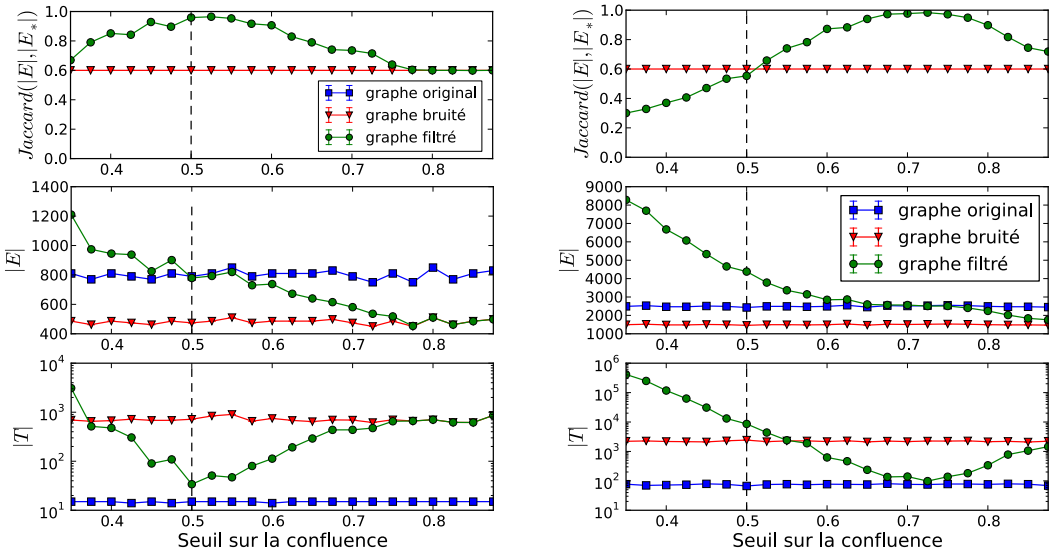
Figure 5.7 – Résultats de la méthode de filtrage sur des graphes artificiels bruités par une suppression aléatoire des arêtes. Le filtrage consiste à ajouter les paires ayant une confluence (calculée pour $t = 2$) strictement supérieure à $\tau^+ = 0.5$. Pour une description en détail de la signification de ces courbes voir la légende de la figure 5.5.

La figure 5.7 donne les résultats de la méthode de filtrage en fonction de la quantité d'arêtes supprimées aléatoirement. Le score de confluence, pour qu'une paire soit ajoutée comme arête, est fixé à $\tau^+ = 0.5$. La confluence est toujours calculée avec $t = 3$.

On observe que les résultats sont très différents pour les deux tailles de graphe. Sur les petits graphes, en figure 5.7a, le nombre d'arêtes des graphes filtrés reste similaire au nombre d'arêtes de départ, et jusqu'à une suppression de 30% des arêtes, le filtrage récupère ces arêtes supprimées (Jaccard$\approx$ 1). À l'inverse sur les grands graphes, en figure 5.7b, la procédure de filtrage engendre un ajout d'arêtes dès le départ lorsqu'il n'y a pas encore de bruit. Dans les graphes « b », il y a donc un nombre important de paires de sommets, non arêtes, ayant un score de confluence supérieur à $\tau^+ = 0.5$.

La figure 5.8 donne les résultats lorsque le seuil τ^+ varie. Le bruit est fixé à 0.4, c'est-à-dire 40% des arêtes ont été supprimées aléatoirement.

On observe sur la figure 5.8a qu'un seuil entre 0.45 et 0.6 permet de réduire convenablement le bruit, alors que sur la figure 5.8b il faut un seuil entre 0.65 et 0.8. En figure 5.8a les paires non-adjacentes ont toutes un score de confluence inférieur à 0.5, alors que les arêtes ont un score supérieur à 0.5. Cette limite autour de 0.5 se déplace vers 0.7 sur la figure 5.8b.



(a) 5 groupes de 10 sommets ($n = 5$) **(b)** 15 groupes de 10 sommets ($n = 15$)

Figure 5.8 – Résultats de la méthode de filtrage par ajout d'arêtes, en fonction du seuil sur des graphes artificiels bruités par une suppression aléatoire des arêtes. La confluence est calculée avec $t = 3$. Le seuil sur la confluence est indiqué sur l'axe des abscisses, les paires de sommets (non-arêtes) ayant un score supérieur au seuil sont ajoutées. Le bruit est ici constant : 40% des d'arêtes de départ sont supprimées aléatoirement. Pour une description en détail de la signification de ces courbes voir la légende de la figure 5.5.

5.4.3.3 Conclusion des évaluations sur les graphes artificiels

Il est possible de résumer les remarques faites sur ces deux premières séries d'évaluations de la manière suivante. Pour les graphes des figures « a » :

- le score de confluence des *arêtes* est supérieur à 0.5 (et proche de 0.5),
- et celui des *non-arêtes* est inférieur à 0.5.

Alors que pour les graphes des figures « b » :

- le score de confluence des *arêtes* est supérieur à 0.7.
- et celui des *non-arêtes* est inférieur à 0.7 (et souvent supérieur à 0.5).

Sur ce type de graphe, la valeur de la confluence dépend donc du nombre de groupes de sommets.

Pour les arêtes cela se comprend bien avec le raisonnement suivant. Plus un graphe comporte de groupes de sommets plus la probabilité limite d'une marche aléatoire ($\pi(v)$, voir eq. (5.31)) tend à être faible. En revanche, la valeur des marches aléatoires à temps courts entre deux sommets du même groupe ne varie pas énormément entre les deux séries de graphes. En effet la taille (et la densité) des groupes de

sommets est la même dans les deux cas. Les probabilités à temps courts ne varient donc pas ou peu, alors que les valeurs limites changent : on comprend donc bien ces variations des valeurs de confluence.

Le fait qu'un certain nombre de paires de sommets non-adjacents aient quand même un score supérieur à 0.5 sur les graphes des figures « b », s'explique par le fait que les graphes utilisés présentent un fort recouvrement (des propriétés) entre les groupes. En effet chaque groupe de sommets a autant de propriétés propres que de propriétés partagées avec un autre groupe. Imaginons par exemple une paire de sommets (a, b) constituée d'un objet a appartenant à un groupe A et d'une propriété b étant une propriété propre à un groupe B. Il est possible que les groupes A et B partagent un ensemble de propriétés. Il existe alors un nombre important de chemins courts entre a et b, et donc une forte confluence entre a et b. Le recouvrement important entre les groupes de sommets dans ces graphes fait que des paires de sommets peuvent avoir une confluence assez forte sans appartenir au même groupe. Ces paires de sommets existent dans les graphes des figures « a », mais la valeur limite des marches aléatoires étant forte, leur confluence reste en dessous de 0.5.

La suppression des arêtes ayant une confluence inférieure à 0.5 est donc efficace, sur ces graphes. Utiliser un seuil plus élevé peut être intéressant, mais au risque de supprimer aussi de « bonnes » arêtes. Aussi, l'utilisation de la méthode pour ajouter des arêtes est possible. Mais le choix du seuil est plus compliqué, un seuil de 0.5 engendrant un nombre trop important d'ajouts. Cela est dû au recouvrement fort existant dans les graphes utilisés, en effet la moitié des propriétés appartient à deux groupes de sommets.

5.4.3.4 Graphes réels documents-mots

Le tableau 5.6 présente le résultat de notre méthode de filtrage sur 25 graphes reliant des documents à des termes. Ces graphes sont issus d'une application présentée au chapitre suivant (section 6.4). Chacun comporte 100 sommets documents (sauf le graphe #20 qui n'en comporte que 40). Ces documents sont les 100 premiers retournés par un moteur de recherche pour une requête donnée. Un lien entre un document et un terme indique que le terme est présent dans le document. Chacune des arêtes est pondérée par un score de BM25, mesurant à quel point un terme est significatif pour un document. Tous les termes des documents ne sont pas présents dans le graphe, car différent filtres sont appliqués. Pour plus de détails sur ces graphes nous renvoyons à la sous-section 6.4.2 du chapitre suivant.

La suppression des arêtes ayant une confluence inférieure à $\tau^- = 0.5$ semble particulièrement efficace. En effet en moyenne 27% des arêtes sont supprimées, et cela résulte en moyenne en une diminution de 64% des concepts. Supprimer ces arêtes semble donc particulièrement efficace pour réduire le nombre de concepts.

Par contre le filtrage par ajout d'arêtes engendre une augmentation importante

du nombre d'arêtes *et* une augmentation très importante du nombre de concepts.

Ces deux observations sont à mettre en parallèle avec les remarques faites précédemment sur des graphes artificiels. En particulier le fait que beaucoup de paires non-adjacentes aient un score supérieur à 0.5 pourrait résulter, comme sur les graphes artificiels, d'un fort recouvrement des termes entre les groupes de documents. Il serait intéressant de voir si un seuil plus contraignant pour l'ajout d'arêtes permet de réduire le nombre de concepts.

5.4.4 Conclusion et perspectives

Nous avons présenté ici une méthode de filtrage de graphe biparti basée sur une mesure de similarité entre sommets. Une première série d'évaluations nous a permis de montrer que cette méthode est efficace pour retirer des arêtes résultant d'un bruit aléatoire sur des graphes artificiels. En revanche l'utilisation de cette méthode pour ajouter des liens semble plus délicate. Des résultats compatibles avec ces conclusions sont obtenus sur des graphes réels : la suppression des arêtes de faible confluence permet de réduire le nombre de concepts, en revanche l'ajout des paires de sommets (non-arêtes) engendre une augmentation importante du nombre de concepts.

Le choix du seuil mériterait une évaluation plus approfondie. En particulier : est-ce que, sur ces graphes réels, un seuil plus restrictif permet de réduire le nombre de concepts ? On pourrait alors imaginer une méthode qui choisisse le seuil de manière à minimiser le nombre de concepts. Toutefois cela poserait un problème de complexité. De plus il suffit d'ajouter toutes les arêtes possibles pour qu'il n'y ait plus qu'un seul concept.

Aussi plutôt que d'utiliser naïvement la confluence pour ajouter toutes les paires ayant un score supérieur à un certain seuil, une approche intéressante pourrait être d'utiliser la confluence (ou une mesure similaire) dans un algorithme de regroupement a posteriori des concepts formels. Par exemple en regroupant toutes les paires de concepts telles que la confluence minimale des liens manquants dans leur union soit inférieure à un certain seuil. Notons que le défaut principal de ces méthodes de regroupement des concepts est qu'elles ne peuvent pas faire disparaître des concepts créés par des arêtes erronées. Dans l'optique d'un usage combiné de notre méthode de pré-traitement et d'une telle méthode de post-traitement par agrégation de concepts, les résultats de la méthode proposée ici sont donc tout à fait encourageants.

Table 5.6 – Résultat de la méthode de filtrage sur les 25 graphes documents-mots. $|V_{doc}|$, $|V_{term}|$, $|E|$ et $|T|$ correspondent respectivement aux nombres de documents, de termes, d'arêtes et de concepts dans les graphes. $|E_{cut-}|$ et $|T_{cut-}|$ sont respectivement les nombres d'arêtes et de concepts après un filtrage supprimant les arêtes de confluence inférieure à 0.5. $|E_{cut+}|$ et $|T_{cut+}|$ donnent ces valeurs après un filtrage ajoutant les paires de sommets de confluence supérieure à 0.5. Enfin $|E_{cut+-}|$ et $|T_{cut+-}|$ donnent ces valeurs pour un filtrage permettant l'ajout (confluence > 0.5) et la suppression (confluence < 0.5). Les confluences sont calculées avec $t = 3$.

requête	$\|V_{doc}\|$	$\|V_{term}\|$	$\|E\|$	$\|T\|$	$\|E_{cut-}\|$	$\|T_{cut-}\|$	$\|E_{cut+}\|$	$\|T_{cut+}\|$	$\|E_{cut+-}\|$	$\|T_{cut+-}\|$
#0 : « mykonos site »	100	328	1 603	982	1 285	374	5 985	187 567	5 699	181 116
#1 : « creation d'objets en fil de fer »	100	410	2 719	7 872	1 910	2 285	9 411	464 550	8 649	304 753
#2 : « taux TSH »	100	468	2 684	7 125	1 583	856	10 478	1 014 688	9 396	428 663
#3 : « risques auditifs »	100	361	1 745	1 311	1 302	594	7 680	187 246	7 299	181 917
#4 : « abeilles et gaucho »	100	469	3 048	11 008	1 938	1 153	10 928	1 555 274	9 821	665 568
#5 : « lichens bioindicateurs »	100	421	1 955	2 297	1 330	635	8 232	211 401	7 664	168 019
#6 : « alsace nature »	100	418	2 165	2 673	1 384	567	7 768	143 373	7 049	76 922
#7 : « Gingko »	100	246	854	302	712	188	3 193	16 259	3 100	10 128
#8 : « protection des mains »	100	403	1 788	965	1 468	433	6 702	117 398	6 451	100 452
#9 : « grues mobiles d'atelier »	100	370	1 991	1 472	1 540	531	7 925	269 642	7 492	188 852
#10 : « noirmoutier camping »	100	373	2 073	2 345	1 406	598	7 881	226 140	7 233	134 199
#11 : « camping car occasion trafic »	100	492	3 110	9 089	2 162	1 043	11 189	454 372	10 257	246 769
#12 : « crampes aux pieds nocturne »	100	461	2 399	3 286	1 698	936	9 573	2 063 929	8 907	1 602 896
#13 : « tarte aux fruits de mer »	100	539	4 466	32 254	2 792	1 919	15 138	4 099 879	13 470	1 631 665
#14 : « lactation regurgitation »	100	284	1 410	807	1 133	419	5 367	45 770	5 153	46 068
#15 : « poème sur combattre la maladie »	100	287	800	252	601	111	3 537	7 705	3 469	7 369
#16 : « le gouffre de poudrey »	100	327	1 622	1 205	1 143	346	7 466	225 820	7 030	141 242
#17 : « meringues fondantes a l'interieur »	100	438	2 985	17 779	1 571	1 106	10 951	2 487 591	9 545	897 536
#18 : « greffe en ecusson »	100	257	1 131	1 201	960	907	4 896	109 024	4 783	116 021
#19 : « bruyeres le chatel »	100	567	3 964	4 513	3 891	4 467	11 360	1 610 967	11 315	1 543 676
#20 : « Hypertyroidie »	41	204	821	726	468	158	2 072	13 034	1 748	5 417
#21 : « Mercilon »	100	476	2 164	2 608	1 847	1 387	9 265	2 698 881	8 966	2 254 140
#22 : « refuge du chatelleret »	100	440	2 170	3 935	1 532	1 045	9 463	4 342 913	8 859	4 344 610
#23 : « le bourg d'oisans hotel »	100	426	1 908	1 383	1 512	489	7 635	51 068	7 283	41 021
#24 : « resultats templiers 2008 »	100	386	1 511	945	1 285	648	6 583	206 621	6 468	210 226

5.5 Conclusion du chapitre

Nous avons présenté dans ce chapitre trois contributions au problème de clustering de graphe bipartis. Tout d'abord nous avons présenté un parallèle entre l'analyse formelle de concepts (étendue) et l'analyse de graphe biparti, ce parallèle apporte un éclairage intéressant sur ce que doit être un cluster dans un graphe. Nous avons ensuite proposé une méthode permettant d'obtenir une partition des objets sans imposer de partition correspondante sur les propriétés. Nous avons montré l'intérêt de l'approche sur un jeu de graphes artificiels et sur un jeu de données réelles. Et enfin nous avons présenté une méthode de pré-traitement d'un graphe biparti en vue de simplifier le résultat de l'analyse formelle de concepts, cette méthode permettant aussi de binariser un graphe biparti pondéré. Nous avons étudié les résultats de cette méthode sur des graphes artificiels et sur des graphes réels. Résultats qui montrent les limites de cette approche, mais aussi son intérêt quand ce pré-traitement ne fait que supprimer des arêtes.

Ce travail de comparaison des différentes méthodes de clustering de graphe est à poursuivre. En particulier, il nous semblerait important d'effectuer une comparaison tant théorique qu'expérimentale des méthodes de bi-clustering et des méthodes dites de détection de communautés (appliquées au graphes bipartis). Cela afin de mieux comprendre les hypothèses implicites faites sur les données, ainsi que les différentes définitions (souvent implicites) données aux clusters. Bien sûr cela pourrait aider à imaginer des nouvelles méthodes plus efficaces. Mais ces différentes familles de méthodes étant souvent utilisées dans des champs d'application différents, un tel parallèle pourrait se montrer productif, même s'il ne débouche pas sur la création de nouvelles méthodes.

Aussi une évaluation plus poussée des méthodes que nous avons proposées serait souhaitable. En particulier une évaluation comparative avec les méthodes de la littérature les plus semblables. Notons que cette évaluation est toutefois prolongée, sur une application réelle, dans le chapitre suivant.

En effet le chapitre suivant présente une application possible du problème de clustering de graphe biparti vu dans ce chapitre. Cette application consiste à catégoriser automatiquement, de manière non supervisée, les résultats d'une recherche de documents.

Chapitre 6

Application à l'organisation des résultats d'une recherche d'information

L'objectif de ce chapitre est de présenter *Kodex*, un système de recherche d'information modulable permettant une classification automatique des résultats d'une recherche de documents.

Ce travail apporte une contribution double : à la fois logicielle et scientifique. En effet *Kodex* est un cadre logiciel modulable, et donc largement réutilisable, pour enrichir d'une classification automatique les résultats d'une recherche. Cette modularité permet de construire facilement tant des applications « réelles », que des scripts d'évaluation. Enfin l'application de méthodes récentes de clustering de graphe biparti (voir chapitre 5) au problème de clustering des résultats d'une recherche d'information est nouvelle et l'évaluation menée montre l'intérêt de l'approche.

L'approche générale de *Kodex* est introduite en section 6.1. Nous présentons ensuite, en section 6.2 un état de l'art des méthodes de clustering appliquées à la recherche de documents. Avant de présenter plusieurs applications, les grandes lignes de l'architecture du cadre applicatif mis en place sont décrite en section 6.3. En section 6.4, une application sur une collection de 2,6 millions de pages web est présentée. Cette application permet de proposer une première évaluation de l'approche, ainsi qu'une comparaison de différentes méthodes de clustering de graphe. En section 6.5, nous présentons deux applications fonctionnelles construites l'une sur la collection en ligne du journal le *Guardian*, l'autre sur la base de publication *DBLP*. Enfin la section 6.6 conclut le chapitre en présentant les pistes d'améliorations envisagées.

6.1 Introduction, clustering de graphe documents-termes

Nous présentons ici les grandes lignes de l'approche que nous développons avec le cadre logiciel *Kodex*. L'idée est de proposer à l'utilisateur une organisation en différents groupes des documents retournés pour une recherche. Cela doit permettre de rendre visible la polysémie de la requête par rapport à la collection de documents, et d'aider l'utilisateur à raffiner sa recherche. Un exemple de cas d'utilisation est donné en figure 6.1. Dans cet exemple, un utilisateur cherche « *japan* » et le système lui indique une liste de documents classés en trois clusters : l'un concernant plutôt les aspects historiques et culturels, un autre à propos d'informations touristiques, et un dernier relevant de l'actualité. *Kodex* est basé sur une méthode de clustering de graphe biparti entre documents et termes. La figure 6.2 résume schématiquement le fonctionnement du système en réponse à une requête.

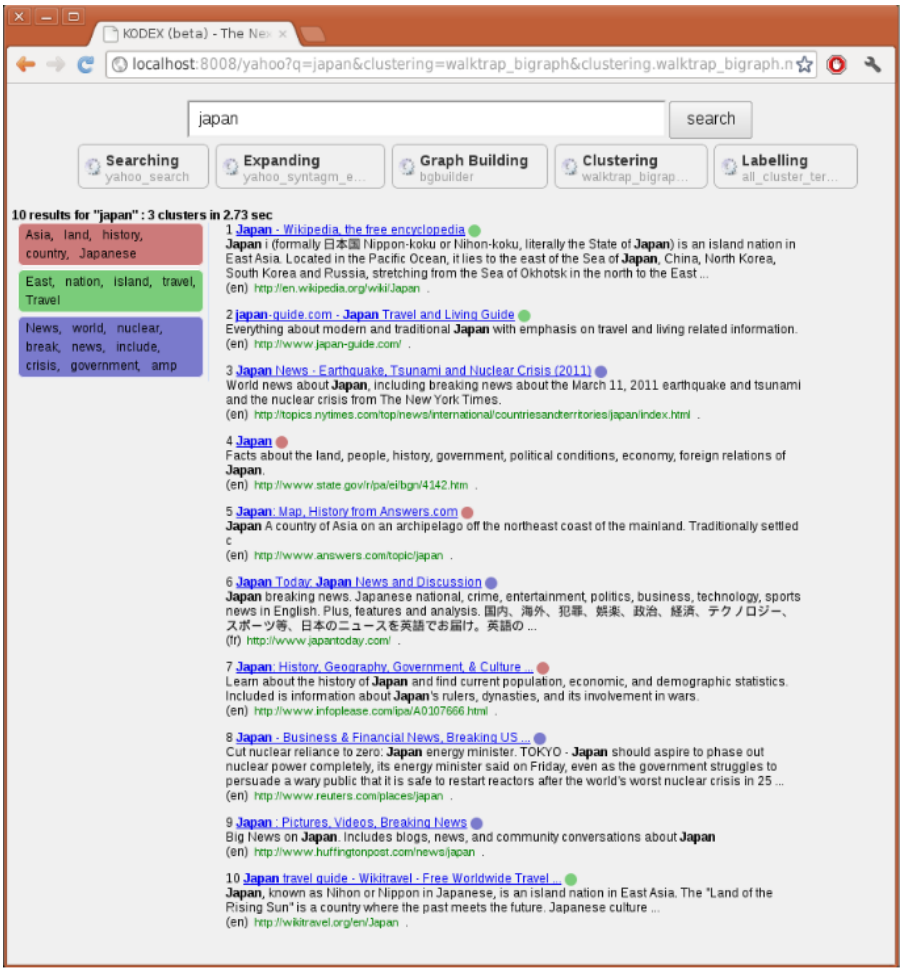


Figure 6.1 – Cas d'utilisation de *Kodex*

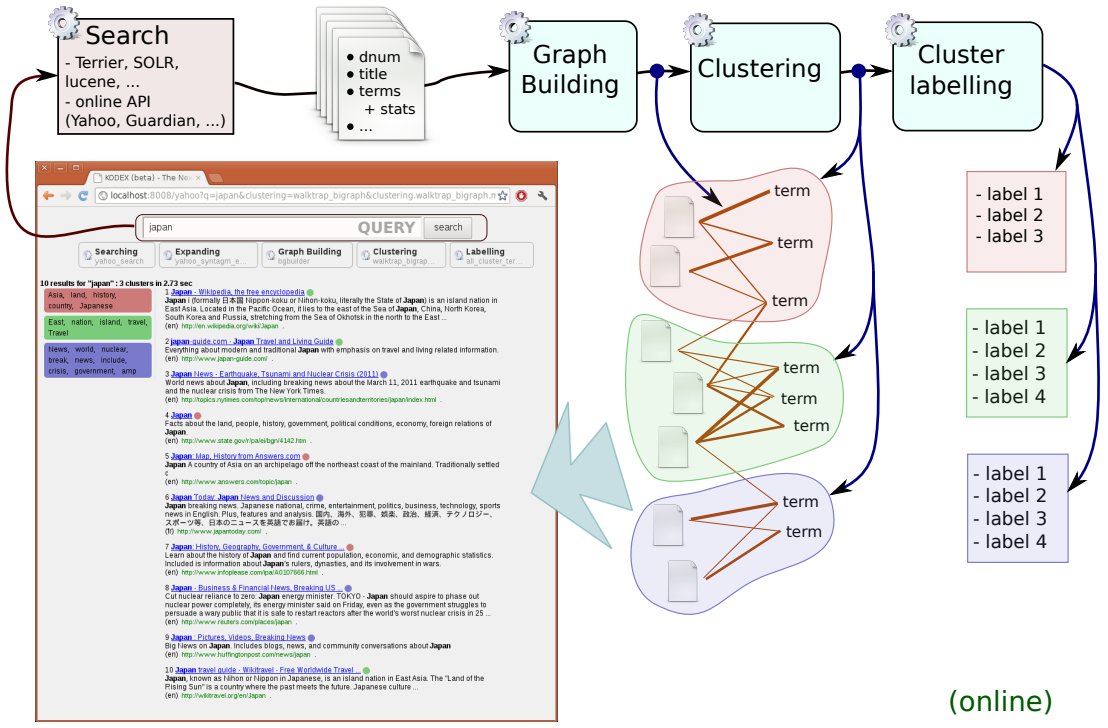


Figure 6.2 – Schéma général de fonctionnement de *Kodex*

Kodex fonctionne par dessus un système de recherche d'information classique (SRI dans la suite) qui retourne une liste ordonnée de résultats pour une requête. C'est la phase *Search* sur la figure 6.2. Étant donnée une telle liste de résultats, *Kodex* dans une première étape (la phase *Graph Building* sur la figure 6.2) construit un graphe biparti entre documents et termes. La méthode de construction du graphe, c'est-à-dire la méthode d'extraction des termes de chaque document, peut varier en fonction des informations disponibles sur les documents. Nous appelons « modélisation » d'un document l'ensemble des termes pondérés ainsi associé à celui-ci. Cette modélisation est la seule représentation des documents utilisée pour construire les clusters. Ce graphe est ensuite découpé par une méthode de clustering de graphe biparti (phase *Clustering* sur la figure 6.2). Enfin, chaque cluster se voit attribuer un certain nombre d'étiquettes (*labels*), c'est la phase *Cluster labelling* sur le schéma.

Le fonctionnement de *Kodex* peut donc être résumé par ces quatre étapes principales : recherche « classique », construction d'un graphe biparti, clustering de celui-ci, et calcul d'étiquettes pour chacun des clusters.

Notons que la chaîne de traitement d'une requête de *Kodex* permet de configurer des étapes intermédiaires de transformation, d'analyse ou de filtrage des données entre chacune de ces quatre étapes principales. Ces étapes intermédiaires permettent par exemple d'interroger un index pour récupérer une modélisation des documents

calculée hors-ligne. Ou encore de filtrer les clusters pour regrouper les –trop– petits clusters dans un cluster « divers ». Cette chaîne de traitement est présentée plus en détail en section 6.3.5.

Pour la construction du graphe, différentes techniques de modélisation des documents sont envisageables. La plus simple est d'utiliser les informations retournées par le SRI. Malheureusement, ces informations se limitent souvent aux titres et à de courts extraits des documents. Parfois des mots-clés peuvent être fournis par le SRI et se révéler pertinents pour le clustering. C'est ce que nous utilisons dans une application mise en place sur le SRI du journal anglais le *Guardian* (présentée en section 6.5.1). Une autre solution consiste à extraire certains termes des documents complets. Pour cela, une méthode simple consiste en trois étapes : extraire tous les termes, les pondérer puis les filtrer. *Kodex* permet de prendre en charge un tel traitement hors-ligne. Nous revenons sur ce point en section 6.3.6.

Les différentes méthodes de clustering de graphe biparti présentées au chapitre précédent peuvent être utilisées ici. Nous utilisons en particulier la méthode introduite au chapitre précédent en section 5.3.

Notons enfin que ce travail, et en particulier la conception et le développement logiciel, a été fait en collaboration avec Yannick Chudy et Benoit Gaillard.

6.2 État de l'art : le clustering appliqué à la recherche d'information

Nous proposons dans cette section un état de l'art des différentes méthodes utilisées pour classifier automatiquement les résultats d'une recherche. Cet état de l'art permet de mettre en exergue les difficultés spécifiques au clustering des résultats d'une recherche de documents par rapport au problème général de clustering. Pour un état de l'art plus complet, nous renvoyons à l'article de Carpineto et al. [2009] [1].

Le clustering des résultats d'un SRI a été introduit pour la première fois par Cutting et al. [1992]. Les clusters y étaient construits en utilisant *Fractionation*. C'est un algorithme de clustering hiérarchique basé sur la modélisation vectorielle des documents [Salton et al., 1975b], introduit avec le système *Scatter/Gather*. Son principal avantage résulte de sa complexité linéaire. En revanche, il ne permet pas de traiter le problème de la polysémie, chaque document étant associé à un seul cluster. De plus, le nombre de clusters produits est un paramètre fixé par l'utilisateur, ce qui nous semble peu réaliste. Ce nombre de clusters ne doit-il pas plutôt dépendre de la polysémie de la requête ? N'existe-t-il pas des requêtes pour lesquelles une organisation des résultats en plusieurs groupes n'a aucun sens ? Ce sont là des points

1. Notons tout de même que les travaux [Boley et al., 1999],[Dhillon, 2001],[Mecca et al., 2007],[Chen et al., 2008] ne sont pas cités dans cette revue.

importants qui ont souvent été mis de côté dans les techniques proposées dans l'état de l'art. Enfin c'est un algorithme centré sur les données (*data-centric*[2], pour reprendre la classification de Carpineto et al. [2009]) et l'étiquetage des clusters n'est pas forcément pertinent et interprétable par l'utilisateur (pour chaque cluster, les termes les plus fréquents sont utilisés comme étiquettes).

Une approche très différente a ensuite été proposée avec *Grouper* [Zamir et Etzioni, 1998, 1999]. Ce système repose sur l'algorithme *STC* (*Suffix Tree Clustering*) qui rassemble les documents sur la base d'une seule propriété bien choisie : *STC* regroupe les documents qui partagent des expressions suffisamment fréquentes (dans l'ensemble des extraits de documents). L'astuce principale de *STC* repose sur l'utilisation d'un arbre de suffixe généralisé pour rechercher les séquences de termes les plus fréquentes en un temps linéaire. L'intérêt d'une telle approche est que les clusters ont une justification clairement lisible pour l'utilisateur final (*description-aware* selon Carpineto et al. [2009]). En revanche, nous doutons de la robustesse de telles méthodes, il suffit en effet de changer l'ordre de deux mots ou de remplacer un mot par un synonyme pour que les séquences les plus fréquentes changent complètement.

Depuis ces deux travaux de référence, de nombreuses autres contributions ont été proposées, dont : *SHOC* [Zhang et Dong, 2004], *SnakeT* [Ferragina et Gulli, 2005], *Lingo* [Osinski et Weiss, 2005] et *Noodles* [Mecca et al., 2007]. Il est assez remarquable que la plupart de ces méthodes (toutes celles citées ici sauf *SnakeT*) se basent sur la technique de *Latent Semantic Indexing* (*LSI*) introduite par Deerwester et al. [1990]. Ceci est compréhensible car, d'une part, en prenant en considération la co-occurrence des termes, la *LSI* répond au problème de la synonymie (c'est-à-dire : rapprocher deux documents même s'ils n'ont aucun mot en commun mais seulement des mots synonymes, voir [Manning et al., 2008, page 378]). D'autre part, la complexité importante de la *LSI* est peu gênante car la taille des données est faible car réduite au sous-ensemble de documents restitués par un SRI.

La majeure partie des approches (toutes sauf *Scatter/Gather* Cutting et al. [1992] et *Noodles* [Mecca et al., 2007]) sont basées sur les extraits de documents plutôt que sur les textes complets. C'est un intérêt évident au regard de la complexité, et cela permet de faire facilement fonctionner les systèmes au-dessus d'un SRI existant en analysant uniquement (et en ligne) les extraits restitués par celui-ci. Par ailleurs, Zamir et Etzioni [1998] n'ont pas observé d'augmentation de la qualité des clusters en utilisant les documents complets plutôt que les simples extraits. On peut cependant penser que ce résultat est dû à un comportement particulier de l'algorithme *STC*. En effet, Mecca et al. [2007] montrent qu'avec une méthode basée sur la *LSI*, les résultats sont significativement meilleurs quand les documents complets sont utilisés.

2. Selon la dénomination de Carpineto et al. [2009], les approches de clustering *data-centric* sont centrées sur la qualité des clusters ne se préoccupant pas de l'interprétabilité des clusters, alors qu'au contraire les méthodes *description-aware* sont centrées sur l'interprétabilité des clusters par l'utilisateur.

Alors que les approches précédentes reposent principalement sur des modélisations vectorielles des documents, quelques auteurs ont traité le problème de clustering des résultats d'un SRI comme un problème de partitionnement de graphe [Boley et al., 1999; Dhillon, 2001]. Le partitionnement d'un graphe consiste à trouver une partition de l'ensemble des sommets qui minimise la taille de la coupe induite (c'est-à-dire le nombre total d'arêtes entre deux groupes). Nous avons vu au chapitre précédent (section 5.1) que ce type d'approche pose problème quand on souhaite faire émerger un découpage pré-existant dans les données.

Le travail récent de Chen et al. [2008] exploite une technique de clustering de graphe (par optimisation de la modularité) sur un graphe de termes construit à partir des co-occurrences de ceux-ci dans les extraits de documents restitués. Le but est de trouver les *word sense communities* qui permettent ensuite de construire des clusters de documents. Comme le signalent les auteurs, la principale originalité de cette approche par détection de communautés est que le nombre de clusters n'est pas un paramètre fixé mais dépend des données. Les méthodes de clustering que nous utilisons conservent cet avantage. Seulement notre approche est différente puisque nous proposons de travailler sur un graphe biparti documents-termes, et non sur un graphe uni-parti de termes.

Nous pouvons remarquer que beaucoup des méthodes de l'état de l'art construisent d'abord des clusters de termes, et y attachent ensuite les documents (c'est par exemple le cas de *Lingo* et de la méthode de Chen et al. [2008]). Nous retrouvons ici, en un sens, la dichotomie proposé par Carpineto et al. [2009] entre approches *description-aware* et approches *data-centric*. Les premières ayant pour avantage des descriptions a priori plus pertinentes, alors que les secondes proposent des clusters, a priori, correspondant mieux aux documents. Bien sûr la description des clusters est primordiale quand on cherche à utiliser une méthode de clustering pour guider un utilisateur dans les résultats d'une recherche. Cela étant dit, la pertinence des clusters vis à vis du jeu de documents concernés n'est pas à minimiser. Imaginons par exemple un résultat comportant deux clusters qui font sens par rapport à leurs étiquettes, mais tels que tous les documents appartiennent à ces deux clusters : l'intérêt d'un tel découpage est limité. On peut aussi se demander si le problème consistant à attacher un ensemble de documents à un cluster de termes n'est pas similaire au problème d'étiquetage, consistant à attacher un ensemble de termes à un cluster de documents ?

L'approche que nous adoptons dans *Kodex* est intermédiaire, puisque nous construisons des clusters contenant à la fois documents et termes. Dans la pratique, utiliser directement les termes des clusters comme étiquettes de ceux-ci pose plusieurs problèmes. Tout d'abord le nombre de termes est souvent important, une technique de sélection est donc nécessaire. De plus les termes les plus centraux des clusters ne sont pas forcément les plus lisibles. Dans la pratique l'approche de *Kodex*

est donc plus *data-centric*.

Notons enfin que le cadre logiciel de *Kodex* permet de développer une approche qui passe d'abord par des clusters de termes, nous avons par exemple pu intégrer l'implémentation de *Lingo* proposée par le logiciel *Carrot*.

6.3 Implémentation logicielle d'une chaîne de traitement modulable (*Kodex*)

Cette section a pour objectif de décrire les grandes lignes de l'architecture du cadre logiciel *Kodex*. Nous présentons tout d'abord les motivations et les besoins auxquels répond ce logiciel. Nous introduisons ensuite les types de données manipulées (sous-section 6.3.2), et le système de composant de traitement et de chaînage de ces composants (sous-sections 6.3.3 et 6.3.4). Enfin les fonctionnements des chaînes de traitement en-ligne (sous-section 6.3.5) et hors-ligne (sous-section 6.3.6) sont exposés.

6.3.1 Objectifs, motivations et choix techniques

Kodex permet d'organiser automatiquement les résultats d'une recherche. En ce sens *Kodex* n'est pas un SRI à proprement parler, mais un « méta-moteur » : il se place au-dessus un SRI existant. Seulement une contrainte supplémentaire s'applique à *Kodex* : le contenu textuel des documents doit être connu pour construire le graphe et construire des groupes de documents. Il faut donc que *Kodex* puisse analyser les documents et construire son propre index en se basant éventuellement sur un SRI. *Kodex* doit donc fournir, en plus d'une chaîne de traitement en-ligne, des outils pour analyser et indexer des documents hors-ligne.

Dès le départ *Kodex* a été conçu pour réaliser des évaluations, mais aussi pour mettre en place des applications fonctionnelles. L'objectif est d'être utilisable tant pour la recherche que pour une « mise en production ».

Afin de mener à bien différentes évaluations, et d'arriver à un prototype fonctionnel performant, *Kodex* a été construit de manière à être extrêmement modulable. Il est important de pouvoir facilement modifier ou changer chacune des parties du traitement, pour pouvoir facilement comparer différentes approches. Aussi chacune des étapes doit pouvoir être confiée à un service externe. *Kodex* doit pouvoir être facilement « branché » à un logiciel tiers. En particulier les parties concernant la recherche d'information « classique » n'étant pas faites par *Kodex*, il faut que différents SRI puissent être utilisés.

Cette modularité des chaînes de traitement doit être accompagnée d'outils pour faciliter, voire automatiser, la construction des applications d'évaluations et de démonstrations. Il est nécessaire que la modification d'un composant d'une chaîne de

traitement ne demande pas de modification des applications qui l'utilisent. Sans cela la modularité de *Kodex* serait toute relative.

Enfin *Kodex* a été conçu pour permettre un prototypage rapide. C'est-à-dire pouvoir rapidement mettre en place et expérimenter de nouvelles méthodes pour chacune des phases du traitement.

Toutes ces motivations nous ont amené au choix de l'architecture présentée dans les sous-sections suivantes. Ces besoins nous ont aussi amené à choisir de développer *Kodex* en *Python*. En effet *Python* permet un développement rapide (donc un prototypage rapide) tout en étant suffisamment solide pour construire des applications de production (par exemple une application web). Aussi la souplesse de *Python* aide à construire facilement une solution modulable et facilement interfaçable avec des logiciels tiers. Le dernier avantage de taille de *Python* est le vaste écosystème de bibliothèques disponibles. En particulier, *Kodex* utilise intensément la bibliothèque *igraph* [Csardi et Nepusz, 2006].

Les deux inconvénients majeurs de Python sont ses faibles performances (face à des langages compilés), et le typage dynamique (qui est aussi l'un de ses avantages) et donc le manque de vérification faite à la compilation. On peut éviter les pièges de ce second problème par une rigueur dans le développement, la création de tests unitaires et une bonne documentation construite au fur et à mesure. Le premier problème n'est pas critique pour nous car les parties les plus coûteuses en temps de calcul sont soit effectuées par des logiciels tiers (codés dans des langages compilés), soit effectuées par des fonctions codées en C et appelées depuis *Python*, en particulier avec les bibliothèques *numpy* et *igraph*.

6.3.2 Types de données principaux

Nous présentons ici les principaux types de données complexes manipulées par *Kodex*.

Documents. Dans les chaînes de traitement de *Kodex*, les documents sont des objets *KodexDoc*. Ces *KodexDoc* comportent des champs libres, et des listes d'éléments. L'ensemble des champs utilisés n'est pas défini par avance, c'est une approche « sans-schéma » (*schema free*). Seulement le champ *docnum* est toujours renseigné, il indique l'identifiant du document dans la collection. Un *KodexDoc* est en fait un dictionnaire *Python* (une table d'association) gérant en plus un mécanisme de listes d'éléments. Ces listes d'éléments sont utilisées typiquement pour indiquer la liste des termes associés au document. Elles peuvent être multiples (par exemple les termes du titre et les termes du corps de texte) et des données peuvent être associées à chacun de ces éléments (par exemple la fréquence dans le document, ou encore la catégorie lexicale).

Graphes. Pour la gestion des graphes, *Kodex* utilise intensément la bibliothèque *igraph*[3] [Csardi et Nepusz, 2006]. C'est une bibliothèque écrite en C et proposant une interface en *Python*. Les graphes construits par *Kodex* sont donc des objets *Graph* définis dans cette bibliothèque.

Clustering. Le résultat des composants de clustering utilise le type `GraphCover` de la bibliothèque *igraph*, mais lors du traitement en ligne, un clustering est convertit en un objet `KodexCover`. Une objet `KodexCover` est une liste de `Cluster`, chaque `Cluster` comportant une liste de documents, une liste de termes et une liste d'étiquettes. En effet cela permet d'indiquer les étiquettes directement dans l'objet clustering.

6.3.3 Chaînes de traitement, chaînes de composants

Les différentes étapes de traitement des données sont effectuées par un chaînage configurable de composants élémentaires. Les données passent successivement dans chacun des composants pour y être traitées. Cela permet une grande modularité et une factorisation efficace des différentes étapes de traitement des données. Nous présentons ici les grandes lignes du fonctionnement de ce système de chaînes de composants. Notons que cette architecture est largement inspirée du SRI libre *Whoosh* [Chaput, 2009].

```
# construction de la méthode de clustering
fca_infomap = GraphOfConcepts() | Infomap()
# exemple d'utilisation
gb = igraph.Graph.Formula("A:B:C--a:b:c, D:E:F--b:c:d:e")
clusters = fca_infomap(gb)
goc = clusters.graph # récupération du graphe objets/concepts
```

Listing 6.1 – Exemple simple de chaîne de traitement

Le listing 6.1 présente un exemple simple de chaîne de traitement correspondant à la méthode de clustering de graphe biparti présentée au chapitre précédent (voir section 5.3). L'objet `fca_infomap` est une chaîne combinant deux composants de base : `GraphOfConcepts()` et `Infomap()`. Le composant `GraphOfConcepts()` transforme un graphe biparti en un graphe objets/concepts et `Infomap()` applique la méthode de clustering *Infomap* et retourne un partitionnement des sommets du graphe. Cette chaîne est exécutée, à la ligne 5, sur le graphe `gb` créé à la ligne précédente. Notons que le partitionnement retourné (`clusters`) est un partitionnement des sommets du graphe objets/concepts et non un partitionnement de `gb`. Le graphe objets/concepts est accessible depuis le clustering (`goc` en ligne 6) ainsi il est possible d'en déduire un partitionnement des sommets de `gb`, cette étape n'est pas montrée ici.

3. http://igraph.sourceforge.net/

Un traitement complexe peut donc, avec ce système, être découpé en une succession de composants élémentaires. Cela permet de construire facilement les chaînes de traitement, et de factoriser efficacement le code de certaines opérations entre différentes chaînes de traitement.

```
class ToUndirected(Composable):
    def __init__(self):
        Composable.__init__(self)

    def __call__(self, graph):
        graph_undir = graph.copy()
        graph_undir.to_undirected()
        return graph_undir
```

Listing 6.2 – Exemple de composant élémentaire

Un composant élémentaire est une classe qui hérite de la classe abstraite `Composable` et qui implémente la méthode `__call__`. Un exemple simple est donnée par le listing 6.2, ce composant d'exemple transforme un graphe dirigé en un graphe non-dirigé.

La méthode `__call__` [4] effectue le traitement en lui même. Elle prend en premier paramètre la donnée d'entrée et retourne la donnée traitée. Les autres paramètres de cette méthode `__call__` correspondent à des options éventuelles, nous allons le voir dans la section suivante.

6.3.4 Gestion des options des composants

Chaque composant peut avoir des options dont les valeurs sont données à l'exécution. *Kodex* propose un système de déclaration de ces options. Ce système s'occupe de dispatcher les valeurs de chaque option aux bons composants dans une chaîne de traitement. Il permet aussi une découverte automatique des options d'un composant ou d'une chaîne de composants. Cela est particulièrement utile pour générer des formulaires *html* de configuration des options dans une application web. De la même manière, cela permet de produire automatiquement un analyseur d'arguments de lignes de commandes pour exposer et configurer facilement ces options dans un programme s'exécutant en ligne de commande.

Le listing 6.3 est une modification du listing 6.1, la chaîne de traitement `fca_infomap` est maintenant appelée avec deux options : `wgt_method` et `nb_trials`. La première option `wgt_method` définit la méthode à utiliser pour agréger les poids du graphe biparti

4. En *Python* la méthode `__call__` est une méthode spéciale appelée quand l'objet est utilisé comme une fonction, par exemple en ligne 5 du listing 6.1 la méthode `__call__` de l'objet `fca_infomap` est appelée, implicitement.

```
# construction de la méthode de clustering
fca_infomap = GraphOfConcepts() | Infomap()
# exemple d'utilisation
gb = igraph.Graph.Formula("A:B:C--a:b:c, D:E:F--b:c:d:e")
clusters = fca_infomap(gb, wgt_method="sum", nb_trials=50)
```

Listing 6.3 – Exemple simple de chaîne de traitement avec options

sur le graphe objets/concepts, c'est une option du composant `GraphOfConcepts()`. La seconde option `nb_trials` est une option du composant `Infomap()`, elle indique le nombre de tentatives d'optimisation qui doivent être effectuées. Ces options sont dispatchées automatiquement aux bons composants élémentaires, sous réserve que deux composants n'aient pas une option de même nom (cela arrive rarement dans la pratique et est vérifié automatiquement lors de la construction d'une chaîne).

```
class ToUndirected(Composable, Optionable):
    def __init__(self, name="to_undirected"):
        Composable.__init__(self)
        Optionable.__init__(self, name=name)
        self.add_enum_option(
            "mode",
            ["collapse", "each", "mutual"],
            "collapse",
            "What to do with multiple edges going between same
               vertex pair",
            str
        )

    def __call__(self, graph, mode="collapse"):
        graph_undir = graph.copy()
        graph_undir.to_undirected(mode=mode)
        return graph_undir
```

Listing 6.4 – Exemple de composant élémentaire avec une option

Le listing 6.4 donne un exemple simple de déclaration d'option pour le composant introduit au listing 6.2. C'est une option à choix multiples permettant de définir comment plusieurs arcs entre deux sommets doivent être agrégés.

Il convient ici de faire une distinction entre les *options* des composants, et ce que nous appellons les *paramètres*. Les *paramètres* sont donnés lors de la construction du composant (comme arguments de la méthode `__init__`). Alors que les *options* ne sont données que lors de l'exécution du composant (comme arguments de la méthode `__call__`). Les *paramètres* permettent donc une configuration « statique » des composants (par exemple l'url du serveur à interroger, pour une phase de *search*). Alors que les *options* permettent une configuration « dynamique », à l'exécution

(par exemple le nombre de documents à retourner). Notons enfin qu'il est possible de forcer la valeur d'une option de manière statique, l'option n'est alors plus exposée et la valeur indiquée est forcée à l'exécution. Cela est utile pour disposer de toutes les options lors d'une phase de mise au point et d'expérimentation, puis de « faire disparaître » certaines de ces options sur une application en production.

6.3.5 Configuration d'une chaîne de traitement en ligne

La chaîne de traitement de requête (en-ligne) de *Kodex* est composée de cinq blocs de traitement :

Searcher : effectue la recherche (en faisant appel à un SRI externe) ;

Expander : traite les documents ; par exemple : enrichit les documents en interrogeant un index ou une base de données, filtre les termes en fonction de différents scores, ou encore extraits les termes du titre avec un *tokeniser* [5] ;

GraphBuilder : construit le graphe ;

Clustering : regroupe les sommets du graphe en différents clusters ;

Labelling : appose des étiquettes à chaque cluster.

Chacun de ces blocs de traitement est réalisé par un composant ou par une chaîne de composants.

Cette chaîne de traitement est construite par un objet `KodexEngineBuilder`. Cet objet permet d'indiquer le ou les composants utilisables pour chacune des étapes, et ensuite de gérer le traitement d'une requête étant donné le nom des composants à utiliser et l'ensemble des options. Le `KodexEngineBuilder` permet aussi à une application l'utilisant de découvrir les composants disponibles et leurs options respectives.

Le listing 6.5 donne un exemple de construction d'un `KodexEngineBuilder`. L'objet `keb` permet d'effectuer une recherche par dessus le SRI *Terrier* [Ounis et al., 2005]. Par soucis de lisibilité cet exemple est simplifié pour ne présenter que les concepts les plus importants. Dans cet exemple un seul composant est disponible pour les phases *search*, *expander* et *graph builder*. Par contre, trois méthodes de *clustering* et deux de *labelling* sont disponibles. Le composant le plus complexe est certainement la chaîne `expander`. Il est d'abord composé de `BulkIndexExpand(idx)` qui importe, pour chaque document, la liste des termes enregistrés dans l'index de *Terrier* ainsi qu'un certain nombre de statistiques et de scores associés. Ensuite un composant `TermSetBuildExpand` permet de regrouper l'ensemble des termes

5. La tokenisation consiste à découper un texte en différents *tokens*, c'est-à-dire à séparer les mots.

```
terrier_server = "localhost:8081"
index_name = "index_test"

keb = KodexEngineBuilder()

# Search
from kodex.extra import terrier
search = terrier.TerrierSearch(terrier_server, index_name)
keb.add_searching(search)

# Expander
idx = terrier.TerrierIndex(terrier_server, index_name)
expander = BulkIndexExpand(idx)
expander |= TermSetBuildExpand(
    fields_to_merge=[
        ("terms_tf", "tf_RD", lambda x, y : x+y),
        ("terms_tf", "df_RD", lambda x, y : x+1),
    ]
)
expander |= OnlineTermsFilter(
    output_field="terms_filtered"
)
expander |= CacheUrlExpand()
keb.add_expanding(expander)

# Graph Builder
from kodex.gbuilder import DocsTermsGraphBuilder
gbuilder = DocsTermsGraphBuilder(
    terms_field="terms_filtered",
    terms_scores_field="terms_bm25"
)
keb.add_graph_building(gbuilder)

# Clustering
from kodex.clustering import WalktrapBigraph, Infomap,
   ConnectedComponents
keb.add_clustering(Infomap())
keb.add_clustering(WalktrapBigraph())
keb.add_clustering(ConnectedComponents())

# Labelling
from kodex.labelling import AllClusterTerms, BaseLineLabelling
keb.add_labelling(AllClusterTerms())
keb.add_labelling(BaseLineLabelling())
```

Listing 6.5 – Exemple de construction d'un *KodexEngineBuilder*

présents dans les documents retournés, et d'agréger certaines statistiques. Ce composant calcule pour chaque terme sa fréquence totale dans les documents retournés (`"tf_RD"`) et le nombre de documents retournés le contenant (`"df_RD"`). Le composant suivant, `OnlineTermsFilter()`, permet de calculer des scores pour chaque couple terme-document et d'appliquer certains filtres. Enfin le composant `CacheUrlExpand()` permet d'ajouter aux documents un champ `"chache_url"` correspondant à l'adresse permettant d'accéder à une copie locale du document en question.

6.3.6 Analyse des documents hors-ligne

La construction du graphe lors d'une réponse à une requête demande une modélisation des documents. C'est-à-dire, une liste de termes (éventuellement pondérés) associée à chaque document. Cette modélisation peut être construite entièrement en ligne à partir des données retournées par le SRI. Seulement la quantité et la qualité des informations sont souvent limitées. Les SRI ne retournent, la plupart du temps, que le titre et un court extrait du document. De plus, aucune statistique n'est alors disponible sur les termes extraits, si ce n'est la fréquence « locale ».

Une autre solution consiste à utiliser le texte complet des documents. Nous utilisons pour cela une approche simple, consistant en trois étapes : i) analyse des textes, c'est-à-dire en extraire des termes (des *tokens*), ii) calcul des fréquences des *tokens* sur la collection complète et iii) sélection, pour chaque document, des meilleurs *tokens* grâce à un score calculé à partir de ces fréquences. Les deux premières étapes doivent être faites hors-ligne, la troisième étape peut être faite tant en-ligne que hors-ligne. Un SRI doit nécessairement effectuer les deux premières étapes pour construire un index inversé et calculer des scores pour les documents. *Kodex* va donc chercher à s'appuyer le plus possible sur un SRI pour gérer ces deux étapes.

Notons que de nombreuses autres solutions seraient envisageables pour extraire des termes, ou *mots-clés* des documents. Par exemple en utilisant des méthodes d'apprentissage automatique [Boniface, 2011] ou encore des méthodes cherchant à relier les documents aux concepts d'une ontologie [Bouidghaghen et al., 2009]. Ces approches plus complexe de « *text mining* » (ou « *keywords extraction* ») n'ont pas été explorées. Nous y reviendrons dans la conclusion, mais ce problème de modélisation des documents est certainement un point clé pour améliorer le système.

6.3.6.1 Tokenisation et analyse hors-ligne des documents

L'analyse des documents par *Kodex* repose sur le système de composants et de chaînes de composants présenté dans les sous-sections précédentes. *Kodex* analyse les documents grâce à une chaîne de trois composants :

```
Reader()| Analyser()| Writer()
```

À partir d'une liste d'identifiants ou d'adresses, le *reader* lit les documents depuis des fichiers ou depuis une base de données. Son rôle est de construire des *KodexDoc* contenant des champs correspondant au plus près à la structure des documents bruts. Le minimum de transformation des données est fait dans cette phase. L'*analyser* a ensuite pour rôle de découper les champs de textes des documents en *tokens* et de compter les fréquences de ces *tokens* (fréquences dans chaque document). Cet *analyser* peut lui même être une chaîne de composants élémentaires. Enfin, le *writer* enregistre les documents analysés dans des fichiers, dans l'index d'un SRI, ou dans une base de données. Il est possible de chaîner plusieurs *writer* pour enregistrer différentes parties des documents sur différents supports.

Notons que les *analysers* ne se limitent pas forcément à une tokenisation. Toute méthode d'analyse ou d'enrichissement des documents est envisageable. Par exemple, il est possible d'extraire tous les mots indiqués en gras dans un texte en *html*.

Notons encore que cette phase d'analyse peut être faite en parallèle. La parallélisation est simple car chaque document est traité indépendamment. Nous utilisons en particulier l'outil *gnu parallel*[6] pour cela.

6.3.6.2 Calcul des fréquences

Certains SRI permettent d'indexer des champs de textes des documents dont le découpage en *tokens* est déjà fait. Le SRI calcule alors les fréquences totales de ces termes. Le plus simple pour cette étape est donc de se reposer sur un SRI permettant cela. En effet aucune adaptation intéressante et spécifique à *Kodex* n'est envisageable ici. Il s'agit simplement de compter, pour chaque terme, le nombre de documents qui l'utilisent et sa fréquence totale dans la collection.

Pour utiliser ainsi un SRI, il suffit que le *writer* de l'étape précédente donne le document à indexer au SRI.

6.3.6.3 Calcul de score et sélection des termes

Cette étape peut se faire en-ligne ou hors-ligne. Une approche entièrement en-ligne pose un problème de performances. Le principal goulot d'étranglement est le chargement en mémoire des termes associés aux n documents retournés, ces informations étant sur le disque dur. À l'inverse une sélection en-ligne à l'avantage de permettre de choisir les termes en connaissant la requête et les autres documents retournés. Nous nous basons donc sur une approche mixte : une première sélection est faite hors-ligne, et une seconde est faite en ligne.

La sélection des termes (ou *token*) hors-ligne est faite avec la chaîne de traitement suivante :

```
MinimalReader()| TermExpander()| TermScoring()| Cut()| Writer()
```

6. http://www.gnu.org/software/parallel/

À partir d'une liste d'identifiants (de documents), le composant `MinimalReader()` construit des *KodexDoc* vides ou ne contenant que le minimum d'information nécessaire. Dans chacun de ces *KodexDoc* « vides », le composant `TermExpander()` charge les termes et leurs statistiques associés depuis l'index du SRI.

Un score pour chaque couple document-terme est ensuite calculé par le composant `TermScoring()`. Ces scores sont stockés dans les *KodexDoc*. Le composant `Cut()` permet ensuite d'appliquer un seuil pour filtrer les termes en fonction de leurs scores. Notons que plusieurs scores peuvent être calculés (par un ou différents composants) et plusieurs seuils sont applicables sur chacun de ces scores.

Enfin un composant `Writer()` sauvegarde les termes sélectionnés (et éventuellement les scores associés) pour chaque document. Ce stockage est fait soit dans une base de données spécifique, soit en modifiant les données enregistrées par le SRI utilisé à l'étape précédente.

La seconde sélection, en-ligne, se fait avec une chaîne de composants semblable à la chaîne `TermScoring()| Cut()`. La différence est que ces composants peuvent alors avoir accès à l'ensemble des documents retournés, et éventuellement à la requête.

Notons ici l'un des avantages de la structure modulable de *Kodex* : pour une phase de mise au point ou d'expérimentation, il est possible d'utiliser en-ligne la partie centrale de la chaîne de traitement prévue pour être utilisée hors-ligne. C'est-à-dire la chaîne :

```
TermExpander()| TermScoring()| Cut()
```

Le temps de réponse pour une requête devient alors trop important pour une application réelle. Mais cela permet, par exemple, de lancer des jeux de tests et d'évaluations sans avoir à effectuer une passe complète sur la collection et sans avoir à stocker les sélections intermédiaires.

6.4 Évaluation sur une collection de deux millions de documents

Nous présentons dans cette section une application de *Kodex* sur une collection de 2,6 millions de pages web. Cette application a été mise en place dans le cadre du programme de recherche européen *Quaero*[7] afin de proposer une évaluation de *Kodex*. Après avoir présenté la collection et la configuration de *Kodex* mise en place, nous présentons les résultats de cette évaluation.

7. http://www.quaero.org/

6.4.1 Collection et jeu d'évaluation

La collection utilisée est composée de 2,6 millions de pages Web issues du domaine français (.fr) et aspirées en 2008 par le moteur de recherche *Exalead*[8]. Chacune de ces pages a été transcrite au format TREC [Harman, 2005], en extrayant uniquement le texte brut (suppression de toutes les balises *html*), l'url et le titre.

Pour l'évaluation, un corpus de 25 *sujets* (*topics*) a été constitué. Un *sujet* comprend une requête textuelle (`Title`) et un texte explicitant le besoin en information (`Narrative`). Ces derniers sont des besoins réels qui ont été soumis par des utilisateurs d'*Exalead* (extraits du log de ce moteur de recherche). Le listing 6.6 donne un exemple de *topic* codé en XML. Pour chacun de ces 25 *topics*, des *jugements de pertinence* ont été recueillis en suivant une procédure similaire à celle utilisée dans TREC [Harman, 2005]. Il a été constitué un *pool* de résultats issus de 144 configurations différentes du SRI *Terrier* [Ounis et al., 2005]. Ces configurations ont été construites en utilisant différentes formes d'indexation, différents modèles de recherche, et en réalisant ou non l'expansion de requêtes. Ce *pool* a ensuite été évalué manuellement pour identifier les documents pertinents de chaque requête.

```
<Topic Topic_Id="17">
  <Title>le gouffre de poudrey</Title>
  <Narrative>On souhaite un minimum d'information sur la
     localisation, ou la taille, ou les caractéristiques du
     gouffre de Poudrey. La seule comparaison de ce gouffre à un
     monument (par exemple) n'est pas pertinente.
  </Narrative>
</Topic>
```

Listing 6.6 – Exemple de requête d'évaluation en XML

6.4.2 Configuration de *Kodex*

Nous décrivons ici la configuration de *Kodex* mise en place sur cette collection.

6.4.2.1 Collection, indexation et recherche

Nous avons utilisé le SRI *Terrier* [Ounis et al., 2005] pour indexer les documents et effectuer la phase de recherche. Ce SRI a l'avantage de proposer beaucoup des modèles de recherche de l'état de l'art, et de fournir un système d'indexation pour les documents au format TREC.

Pour les évaluations de *Kodex*, nous avons choisi la configuration de *Terrier* suivante. Les documents ont été indexés en appliquant une radicalisation (*stemming*)

8. http://www.exalead.fr/

par l'algorithme « Snowball » de Porter [2001] pour le français. Les mots vides ont été éliminés. Pour la recherche le modèle de pondération *Okapi BM25* a été utilisé avec des paramètres par défaut [Manning et al., 2008, chap. 11]. Le nombre de documents retournés pour chaque requête a été fixé à : $n_D = 100$.

6.4.2.2 Modélisation des documents et construction du graphe

La modélisation des documents pour la construction du graphe a été faite à partir des termes extraits par *Terrier* pour la phase de recherche. C'est-à-dire les termes ont été extraits des documents par une tokenisation et une radicalisation par l'algorithme de Porter [2001] pour le français. Les termes ayant une *fréquence documentaire*[9] inférieure à $f_{df} = 100$ sont éliminés. Lors d'une requête, pour chaque terme est calculé le score maximale de *BM25* qu'il obtient parmi les documents retournés. Les termes ayant un tel score inférieur à $f_{BM25} = 5$ sont éliminés. Enfin le nombre de termes total du graphe est limité à $n_T = 1000$, en utilisant ce score de *BM25* maximal pour ordonner les termes. Notons que le premier filtre peut être appliqué hors-ligne (voir sous-section 6.3.6) alors que les deux dernières sélections sont forcément faites en-ligne.

Le graphe biparti construit peut être pondéré de différentes manières. Nous avons évalué les pondérations entre documents et termes suivantes :

one, le graphe n'est pas pondéré, chaque lien a un poids de 1,

tf, le nombre d'occurrence du terme dans le document ($tf_{doc,term}$) est utilisée,

tfidf, mesure de tf-idf classique :

$$tfidf(doc, term) = tf_{doc,term} * \log\left(\frac{N}{df_{term}}\right)$$

où N indique le nombre de documents dans la collection et df_{term} est la fréquence documentaire du terme (c'est à dire le nombre de documents contenant le terme),

tfidf$_{\mathbf{RD}}$, variante de la mesure de tf-idf, la partie idf est calculée sur les documents retournés (RD) :

$$tfidf_{RD}(doc, term) = tf_{doc,term} * \log\left(\frac{n_D}{df_{RD}(term)}\right)$$

où n_D indique le nombre de documents retournés, et $df_{RD}(term)$ le nombre de documents retournés contenant le terme,

9. Nombre de documents contenant le terme (*document frequency* en anglais).

logfratio, score naïf :

$$logfratio(doc, term) = \log(\frac{tf_{doc,term}}{TF(term)}) - \log(\frac{len(doc)}{N})$$

où $TF(term)$ est la fréquence totale du terme dans la collection, $logfratio$ est donc le ratio entre la fréquence du terme dans le document par rapport à sa fréquence totale et la taille du document par rapport au nombre total de documents,

bm25, score de *BM25* avec paramètres par défaut, voir [Manning et al., 2008, chap. 11].

6.4.2.3 Clustering du graphe biparti

Pour cette évaluation, différentes méthodes de clustering de graphe biparti ont été utilisées. Ces méthodes sont présentées dans le chapitre précédent (voir chapitre 5), nous les résumons ici rapidement.

WalktrapB : adaptation de la méthode *Walktrap* de Pons et Latapy [2006] pour des graphes bipartis, voir [Navarro et al., 2011] pour plus de détails,

Infomap : méthode de clustering basée sur la compression de la description de la trajectoire d'un marcheur aléatoire [Rosvall et Bergstrom, 2008], Pour une description plus en détail voir section 5.1.3,

fca.Infomap : méthode *Infomap* appliquée sur le graphe documents-concepts formels, c'est la méthode présentée au chapitre précédent en section 5.3.

confl.fca.Infomap, confl.fca.WalktrapB : méthodes *Infomap* et *WalktrapB* appliquées sur le graphe documents-concepts formels, mais cette fois construit après un filtrage du graphe documents-mots par la méthode présentée en section 5.4.

Pour chacune des méthodes, le post-traitement suivant est appliqué : tous les clusters comportant moins de $n_{doc-min} = 5$ documents sont regroupés dans un seul cluster « divers ». Cela permet d'éviter d'avoir des clusters trop petits ce qui est souhaitable sur une application réelle, mais aussi pour l'évaluation que nous présentons dans la sous-section suivante. En effet il est « facile » pour un petit cluster (un ou deux documents par exemple) d'avoir une précision très forte. Comme nous allons le voir, l'évaluation choisit le meilleur cluster grâce à la précision, l'un des biais possible de l'évaluation est donc de ne prendre en compte que des « mini »-clusters. Regrouper ainsi ces petits clusters permet d'éviter ce biais.

6.4.3 Évaluation

L'évaluation que nous présentons ici permet de mesurer dans quelle mesure un des clusters retourné par *Kodex* regroupe les documents pertinents. L'hypothèse est que si les clusters sont « corrects », ils correspondent chacun à une « interprétation » de la requête (ou à un « sous-sens » de celle-ci) par rapport à la collection de documents. Le sens de la requête entendu par l'utilisateur doit donc correspondre à l'un de ces clusters.

Plus pragmatiquement, *Kodex* peut avoir de l'intérêt pour un utilisateur seulement si cet utilisateur peut trouver plus de documents pertinents dans un cluster *bien choisi* que dans la liste brute retournée par le SRI.

Une première version de cette évaluation a été présentée dans [Navarro et al., 2011]. Nous présentons ici une évaluation plus complète, basée sur le même jeu de données. Il faut noter que les valeurs des résultats présentés ici pour la méthode *WalktrapB* diffèrent légèrement de celles indiquées dans ce papier. Cela est dû à une configuration différente du SRI utilisé. Nous utilisons ici la version 3 de *Terrier* alors que la version 2 était utilisée pour ce premier article. De plus les termes sont ici radicalisés alors qu'ils étaient simplement tronqués lors de la première évaluation. Notons que les nouveaux résultats obtenus avec ces changements sont compatibles avec les conclusions que nous tirions de cette première évaluation.

6.4.3.1 Méthode d'évaluation

Comme nous l'avons vu, cette évaluation a pour but de valider l'hypothèse suivante : à partir d'un ensemble de documents restitués par un moteur de recherche pour une requête, *Kodex* est capable de construire un cluster contenant un maximum de documents pertinents. Pour ce faire, nous exploitons la collection présentée en section 6.4.1.

Afin d'évaluer l'intérêt du clustering de *Kodex* par rapport à une organisation sous forme de liste de documents, nous nous basons sur la procédure MK1-k décrite par Tombros et al. [2002]. Cette évaluation se base sur les deux ensembles de documents e_{kodex} et e_{SRI} provenant respectivement de *Kodex* et du SRI utilisé. L'ensemble e_{kodex} contient les documents du « meilleur cluster » de *Kodex* (contenant m documents) et e_{SRI} contient les m premiers documents de la liste de *Terrier*. Nous décrivons plus bas comment ce « meilleur cluster » est déterminé. Par exemple, si le « meilleur cluster » de *Kodex* e_{kodex} contient 18 documents alors e_{kodex} est comparé à e_{sri} qui contient les 18 documents les mieux classés par *Terrier*. Les ensembles e_{kodex} et e_{SRI} sont alors comparables grâce aux mesures rappel (R), précision (P) et F_1 définies comme suit, pour un système s ($\in \{Kodex, SRI\}$) et un topic q [Jardine et van Rijsbergen, 1971].

$$R(s,q) = \frac{\text{Nombre de documents pertinents dans } e_s \text{ pour } q}{\text{Nombre de documents pertinents dans le top-100 du SRI pour } q} \quad (6.1)$$

$$P(s,q) = \frac{\text{Nombre de documents pertinents dans } e_s \text{ pour } q}{\text{Nombre de documents dans } e_s \text{ pour } q} \quad (6.2)$$

$$F_1(s,q) = \frac{2 \cdot R(s,q) \cdot P(s,q)}{R(s,q) + P(s,q)} \quad (6.3)$$

Pour identifier le « meilleur » cluster (e_{kodex}) une possibilité serait de montrer tous les clusters à des utilisateurs pour qu'ils sélectionnent celui qui leur semble le plus pertinent. On retiendrait alors le cluster le plus consensuel.

Pour ne pas introduire un facteur (humain) externe à l'évaluation et afin de garantir sa reproductibilité, nous avons plutôt opté pour un critère objectif de sélection de ce meilleur cluster. La mesure de *précision* nous est apparue la plus adaptée car le « meilleur » cluster est celui ayant la plus forte concentration de documents pertinents.

Pour chaque requête la procédure d'évaluation consiste donc à calculer les scores de rappel, précision et F-score pour le cluster de meilleure précision, ainsi que sur la liste de documents retournés par le SRI coupée à la même taille. La moyenne de ces scores est ensuite calculée pour les 25 requêtes.

6.4.3.2 *Kodex* par rapport au SRI seul, et par rapport à *Carrot*

Le tableau 6.1 présente les résultats pour quatre méthodes de clustering différentes. La signification de chaque colonne est indiquée dans la légende. Les deux premières lignes correspondent à la chaîne *Kodex* configurée comme indiqué précédemment, avec les méthodes de clustering *Infomap* et *WalktrapB*. Dans ces deux cas le graphe est pondéré par le score de *BM25*. Les deux lignes suivantes indiquent les résultats avec un clustering effectué par le logiciel *Carrot* (voir section 6.2) qui implémente deux algorithmes (*k-means* et *lingo*). Notons qu'avec *Carrot* la même chaîne de traitement des documents est appliquée pour le filtrage des termes, par contre la pondération des termes n'est pas prise en compte. Le regroupement des petits clusters dans un cluster « divers » est aussi effectué lors du clustering par *Carrot*.

Notons que <F> est la moyenne des F-scores des meilleurs clusters, il ne correspond donc pas au F-score que l'on pourrait calculer à partir de <R> et <P>.

On observe, par exemple sur la configuration *Infomap* que les meilleurs clusters produits contiennent en moyenne 48% de bons documents (précision), et que cela représente en moyenne 45% de l'ensemble des bons documents retournés par le SRI (rappel). Sur cette configuration, ces mêmes meilleurs clusters comportent en

Table 6.1 – Résultats de *Kodex* et de *Carrot* par rapport à la liste ordonnée. $<|C|>$ indique le nombre moyen de clusters, $<|e_s|>$ la taille moyenne du meilleur cluster. <P>, <R> et <F> sont respectivement la précision, le rappel et la F-mesure moyenne du meilleur cluster. Entre parenthèses est indiqué l'écart-type. Enfin +P, +R et +F indiquent le pourcentage d'amélioration sur <P>, <R> et <F> par rapport à une liste de même taille retournée par le SRI.

	$<\|C\|>$	$<\|e_s\|>$	<P>	<R>	<F>	+P	+R	+F
Infomap	3.56	29.76	0.48 (0.33)	0.45 (0.37)	0.27 (0.14)	+30.5%	+17.4%	+24.8%
WalktrapB	4.52	25.84	0.44 (0.28)	0.51 (0.32)	0.33 (0.17)	+18.2%	+30.8%	+21.7%
Carrot (k-means)	9.64	17.60	0.51 (0.27)	0.38 (0.25)	0.34 (0.13)	+20.7%	+24.4%	+20.4%
Carrot (lingo)	11.04	10.48	0.53 (0.30)	0.29 (0.24)	0.27 (0.10)	+15.2%	+29.5%	+12.8%

moyenne un peu moins de 30 documents. Cela représente une amélioration moyenne de 30% pour la précision, de 17% pour le rappel et de 24% pour le F-score par rapport aux premiers documents retournés par le SRI.

Pour les différentes mesures, les résultats comparatifs (+P, +R et +F) montrent clairement l'amélioration apportée par rapport aux top-k documents de la liste du SRI. Ce premier résultat valide donc l'intérêt de la méthode.

On observe que les deux algorithmes de *Carrot* tendent à produire plus de clusters, plus petits, que *Infomap* et *WalktrapB*. Il en résulte que *Carrot* produit des clusters de meilleures précisions mais avec de moins bons rappels que ceux construits par *Kodex*. Dans une moindre mesure, on observe aussi que *Infomap* tend à avoir de meilleures précisions, alors que *WalktrapB* privilégie le rappel.

Notons que la précision et le rappel moyen de ces quatre systèmes sont aussi indiqués sur la figure 6.3.

6.4.3.3 Comparaison de la méthode de pondération du graphe

Nous comparons ici les résultats des méthodes de clustering *Infomap* et *WalktrapB*. Les résultats sont aussi indiqués dans le tableau 6.2 et les performances de chaque configuration, en termes de précision et rappel moyen, sont représentées de manière plus lisible sur la figure 6.3. Chaque point indique la précision (ordonnée) et le rappel moyen (abscisse) d'une configuration donnée. Les carrés (bleus) correspondent à la méthode de clustering *Infomap* (avec différentes pondérations), les losanges (rouges) la méthode *WalktrapB* (là aussi avec différentes pondérations) et les triangles (violets) indiquent que le clustering est fait avec *Carrot*. Enfin les croix noires reliées par une ligne indiquent les performances de *Terrier* sur les k premiers documents. A chaque croix un document de plus est pris en compte, et les croix correspondant à une liste de 20 puis de 30 documents sont indiquées.

La remarque faite précédemment est clairement observable sur la figure 6.3 : *lingo* et *k-means* privilégient la précision par rapport au rappel. Et cela plus que ne le fait *Infomap*. *Infomap* à son tour est dans le même cas par rapport à *WalktrapB*.

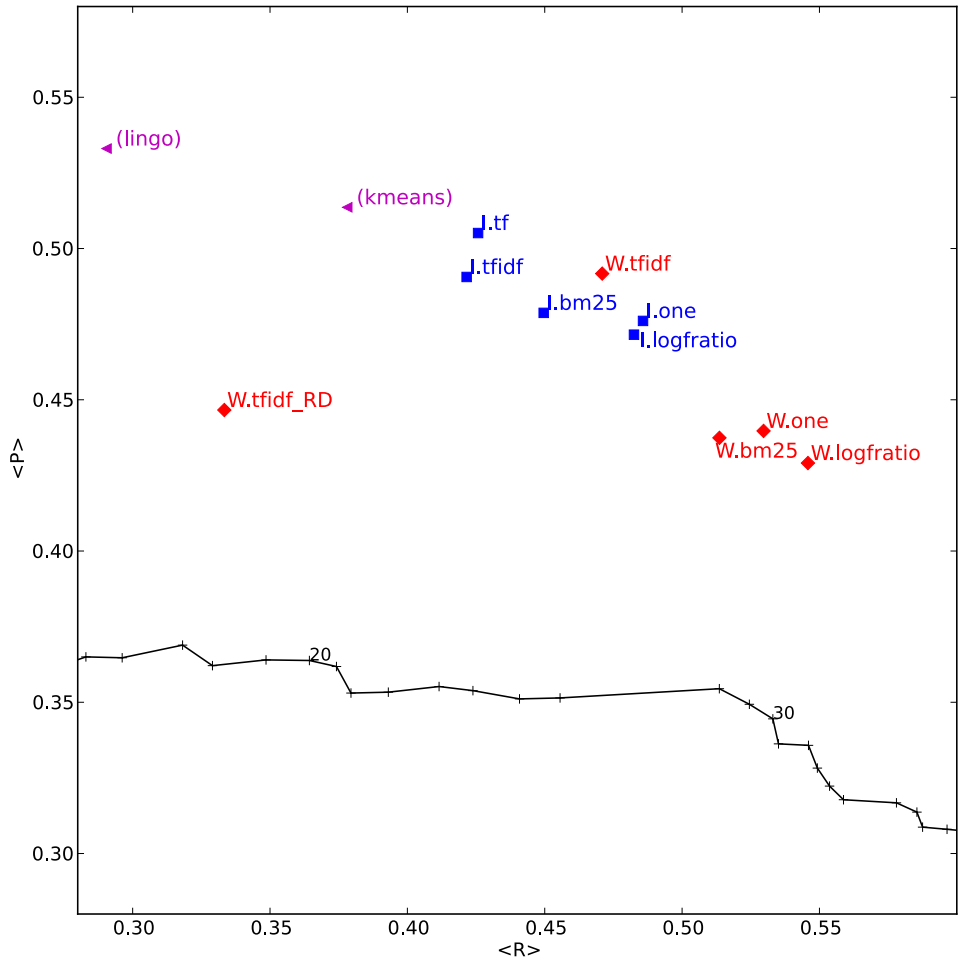


Figure 6.3 – Précision et rappel moyen du meilleur cluster pour différentes pondérations du graphe. I indique l'utilisation de *Infomap*, W celle de *WaltrapB*. La ligne noire indique les performances de la liste ordonnée du SRI, chaque point correspondant à la prise en compte d'un nouveau document, les points correspondant à une liste de 20 et 30 documents sont indiqués.

Table 6.2 – Resultats de *Kodex* en fonction de la pondération du graphe <$|C|$> indique le nombre moyen de clusters, <$|e_s|$> la taille moyenne du meilleur cluster. <P>, <R> et <F> sont respectivement la précision, le rappel et la F-mesure moyenne du meilleur cluster. Entre parenthèses est indiqué l'écart-type.

		$\|C\|$	<$\|e\_s\|$>	<P>	<R>	<F>
Infomap	one	3.88	31.24	0.48 (0.33)	0.49 (0.39)	0.26 (0.13)
	tf	3.40	26.60	0.51 (0.34)	0.43 (0.37)	0.25 (0.14)
	tfidf	3.28	27.92	0.49 (0.31)	0.42 (0.37)	0.26 (0.14)
	tfidf_{RD}	1.88	77.60	0.29 (0.25)	0.86 (0.32)	0.31 (0.21)
	logfratio	3.64	32.44	0.47 (0.33)	0.48 (0.39)	0.26 (0.14)
	BM25	3.56	29.76	0.48 (0.33)	0.45 (0.37)	0.27 (0.14)
WalktrapB	one	4.64	26.40	0.44 (0.26)	0.53 (0.30)	0.35 (0.18)
	tf	3.04	64.20	0.33 (0.25)	0.81 (0.30)	0.37 (0.23)
	tfidf	4.96	20.92	0.49 (0.30)	0.47 (0.26)	0.36 (0.18)
	tfidf_{RD}	4.40	17.96	0.45 (0.30)	0.33 (0.26)	0.25 (0.13)
	logfratio	4.60	27.40	0.43 (0.25)	0.55 (0.30)	0.36 (0.17)
	BM25	4.52	25.84	0.44 (0.28)	0.51 (0.32)	0.33 (0.17)

Concernant le choix des pondérations, on observe tout d'abord que la mesure $tfidf_{RD}$ ne fonctionne ni pour *walktrap* ni pour *Infomap*. Avec *Infomap* cela tend à créer un seul gros cluster (77,6 sommets en moyenne, sur 100) et *WalktrapB* les performances sont clairement moins bonnes qu'avec les autres pondérations.

On remarque aussi que la pondération tf se comporte sur *WalktrapB* comme $tfidf_{RD}$ sur *Infomap* : un cluster regroupant la quasi totalité des documents est construit. Par contre sur *Infomap*, tf produit des résultats comparables aux autres pondérations.

Notons ensuite que pour *Walktrap* la pondération $tfidf$ est remarquable dans le sens où elle amène une meilleure précision et un moins bon rappel que les trois autres pondérations (*one*, $logfratio$ et $bm25$).

Mis à part ces différences, les mesures observées ici ne permettent pas de mettre en évidence des comportements différents entre ces pondérations sur les deux méthodes de clustering utilisées. Le plus surprenant est certainement que la pondération *one*, c'est-à-dire une absence de pondération, produit un résultat comparable par exemple à la mesure *BM25*. Cela s'explique très certainement à cause du filtrage des termes effectué en amont (voir sous-section 6.4.2). Seul les termes ayant des scores élevés ont été gardés et donc la pondération utilisée lors du clustering peut difficilement améliorer les résultats. Nous avons vu, par contre, que la pondération peut se retrouver contre-productive. Enfin *Infomap* et *WalktrapB* ne se comportent pas de la même manière avec ces pondérations.

Table 6.3 – Résultat de l'utilisation de l'AFC en amont du clustering, La pondération du graphe est faite par le score de *BM25*. <$|C|$> indique le nombre moyen de clusters, <$|e_s|$> la taille moyenne du meilleur cluster. <P>, <R> et <F> sont respectivement la précision, le rappel et la F-mesure moyenne du meilleur cluster. Entre parenthèses est indiqué l'écart-type.

		$\|C\|$	<$\|e\_s\|$>	<P>	<R>	<F>
fca.Infomap	one	4.32	26.48	0.50 (0.34)	0.49 (0.34)	0.31 (0.16)
	min	4.56	28.36	0.51 (0.34)	0.54 (0.34)	0.33 (0.17)
	max	4.12	30.36	0.47 (0.32)	0.54 (0.35)	0.32 (0.15)
	sum	3.44	39.08	0.44 (0.34)	0.59 (0.37)	0.30 (0.15)
	avg	4.40	26.72	0.52 (0.34)	0.50 (0.33)	0.31 (0.15)
	prod	1.08	97.16	0.24 (0.18)	1.00 (0.00)	0.35 (0.22)
confl.fca.Infomap	one	5.56	15.52	0.54 (0.33)	0.37 (0.32)	0.29 (0.15)
	min	5.56	14.92	0.55 (0.31)	0.36 (0.32)	0.29 (0.14)
	max	5.56	16.72	0.52 (0.32)	0.38 (0.33)	0.28 (0.15)
	sum	4.68	27.00	0.49 (0.35)	0.45 (0.38)	0.28 (0.17)
	avg	5.72	17.16	0.56 (0.33)	0.38 (0.33)	0.28 (0.15)
	prod	5.92	16.96	0.52 (0.32)	0.40 (0.33)	0.30 (0.16)
confl.fca.WalktrapB	one	6.64	13.32	0.53 (0.28)	0.37 (0.26)	0.32 (0.14)
	min	6.84	15.80	0.51 (0.31)	0.39 (0.27)	0.32 (0.15)
	avg	6.96	15.60	0.51 (0.29)	0.41 (0.29)	0.33 (0.17)

6.4.3.4 Passage par l'AFC et la confluence

Nous évaluons ici les méthodes de clustering passant par le graphe objets-concepts (ici documents-concepts) introduites au chapitre précédent (sections 5.3 et 5.4). La figure 6.4 présente de manière identique à la figure 6.3 les performances de différentes méthodes utilisant l'Analyse Formelle de Concepts (AFC). Ces résultats sont aussi indiqués dans le tableau 6.3. Pour toutes les méthodes présentées ici, le graphe documents-termes est pondéré par *BM25*. Les symboles pleins indiquent l'usage des méthodes de base, sans passer par l'AFC (*WalktrapB* et *Infomap*). Les symboles grisés correspondent à un clustering fait sur le graphe documents-concepts construit directement depuis le graphe documents-termes, enfin les symboles blancs correspondent à un clustering fait sur un graphe documents-concepts construit après un filtrage par la confluence du graphe documents-termes. Ce filtrage consiste à supprimer tous les liens ayant une confluence à temps $t = 3$ inférieure à 0.5.

On observe tout d'abord que le passage par le graphe des concepts (sans filtrage préalable) permet d'améliorer assez clairement les résultats d'*Infomap*. En effet, par rapport à *Infomap.bm25* (tableau 6.2), le rappel moyen et la précision moyenne augmentent pour les méthodes *fca.Infomap* avec les pondérations *avg*, *one* et *min*, Le passage par le graphe objets-concepts semble donc être intéressant ici.

Ce sont les méthodes d'agrégation des poids *min* et *avg* qui sont les plus performantes. Le méthode *prod* ne fonctionne pas : tous les sommets se retrouvant dans

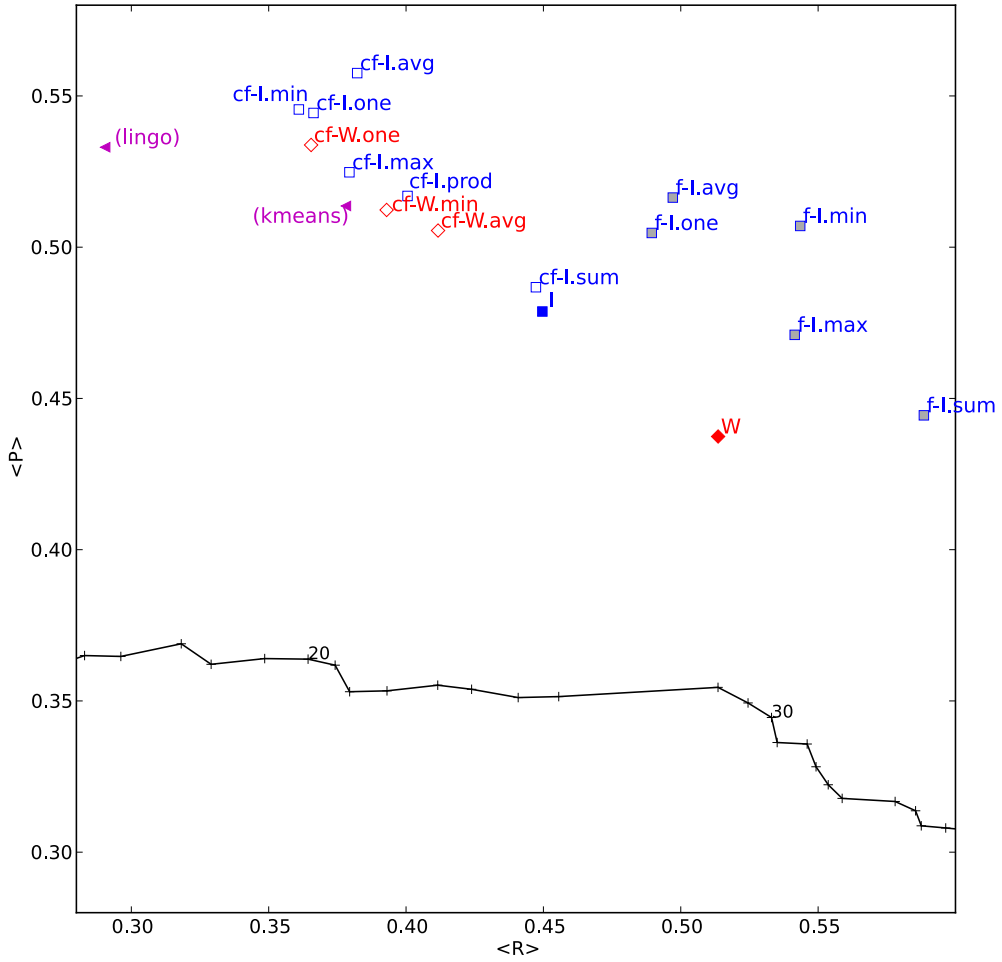


Figure 6.4 – Précision et rappel moyen du meilleur cluster, par un clustering du graphe documents-concepts. Pour améliorer la lisibilité, les noms des méthodes ont été abrégés : I pour *Infomap*, f-I pour *fca.Infomap*, cf-Infomap pour *confl.fca.Infomap*, de même W indique *WalktrapB*. La ligne noire indique les performances de la liste ordonnée du SRI, chaque point correspondant à la prise en compte d'un nouveau document, les points correspondant à une liste de 20 et 30 documents sont indiqués.

un seul cluster ($|C| = 1.08$).

Notons aussi que ce passage par l'AFC engendre davantage de clusters plus petits (4.56 clusters en moyenne pour *fca.Infomap.bm25.min* contre 3.56 pour *Infomap.bm25*).

Le filtrage du graphe par la confluence, comme nous l'avons vu au chapitre précédent, permet de réduire considérablement le nombre de concepts (voir le tableau 5.6 en section 5.4.3). Nous voyons ici que ce filtrage du graphe permet d'améliorer clairement la précision moyenne du meilleur cluster, mais au prix d'une dégradation du rappel. Le gain n'est donc pas évident. Notons tout de même que *confl.fca.Infomap.avg* donne un rappel semblable à la méthode *k-means* de *Carrot* mais avec une précision clairement supérieure. Nous observons ici que ce filtrage amène *Infomap* à construire plus de clusters, plus petits (par exemple 5.56 clusters en moyenne pour *confl.fca.Infomap.min* contre 4.56 pour *fca.Infomap.min*). Le filtrage coupe des liens « faibles » dans le graphe de départ, cela permet visiblement de mieux séparer certains groupes, et d'avoir ainsi davantage de clusters.

6.4.4 Conclusions et perspectives

Nous avons présenté dans cette section une évaluation de *Kodex* sur une collection de 2,6 millions de pages web. Nous avons pu montrer la capacité de l'approche à regrouper dans un cluster plus de documents pertinents qu'il y en a dans la liste ordonnée de même taille. Cette évaluation ne constitue toutefois qu'une première étape, elle présente en effet plusieurs limites.

Tout d'abord, cette évaluation se base sur l'hypothèse selon laquelle les documents pertinents doivent être regroupés dans un seul cluster. Cette hypothèse a l'avantage de permettre une évaluation en ne connaissant « que » la liste des documents pertinents pour un certain nombre de requêtes. Seulement il est envisageable qu'un clustering soit correct et informatif alors que les documents pertinents sont répartis dans plusieurs clusters. Typiquement dans le cas où l'utilisateur cherche à faire une recherche « en largeur » et donc cherche des documents (de qualité) dans différents clusters. Il est aussi possible que le découpage proposé par le système soit pertinent sans qu'il ne corresponde au découpage « attendu » par l'utilisateur. Imaginons par exemple une recherche sur un médicament, l'utilisateur cherchant des informations sur les effets secondaires. Un clustering classant les résultats en fonction du type de publication (forums médicaux, sites gouvernementaux, publications scientifiques, sites de laboratoires, etc.) peut être intéressant, mais des documents pertinents pour l'utilisateur peuvent alors être présents dans chacun de ces groupes.

Cette évaluation présente aussi une limitation plus pragmatique. En effet nous mesurons à quel point *un* cluster regroupe bien un nombre important de documents pertinents, mais cela ne nous dit rien sur les autres clusters. Est-ce que ces autres clusters sont pertinents ?

Le problème soulevé ici est que cette évaluation est indirecte par rapport à l'objectif visé par le système. Une évaluation plus directe consisterait « simplement » à vérifier pour un ensemble de documents donnés si les groupes construits automatiquement forment un découpage intelligible.

Un autre défaut de cette évaluation est dû à la moyenne faite sur les 25 requêtes. Nous avons pu observer que l'écart-type, associé à ces moyennes, est assez élevé. Cela indique que les résultats varient beaucoup entre les différentes requêtes. Il est alors naturel de se demander si toutes les configurations évaluées se comportent de la même manière sur les différentes requêtes. Indépendamment de la performance moyenne, est-ce que certaines configurations ne sont pas plus efficaces pour certains types de requêtes ? Le problème étant ensuite de pouvoir caractériser ces différents « types » de requêtes.

Enfin un des enjeux majeurs d'un tel système de classification automatique de résultats est l'étiquetage des clusters. Il faut en effet que la signification des groupes puisse être rapidement identifiée par l'utilisateur. Un travail, non présenté ici, a été entamé [10] sur ce point dans *Kodex* par Benoit Gaillard. La principale difficulté étant que cette évaluation ne permet pas d'évaluer la qualité d'un étiquetage des clusters.

6.5 Deux applications de *Kodex*

Nous présentons ici deux implémentations de *Kodex* qui ont été mises en place [11] en utilisant les interfaces de programmation (*API*) des moteurs de recherche du journal *Guardian* et de la base de publication *DBLP*.

Pour chacune de ces applications nous présentons la chaîne de traitement mise en place ainsi qu'une capture d'écran donnant une idée des résultats. Pour le moment, aucune évaluation n'a été construite sur ces applications. Notons que ces deux applications doivent être considérées comme des preuves de concept. Leur intérêt principal est de montrer le potentiel de *Kodex* sur une application réelle.

6.5.1 Application sur l'API de recherche du *Guardian*

Le site internet du journal anglais le *Guardian* propose une interface de développement (API) permettant d'interroger le moteur de recherche du journal [12]. Nous avons mis en place une application *Kodex* en utilisant cette API.

Pour cela nous avons développé un composant *Kodex* qui interroge l'API du *Guardian* et transforme les résultats obtenus en objets `KodexDoc` (voir section 6.3). Ce composant exploite quelques options disponibles sur cette API (la méthode de

10. Travail non encore publié.
11. Ces applications sont accessibles à l'adresse : http://www.irit.fr/kodex.
12. http://www.guardian.co.uk/open-platform/content-api-content-search-reference-guide

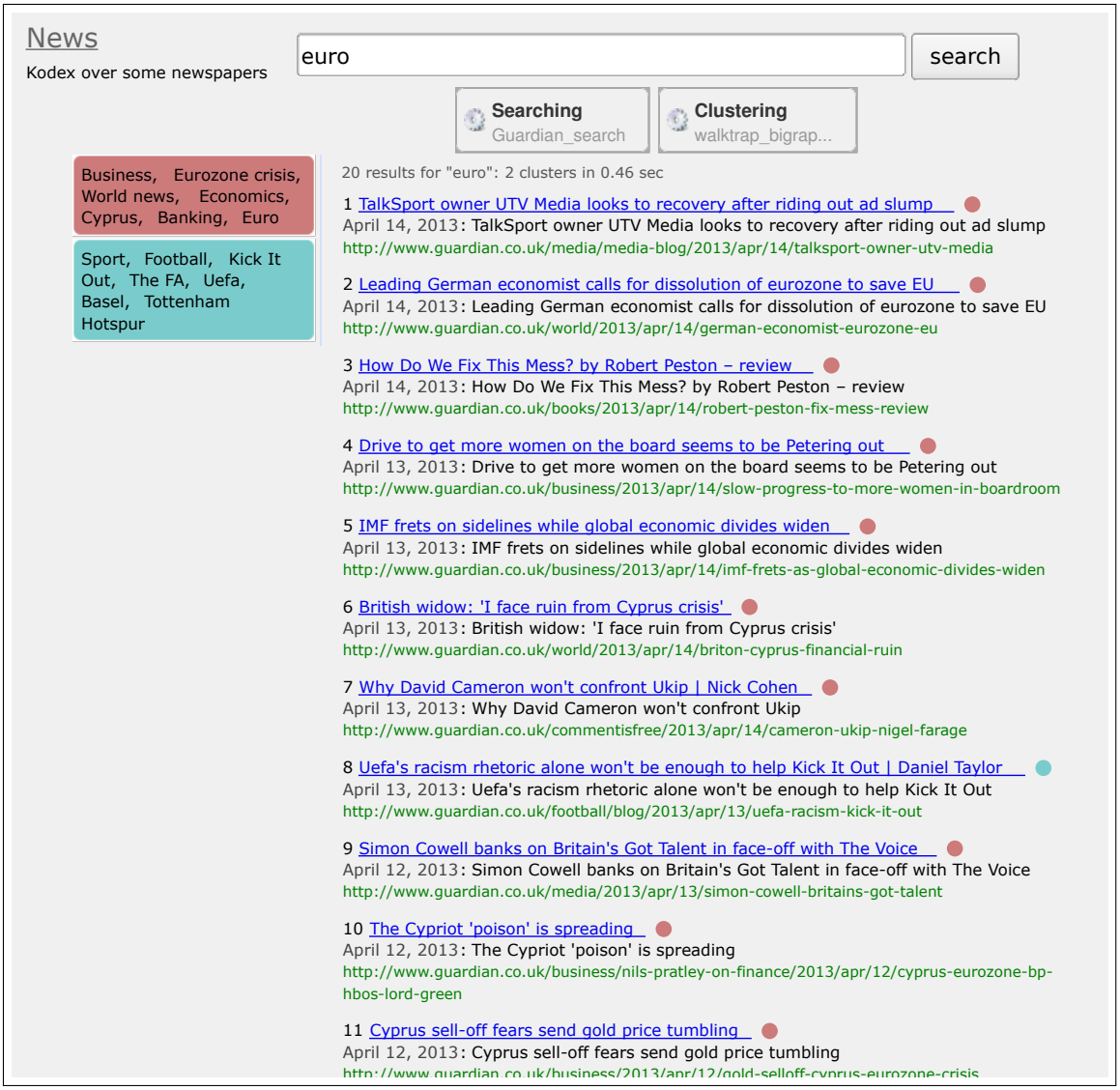


Figure 6.5 – Exemple d'application de *Kodex* par dessus le *Guardian*.
Pour la requête « euro », On observe que deux clusters sont crées : un premier en rapport avec la monnaie, un second avec le championnat de football.

classement des résultats, et le nombre de résultats). Le résultat de la recherche, retourné par le web-service du *Guardian*, indique pour chaque document, en plus des informations classiques (titre, résumé, auteur, date, url, etc.), une liste de mots-clés apposée sur chaque document par les journalistes ou la rédaction. Ce sont ces mots-clés qui sont utilisés pour construire le graphe biparti. L'application laisse ensuite le choix entre différentes méthodes de clustering. L'étiquetage des clusters peut être fait de différentes manières, dans tous les cas ce sont ces mots-clés qui sont utilisés. Une méthode simple, et assez efficace pour cette application, consiste à utiliser tous les mots-clés regroupés avec les documents dans les clusters.

La figure 6.5 présente une copie d'écran réalisée pour la requête « euro ». Une liste de 20 documents est présentée (coupée sur la figure), et deux clusters sont indiqués sur la gauche. On observe que ces deux clusters correspondent à deux emplois d'« euro » : la coupe d'europe de football et la monnaie européenne. L'appartenance des documents aux clusters est indiquée par des pastilles disposées après le titre de chaque document.

Plusieurs améliorations sont envisageables. La première est de mettre en place une évaluation afin de pouvoir mesurer objectivement comment des modifications de la chaîne de traitement améliore (ou dégrade) les résultats.

Il serait possible d'utiliser les titres et les extraits des articles pour ne pas modéliser les documents uniquement avec les mots-clés. Une pondération donnant plus d'importance aux mots-clés serait certainement utile.

Une autre amélioration importante pourrait résider dans une modification de la présentation des résultats. En l'état, par défaut, les 20 articles les plus récents répondant à la requête sont retournés et clusterisés. Il serait possible de demander à l'API du *guardian* plus de documents, de construire les clusters et ensuite de n'afficher que 20 documents en les prenant parmi chaque cluster. L'utilisateur n'aurait donc toujours que 20 documents listés mais des documents plus diversifiés. Le choix d'un cluster (par un clic) permettant d'afficher plus d'articles appartenant à ce cluster.

6.5.2 Application sur l'API de recherche de *DBLP*

Une démonstration de *Kodex* a été mise en place en utilisant l'interface de développement (API) proposée par la base de publications scientifiques *DBLP*[13]. Ce web-service retourne, pour une requête, une liste d'articles scientifiques correspondant. Pour chaque article diverses informations sont disponibles : titre, date, liste des auteurs, ...

Un composant *Kodex* de recherche interroge donc cette API, et transforme le résultat en une liste de `KodexDoc`. Dans cette première version, le graphe est construit en utilisant les listes d'auteurs. Chaque document est relié à ses auteurs, ainsi un

13. http://www.dblp.org/search/

Figure 6.6 – Exemple d'application de *Kodex* par dessus *DBLP*

graphe biparti articles-auteurs est construit. Ce graphe peut ensuite être clusterisé avec différentes méthodes. Enfin l'ensemble des auteurs des documents de chaque cluster est utilisé pour étiqueter les clusters.

La figure 6.6 présente une copie d'écran obtenue pour une recherche des articles dont Paul Erdös est auteur. La liste est limitée aux 100 articles les plus récents. On observe que 20 clusters ont été construits.

Cette application permet donc de visualiser quels sont les collaborateurs d'un auteur, et comment se regroupent ses collaborations. La principale limitation est qu'aucune donnée thématique n'est utilisée. Il est probable que les clusters correspondent à des thématiques différentes (différentes collaborations pour différentes thématiques). Mais quand bien même ce serait le cas, il n'est pas évident de comprendre quelles sont les thématiques de chaque cluster. Extraire des informations sur les thématiques abordées dans les articles n'est pas trivial. D'autant plus que la seule information thématique disponible facilement est le titre des articles.

Aussi la donnée exploitée est certainement « trop » locale. Imaginons par exemple un auteur A qui aurait collaboré une fois avec un auteur B et une fois (dans un autre article) avec un auteur C et que ces deux auteurs B et C aient écrit un nombre important d'articles ensemble. Si l'on ne connaît que les articles de A (comme c'est le cas pour le moment) alors on ne sait pas qu'il est possible de rapprocher B et C. Techniquement il faudrait disposer localement (i.e. sur le serveur de *Kodex*) de la totalité de la base de données pour pouvoir enrichir le graphe d'articles non retournés par la recherche mais pertinents vis à vis des auteurs présents. Cela demande une mise en place technique plus conséquente, néanmoins envisageable car l'ensemble de la base *DBLP* est disponible au téléchargement.

6.6 Conclusions du chapitre

Dans ce chapitre, nous avons présenté *Kodex*, un cadre logiciel modulable permettant de construire des applications de catégorisation automatique des résultats d'une recherche en utilisant des méthodes de clustering de graphe biparti.

Nous avons tout d'abord présenté un état de l'art des solutions proposées dans la littérature pour répondre à ce problème de catégorisation automatique des résultats d'une recherche. Ensuite l'architecture de *Kodex* a été exposée. Cette architecture modulable permet de mettre en place facilement une chaîne de traitement pour construire un clustering des résultats d'une recherche. *Kodex* permet aussi d'effectuer des pré-traitements hors-ligne.

Une application d'évaluation sur 2,6 millions de pages web a été présentée. Cette évaluation a permis de montrer que cette approche améliore les résultats par rapport à une simple liste ordonnée. Différentes méthodes de clustering ont pu être comparées. Il a été montré que notre approche est préférable aux méthodes implémentées

dans le logiciel *Carrot*. Aussi la méthode se basant sur le graphe objets-concepts présentée au chapitre précédent donne les meilleurs résultats.

Plusieurs pistes de travail futur sont envisagées. La première concerne l'évaluation. Nous avons vu les limites de l'évaluation présentée ici. Une nouvelle évaluation a été mise en place dans le cadre du projet Quaero et on devrait bientôt pouvoir en tirer des conclusions. Cette évaluation propose en particulier une tâche pour laquelle les documents retournés par un SRI ont été classés à la main en différentes catégories.

Une autre idée pour faciliter l'évaluation est de construire un système pour aider à annoter les résultats de *Kodex*. L'idée est de proposer un système pour faciliter la construction d'un jeu d'évaluation en capitalisant le temps passé à tester et à mettre au point une application *Kodex*. En effet, lors d'une phase de mise au point, on observe souvent de très bons résultats pour certains réglages, résultats qui ne sont pas nécessairement aussi bons avec d'autres paramètres. Il faudrait donc pouvoir sauvegarder aisément ces « bons » résultats, en y apportant éventuellement les quelques corrections qui semblent nécessaires. Ces requêtes enregistrées pourraient ainsi être rejouées et évaluées automatiquement, et constituer, a minima, des tests de non-régression. En effet un jeu de test ainsi construit n'aurait pas la même impartialité qu'une évaluation tierce de type TREC. Cela pourrait toutefois permettre une comparaison automatique utile et valable entre différents réglages ou algorithmes. L'autre avantage est que ces jeux d'évaluation seront construits directement sur les données des applications visées, et donc correspondent au plus près aux résultats souhaités.

En parallèle à cette évolution nécessaire des systèmes d'évaluation, de nombreuses améliorations sont envisageables dans les chaînes de traitement de *Kodex*. La modélisation des documents est certainement le point qui mérite le plus d'attention. En effet nous avons vu que la modélisation faite dans les applications présentées est relativement simple. En particulier sur l'application d'évaluation, la modélisation est faite par une radicalisation des termes des documents puis un filtrage de ceux-ci à l'aide de différentes pondérations.

Plusieurs améliorations sont possibles. La première concerne la construction des *tokens* utilisés. Une modélisation des documents autorisant des expressions multi-mots serait nécessaire pour l'étiquetage des clusters, mais cela serait certainement utile aussi pour la phase de clustering. Par exemple cela pourrait éviter de rapprocher des documents parlant de « Jacques Brel » et des documents à propos de « Jacques Cartier », ou encore des documents utilisant l'expression « tour d'ivoire » et d'autres qui parleraient, par exemple, d'un réseau de trafic d'« ivoire » qui utilise une « tour de guet » pour surveiller ces installations. À l'inverse cela permettrait de rapprocher plus fortement des documents employant les mêmes expressions. Le sens d'une expression multi-mots est moins ambigu que ceux de chacun de ces com-

posants pris indépendamment. Un système de tokenisation qui puisse extraire ces expressions multi-mots pourrait donc améliorer les performances de *Kodex*.

Il serait possible aussi de désambiguïser, hors-ligne, les *tokens* associés à un document. Par exemple différencier dans le modèle d'un document si « poulet » est utilisé pour parler de manière argotique de la police, ou pour s'exprimer sur un élevage de volailles.

Aussi un système plus complexe de sélection des *tokens* peut être envisagé. En utilisant, par exemple, des méthodes d'apprentissage automatique. Cela peut être pertinent sur des collections de documents assez homogènes.

Enfin un autre point d'amélioration important concerne le recouvrement entre clusters. Pour le moment les méthodes de clustering utilisées assignent chaque document à un seul cluster. Il semble pourtant souhaitable que certains documents apparaissent dans plusieurs clusters. Ce problème de clustering avec recouvrement n'est pas simple à résoudre. En effet les méthodes proposant un clustering avec recouvrement ont souvent comme défaut que le nombre de clusters retournés explose (voir chapitre précédent). Or sur cette application, l'objectif est bien de garder un nombre limité de clusters, quitte à perdre ou à contredire une partie de l'information disponible.

Conclusion

Nous nous sommes intéressés dans cette thèse à l'exploitation de données de « terrain » considérées sous forme de graphes : comment utiliser efficacement la structure de ces graphes, dits de terrain, pour en extraire une information « résumée » et pertinente en vue de diverses applications ? Dans une première partie l'information extraite prend la forme d'une mesure de similarité entre sommets, puis dans une seconde partie, d'un regroupement des sommets.

La première des contributions faites dans ce travail est l'introduction d'une nouvelle mesure de similarité entre sommets d'un graphe : la *confluence*. Cette mesure est basée sur des marches aléatoires en temps courts, elle a donc l'avantage de présenter une complexité temporelle raisonnable. Aussi la valeur de la *confluence* est normalisée entre zéro et un, et présente un seuil naturel à 0.5. Nous avons montré que ce seuil peut se comprendre soit par rapport à la limite en temps infini de la marche aléatoire permettant le calcul, soit par rapport à la valeur de cette même marche sur un graphe aléatoire équivalent. La confluence est, à notre connaissance, la seule similarité présentant ces caractéristiques.

Cette similarité est exploitée par la deuxième contribution. En effet nous introduisons GUD une méthode de comparaison de graphes se basant sur la *confluence*. GUD est une distance d'édition robuste entre graphes partageant les mêmes sommets. En comparant deux graphes, une arête présente seulement dans un des graphes indique un conflit. Pourtant ce conflit n'est pas de la même intensité si les deux sommets correspondants sont proches (bien que non adjacents) ou s'ils sont distants. Nous avons donc introduit une mesure qui évalue la « force » des conflits en utilisant la *confluence*. À notre connaissance, ce problème de comparaison de graphes dans sa forme particulière où les deux graphes partagent les mêmes sommets est nouveau. Nous avons par ailleurs pu montrer son intérêt sur une application de comparaison de ressources lexicales.

Une troisième contribution concerne le problème de clustering de graphe biparti. Cette contribution est double : nous avons introduit une nouvelle procédure de clustering ainsi qu'une méthode de prétraitement. La procédure de clustering permet, en deux étapes, de construire un partitionnement des sommets d'un type sans imposer de partitionnement correspondant sur les sommets de l'autre type. La méthode de prétraitement utilise la *confluence* pour simplifier un graphe biparti en amont de

l'utilisation d'une méthode de clustering. Nous avons montré l'intérêt de ces deux méthodes sur différents jeux d'évaluations artificiels et issus de données réelles.

Enfin deux contributions applicatives ont été présentés : le système d'enrichissement semi-automatique de ressources lexicales *Wisigoth* et le système de classification automatique des résultats d'une recherche *Kodex*. *Wisigoth* permet d'assister la construction de ressources lexicales en proposant à l'utilisateur des candidats synonymes. Ces candidats sont calculés en utilisant une mesure de similarité entre sommets d'un graphe, la ressource déjà existante étant employée comme donnée source. Appliquée au dictionnaire collaboratif *Wiktionary*, nous avons montré que le système permet de proposer des candidats pour la quasi totalité des entrées et que 30 à 60% des propositions permettent d'ajouter une relation de synonymie pertinente d'après une ressource étalon externe.

Le système *Kodex* permet d'organiser automatiquement les résultats d'une recherche d'information et ainsi d'exposer à l'utilisateur la polysémie éventuelle de la requête. C'est une application directe des méthodes de clustering de graphe biparti. Nous avons présenté les grandes lignes de l'architecture logicielle modulable mise en place. Aussi une campagne d'évaluation sur une collection de plus de deux millions de pages web a été présentée. Cette évaluation a permis de montrer l'intérêt de l'approche, tant par rapport à une simple liste ordonnée que par rapport à des systèmes de clustering de documents de l'état de l'art. Enfin deux applications en ligne basées sur *Kodex* ont été introduites.

En marge de ces différentes contributions, ce rapport présente un travail de synthèse de l'état de l'art des similarités entre sommets et de celui concernant le clustering de graphe biparti. En effet, nous avons fourni un panorama aussi complet que possible des mesures de similarité entre sommets d'un graphe. Nous avons aussi proposé une comparaison expérimentale de ces mesures. Cette comparaison permet de mieux cerner les différences et ressemblances qui peuvent exister entre ces mesures. Et enfin nous proposons, dans la seconde partie, un tour des différentes familles d'approches pouvant se comprendre en termes de clustering de graphe biparti. Nous avons en particulier développé un parallèle entre l'analyse formelle de concept et l'analyse de graphe biparti. Ce parallèle forme le socle nécessaire aux contributions faites dans le chapitre correspondant.

De nombreuses perspectives ont été évoquées dans ce rapport, nous ne les répétons donc pas ici. Nous souhaitons tout de même insister sur le fait que le travail de synthèse entamé dans ce rapport peut être continué. Le nombre impressionnant d'approches et de famille d'approches se rapportant, ou pouvant se rapporter, à la métrologie des graphes de terrain rend cette tâche tant difficile que nécessaire. Par exemple, comme nous l'avons déjà remarqué, un parallèle plus approfondi entre les méthodes de bi-clustering (utilisées en particulier en génétique) et les méthodes de clustering de graphe (et de graphe biparti) serait profitable. En plus de permettre

des transferts de connaissances entre les différentes communautés développant les méthodes mises en parallèle, de tels parallèles permettent de mieux comprendre les hypothèses parfois implicites faites par les méthodes, et donc amènent naturellement à la construction de meilleures approches.

Bibliographie

L. Adamic et E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3) : 211–230, 2003.

G. Adda, B. Sagot, K. Fort, et J. Mariani. Crowdsourcing for Language Resource Development : Critical Analysis of Amazon Mechanical Turk Overpowering Use. In *LTC 2011 : Proceedings of the 5th Language and Technology Conference*, Poznan, Poland, 2011.

R. Agrawal, T. Imielinski, et A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD Inter. Conf. on management of data*, pages 207–216, New York, 1993.

O. Allali, C. Magnien, et M. Latapy. Internal link prediction : A new approach for predicting links in bipartite graphs. *Intell. Data Anal.*, 17(1) :5–25, 2013.

M. M. Babu. Introduction to microarray data analysis. In R. P. Grant, editor, *Computational Genomics : Theory and Application*. Norwich, Horizon Press, 2004.

A. L. Barabási et R. Albert. Emergence of scaling in random networks. *Science*, 286 (5439) :509–512, 1999.

M.J. Barber. Modularity and community detection in bipartite networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 76(6), December 2007.

M. Barbut et B. Monjardet. *Ordre et classification : algèbre et combinatoire*. Hachette, 1970.

R Belohlavek. Similarity relations in concept lattices. *Journal of Logic and Computation*, 10(6) :823–845, 2000.

R. Belohlavek. *Fuzzy Relational Systems : Foundations and Principles*. Kluwer Academic Publishers, 2002.

R. Belohlavek, J. Dvořák, et J. Outrata. Fast factorization by similarity in formal concept analysis of data with fuzzy attributes. *Journal of Computer and System Sciences*, 73(6) :1012 – 1022, 2007.

C. Berge. *Graphes et hypergraphes*. Monographies universitaires de mathématiques. Dunod, Paris, 1970.

C. E. Bichot et P. Siarry. *Partitionnement de graphe*. Lavoisier, 2010.

V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, et P. V. Dooren. A measure of similarity between graph vertices : Applications to synonym extraction and web searching. *SIAM Rev.*, 46 :647–666, 2004. ISSN 0036-1445.

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, et D.-U. Hwang. Complex networks : Structure and dynamics. *Physics Reports*, 424(4–5) :175–308, 2006.

D. Boley, M. Gini, R. Gross, E.H. (Sam) Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, et J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3) :329–341, December 1999.

B. Bollobás. *Modern Graph Theory*. Graduate texts in mathematics. Springer, 1998.

Y. Boniface. Sluci : outil open source de text-mining en français, 2011. http://sulci.fr/.

O. Bouidghaghen, M. Boughanem, H. Prade, et I. Mallak. A fuzzy logic approach to topic extraction in texts. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 17(Supplement-1) :81–112, 2009.

Didier Bourigault. *Un Analyseur Syntaxique Opérationnel : SYNTEX*. Mémoire d'habilitation à diriger des recherches, Université de Toulouse, 2007.

S. Brin et L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7) :107–117, 1998.

M. Brunello. The Creation of Free Linguistic Corpora from the Web. In *Proceedings of WAC5 : 5th Workshop on Web As Corpus*, pages 37–44, San Sebastian, 09 2009.

S. Busygin, O. Prokopyev, et P. M. Pardalos. Biclustering in data mining. *Computers and Operations Research*, 35(9) :2964–2987, September 2008. ISSN 0305-0548.

C. Carpineto, S. Osiński, G. Romano, et D. Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3) :1–38, 2009.

P. J. Carrington, J. Scott, et S. Wasserman, editors. *Models and methods in social network analysis*. Cambridge University Press, 2005.

L. Cerf, P.N. Mougel, et J.F. Boulicaut. Agglomerating local patterns hierarchically with alpha. In *Proc. of the 18th ACM Conf. on Information and Knowledge Management*, CIKM'09, pages 1753–1756, New York, 2009.

S.H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *Int. Journal of Mathematical Models and Methods in Applied Sciences*, 1(4) :300–307, 2007.

A.K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky, et P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *computational complexity*, 6(4) :312–340, 1996.

M. Chaput. Whoosh : a full-text indexing and searching library implemented in pure python, 2009.
https://bitbucket.org/mchaput/whoosh/wiki/Home.

P. Chebotarev et E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9) :1505–1514, 1997.

P. Chebotarev et E. Shamis. On proximity measures for graph vertices. *Automation and remote control*, 59(10) :1443–1459, 1998.

M. Chein et M.-L. Mugnier. *Graph-based Knowledge Representation : Computational Foundations of Conceptual Graphs*. Springer, 1 edition, 2008.

J. Chen, O. R. Zaïane, et R. Goebel. An unsupervised approach to cluster web search results based on word sense communities. In *Web Intelligence*, pages 725–729, 2008.

H. Cheng, P. S. Yu, et J. Han. Ac-close : Efficiently mining approximate closed itemsets by core pattern recovery. In *Proc. of the 6th Inter. Conf. on Data Mining*, ICDM'06, pages 839–844, Washington, 2006.

S. S. Choi, S. H. Cha, et C. Tappert. A Survey of Binary Similarity and Distance Measures. *Journal on Systemics, Cybernetics and Informatics*, 8(1) :43–48, 2010.

A. Clauset, M. E. J. Newman, et C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6), Dec 2004.

A. Clauset, C. Shalizi, et M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4) :661–703, 2009.

G. Csardi et T. Nepusz. The igraph software package for complex network research. *InterJournal Complex Systems*, 1695, 2006.

J. R. Curran et M. Moens. Improvements in Automatic Thesaurus Extraction. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition*, pages 59–66, 2002.

D. R. Cutting, D. R. Karger, J. O. Pedersen, et J. W. Tukey. Scatter/Gather : a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329, Copenhagen, Denmark, 1992. ACM.

L. Danon, A. Díaz-Guilera, J. Duch, et A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, 2005(09) : P09008, 2005.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, et R. A. Harshman. Indexing by Latent Semantic Analysis. *J. Am. Soc. Inf. Sci.*, 41(6) :391–407, 1990.

J. -C Delvenne, S. N Yaliraki, et M. Barahona. Stability of graph communities across time scales. *Proc. of the National Academy of Sciences of the USA*, 107(29) :12755–12760, 2010.

I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, San Francisco, 2001. ACM.

L. R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3) :297–302, July 1945.

R. Diestel. *Graph Theory*. Graduate texts in mathematics. Springer, second edition, 2000.

Y. Djouadi, D. Dubois, et H. Prade. Graduality, uncertainty and typicality in formal concept analysis. In C. Cornelis, G. Deschrijver, M. Nachtegael, S. Schockaert, et Y. Shi, editors, *35 years of Fuzzy Set Theory*, pages 127–147. Springer, 2010a.

Y. Djouadi, D. Dubois, et H. Prade. Possibility theory and formal concept analysis : Context decomposition and uncertainty handling. In E. Hüllermeier, R. Kruse, et F. Hoffmann, editors, *Computational Intelligence for Knowledge-Based Systems Design, Proc. 13th Inter. Conf. on Information Processing and Management of Uncertainty (IPMU 2010), Dortmund, June 28 - July 2*, volume 6178 of *LNAI*, pages 260–269. Springer, 2010b.

D. Dubois, F. Dupin de Saint-Cyr, et H. Prade. A possibility theoretic view of formal concept analysis. *Fundamenta Informaticae*, 75(1) :195–213, 2007.

D. Dubois et H. Prade. Possibility theory and formal concept analysis in information systems. In *Proc. 13th International Fuzzy Systems Association World Congress IFSA-EUSFLAT*, Lisbon, July 2009.

D. Dubois et H. Prade. Possibility theory and formal concept analysis : Characterizing independent sub-contexts. *Fuzzy Sets and Systems*, 196 :4–16, 2012.

P. Erdős et A. Rényi. On random graphs I. *Publicationes Mathematicae (Debrecen)*, 6 :290–297, 1959.

P. Erdős et A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5 :17–61, 1960.

K. Eren, M. Deveci, O. Küçüktunç, et Ü. V. Çatalyürek. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in Bioinformatics*, 2012.

L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8 :128–140, 1741.

J.L. Faulon. Isomorphism, automorphism partitioning, and canonical labeling can be solved in polynomial-time for molecular graphs. *Journal of Chemical Information and Computer Sciences*, 38(3) :432–444, 1998.

C. Fellbaum, editor. *WordNet : An Electronic Lexical Database*. MIT Press, 1998.

P. Ferragina et A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 801–810, Chiba, Japan, 2005. ACM.

A. Formica. Concept similarity in formal concept analysis : An information content approach. *Knowl.-Based Syst.*, 21(1) :80–87, 2008.

F. Fouss, K. Francoisse, L. Yen, A. Pirotte, et M. Saerens. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural Networks*, 31(0) :53 – 72, 2012.

F. Fouss, A. Pirotte, J.M. Renders, et M. Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3) :355–369, March 2007.

F. Fouss, L. Yen, A. Pirotte, et M. Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 863–868, 2006.

L. C. Freeman. Some Antecedents of Social Network Analysis. *Connections*, 19(2) : 39–42, 1996.

H. Fu et E. Mephu Nguifo. A parallel algorithm to generate formal concepts for large data. In Peter W. Eklund, editor, *Concept Lattices, Prooc. 2nd Int. Conf. on Formal Concept Analysis (ICFCA 2004) Sydney*, volume 2961 of *LNCS*, pages 394–401. Springer, 2004.

B. Gaillard, B. Gaume, et E. Navarro. Invariants and variability of synonymy networks : Self mediated agreement by confluence. In *Proc. of TextGraphs-6 : Graph-based Methods for NLP*, pages 15–23, Portland, 2011. ACL.

G. Gan, C. Ma, et J. Wu. *Data clustering - theory, algorithms, and applications*. SIAM, 2007.

B. Ganter et R. Wille. *Formal Concept Analysis*. Springer-Verlag, 1999.

Xinbo Gao, Bing Xiao, Dacheng Tao, et Xuelong Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1) :113–129, 2010.

B. Gaume, E. Navarro, et H. Prade. A parallel between extended formal concept analysis and bipartite graphs analysis. In Eyke Hüllermeier, Rudolf Kruse, et Frank Hoffmann, editors, *Computational Intelligence for Knowledge-Based Systems Design*, volume 6178 of *Lecture Notes in Computer Science*, pages 270–280. Springer Berlin / Heidelberg, June 2010.

B. Gaume, E. Navarro, et H. Prade. Clustering bipartite graphs in terms of approximate formal concepts and sub-contexts. *International Journal of Computational Intelligence Systems*, 6(6) :1125–1142, 2013.

J Giles. Internet Encyclopaedias go Head to Head. *Nature*, 438 :900–901, 2005.

R. Gupta, G. Fang, B. Field, M. Steinbach, et V. Kumar. Quantitative evaluation of approximate frequent pattern mining algorithms. In *Proc. of the 14th ACM SIGKDD Inter. Conf. on Knowledge discovery and data mining*, KDD '08, pages 301–309, New York, 2008.

T. Gärtner, P. Flach, et S. Wrobel. On graph kernels : Hardness results and efficient alternatives. In B. Schölkopf et M. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *LNCS*, pages 129–143. Springer Berlin Heidelberg, 2003.

V. Hanoka et B. Sagot. Wordnet extension made simple : A multilingual lexicon-based approach using wiki resources. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, 2012. European Language Resources Association (ELRA).

F. Harary. *Graph theory and theoretical physics*. Academic P., 1967.

D. K. Harman. The trec test collections. In Ellen M. Voorhees et Donna K. Harman, editors, *TREC : Experiment and Evaluation in Information Retrieval*, chapter 2, pages 21–53. 2005.

M. Hasan et M. J. Zaki. A survey of link prediction in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 243–275. Springer US, 2011.

H. He et A. K. Singh. Closure-tree : An index structure for graph queries. In *Proc. of the 22th IEEE Int. Conf. on Data Engineering (ICDE)*, april 2006.

M. A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, July 1992.

K. Heylen, Y. Peirsman, D. Geeraerts, et D. Speelman. Modelling Word Similarity : an Evaluation of Automatic Synonymy Extraction Algorithms. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008.

T. Hu, C. Qu, C. Lim T., S. Yuan Sung, et W. Zhou. Preserving patterns in bipartite graph partitioning. In *18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006), Washington, USA*, pages 489–496. IEEE Computer Society, 2006.

T. Hughes et D. Ramage. Lexical Semantic Relatedness with Random Graph Walks. In *Proceedings of EMNLP-CoNLL*, pages 581–589, 2007.

T. Ito, M. Shimbo, T. Kudo, et Y. Matsumoto. Application of kernels to link analysis. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD'05, pages 586–592, New York, USA, 2005. ACM.

C. Jacquin, E. Desmontils, et L. Monceaux. French EuroWordNet Lexical Database Improvements. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing (CICLING)*, pages 12–22, 2007.

N. Jardine et C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Inform. Stor. Retr.*, 7(5) :217–240, 1971.

N. Jay, F. Kohler, et A. Napoli. Analysis of social communities with iceberg and stability-based concept lattices. In R. Medina et S. A. Obiedkov, editors, *Proc. 6th Inter. Conf. on Formal Concept Analysis (ICFCA'08), Montreal*, volume 4933 of *LNCS*, pages 258–272. Springer, 2008.

G. Jeh et J. Widom. Simrank : a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD int. conf. on knowledge discovery and data mining*, KDD'02, pages 538–543. ACM, 2002.

H. Jeong, S. P. Mason, A. L. Barabási, et Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833) :41–42, May 2001.

H. Jiang, H. Wang, P.S. Yu, et S. Zhou. Gstring : A novel approach for efficient search in graph databases. In *Proc. of the 23th IEEE Int. Conf. on Data Engineering (ICDE)*, pages 566–575, 2007.

Andrew B. Kahng, Jens Lienig, Igor L. Markov, et Jin Hu. *VLSI Physical Design - From Graph Partitioning to Timing Closure.* Springer, 2011.

J. Kandola, J. Shawe-taylor, et N. Cristianini. Learning semantic similarity. In *In NIPS 15*, pages 657–664, 2003.

L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1) : 39–43, 1953.

R. Khanin et E. Wit. How scale-free are biological networks. *Journal of Computational Biology*, 13(3) :810–818, 2006.

A. Kilgarriff. Gold Standard Datasets for Evaluating Word Sense Disambiguation Programs. *Computer Speech & Language*, 12(4) :453–472, 1998.

M. Klimushkin, S. A. Obiedkov, et C. Roth. Approaches to the selection of relevant concepts in the case of noisy data. In L. Kwuida et B. Sertkaya, editors, *Proc. 8th Inter. Conf. on Formal Concept Analysis (ICFCA'10), Agadir*, volume 5986 of *LNCS*, pages 255–266. Springer, 2010.

R. Kondor et J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pages 315–322, 2002.

P. Krajca, J. Outrata, et V. Vychodil. Parallel recursive algorithm for fca. In S. O. Kuznetsov R. Belohlavek, editor, *Proc. CLA 2008, CEUR WS*, pages 71–82, 2008.

J. Kunegis et A. Lommatzsch. Learning spectral graph transformations for link prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 561–568. ACM, 2009.

S. O. Kuznetsov, S. A. Obiedkov, et C. Roth. Reducing the representation complexity of lattice-based taxonomies. In U. Priss, S. Polovina, et R. Hill, editors, *Conceptual Structures : Knowledge Architectures for Smart Applications, Proc. 15th Inter. Conf. on Conceptual Structures (ICCS'07), Sheffield*, volume 4604 of *LNCS*, pages 241–254. Springer, 2007.

M. Lafourcade. Making People Play for Lexical Acquisition with the JeuxDeMots prototype. In *SNLP'07 : 7th International Symposium on Natural Language Processing*, Pattaya, Thailand, 12 2007.

A. Lancichinetti et S. Fortunato. Community detection algorithms : A comparative analysis. *Phys. Rev. E*, 80(5) :056117, November 2009.

A. Lancichinetti, S. Fortunato, et F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78 :046110, Oct 2008.

M. Latapy, C. Magnien, et N. Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1) :31–48, 2008.

S. Lehmann, M. Schwartz, et L. K. Hansen. Biclique communities. *Phys. Rev. E*, 78 (1) :016108–9, 2008.

E. A. Leicht, P. Holme, et M. E. J. Newman. Vertex similarity in networks. *Phys. Rev. E*, 73 :026120, Feb 2006.

V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8) :707–710, 1966.

D. Liben-Nowell et J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7) : 1019–1031, 2007. ISSN 1532-2890.

W. Liu et L. Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5) :58007, 2010.

X. Liu et T. Murata. Evaluating community structure in bipartite networks. In A. K. Elmagarmid et D. Agrawal, editors, *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010, Minneapolis, USA*, pages 576–581. IEEE Computer Society, 2010.

L. Lü et T. Zhou. Link prediction in complex networks : A survey. *Physica A : Statistical Mechanics and its Applications*, 390(6) :1150–1170, 2011.

C. D. Manning, P. Raghavan, et H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

C. D. Manning et H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

G. Mecca, S. Raunich, et A. Pappalardo. A new algorithm for clustering search results. *Data & Knowledge Engineering*, 62(3) :504–522, September 2007.

S. Milgram. The small world problem. *Psychology Today*, 2 :60–67, 1967.

R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, et U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663) : 1538–1542, 2004.

M. Mohri, A. Rostamizadeh, et A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

G. C. Murray et R. Green. Lexical Knowledge and Human Disagreement on a WSD Task. *Computer Speech & Language*, 18(3) :209–222, 2004.

P. Naïm, P.H. Wuillemin, P. Leray, O. Pourret, et A. Becker. *Réseaux bayésiens*. Eyrolles, 3 edition, 2011.

E. Navarro, Y. Chudy, B. Gaume, G. Cabanac, et K. Pinel-Sauvagnat. Kodex ou comment organiser les résultats d'une recherche d'information par détection de communautés sur un graphe biparti ? In *CORIA'11, Avignon*, pages 25–40. ARIA, mars 2011.

E. Navarro, B. Gaume, et H. Prade. Comparing and fusing terrain network information. In E. Hüllermeier, S. Link, T. Fober, et B. Seeger, editors, *SUM*, volume 7520 of *LNCS*, pages 459–472. Springer, 2012a.

E. Navarro, H. Prade, et B. Gaume. Clustering sets of objects using concepts-objects bipartite graphs. In Eyke Hüllermeier, Sebastian Link, Thomas Fober, et Bernhard Seeger, editors, *SUM*, volume 7520 of *LNCS*, pages 420–432. Springer, 2012b.

E. Navarro, F. Sajous, B. Gaume, L. Prévot, S. Hsieh, I. Kuo, P. Magistry, et C.R. Huang. Wiktionary and NLP : Improving Synonymy Networks. In *Proceedings of the ACL-IJCNLP Workshop on The People's Web Meets NLP : Collaboratively Constructed Semantic Resources*, pages 19–27, Singapore, August 2009.

M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E*, 67 :026126, Feb 2003a.

M. E. J. Newman. The structure and function of complex networks. *SIAM review*, 45 (2) :167–256, 2003b.

M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74 :036104, Sep 2006.

M. Ohlrich, C. Ebeling, E. Ginting, et L. Sather. Subgemini : Identifying subcircuits using a fast subgraph isomorphism algorithm. In *30th Conference on Design Automation*, pages 31 – 37, 1993.

Y. Ollivier et P. Senellart. Finding Related Pages Using Green Measures : an Illustration with Wikipedia. In *AAAI'07 : Proceedings of the 22nd national conference on Artificial intelligence*, pages 1427–1433. AAAI Press, 2007.

S. Osinski et D. Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3) :48–54, 2005.

I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, et D. Johnson. Terrier Information Retrieval Platform. In *ECIR'05 : Proceedings of the 27th European Conference on IR Research*, volume 3408 of *LNCS*, pages 517–519. Springer, 2005.

G. Palla, I. Derenyi, I. Farkas, et T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043) :814–818, 2005.

M. Palmer, H. T. Dang, et C. Fellbaum. Making Fine-Grained and Coarse-Grained Sense Distinctions, Both Manually and Automatically. *Natural Language Engineering*, 13(2) :137–163, 2007.

P. Pantel et M. Pennacchiotti. Espresso : Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the International Conference on Computational Linguistics*, pages 113–120, Sydney, 17th–21st July 2006. ACL Press.

N. Pasquier, Y. Bastide, R. Taouil, et L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1) :25–46, 1999. ISSN 0306-4379.

P. Pons. *Détection de communautés dans les grands graphes de terrain*. PhD thesis, 2007.

P. Pons et M. Latapy. Computing communities in large networks using random walks (long version). *J. of Graph Algorithms and Applications (JGAA)*, 10(2) :191–218, 2006.

M. F. Porter. Snowball : A language for stemming algorithms, 2001. http://snowball.tartarus.org/texts/introduction.html.

N. Pržulj, D. G. Corneil, et I. Jurisica. Modeling interactome : scale-free or geometric ? *Bioinformatics*, 20(18) :3508–3515, 2004.

E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, et A. L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586) :1551–1555, 2002.

F. Ricci, L. Rokach, B. Shapira, et P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

G. Robins et M. Alexander. Small worlds among interlocking directors : Network structure and distance in bipartite graphs. *Comput. Math. Organ. Theory*, 10(1) : 69–94, May 2004.

A. Robles-Kelly et E. R. Hancock. Graph edit distance from spectral seriation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3) :365–378, 2005.

M. Rosvall et C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4) : 1118–1123, 2008.

B. Sagot et D. Fišer. Building a Free French Wordnet from Multilingual Resources. In *Proceedings of OntoLex 2008*, Marrakech, 2008.

F. Sajous, E. Navarro, et B. Gaume. Enrichissement de lexiques sémantiques approvisionnés par les foules : le système wisigoth appliqué à wiktionary. *TAL*, 52(1) : 11–35, 2011a.

F. Sajous, E. Navarro, B. Gaume, L. Prévot, et Y. Chudy. Semi-automatic endogenous enrichment of collaboratively constructed lexical resources : Piggybacking onto wiktionary. In Hrafn Loftsson, Eiríkur Rögnvaldsson, et Sigrún Helgadóttir, editors, *Advances in Natural Language Processing*, volume 6233 of *LNCS*, pages 332–344. Springer Berlin / Heidelberg, 2010.

F. Sajous, E. Navarro, B. Gaume, L. Prévot, et Y. Chudy. Semi-automatic enrichment of crowdsourced synonymy networks : the wisigoth system applied to wiktionary. *Language Resources and Evaluation*, pages 1–34, 2011b. ISSN 1574-020X. To appear in Language Resources & Evaluation, special issue on Collaboratively Constructed Lexical Resources.

G. Salton, A. Wong, et C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11) :613–620, November 1975a.

G. Salton, A. Wong, et C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11) :613–620, November 1975b.

A. Sanfeliu et K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-13 (3) :353–362, 1983.

G. Saporta. *Probabilités, analyses des données et statistiques*. Editions Technip, 2006.

S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1) :27–64, 2007. ISSN 1574-0137.

S. Sekine. We Desperately Need Linguistic Resources ! –Based on the Users' Point of View. FLaReNet Forum. Barcelona, Feb 2010.

A. Serin et M. Vingron. Debi : Discovering differentially expressed biclusters using a frequent itemset approach. *Algorithms for Molecular Biology*, 6(1) :18, 2011. ISSN 1748-7188.

R. Snow, B. O'Connor, D. Jurafsky, et A. Y. Ng. Cheap and Fast—but is it good ? : Evaluating Non-Expert Annotations for Natural Language Tasks. In *EMNLP '08 : Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Morristown, NJ, USA, 2008. ACL.

T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biol. Skr.*, 5 :1–34, 1948.

C. Soria, M. Monachini, F. Bertagna, N. Calzolari, C. R. Huang, S. K. Hsieh, A. Marchetti, et M. Tesconi. Exploring Interoperability of Language Resources : the Case of Cross-Lingual Semi-Automatic Enrichment of Wordnets. *Language Resources and Evaluation*, 40(1) :87–96, 2009.

P. N. Tan, M. Steinbach, et V. Kumar. *Introduction to Data Mining*. Addison-Wesley, Boston, 2005.

A. Tombros, R. Villa, et C. J. van Rijsbergen. The effectiveness of query-specific hierarchic clustering in information retrieval. *Inf. Process. Manage.*, 38(4) :559–582, 2002.

N. Trinajstić. *Chemical graph theory*. CRC Press, 1983.

D. Tufis. Balkanet Design and Development of a Multilingual Balkan Wordnet. *Romanian Journal of Information Science and Technology*, 7(1–2), 2000.

R. van der Hofstad. Random graphs and complex networks. University Lecture, 2013.

L. Van der Plas et G. Bouma. Syntactic Contexts for Finding Semantically Related Words. In Ton van der Wouden, Michaela Poß, Hilke Reckman, et Crit Cremers, editors, *Computational Linguistics in the Netherlands 2004 : Selected papers from the fifteenth CLIN meeting*, volume 4 of *LOT Occasional Series*. Utrecht University, 2005.

U. von Luxburg, A. Radl, et M. Hein. Getting lost in space : Large sample analysis of the resistance distance. In *24th Conference on Neural Information Processing Systems (NIPS'2010), Vancouver*, pages 2622–2630, 2010.

H. Voormann et U. Gut. Agile Corpus Creation. *Corpus Linguistics and Lingustic Theory*, 4(2) :235–251, 2008.

P. Vossen, editor. *EuroWordNet : a Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

S. Wasserman et K. Faust. *Social Network Analysis : Methods and Applications*. Cambridge University Press, 1994.

D. Watts et S. Strogatz. Collective dynamics of'small-world' networks. *Nature*, 393 : 440–442, 1998.

T. Weale, C. Brew, et E. Fosler-Lussier. Using the Wiktionary Graph Structure for Synonym Detection. In *Proceedings of the ACL-IJCNLP Workshop on The People's Web Meets NLP : Collaboratively Constructed Semantic Resources*, pages 28–31, Singapore, August 2009.

R. Wille. Restructuring lattice theory : An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, volume 83 of *NATO Advanced Study Institutes Series*, pages 445–470. Springer Netherlands, 1982.

C. Yang, U. Fayyad, et P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proc. of the seventh Inter. Conf. on Knowledge Discovery and Data mining*, pages 194–203, 2001.

O. Zamir et O. Etzioni. Web document clustering : a feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54, Melbourne, Australia, 1998. ACM.

O. Zamir et O. Etzioni. Grouper : a dynamic clustering interface to web search results. In *Proceedings of the eighth international conference on World Wide Web*, pages 1361–1374, Toronto, Canada, 1999. Elsevier North-Holland, Inc.

T. Zesch. What's the Difference ? –Comparing Expert-Built and Collaboratively-Built Lexical Semantic Resources. FLaReNet Forum. Barcelona, Feb 2010.

T. Zesch et I. Gurevych. Wisdom of Crowds versus Wisdom of Linguists - Measuring the Semantic Relatedness of Words. *Journal of Natural Language Engineering.*, 16 (01) :25–59, Jan 2010.

D. Zhang et Y. Dong. Semantic, hierarchical, online clustering of web search results. In *Advanced Web Technologies and Applications*, pages 69–78. 2004.

T. Zhou, L. Lü, et Y.-C. Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4) :623–630, 2009.

Table des figures

Liste des tableaux

Index

Résumé

Cette thèse s'organise en deux parties : une première partie s'intéresse aux mesures de similarité (ou de proximité) définies entre les sommets d'un graphe, une seconde aux méthodes de clustering de graphe biparti.

Une nouvelle mesure de similarité entre sommets basée sur des marches aléatoires en temps courts est introduite. Cette méthode a l'avantage, en particulier, d'être insensible à la densité du graphe. Il est ensuite proposé un large état de l'art des similarités entre sommets, ainsi qu'une comparaison expérimentale de ces différentes mesures. Cette première partie se poursuit par la proposition d'une méthode robuste de comparaison de graphes partageant le même ensemble de sommets. Cette méthode est mise en application pour comparer et fusionner des graphes de synonymie. Enfin une application d'aide à la construction de ressources lexicales est présentée. Elle consiste à proposer de nouvelles relations de synonymie à partir de l'ensemble des relations de synonymie déjà existantes.

Dans une seconde partie, un parallèle entre l'analyse formelle de concepts et le clustering de graphe biparti est établi. Ce parallèle conduit à l'étude d'un cas particulier pour lequel une partition d'un des groupes de sommets d'un graphe biparti peut-être déterminée alors qu'il n'existe pas de partitionnement correspondant sur l'autre type de sommets. Une méthode simple qui répond à ce problème est proposée et évaluée. Enfin Kodex, un système de classification automatique des résultats d'une recherche d'information est présenté. Ce système est une application en RI des méthodes de clustering vues précédemment. Une évaluation sur une collection de deux millions de pages web montre les avantages de l'approche et permet en outre de mieux comprendre certaines différences entre méthodes de clustering.

Mots clés

graphes de terrain - similarité - comparaison de graphes - marche aléatoire - clustering - analyse formelle de concepts - ressources lexicales - recherche d'information

Abstract

This thesis is organized in two parts : the first part focuses on measures of similarity (or proximity) between vertices of a graph, the second part on clustering methods for bipartite graph.

A new measure of similarity between vertices, based on short time random walks, is introduced. The main advantage of the method is that it is insensitive to the density of the graph. A broad state of the art of similarities between vertices is then proposed, as well as experimental comparisons of these measures. This is followed by the proposal of a robust method for comparing graphs sharing the same set of vertices. This measure is shown to be applicable to the comparison and merging of synonymy networks. Finally an application for the enrichment of lexical resources is presented. It consists in providing candidate synonyms on the basis of already existing links.

In the second part, a parallel between formal concept analysis and clustering of bipartite graph is established. This parallel leads to the particular case where a partition of one of the vertex groups can be determined whereas there is no corresponding partition on the other group of vertices. A simple method that addresses this problem is proposed and evaluated. Finally, a system of automatic classification of search results (Kodex) is presented. This system is an application of previously seen clustering methods. An evaluation on a collection of two million web pages shows the benefits of the approach and also helps to understand some differences between clustering methods.

Keywords

complex networks - terrain networks - similarity - graph comparison - random walks - clustering - formal concept analysis - lexical resources - information retrieval