Graphes linguistiques multiniveau pour l'extraction de connaissances : l'exemple des collocations

Vincent Archer

THÈSE
présentée par

Vincent ARCHER

Pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER – GRENOBLE 1
(ARRÊTÉS MINISTÉRIELS DU 5 JUILLET 1984 ET DU 30 MARS 1992)

Discipline
INFORMATIQUE

École doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique

Graphes linguistiques multiniveau pour l'extraction de connaissances : l'exemple des collocations

24 septembre 2009

Jury :

Président :	Catherine Berrut
Rapporteurs :	Yves Lepage
	Denis Maurel
	Eric Wehrli
Examinateur :	Bruno Gaume
Directeur :	Christian Boitet
Co-directeur :	Gilles Sérasset

Thèse préparée au sein de l'équipe GETALP du Laboratoire d'Informatique de Grenoble

Remerciements

Je tiens à remercier toutes les personnes grâce à qui j'ai pu mener à bien cette thèse.

Je suis, en premier lieu, reconnaissant à ceux qui l'ont dirigée : Christian Boitet et Gilles Sérasset.

Gilles a encadré mes recherches depuis mon premier stage dans l'équipe, en 2003. Je lui exprime ma pleine gratitude (tiens, une collocation !) pour toutes ses remarques et tous ses conseils, pour sa disponibilité, et en particulier pour m'avoir aidé à (re)prendre de la confiance en moi et du recul quand cela a été nécessaire.

Merci à Christian pour tous ses commentaires, ses corrections sur les différentes versions de la thèse, et pour sa rigueur et son exigence, qui ont permis de chasser de ce document nombre d'ambiguïtés et d'imprécisions. Je lui sais également gré pour ses relances répétées à propos d'un stage à Tokyo au printemps 2007, qui finirent par vaincre les réticences du timide que je suis.

Je tiens à remercier les rapporteurs de cette thèse, messieurs Yves Lepage, Denis Maurel, et Eric Wehrli, dont les remarques m'ont permis d'améliorer le contenu de ce document. Je remercie également, pour avoir accepté de participer au jury, madame Catherine Berrut, présidente, et monsieur Bruno Gaume, examinateur.

Je suis reconnaissant à tous les membres de *GETALP*, doctorants et permanents. Je remercie aussi les autres gens que j'ai pu côtoyer au *LIG* (…et au *CLIPS*), en particulier Jean-Pierre Chevallet qui m'a accueilli pour mon tout premier stage, en première année de Magistère, fin 2002.

Je veux également exprimer ma gratitude à Akiko Aïzawa d'avoir encadré mon stage au *National Institute of Infomatics* de Tokyo, ainsi qu'à ceux que j'ai croisé là-bas. S'il m'a été utile au niveau de la recherche, ce stage m'a avant tout beaucoup apporté sur un plan personnel, me prouvant que j'étais capable d'évoluer tranquillement trois mois dans un pays dont je ne connaissais ni la langue ni même l'alphabet.

Merci aussi à tous ceux que j'ai côtoyés depuis que je suis ATER sur la Côte d'Azur, à l'IUT à Cannes et à Nice, comme au laboratoire I3S au Sophia Antipolis, et particulièrement à l'équipe *Ressources Linguistiques* de Jacques Farré.

Merci à l'école publique française, que j'ai suivie de la maternelle au doctorat.

Je tiens enfin à exprimer ma grande reconnaissance à mes proches pour leur soutien.

À ma famille, mes parents et mes grands-parents, qui ont toujours cru en moi, même dans les moments plus difficiles. À mon cousin Guillaume, pour nos grandes balades à vélo, nos longues discussions, et ses qualités d'écoute. À mes amis (Philippe, François, Yvan, Damien, Sylvain, les deux Julien, Anthony), pour tous les moments salutaires de détente et tous les sourires que je leur dois. Aux copines de msn (Sophia, Élise). À Valérie.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Le développement de l'informatique a rapidement mené à la volonté d'utiliser la puissance de calcul des ordinateurs pour reproduire des tâches jusqu'alors réalisées manuellement, en espérant obtenir, pour certains problèmes, des solutions plus rapides et moins coûteuses que le travail humain. L'un des premiers domaines concernés par cette idée fut le traitement des langues, et plus particulièrement la traduction.

Malgré plusieurs décennies de recherche, ce problème est encore loin d'être résolu : les logiciels dédiés à des tâches linguistiques (en particulier la traduction) donnent toujours des résultats peu satisfaisants pour une utilisation générale, sans contrainte sur les données à traiter : la traduction automatique d'un article de l'anglais vers le français décevra souvent l'utilisateur. Ce sont pourtant des langues assez proches (notamment dans la syntaxe des phrases), pour lesquelles la traduction devrait être plus facile qu'avec d'autres couples de langues.

Pour traduire correctement une phrase de l'anglais vers le français, l'idéal serait d'analyser la phrase anglaise pour en connaître le sens, puis générer une phrase française qui exprime correctement ce sens. Mais le traitement du sens est une tâche compliquée : l'analyse sémantique n'est actuellement pas satisfaisante. Les systèmes de traduction cherchent donc à contourner le problème. Parce qu'on n'est pas capable de modéliser efficacement tous les mécanismes constituant la traduction, on doit effectuer des simplifications dans la réalisation d'un processus automatique effectuant cette tâche. Cela provoque des imprécisions, qui expliquent les importants taux d'erreur constatés.

En particulier, les systèmes de traduction manquent, pour la plupart, de ressources linguistiques : s'ils peuvent donner de bons résultats dans des « sous-langages », quand ils traitent un vocabulaire limité (comme le système METEO qui traduit les bulletins météorologiques au Canada), ils fléchissent dès qu'ils doivent travailler avec un langage ouvert, avec des ambiguïtés, des mots et des expressions dont le sens dépend du contexte dans lequel ils sont employés.

Un cas d'expression particulière est celui des *locutions*. Il s'agit d'expressions comme « fruit de mer » ou « pomme de terre ») dont le sens ne peut être déduit de ses composants. Les systèmes traduisent généralement bien les locutions entre le français et l'anglais parce qu'il s'agit d'expressions ayant un sens propre et qu'on peut lister.

Les *collocations* sont une autre sorte d'expression particulière, où l'un des termes exprime un sens en fonction d'un autre. Par exemple, dans « peur bleue », « bleue » n'a pas son sens habituel (couleur) mais a été choisi pour exprimer l'intensification de « peur ». Ce choix diffère selon le mot sur lequel on veut appliquer la modification. Ainsi, on peut également exprimer l'intensification en disant qu'une « pluie » est « battante », que des « applaudissements » sont « nourris », etc.

Les systèmes de traduction font beaucoup plus d'erreurs sur les collocations que sur les locutions. Cela vient des ressources linguistiques disponibles : les dictionnaires utilisés par les systèmes à règles listent bien plus fréquemment les locutions que les collocations ; les systèmes statistiques peinent également à repérer les fragments de phrase qui pourraient permettre une traduction correcte des collocations.

Introduction

Le problème des collocations est en effet plus complexe à traiter dans sa globalité, parce que celles-ci sont bien plus nombreuses que les locutions, et qu'elles nécessitent une meilleure connaissance de la langue pour les traiter. Il faudrait que les dictionnaires bilingues puissent donner une traduction de toutes les collocations. Mieux, il suffirait que les dictionnaires monolingues puissent associer aux collocations les modifications de sens correspondantes : si on sait qu'un fumeur intense est « gros » en français, mais lourd en anglais (« heavy smoker ») ou fort en allemand (« starker Raucher »), il n'est plus nécessaire de connaître les traductions directement entre les collocations[1].

Ce manque de ressources s'explique avant tout par la complexité de leur création. Deux objectifs s'opposent : la rapidité de production des informations, et leur précision. En effet, il faut absolument que les ressources linguistiques produites soient précises, correctes, et complètes. Cela nécessite le travail de spécialistes de la langue, et c'est une tâche qui prend beaucoup de temps. Il est également possible de créer des ressources linguistiques de manière automatique (de moins bonne qualité), en extrayant les occurrences de phénomènes à traiter, et en se basant sur des patrons ou sur la statistique. Certains systèmes de traduction (comme *Google Translate*) reposent sur des correspondances bilingues produites de manière statistique, à partir de corpus bilingues parallèles (contenant des documents traductions l'un de l'autre), tentant de combler le manque de précision en étant plus complet.

La meilleure approche à suivre pour obtenir des ressources de qualité et en grand nombre, est de concilier au mieux les travaux automatiques et manuels, en proposant des outils automatiques facilitant la tâche des linguistes, voire en leur faisant guider les processus automatiques.

En pratique, malgré les bonnes volontés, il reste difficile d'effectuer une telle production « hybride ». L'une des principales raisons en est la non-adaptation des outils au public visé. Le traitement automatique des langues, et en particulier l'extraction de connaissances linguistiques, nécessite encore, avec la plupart des outils actuels, une certaine connaissance en informatique que la plupart des linguistes n'ont pas. Il faut donc réaliser des outils informatiques pour le traitement les langues qui soient les plus accessibles possibles, limitant le plus possible les connaissances informatiques nécessaires.

Ces outils doivent donc être génériques, et s'adapter au contenu et à la tâche plutôt que demander l'effort inverse à l'utilisateur. Pour cela, ils doivent reposer sur un modèle clair de description des données qui puisse, en lui-même, permettre de représenter *simplement* des données de formes diverses (issues de corpus, de dictionnaires), etc., y compris celles auxquelles on ne pense pas nécessairement lors de la spécification : le modèle doit être adaptable (de manière à ce qu'on puisse réaliser la modélisation des données simplement et rapidement). Le même souci doit être considéré en ce qui concerne le processus : un bon outil doit fournir des fonctionnalités efficaces et simples, mais aussi, afin de ne pas être trop contraignant, permettre d'en introduire facilement de nouvelles : dans le même esprit que précédemment, l'outil doit être utilisable pour d'autres buts que ceux pour lesquels il a été conçu.

[1]Cette idée a été développée dans la Théorie Sens-Texte de Mel'čuk à Moscou dans les années 1960 (Žolkovskij & Mel'čuk, 1967)

La première partie de cette thèse étudie en détail le problème de la collecte des collocations.

Tout d'abord, nous présentons dans le premier chapitre ce qu'est précisément ce phénomène linguistique, les conséquences qu'il a sur le traitement automatique des langues, et une première spécification du problème de collecte à résoudre.

Dans le deuxième chapitre, nous étudions les principaux modèles de lexiques existants, et nous intéressons plus particulièrement à leur capacité éventuelle à modéliser des collocations.

Le troisième chapitre est un état de l'art des méthodes déjà proposées et expérimentées pour la collecte de collocations (et plus généralement des « expressions multi-mots »), utilisant l'extraction, l'apprentissage automatique, ou la contribution.

La seconde partie s'intéresse au problème de l'extraction de connaissances linguistiques en général, et aux solutions que nous y apportons.

Nous présentons, dans le quatrième chapitre, des expérimentations monolingues et bilingues d'extraction de collocations, et en analysons les besoins.

Le cinquième chapitre répond aux besoins observés, en proposant de considérer l'extraction d'information comme une manipulation de graphes (multiniveau), et en introduisant le « cahier des charges » que devra respecter un modèle basé sur cette idée.

Dans le sixième chapitre, nous présentons *MuLLinG*, notre modèle générique de graphes linguistiques multiniveau, ainsi que tous les opérateurs de base relatifs à ce modèle, qui permettront de décrire une extraction de connaissances linguistiques comme une succession d'opérations simples de manipulation du graphe représentant la ressource.

La troisième et dernière partie concerne les applications du modèle.

Le septième chapitre présente la mise en œuvre du modèle sous la forme d'une bibliothèque, écrite en C++, et stockant les graphes dans le format GraphML.

Nous nous intéressons ensuite, dans le huitième chapitre, à la réalisation d'expérimentations monolingues et bilingues d'extraction de collocations, afin de montrer que les processus de collecte concernés peuvent être facilement décrits dans notre modèle.

Enfin, nous présentons, dans le neuvième chapitre, des expérimentations utilisant notre modèle pour pondérer des relations de traductions de référence : cela a pour but de montrer que notre modèle peut être utilisé pour d'autres tâches, notamment en combinant plusieurs sources d'information.

Partie I - Le problème de la collecte de collocations

Chapitre I – Motivation : la collecte de collocations

Nous nous concentrons tout d'abord, pour étudier le problème des ressources linguistiques, sur un phénomène particulier, celui des collocations (des expressions composées de deux termes dont l'un a été choisi pour exprimer un sens qui dépend de l'autre terme). Dans ce chapitre, nous essayerons tout d'abord de situer ce phénomène dans le large domaine de la linguistique. Ensuite, nous définirons plus précisément la collocation, ses différents types, ses propriétés, puis ses conséquences sur la traduction de textes. Enfin, après nous être intéressé à la manière dont une collocation doit être décrite, nous donnerons une première spécification du problème que nous souhaitons résoudre, à savoir la collecte de collocations[2] à partir de corpus, en utilisant des outils informatiques basés sur des processus automatiques efficaces.

1 Introduction aux problèmes linguistiques

Les travaux présentés dans cette thèse se situent dans le domaine du traitement automatique des langues : il s'agit de résoudre des problèmes linguistiques à l'aide d'outils informatiques. Cette section, à destination des non-initiés, présente les notions de base nécessaires à la compréhension des considérations linguistiques à la base des travaux que nous présentons.

1.1 Les champs d'étude de la linguistique théorique

La *linguistique théorique* a pour but la modélisation des phénomènes universels liés au langage. On retrouve dans la plupart des théories cinq principaux champs d'étude : la phonologie, la morphologie, la syntaxe, la sémantique, et la pragmatique.

1.1.1 La phonologie

La phonologie étudie le système sonore associé à une langue.

Les chaînes parlées peuvent être découpées en *phonèmes* (la plus petite unité distinctive isolable). Par exemple, « arbre » a pour transcription phonologique `/arbr/`, c'est à dire qu'il s'agit de la suite des phonèmes `/a/`, `/r/`, `/b/` et `/r/`. Un phonème n'est pas forcément associé à un seul « phonon » : en français, qu'on roule ou non les r, cela correspond au même phonème `/r/` parce que le mot garde le même sens dans les deux cas (les différentes prononciations d'un même phonème sont appelées *allophones*), alors qu'il y a deux phonèmes en arabe.

[2]La collecte de collocations peut se faire de différentes manières : extraction à partir de corpus, apprentissage automatique, contribution humaine.

Le problème de la collecte de collocations

La correspondance entre sons et phonèmes est relative à une langue : par exemple, les sons `[s]` et `[z]` sont associés à des phonèmes différents en français ou en anglais, alors qu'ils sont allophones en espagnol.

Il est à noter que Miller (2000) affirme que les langues des signes ont des phénomènes aux propriétés « comparables à celle de la phonologie des langues orales ».

1.1.2 La morphologie

La morphologie est l'étude de la structure des mots.

Les *morphèmes* sont les unités porteuses de sens élémentaires d'un énoncé. Le morphème qui contient le sens principal d'un mot est le *radical*. Il permet de former des mots lorsqu'il est combiné avec d'autres morphèmes. Un morphème peut avoir plusieurs réalisations différentes, des *allomorphes*, comme « -s » et « -x » pour le pluriel, ou « all- », « ir- » et « v- » pour le verbe « aller » (supplétivisme).

Par exemple, le mot « incassables » peut être décomposé en 4 morphèmes : le radical « cass », composé avec « in- », « -able » et le morphème « pluriel ».

Il y a plusieurs façons possibles de combiner des morphèmes :

- la *flexion*, qui permet d'exprimer des traits grammaticaux à travers l'ajout de morphèmes, tels que la déclinaison des noms, adjectifs et pronoms (genre, nombre, cas) ou la conjugaison des verbes (personne, nombre, genre, temps, voix, mode, etc.), cela ne modifiant pas le sens de la racine du mot ; dans notre exemple, « -s » sert à exprimer le nombre pluriel.
- la *dérivation*, qui permet de modifier le sens ou la catégorie grammaticale du radical ; dans notre exemple, le suffixe « -able » sert à dériver un adjectif de potentialité passive (« cassable ») à partir du radical verbal « cass », alors que le préfixe « in- » permet d'obtenir un mot dont le sens est opposé.
- la *composition*, qui crée de nouveaux mots à partir de plusieurs radicaux :
    - dans certains cas, cela donne un nouveau mot (dont les composants sont concaténés, séparés par une apostrophe ou un trait d'union, etc.), par exemple « portemanteau », basé sur la composition du radical « porte » et du radical « manteau »).
    - dans d'autres cas, on obtient des expressions linguistiques complexes dont le sens est différent de la composition des sens de leurs composants (« pomme de terre », « eau de vie », etc.).

L'ensemble des formes fléchies d'un même terme (avec éventuellement différents radicaux allomorphes) forme un *lexème*. Par exemple, le lexème « GENTIL » regroupe les mots « gentil », « gentille », « gentils », et « gentilles ». On appelle *lemme* la forme choisie pour représenter un lexème (en français : le masculin singulier pour les adjectifs et les noms, et l'infinitif pour les verbes).

On appelle *locution* le regroupement d'expressions linguistiques complexes qui ne diffèrent que par la flexion. Grossman & Tutin (2003) distinguent les locutions *imagées* (celles dont on peut deviner quelle métaphore a pu conduire à leur formation, comme « pomme de terre ») et *opaques* (celles dont le mécanisme ne peut être facilement résolu : quel rapport entre un bon cuisinier et un « cordon » qui serait « bleu » ?).

1.1.3 La syntaxe

La *syntaxe* étudie la manière dont les mots se combinent, et cherche à expliciter les règles qui encadrent cette combinaison ; elle correspond à la *grammaire* d'une langue.

On parle de *syntagme* (en anglais *phrase*) ou de *constituant* pour désigner une unité syntaxique de la phrase (plus grande que le mot, plus petite que la phrase); les propriétés syntaxiques d'un syntagme (comment il peut se combiner) sont dérivées de sa tête (aussi appelée noyau). On dit qu'un syntagme est *nominal* si sa tête est un nom (« fruit mûr »), *verbal* pour un verbe (« manger un gâteau »), etc.

Les syntagmes peuvent être plus ou moins complexes, emboîtés les uns dans les autres. Par exemple, « j'ai acheté une nouvelle voiture » est une phrase (qui se réduit à une proposition), mais aussi un syntagme verbal (dont le noyau est « ai acheté »), qui contient un syntagme nominal, « nouvelle voiture » (dont le noyau est « voiture »).

1.1.4 La sémantique

La *sémantique* est l'étude des sens, et notamment de la manière dont ils peuvent se combiner pour former le sens global d'une phrase, à travers les formes qui les expriment.

L'unité sémantique élémentaire est la *lexie*, auquel on peut associer une entrée de dictionnaire. Plusieurs lexies (de sens distincts) peuvent correspondre au même mot[3] ; on dit alors que ce mot est *polysémique*.

Le *lexique* d'une langue est l'ensemble des lexies employées dans celui-ci.

Lorsque les lexies renvoient à des concepts qui ont au moins un argument, on appelle les appelle *prédicats sémantiques* (par exemple, « les bras de X » ou « X vend Y à Z pour W »). Si les concepts auxquels renvoient les lexie n'ont aucun argument (comme « balle », « tomate »), ce sont des *objets sémantiques*. Les arguments d'un prédicat peuvent être des objets sémantiques ou d'autres prédicats.

L'analyse sémantique d'une phrase consiste donc en la reconnaissance d'une hiérarchie de prédicats (et d'arguments) sémantiques. C'est la plus ardue des tâches présentées jusqu'ici : on y manipule des concepts aux frontières floues, difficilement dénombrables (même si certains essayent de modéliser les connaissances existantes dans les ontologies), ce qui rend compliquée l'automatisation de la tâche.

1.1.5 La pragmatique

La *pragmatique* est l'étude de la manière dont la langue est utilisée pour la production des énoncés (Morris, 1938) : l'analyse sémantique d'une phrase n'est en effet pas toujours suffisante, il est parfois nécessaire de connaître le contexte pour choisir correctement la bonne interprétation de la phrase.

La pragmatique linguistique contient notamment la théorie des actes de langages (Austin, 1962) ; celle-ci distingue les actes :

- *locutoires*, accomplis quand on dit quelque chose ;

[3]Par exemple, pour le nom polysémique « adresse », la lexie correspondant au sens « indication du domicile » est distincte de celle signifiant « habileté » ; elles constituent d'ailleurs deux entrées différentes dans les dictionnaires.

- *illocutoires*, accomplis à cause de ce qu'on dit, en le disant (par exemple « promettre ») ;
- *perlocutoires* , accomplis par le fait de dire quelque chose (par exemple « convaincre ») ;

L'analyse de la pragmatique d'un énoncé consiste donc à trouver les actes réalisés par celui-ci. Cette tâche est assez difficile, les énoncés étant souvent basés sur des présuppositions (non explicitées).

1.2 Une première définition des collocations

Avant de décrire précisément les collocations dans la suite, nous en donnons d'abord une première définition informelle.

La collocation est une expression composée (généralement) de deux termes, dont l'un a été choisi pour exprimer un sens qui dépend de l'autre terme, comme par exemple « pluie battante » (où « battante » exprime l'intensification) ou « conseil précieux » (où « précieux » indique l'appréciation positive).

Plus rarement, la collocation peut être constituée de plus de deux mots, comme par exemple « pleuvoir comme vache qui pisse » qui permet d'exprimer l'intensité de « pleuvoir ».

1.3 Où s'inscrivent les collocations ?

1.3.1 Un problème lié à la syntaxe et à la sémantique

Telles que nous les avons présentées, les collocations concernent aux moins deux mots juxtaposés dans la phrase. Il s'agit donc de syntagmes particuliers, qu'on peut découvrir lors d'une analyse *syntaxique*.

Une collocation est caractérisée par la modification du sens (éventuellement nulle) qui lui est associée : on ne peut donc distinguer les collocations des autres syntagmes sans résoudre auparavant un problème sémantique (c'est la principale embûche lorsqu'on veut réaliser un programme informatique capable de reconnaître les collocations).

Le sens d'une collocation peut d'ailleurs être vu comme un prédicat dont l'unique argument serait le terme qui est employé librement (l'autre terme étant choisi en fonction de celui-ci pour exprimer un sens particulier).

1.3.2 Lexicologie et lexicographie

Les collocations peuvent également être reliées à d'autres domaines d'étude, plus ou moins liés aux précédents.

La *lexicologie* est l'étude du lexique d'une langue, et notamment des liens entre les unités qui le forment, qui peuvent, d'après Saussure (1916), être de deux types :[4]

[4]La théorie de la *grammaire de construction radicale* (Croft, 2001) rejette le caractère universel des relations syntaxiques, affirmant qu'elles sont spécifiques à des *constructions* (couples forme-sens dont un aspect n'est pas prédictible).

- les rapports *associatifs*, ou *paradigmatiques* (Hjelmslev, 1971) correspondent aux liens sémantiques entre lexies, comme la synonymie, l'antonymie, l'hyponymie (est une sorte de), la méronymie (en inclus dans), etc.
- les rapports *syntagmatiques* sont, comme leur nom l'indique, les relations combinatoires de lexies qui forment des syntagmes.

Les collocations sont donc des relations syntagmatiques[5], avec une combinatoire particulière.

La *lexicographie* ne doit pas être confondue avec la lexicologie ; il s'agit de la science qui s'intéresse à la réalisation de dictionnaires. Elle est également concernée par les collocations puisque celles-ci (et plus généralement les liens syntagmatiques) font partie des informations qu'on souhaite trouver dans un dictionnaire, le choix des lexies utilisées pour les composer étant figé par la langue et non prédictible par une règle.

2 Le phénomène collocatif

Après avoir rappelé les notions linguistiques de base, nous pouvons décrire plus précisément ce que sont les collocations, et ce qui permet de les distinguer des autres groupes de mots.

2.1 Définition

Le sens donné à « collocation » varie beaucoup dans la littérature scientifique. Il est donc essentiel, avant de rentrer dans les détails de nos travaux, de donner la définition (qui se place dans une vision plus restrictive de la collocation) à laquelle ceux-ci se rapportent.

2.1.1 Une co-occurrence particulière

Sinclair et al. (1970) définissent la *collocation* comme :

> « the occurrence of two items in a context within a specified environment. »
>
> (« l'occurrence de deux unités dans un contexte, à l'intérieur d'un environnement spécifié. »)

Cela correspond à ce qu'on appelle habituellement en français une *co-occurrence*. Mais Sinclair introduit également la notion de *collocation significative* :

> « *Significant collocation* is regular collocation between two items, such as they co-occur more often than their respective frequencies »
>
> (« Une *collocation significative* est une collocation habituelle entre deux unités, telle qu'elles apparaissent ensemble plus souvent que leurs fréquences respectives »).

Cette description, qui réduit déjà le champ d'exploration, reste néanmoins vague ; en particulier, elle ne fixe aucune limite à la composition du sens au sein de ces co-occurrences. Elle n'est donc pas suffisante pour soutenir les travaux de cette thèse.

[5]Une collocation se présente toujours sous la forme d'un syntagme.

2.1.2 Expression semi-figée

On peut envisager de classer les expressions selon la manière dont se combinent les mots qui les composent :

- les expressions *figées* (les *locutions*), dont le sens n'est pas du tout compositionnel, comme « fruit de mer », ou « casser sa pipe » ;
- les expressions *libres*, où le sens est compositionnel, les termes étant employés indépendamment l'un de l'autre (« petit chien », « vélo rouge ») et facilement substituables ;
- les expressions *semi-figées*, entre les deux précédentes :
  - une partie de la co-occurrence est libre et ne prend pas de sens particulier en fonction de l'autre (« peur » a son sens habituel dans « peur bleue »),
  - l'autre partie est figée, parce qu'elle est choisie en fonction de la première pour exprimer le sens précis (on choisira « bleue » pour exprimer l'intensification de « peur », « grièvement » pour celle de « blessé », etc.)

Le phénomène auquel nous nous intéressons, celui que nous appelons collocation, est celui des expressions semi-figées.

Nous excluons de nos travaux les *collocations grammaticales*, « combinaisons d'un mot lexical et d'un mot grammatical » (Williams, 2003), très fréquentes en anglais où elles sont appelées *phrasal verbs* (« to look up », « to get out », etc.) mais qu'on rencontre aussi en français (« compter avec »)[6]. Nous préférons nous concentrer sur l'extraction de collocations lexicales, celles qui font intervenir deux termes porteurs de sens. La raison en est qu'il s'agit de deux problèmes assez différents qui mènent à des solutions différentes, et que les *phrasal verbs* posent moins de problèmes à la traduction (on peut généralement les traiter comme un verbe avec un régime propre).

2.1.3 Base et collocatif

Mel'čuk (2003) propose les caractéristiques qui permettent de reconnaître une collocation d'une autre co-occurrence :

> « L'expression AB ayant le sens (S) est appelée une collocation si et seulement si les conditions suivantes sont simultanément remplies :
>
> 1. 'S' ⊇ 'A'[7] ;
> 2. A est sélectionné par le locuteur de façon régulière et non contrainte[8] ;

[6]Cela correspond également aux « verbes à particule séparable » de l'allemand, comme par exemple « anrufen » (appeler au téléphone), formé en rajoutant la particule « an » au verbe « rufen » (appeler) ; cette particule est détachée du reste du verbe à la conjugaison : « sie *rief* mich *an* » (« elle m'a appelé au téléphone »).

[7]Le sens de *A* est inclus dans le sens *(S)* de l'expression *AB*.

[8]Mel'čuk parle de choix *régulier* lorsque la lexie est choisie pour exprimer un de ses sens habituels, qu'on pourrait trouver dans un dictionnaire, et de choix *non contraint* lorsque la lexie est choisie indépendamment des autres lexies.

3. B n'est pas sélectionné de façon régulière et non contrainte, mais en fonction de A et du sens à exprimer.[9] »

Puisque les deux composantes d'une collocation ont des propriétés différentes, Hausmann (1989) a introduit la terminologie suivante :

- l'élément autonome (libre) est appelé *base* de la collocation ;
- l'élément qui dépend de l'autre est nommé *collocatif*.

Ainsi, dans notre exemple, « peur » est la base, et « bleue » le collocatif.

2.1.4 La définition retenue

Après avoir cité plusieurs définitions, nous devons en retenir une seule, qui fera foi dans le reste de ce document. La définition de Mel'čuk contient sans doute une trop grande restriction ; Tutin & Grossmann (2002) jugent en effet qu'il est :

> « plus pertinent de parler d'association de constituants (plutôt que de lexies) pour englober des exemples comme les suivants : *fort comme un turc* ; *un bruit à crever les tympans* ; pour lesquels l'élément qualifiant se réalise par un syntagme (par exemple, *fort*, *bruit*), et avec un syntagme (*comme un turc*, *à crever les tympans*) »

Afin de tenir compte de telles expressions et de les englober dans l'ensemble des collocations que nous considérerons, nous utiliserons dans nos travaux la définition d'une collocation donnée par Kahane & Polguère (2001) :

> « une expression linguistique faite d'au moins deux composants :
>
> 1. la *base* de la collocation : une unité lexicale complète qui est « choisie » librement par le locuteur
> 2. le *collocatif* : une unité lexicale ou une expression multi-lexicale qui est choisie d'une manière (en partie) arbitraire pour exprimer un sens donné et/ou une structure grammaticale dépendant du choix de la base ».

2.1.5 Cette co-occurrence est-elle une collocation ?

Il n'est pas toujours facile de distinguer rapidement les collocations des autres co-occurrences habituelles. Pourquoi « respecter profondément » est-elle une collocation et pas « voiture chère » ?

D'après (Mel'čuk et al., 1995), cela dépend du sens propre de la collocation (celui amené par le collocatif présumé). Il ne peut y avoir de collocation dont le collocatif aurait le sens « avoir un prix élevé » puisque cela peut toujours être exprimé par « cher », le choix étant donc indépendant de l'autre terme. Au contraire, le sens d'une collocation peut bien être l'intensification de sa base, puisque le choix du terme l'exprimant varie selon la base (« pluie *battante* », « peur *bleue* », « maladie *grave* », etc.).

[9] B peut donc être sélectionné de façon régulière (et donc avoir son sens habituel) si le choix est contraint. C'est par exemple le cas pour la collocation « grièvement blessé », où « grièvement », qui a son sens habituel, est bien choisi en fonction de « blessé » : pour exprimer le même sens avec « malade », on utilisera « gravement ».

2.2 Types de collocation

2.2.1 Modification du sens

Le premier type de collocation, qui correspond aux exemples donnés jusqu'ici, est celui où le sens de la base est modifié par le collocatif. Certaines collocations de ce type sont très fréquentes (intensification, appréciation positive, etc.), d'autres plus rares : « café noir » exprime le sens « sans produit laitier », qui est bien une collocation puisque le choix de « noir » a été fait en fonction de « café » (on ne peut dire « *thé noir ») (Mel'čuk, 2003).

2.2.2 Verbe support

D'après Polguère (2004) :

> « Un *verbe support* est un collocatif verbal sémantiquement vide dont la fonction linguistique est de verbaliser une base nominale, c'est à dire de la faire fonctionner dans la phrase comme si elle était elle-même un verbe ».

Dans une phrase à verbe support, ce n'est pas le verbe qui a la fonction de prédicat de la phrase, mais le verbe ainsi « supporté » (Gross, 1981).

Un verbe support peut avoir sa base comme objet (« *prendre* une douche », « *commettre* un crime »), mais aussi, plus rarement, comme sujet (« le silence *règne* »).

2.3 Propriétés

Les collocations ont plusieurs propriétés spécifiques, qui influencent leur traitement.

2.3.1 Effectif

Tout d'abord, les collocations étant des co-occurrences significatives, elles se produisent plus souvent que par hasard. On rencontre par exemple plus souvent « pluie battante » ou « grosse pluie » que « grande pluie », « pluie grave », « pluie rose », etc.

Cette propriété est d'une grande utilité dans des tâches de collecte de collocations. On peut en effet envisager d'utiliser des mesures statistiques comme l'information mutuelle (la probabilité effective de co-occurrence divisée par la probabilité que la co-occurrence se produise par hasard), ou d'autres similaires. Il faut toutefois garder à l'esprit qu'une co-occurrence apparaissant fréquemment n'est pas nécessairement une collocation.

2.3.2 Décodabilité, prédictibilité

La *décodabilité* d'une expression est la possibilité de trouver son sens uniquement à partir des mots qui la forment[10]. La *prédictibilité* est, à l'inverse, la possibilité de trouver quelle est l'expression correcte à partir du sens que l'on souhaite exprimer.

[10]On utilise également souvent *compositionnalité* de manière équivalente.

Ces deux problèmes sont importants pour les apprenants d'une langue, ou pour un logiciel qui tente de modéliser la langue : que signifie « driving rain » ? Comment exprime-t-on une peur intense en anglais ?

Tutin & Grossmann (2002) proposent un classement des collocations selon leur décodabilité et leur prédictibilité :

- les collocations *opaques*, comme « peur bleue », non prédictibles et difficilement décodables (le sens du collocatif n'est pas celui qu'on lui connaît habituellement, et il n'y aucun indice – métaphore, métonymie, comparaison – qui pourrait permettre de le trouver) ;
- les collocations *transparentes*, facilement décodables (comme « faim de loup » ou « grièvement blessé », ou les verbes support), mais difficilement prédictibles (pourquoi dit-on « grièvement blessé » et « gravement malade », mais pas « *grièvement malade » ?) ;
- les collocations *régulières*, où le collocatif ne se définit qu'à partir de la base (« année bissextile », « nez aquilin »)[11] ou a, au contraire, un sens très générique (« grande tristesse »), si bien qu'elles semblent prédictibles.

Les collocations sont donc plus ou moins décodables, mais peu (voire pas du tout) prédictibles, ce qui peut causer problème lors de tâches de traitement des langues.

2.3.3 Degré

Le *degré* d'une collocation est l'importance de la modification induite par le collocatif. Par exemple, le degré de « grosse pluie » ou de « pluie battante » est plus faible que celui de « pluie torrentielle » ou « pluie diluvienne ». Il ne concerne pas les verbes supports, ceux-ci ne modifiant pas la base.

Bien que secondaire, le degré d'une collocation doit également être analysé pour permettre de traiter plus précisément la collocation.

3 Conséquences sur les traitements multilingues

Les problèmes rencontrés pour traiter les collocations viennent de leur caractère arbitraire. Comment les décoder et, surtout, les prédire, si on ne peut formuler de règles précises correspondant à leur production ?

3.1 Les collocations en traduction automatique

Le choix du collocatif étant arbitraire, il est souvent différent selon les langues. Par exemple, quelqu'un qui fume beaucoup est un « gros fumeur » en français, un « heavy smoker » (fumeur « lourd ») en anglais, et un « starker Raucher » (fumeur « fort ») en allemand.

[11]On ne rencontre (sauf fantaisie littéraire) ces collocatifs qu'avec une seule base.

Pour traduire correctement une collocation, l'idéal serait donc d'être capable de la décoder dans la langue source, puis de générer une collocation dans la langue cible qui exprime le même sens.

3.1.1 Source d'erreur

En réalité, les systèmes de traduction automatique n'ont généralement pas les informations linguistiques nécessaires pour remonter jusqu'au sens, ni pour ensuite générer la bonne phrase à partir du sens. Leur tâche est de trouver les mots qui expriment un sens équivalent aux mots de la phrase de départ, en tenant compte du fait qu'ils n'ont pas forcément le même contexte d'usage.

La manière naïve de traduire une phrase est de découper la phrase en lexies (et de résoudre toutes les ambiguïtés qui peuvent se présenter), d'associer à chaque lexie des traductions possibles, et de générer la combinaison de traductions qui est la plus probable.

Cette traduction mot à mot (ou plutôt terme à terme, puisque les traducteurs sont généralement capables de reconnaître que la locution « pomme de terre » est une seule unité lexicale, et de la traduire en anglais par « potato »), ne permet pas de traduire correctement les collocations, puisque les collocatifs ne sont souvent pas traductions l'un de l'autre : puisqu'un dictionnaire français-anglais n'a pas d'entrée « battante »-« driving », ce type de système ne pourra proposer « driving rain » comme traduction de « pluie battante »[12].

De ce fait, les collocations sont, avec les erreurs d'analyse syntaxique, les principales sources d'erreur des systèmes de traduction automatique.

De plus, il est à noter que si la traduction d'une collocation est le plus souvent une autre collocation dont les bases sont traductions l'une de l'autre, ce n'est pas toujours le cas. Si l'on peut parfois traduire « peur bleue » par « strong fear » en anglais, la traduction la plus fréquente, la plus proche, se fait par des expressions ayant une autre forme, comme « to be scarred stiff » ou « to be deathly afraid » (des formes proches de « être mort de peur » en français).

3.1.2 Architectures computationnelles en traduction automatique

On peut considérer deux types d'approches computationnelles pour la traduction : les méthodes « expertes » et les méthodes « empiriques » (Boitet, 2008). Il s'agit de la façon dont on réalise le passage d'une représentation intermédiaire de l'énoncé à la suivante. La séquence et la nature des représentations intermédiaires utilisées correspond à l'architecture linguistique utilisée.

3.1.2.1 Approches « expertes »

Il y a actuellement deux grandes façons de faire de la traduction automatique. La première est basée sur l'utilisation de « langages spécialisés pour la programmation linguistique » (LSPL) aux différentes étapes (analyse morphologique, analyse syntaxique, traduction des données issues de l'analyse, etc.) C'est le cas de systèmes comme *Systran*, *Reverso* ou *ProMT*, qui

[12]Plus généralement, la traduction mot à mot est trop simpliste pour fonctionner correctement, même si on était capable de traduire correctement les collocations : les mécanismes de génération des phrases sont différents selon les langues et ne peuvent être ignorés lors de la traduction.

utilisent des *règles de traduction* (qui s'appliquent selon le contexte) qui doivent être écrites par les linguistes. Les collocations sont donc gérées à travers des règles, comme les locutions (si aucune règle ne correspond à la collocation, elle sera traduite mot à mot).

On peut citer comme exemple le système *ATLAS* de Fujitsu (Uchida, 1989) du japonais vers l'anglais et inversement : le dictionnaire général de la version actuelle (*V14*) contient 1,43 million d'entrées de l'anglais vers le japonais et autant dans l'autre sens, les dictionnaires techniques contiennent 2,81 millions d'entrées de l'anglais vers le japonais et 2,76 millions dans l'autre sens (Fujitsu, 2007).

3.1.2.2 Approches « empiriques »

D'autres systèmes de traduction automatique (TA) n'utilisent pas de ressources produites par des experts, mais d'autres méthodes, plus empiriques, parmi lesquelles on peut distinguer :

- la TA *basée sur l'exemple*, qui utilise directement des exemples de traduction ;
- la TA *statistique*, qui se base sur un modèle probabiliste, produit à partir d'un corpus de « segments parallèles » (phrases ou titres traductions l'un de l'autre), pour estimer quelle est la traduction la plus probable.

Pour mieux comprendre l'idée de la TA statistique, à la base de systèmes comme *Google Translate*, considérons un fragment de la phrase à traduire, et repérons les segments « parallèles » à ceux dans lesquels il apparaît ; le fragment apparaissant le plus fréquemment dans ces segments est probablement la traduction du fragment de départ. Une collocation sera donc bien traduite si le système est capable de lui faire correspondre statistiquement une traduction correcte. On peut voir l'utilisation de la statistique comme une autre façon de produire davantage de règles, mais de moins bonne qualité.

3.1.3 Comparatif

Si les bonnes règles existent, une traduction à base de règles est sans doute meilleure qu'une traduction statistique, puisque se basant sur des données validées par des linguistes ; le problème de cette approche vient essentiellement de l'incomplétude et de l'inexactitude éventuelle des dictionnaires, surtout en ce qui concerne les tournures, les idiomes, etc., ce qui montre l'intérêt de la collecte de collocations. S'il s'agit de traduire des données non couvertes par les règles de traduction, les systèmes statistiques sont généralement meilleurs.

3.1.3.1 Tests

Pour vérifier ces suppositions, nous avons effectué plusieurs tests de traduction, en allemand et en anglais, de phrases contenant des collocations, sur différents sites proposant des services de traduction automatique : *Yahoo! Babelfish* (qui utilise Systran), *Google Translate*, *Reverso* (de *Softissimo*), et *ProMT Translator*[13], à un an d'intervalle (le 17 avril 2008 et le 12 avril 2009).

Systran, Reverso, ProMT proposent des logiciels payants basés sur leurs moteurs de traduction respectifs. Bien paramétrés, ces logiciels permettent d'obtenir une meilleure qualité de traduction que les services Web correspondants.

[13] *Reverso* utilisait précédemment le moteur de traduction de *ProMT* ; ce n'est plus le cas dans la version actuelle.

Phrase originale	**Yahoo! Babelfish**[14] (Systran)	**Reverso**[15]	**ProMT Translator**[16]	**Google Translate**[17]	
	avril 2008 / avril 2009 : pas de changement			**avril 2008**	**avril 2009**
Du français vers l'anglais					
*La parade a eu lieu sous un **soleil de plomb**.*	The parade took place under a blazing sun.	The parade took place under a blazing sun.	Parade took place under a blazing sun.	The parade took place under a scorching sun.	The parade took place under a blazing sun.
*Je lui **rendrai visite** demain*	I will visit him tomorrow	I shall visit him (her) tomorrow	I will visit him tomorrow.	I will be travelling tomorrow visit.	I visit him tomorrow.
*Jean est un **gros fumeur***	Jean is a large smoker	Jean is a big smoker	Jean is a big smoker.	John is a heavy smoker	John is a heavy smoker
*Je suis sorti sous une **pluie battante***	I left under a beating rain	I went out under a pouring rain	I went out under a brand rain.	I got out in rain	I went out in a pouring rain
*J'ai eu une **peur bleue***	I had a blue fear	I had a strong fear	I was scared to death	I had a fear blue	I had a fright
*C'est d'une **bêtise insondable***	It is of an unsoundable silly thing	It is of a fathomless stupidity.	It is of an immense stupidity.	It is an unfathomable stupidity.	This is an unfathomable stupidity
Du français vers l'allemand					
*C'est un **gros fumeur***	Es ist ein großer Raucher	Das ist ein großer Raucher	Das ist ein großer Raucher	Es ist ein großer bereich	Es ist ein starker Raucher
*J'ai eu une **peur bleue***	Ich habe eine blaue Angst gehabt	Ich habe eine Heidenangst gehabt	Ich hatte eine blaue Angst.	Ich hatte eine blaue Angst	Ich hatte Angst blau

Tableau 1: Exemples de traductions de collocations par Systran, Google Translate, Reverso et ProMT

De plus, il existe de nombreux autres systèmes de traduction automatique, comme *PAHO-MTS*, *ATLAS*, *The Translator*, *Neon*, etc. Les systèmes de traduction utilisés dans ces tests ne sont donc sans doute pas les meilleurs, mais ils ont l'avantage de proposer des services

[14]http://fr.babelfish.yahoo.com/

[15]http://www.reverso.net/

[16]http://www.online-translator.com/

[17]http://www.google.fr/language_tools?hl=fr

gratuits de traduction sur Internet, sur lesquels on peut mener facilement de tels tests comparatifs.

Les résultats obtenus sont présentés dans le tableau 1. Les cases blanches correspondent à une traduction correcte de la collocation (même si toute la phrase n'est pas traduite correctement), les cases grisées à une traduction incorrecte.

En effet, nous évaluons seulement si la traduction de la collocation est bien effectuée (que ce soit sous la forme d'une autre collocation, d'une autre expression, ou d'un simple terme, pour peu que le sens soit respecté) et ne tenons pas compte d'éventuelles autres erreurs, notamment de syntaxe.

3.1.3.2 Observations

En 2008, lors de la première partie de nos tests, les systèmes qui donnaient les meilleurs résultats (pour la traduction des collocations) sur les phrases testées, étaient ProMT et Reverso, basés sur la traduction par règles. La traduction statistique de Google donnait parfois des résultats intéressants (« heavy smoker », « unfathomable stupidity »), mais de manière trop rare. Enfin, si Systran est sans doute le plus connu de tous ces systèmes, c'est celui qui donnait les moins bons résultats, traduisant mot à mot la plupart des collocations testées.

Un an plus tard, les systèmes basés sur les règles, n'ayant pas subi de mise à jour majeure entre temps, renvoient des résultats identiques (pour les phrases testées) à ceux obtenus lors de la première partie des tests. Au contraire, les résultats de Google Translate ont beaucoup évolué, dans le bon sens ; cela peut venir du fait que le système utilise davantage de corpus parallèles (améliorant la probabilité de trouver le fragment à traduire dans les corpus, et réduisant les ambiguïtés) ou d'une amélioration de l'algorithme effectuant le traitement statistique. Cela montre l'intérêt des systèmes utilisant la statistique : Google Translate, pourtant bien plus récent que ses concurrents, donne aujourd'hui, sur les quelques tests effectués, des résultats d'une qualité comparable aux systèmes à règles comme Reverso ou ProMT.

Les observations effectuées ne permettent pas d'évaluer la qualité intrinsèque de ces systèmes ; elles sont faites sur des exemples simples et ne peuvent être généralisées. L'évaluation des systèmes de traduction automatique est d'ailleurs un domaine de recherche à part entière.

3.2 Une base de collocations ?

La raison pour laquelle ces systèmes ont des résultats moyens, voire médiocres, lorsqu'ils rencontrent des collocations, est qu'ils ne les considèrent pas comme un problème spécifique avec des solutions spécifiques. La tâche réalisée par les systèmes de traduction automatique, qu'ils soient basés sur la statistique ou sur de l'expertise, est relativement compliquée, puisque devant associer un énoncé d'une langue à un énoncé d'une autre langue, et qu'il est difficile d'être à la fois complet et précis, en particulier à causes des très nombreuses ambiguïtés (aussi bien au niveau monolingue qu'au niveau bilingue).

Comment résoudre ce problème ? La solution préconisée par Mel'čuk (à la fin des années 1950) est de décrire les collocations en fonction de la base et du sens exprimé par le collocatif, pour aider à d'obtenir des traductions de meilleure qualité (si un système sait qu'une

« pluie *battante* » est intense, et qu'en anglais « rain » est intensifié par « heavy » ou « driving », il peut traduire correctement cette collocation).

Certes, il s'agit de décrire le sens d'une expression, ce qui ne semble *a priori* pas facile ; cette tâche monolingue demande en réalité moins d'expertise que l'écriture de règles bilingues (tout locuteur natif du français sait qu'une « pluie battante » est intense).

De plus, et c'est peut-être l'argument principal, si l'effort peut sembler trop grand (et inutile) pour une traduction bilingue, cette méthode prend tout son intérêt dans le cas d'un système de traduction multilingue, l'introduction d'une nouvelle langue ne nécessitant plus de pouvoir traduire les collocations entre cette langue et toutes les langues existantes du système, mais simplement de pouvoir coder et décoder le sens d'une collocation dans cette nouvelle langue.

3.3 Bicollocation

Créer une base de collocations monolingues pourrait permettre d'aider le traitement de collocations, mais on peut également envisager la production de règles de traduction « améliorées », qui tiendraient compte des particularités du phénomène collocatif, et de ses conséquences pour la traduction.

C'est pourquoi nous introduisons ici la notion de bicollocation.

> *Une **bicollocation** est un couple de collocations :*
>
> - *dont les bases sont traductions l'une de l'autre ;*
> - *dont les collocatifs expriment, lorsqu'ils sont utilisés avec leur base, exactement le même sens (il n'est pas nécessaire que les collocatifs soient traduction l'un de l'autre).*

(« pluie battante », « driving rain ») est un exemple de bicollocation.

Les collocatifs doivent avoir exactement la même fonction, en particulier porter sur la même composante du sens de la base. Cette définition exclut les cas où la collocation ne se traduit pas aussi simplement par une autre collocation dont la base est traduction. Le but ici n'est pas de proposer une définition adaptée à tous les cas, mais au contraire de se concentrer sur un cas précis (et très fréquent), et sur la manière dont on peut le traiter.

4 Description

Après avoir présenté ce que sont les collocations et les bicollocations, nous pouvons nous intéresser de plus près à leur description : en quoi des collocations peuvent-elles être différentes entre elles ? Quelles propriétés faut-il modéliser dans une représentation des collocations (et des bicollocations) ?

4.1 Questions

4.1.1 Collocation

Grossmann & Tutin (2003) proposent plusieurs points pour caractériser la description d'une collocation :

- *description sémantique* : que signifie la collocation ?
- propriétés *syntaxiques* : le collocatif doit-il être placé avant la base (« grosse bagarre »), après (« bagarre générale »), ou peut-on l'employer dans les deux cas (« violente bagarre », « bagarre violente ») ? peut-on séparer la base de son collocatif (on peut parler d'une « victoire très nette » mais pas d'une « *peur très bleue ») ?
- précisions sur la *relation sémantique* entre le collocatif et la base : sur quelle composante du sens s'applique le collocatif (« blessure profonde » -qui porte sur sa taille- vs. « blessure cuisante » -qui porte sur la douleur causée - Kahane, 2003) ? Quel est son degré d'intensité (« pluie battante » vs. « pluie torrentielle ») ou son degré de réalisation (on « accepte une invitation » avant d'y « donner suite » - Mel'čuk et al., 1995) ?
- précisions sur la *relation syntaxique* : est-ce la base qui régit le collocatif, ou bien l'inverse ?[18] Est-ce que le collocatif est un simple modificateur ?

4.1.2 Bicollocation

Pour décrire une bicollocation, il faut le faire pour chacune de ses parties. On doit ainsi analyser les liens monolingues entre base et collocatif de la manière décrite précédemment. En particulier, l'aspect sémantique de ces deux liens doit correspondre.

Il reste le lien de traduction entre les deux bases. Celui-ci doit être caractérisé de manière *sémantique*, en particulier sur l'approximation de la correspondance : le sens des deux bases est-il réellement identique ? Il est en effet fréquent qu'un mot d'une certaine langue décrive plusieurs concepts auxquels correspondent plusieurs mots dans une autre langue (par exemple, le mot anglais « river » correspond à la fois à « rivière » et « fleuve » en français : il ne peut donc être considéré comme une traduction exacte ni de l'un, ni de l'autre).

4.2 Comment répondre ?

Quelles tâches faut-il réaliser (de manière manuelle ou automatique) pour répondre aux questions que pose la description des collocations ?

4.2.1 Aspects syntaxiques

Une solution pour caractériser le régime qui gouverne la relation syntaxique est l'utilisation d'un analyseur syntaxique.

[18]Le plus souvent, c'est la base qui régit le collocatif (« pluie battante », « gravement malade ») ; dans le cas des verbes support, c'est le collocatif verbal qui régit la base nominale (« prendre une douche », « commettre un crime »).

Les propriétés liées à la syntaxe ont l'avantage d'être détectables à l'usage. Avoir une grande base de documents permettrait de donner un avis sur le figement syntaxique. Il serait alors possible de tester si les différentes combinaisons syntaxiques possibles entre la base et le collocatif y sont employées ou non, et donc de prévoir leur validité. Par exemple, pour une collocation nom-adjectif comme « peur bleue », on peut tester l'existence de « *bleue peur », « *peur très bleue », « la *peur est bleue »[19]. Pour une collocation qui fait intervenir un verbe support, on peut tester le passage à la forme passive (« un crime a été commis »). Cela poserait toutefois un problème important, celui de la couverture du corpus utilisé.

4.2.2 Aspects sémantiques

La description sémantique de la collocation, comme l'estimation de son degré, peut se faire de manière manuelle, par un non-spécialiste, pour les types de collocations les plus fréquents : tout locuteur natif du français sait qu'une « pluie battante » est intense, mais moins qu'une « pluie torrentielle ». Il n'est par contre pas possible d'utiliser une telle évaluation humaine pour déterminer sur quelle composante porte le sens, puisqu'il s'agit de nuances plus compliquées, et qu'on ne saurait pas les stocker (cela nécessiterait une définition très précise du sens de chaque base potentielle, avec les composantes qui en forment le sens).

Le caractère absolu ou approximatif de la traduction entre les bases d'une bicollocation peut, lui, être prédit à l'aide de dictionnaires bilingues, censés contenir ce genre d'information.

5 Quel problème résoudre ?

5.1 Découvrir des collocations

Nous avons présenté plusieurs problèmes liés aux collocations que nous pourrions traiter. Leur description est sans doute très intéressante à effectuer, et très utile, mais cela n'est pas le problème principal : avant de pouvoir donner des détails sur des collocations, il faut déjà en avoir. Or, nous l'avons vu, les systèmes de traduction automatique pèchent notamment à cause de leur manque de ressources, en particulier sur les collocations.

Le problème qu'il faut résoudre est donc celui de la collecte des collocations.

Une collocation peut être vue comme un objet à 3 composantes : la base, le collocatif, et le sens exprimé. Collecter les collocations, c'est donc être capable de trouver ce genre de triplets ; avoir simplement des couples de termes (base et collocatif) n'est pas suffisant : une information incomplète est inutile.

5.1.1 Sources

Sur quoi peut-on se baser pour compléter les ressources linguistiques avec des informations concernant les collocations ? Que font généralement les lexicographes (créateurs de dictionnaires) quand ils doivent écrire une définition complète d'un mot ?

[19]Cela pourrait être fait en testant les modification possibles de l'arbre syntaxique correspondant.

La première solution est de faire appel aux cerveaux humains. Cependant, si cela permet d'obtenir des résultats de qualité, cela reste une tâche lente et difficilement exhaustive. C'est pourquoi les lexicographes utilisent fréquemment des corpus de documents, afin d'y trouver les occurrences du mot à définir et de lister plus facilement les différents emplois d'un terme. Suivant une idée similaire, on peut utiliser les corpus de documents comme source où apparaissent de nombreuses collocations. De même, l'utilisation de corpus bilingues, avec des documents traductions l'un de l'autre, peut permettre de faire émerger des bicollocations.

Si, au contraire, on disposait de ressources importantes sur les collocations, on pourrait envisager d'appliquer des processus d'apprentissage automatique sur celles-ci, pour établir quels sont réellement les critères discriminants des collocations recherchées pour améliorer le processus de collecte.

5.1.2 Mesure de qualité

Lors d'un processus de collecte de collocations, de nombreux candidats sont produits. Il s'agit ensuite de les trier (de manière manuelle et automatique) selon qu'ils sont ou non des collocations. C'est pourquoi nous introduisons les notions de collocabilité et de bicollocabilité.

> *La* collocabilité *d'un couple de termes est sa capacité à former une collocation.*
>
> *La* bicollocabilité *d'un quadruplet de termes est sa capacité à former une bicollocation.*

La collecte de collocations devra donc contenir une étape d'évaluation de la collocabilité (dans le cas monolingue) ou de la bicollocabilité (dans le cas bilingue) supposée des candidats. Ces mesures peuvent prendre une valeur binaire (est-ce correct ou non ?), ou une valeur scalaire (basée sur le respect d'un plus ou moins grand nombre de critères).

5.2 Utilisation de l'outil informatique

L'analyse de phénomènes complexes comme celui des collocations requiert beaucoup de temps pour obtenir des résultats précis, manuellement. Si on ne peut utiliser l'informatique pour faire une telle analyse complète, faute de qualité suffisante, on peut en revanche chercher à faciliter la tâche humaine grâce à la rapidité de traitement qu'offre l'utilisation de processus automatiques.

Ainsi, la collecte de collocations à partir de corpus de documents peut sans doute se faire à l'aide d'outils informatiques capables d'automatiser des processus d'extraction de candidats, voire d'un éventuel classement.

L'effort de facilitation de la tâche par les ordinateurs peut également se faire par la réalisation d'interfaces homme-machine facilitant la collecte. Pour une collaboration humaine, cela passe par la réalisation d'un logiciel ou d'une page Web facilitant la description des collocations. Dans tous les cas (processus humains ou automatiques), la tâche doit pouvoir être facilement supervisée par un spécialiste, grâce à une interface pratique.

Nous étudierons, dans le chapitre suivant, les différentes façons existantes de détecter les collocations, ce qui nous permettra, en nous focalisant sur l'une d'entre elles, de préciser la

manière dont nous comptons améliorer le traitement des collocations grâce à l'outil informatique.

Conclusion

Nous avons présenté dans ce chapitre la notion de *collocation*, ses propriétés, sa grande importance pour les systèmes de traitement automatique des langues, et en premier lieu pour les systèmes de traduction automatique. En effet, ces derniers manquent de ressources concernant ce phénomène linguistique, ce qui provoque d'importantes erreurs de traduction. Un traitement correct des collocations par ces systèmes passe, en particulier dans le cadre multilingue, par l'utilisation de ressources modélisant leur sens (afin de pouvoir analyser le sens d'une collocation d'une langue, et générer, à partir de ce sens, la collocation correspondante dans l'autre langue). Il faut pouvoir combler ce manque de ressources, en fournissant les outils informatiques nécessaires à leur production.

Idéalement, de manière à faciliter leur utilisation dans des tâches de traitement des langues, les connaissances linguistiques collectées (comme les collocations) doivent être inscrites dans une description précise du lexique. Nous devons donc maintenant étudier les modèles de lexique existants, de manière à retenir celui qui convient le mieux à l'étude des collocations.

Chapitre II - Un cadre théorique pour le traitement des collocations

La collecte de connaissances linguistiques (et de collocations en particulier), quelque soit la manière dont la réalise, gagne à se placer dans le cadre d'un modèle de description du lexique. En effet, dans de tels modèles, la représentation du lexique, de ses éléments, et des liens syntaxiques ou sémantiques pouvant exister entre les éléments, est spécifiée pour être la plus précise possible, mais aussi, le plus souvent, pour permettre une utilisation facile dans des tâches de traitement automatique des langues.

Nous présenterons tout d'abord les principaux modèles de lexique existants (fonctionnant par énumération ou par regroupement), en étudiant particulièrement leur capacité éventuelle à décrire efficacement les collocations. Nous effectuerons ensuite un comparatif entre ces modèles, pour justifier le choix du modèle des fonctions lexico-sémantiques. Enfin, nous décrirons plus précisément la théorie sens-texte, dont est issu le modèle des fonctions lexico-sémantiques.

1 Lexiques énumératifs

Nous présentons d'abord les modèles de lexiques énumératifs, c'est à dire ceux qui décrivent chaque élément du lexique.

1.1 Lexicologie explicative et combinatoire - Fonctions lexico-sémantiques

La *lexicologie explicative et combinatoire* est une application de la *théorie sens-texte* d'Igor Mel'čuk (1997) à la description des éléments du lexique. Elle se découpe en trois parties :

- une section *sémantique* : définition des différents sens que peut prendre un vocable
- une section *syntactico-combinatoire* (pour chaque acception) : schéma de régime
- une section *lexico-combinatoire*, sous la forme de fonctions lexico-sémantiques.

Les *fonctions lexico-sémantiques* (FLS) font partie de la théorie sens-texte ; leur rôle est de décrire les différents types de liens lexicaux (Mel'čuk et al., 1995) :

> « une fonction lexicale [=FL] est une fonction au sens mathématique ; elle peut être représentée par la formule traditionnelle :
>
> **f**(x)=y
>
> où x est l'*argument* de la fonction et y sa *valeur*. »

(l'argument est aussi appelé *mot-clé* pour éviter des ambiguïtés). Par exemple : `'intensification'(pluie)=battante`

Plus précisément, une FLS est une fonction vers une partie du lexique, un *ensemble* de « valeurs » possibles. Par convention (et pour ne pas surcharger l'écriture), l'ensemble des valeurs d'une FLS est simplement listé (sans être entouré d'accolades).

Les fonctions lexico-sémantiques permettent la modélisation des relations syntagmatiques (collocations) et des relations paradigmatiques (par exemple, `'synonyme'(voiture) = automobile`).

Au sein de cette théorie, 52 FLS standard ont été décrites (Mel'čuk et al., 1995), notamment lors de la création de dictionnaires basés sur cette modélisation du lexique. Nous allons décrire ici quelques FLS, les plus communes (certains des exemples sont issus de ces dictionnaires).

Avant cela, il nous faut présenter les notions d'*actant sémantique* et d'*actant syntaxique profond* utilisées dans le modèle des fonctions lexico-sémantiques.

Un *actant sémantique* est une expression qui correspond à un argument du prédicat (Mel'čuk et al., 1995). Par exemple, le prédicat « donner » a 3 arguments ; ses actants sémantiques sont :

- *X* : celui qui donne ;
- *Y* : celui qui reçoit ;
- *Z* : ce qui est donné par *X* à *Y*.

Un *actant syntaxique profond* est un syntagme qui exprime un actant sémantique. (Mel'čuk et al., 1995). Les actants syntaxiques sont numérotés ; par exemple, pour une lexie verbale l'actant *I* est le sujet grammatical, l'actant *II* le complément d'objet principal, etc.

L'ordre des actants syntaxiques n'est bien sûr pas toujours le même que celui des actants sémantiques correspondants.

1.1.1 Fonctions lexico-sémantiques paradigmatiques

Les FLS paradigmatiques représentent les liens sémantiques qui peuvent exister entre lexies.

`Syn` associe à une lexie ses synonymes exacts ou approximatifs. Dans le second cas, une notation ensembliste est utilisée pour décrire les recouvrements de sens (`Syn⊂` pour les hyperonymes, `Syn⊃` pour les hyponymes, `Syn∩` s'il y a seulement une intersection de sens entre la valeur et l'argument) :

```
Syn(poitrine)=thorax
Syn⊂(poitrine)=sein
Syn⊃(poitrine)=torse
Syn∩(poitrine)=buste[20]
```

`Anti` associe à une lexie ses antonymes (contraires) :

```
Anti(riche)=pauvre
Anti(petit)=grand
```

[20]Ces exemples sont issus de (Mel'čuk et al., 1992).

Conv associe à une lexie son conversif. Deux lexies sont le *conversif* l'une de l'autre si elles représentent la même situation, et si leurs listes d'actants syntaxiques sont une permutation l'une de l'autre. Par exemple mari et femme (dans le sens d'épouse) sont des conversifs : « Marc est le mari de Laure » est équivalent à « Laure est la femme de Marc ». Pour indiquer quelle permutation effectuer, des indices sont rajoutés à la FLS :

$Conv_{3241}$(acheter)=vendre[21]

S_0 et **V_0** associent à une lexie la contrepartie, respectivement nominale et verbale, de celle-ci. Une contrepartie n'est pas forcément une dérivation morphologique :

S_0(pleuvoir)=pluie
S_0(dormir)=sommeil
V_0(payant)=payer
V_0(sommeil)=dormir

1.1.2 Fonctions lexico-sémantiques syntagmatiques

Les fonctions lexico-sémantiques permettent également de représenter des collocations. Dans ce cas, le mot-clé est la base, et la valeur est le collocatif (les collocatifs s'il y a plusieurs possibilités).

Magn est la fonction qui représente l'intensification :

Magn(pluie)=grosse, battante, torrentielle

Bon représente une appréciation positive :

Bon(conseil)=bon, excellent, précieux, salutaire

$Oper_i$, **$Func_i$**, **$Labor_{ij}$**, sont les différentes fonctions qui permettent de représenter les verbes support, dont le mot-clé est une lexie nominale prédicative[22] :

- **$Oper_i$** : le mot-clé est *complément d'objet direct* du verbe, et le sujet grammatical est le *i*-ème actant syntaxique profond du prédicat. Par exemple, pour le prédicat « conseil » (« Pierre [=I] conseille cette voiture [=II] à Jean [=III] »), on a :

$Oper_1$(conseil)=donner	(« Pierre donne un conseil »)
$Oper_3$(conseil)=recevoir	(« Jean reçoit un conseil »)

- **$Func_i$** : le mot-clé est *sujet grammatical* du verbe, et le complément d'objet est le *i*-ème actant syntaxique profond du prédicat[23]. Par exemple, pour le prédicat « analyse » (« l'analyse par Paul [=I] du vote [=II] »), on a :

$Func_1$(analyse)=provenir [de N]	(« l'analyse provient de Paul »)
$Func_2$(analyse)=concerner [N], traiter de [N], porter sur [N]	(« l'analyse concerne le vote »)

[21] « X1 achète X2 à X3 pour X4 » est équivalent à « X3 vend X2 à X1 pour X4 »

[22] C'est-à-dire une lexie nominale s'interprétant comme un prédicat, comme *obligation* : X1 a une obligation X2 par rapport à X3.

[23] Il ne s'agit pas nécessairement d'un complément d'objet direct.

- **$Func_0$** : le mot-clé est *sujet grammatical* du verbe intransitif, qui n'a donc pas de complément d'objet :

$Func_0$(silence)=régner (« le silence règne »);

- **$Labor_{ij}$**: le mot-clé est le *deuxième complément (fortement régi) d'objet* du verbe, le sujet grammatical est le *i*-ème actant syntaxique profond du prédicat, et le complément d'objet direct est le *j*-ème argument. Par exemple, pour le prédicat « estime » (« estime de Pierre pour Jean »), on a :

$Labor_{12}$(estime)= tenir [N en ~] (« Pierre tient Jean en estime »).

1.1.3 Fonctions lexico-sémantiques irrégulières

Si 52 FLS standard ont été décrites (la liste complète est présentée dans l'annexe A), ce n'est évidemment pas assez pour être exhaustif. D'autres fonctions comblent ce manque.

Les *fonctions lexico-sémantiques complexes* sont des combinaisons de FLS simples. Par exemple, **AntiBon** est le contraire d'une évaluation positive, donc une évaluation négative :

```
AntiBon(conseil)=mauvais, pernicieux.
```

D'autres cas sont plus compliqués :

```
S0IncepPredPlus(coût)=augmentation
```

est ainsi la contrepartie nominale (S_0) du verbe qui exprime le commencement (Incep) du fait d'être (Pred) d'un « coût » plus élevé (Plus).

Attention, il ne s'agit pas d'une composition de fonctions au sens mathématique du terme :

```
IncepOperi(feu)=ouvrir
Incep(Operi(feu))= Incep(faire[24])=commencer (Mel'čuk., 2003).
```

Les *configurations de fonctions lexico-sémantiques* sont des suites de FLS qui s'appliquent en même temps et indépendamment. Par exemple, on peut décrire le fait que « paradisiaque » est le collocatif qui permet de qualifier une « joie » à la fois bonne et intense par :

```
Bon+Magn(joie)=paradisiaque
```

Enfin, les *fonctions lexico-sémantiques non standard* correspondent aux cas où le sens est si spécifique qu'on ne peut l'exprimer avec des FLS standard, même en les combinant. Pour pouvoir tout de même décrire ces cas, il faut alors introduire ces FLS spécifiques. Par exemple :

```
à peine cuit(steak)=saignant
peu cuit(steak)=bleu
cuit(steak)= à point                                    (Mel'čuk et al., 1995).
```

[24]Car Operi(feu)=faire.

1.1.4 Applications : le DEC et le DiCo

La théorie sens-texte, et plus spécialement la partie concernant les fonctions lexico-sémantiques, a été mise en œuvre pour la réalisation d'un dictionnaire papier nommé « *Dictionnaire explicatif et combinatoire du français* » (DEC), dont quatre volumes ont déjà été publiés (Mel'čuk et al., 1984, 1988, 1992, 1999)[25]. C'est d'ailleurs cette tâche lexicographique qui a permis, grâce aux nombreux cas étudiés pour cela, de définir les FLS standard. Des « DEC » ont également été réalisés en russe, en anglais, et en japonais.

Le *Dictionnaire de Cooccurrences* (DiCo) est la version informatisée (et simplifiée, en mettant de côté les aspects trop sémantiques) du DEC, qui sert également comme base d'un lexique d'apprentissage du vocabulaire (Mel'čuk & Polguère, 2006). Il est consultable en ligne à travers l'interface DiCouèbe[26].

Chaque entrée du DiCo est découpée en 7 rubriques :

- nom de la lexie ;
- propriétés grammaticales (partie du discours, genre des noms, etc.) ;
- formule sémantique (par exemple, la formule sémantique de `adhésion.II.1` est `action mentale : ~ de personne X à idée y`);
- le régime d'emploi de la lexie ;
- les fonctions lexico-sémantiques associées ;
- des exemples d'emploi ;
- les expressions idiomatiques dans lesquelles la lexie apparaît.

Le fait que DiCo soit une base de données lui procure un avantage notable sur le DEC : on peut se limiter à la description des fonctions lexico-sémantiques jugées essentielles pour couvrir plus rapidement le lexique, au contraire du DEC dans lequel toutes les entrées doivent être présentées de manière complète en une seule fois (dans un seul volume).

La structure simplifiée de DiCo est également utilisée (à quelques aménagements près) dans les volumes monolingues de la base lexicale multilingue *Papillon-NADIA* (Mangeot et al., 2003).

1.1.5 Modélisation des collocations

Les fonctions lexico-sémantiques permettent de modéliser les différents types de liens de la langue, en particulier les liens syntagmatiques. On peut donc utiliser les FLS pour modéliser les collocations. Par exemple :

```
Magn(peur)=bleue
```

[25]Le DEC a été informatisé de façon structurée (Sérasset, 1997).

[26]http://olst.ling.umontreal.ca/dicouebe/

1.2 WordNet

WordNet est une base de données lexicales (Fellbaum, 1998), développée à l'université de Princeton sous la direction de George A. Miller, distribuée gratuitement[27]. Elle consiste en un réseau de *synsets* (groupement particulier de termes du lexique).

D'autres bases ont été développées en utilisant la même structure, telles *EuroWordNet* (payante) pour de nombreuses langues européennes (Vossen, 2004), *Wolf* (gratuite[28]) pour le français (Sagot et Fišer, 2008), *Hindi Wordnet* (gratuite) pour le hindi (Narayan et al., 2002)[29].

1.2.1 Synsets

La particularité de la base WordNet est l'utilisation de synsets. Un *synset* (synonym set) est un ensemble de quasi-synonymes qu'on peut substituer l'un à l'autre dans un certain contexte sans que cela ne modifie la valeur de la proposition.

Ces quasi-synonymes ne sont pas des mots, mais des acceptions : si un mot est polysémique, ses différents sens sont numérotés, et ce sont ces sens qu'on retrouvera dans des synsets.

Les synsets sont accompagnés d'une description du sens représenté, et d'exemples d'utilisation.

On peut retrouver dans l'annexe C une présentation complète de la représentation d'un synset dans *WordNet*.

1.2.2 Relations entre synsets

Les relations sémantiques présentes dans WordNet relient des synsets entre eux ; il s'agit de liens paradigmatiques. Par exemple, les relations suivantes s'appliquent aux synsets regroupant des noms :

- l'*hyponymie* (« est un type de ») et l'*hyperonymie* (la relation inverse) : par exemple, « chien » est un hyponyme de « mammifère », « animal », « vertébré », etc., et un hypernonyme de « caniche », « berger allemand », « chihuahua », etc.
- la *méronymie* (« est contenu dans ») et l'*holonymie* (« contient ») : par exemple, « genou » est un méronyme de « jambe », et un holonyme de « rotule ».

Il y a également dans WordNet des relations lexicales, non plus entre synsets, mais entre termes, pour indiquer par exemple les phénomènes de dérivation.

1.2.3 Pas de modélisation des collocations

Les systèmes de type WordNet ne contiennent pas de lien syntagmatiques, et ne permettent donc pas de représenter les collocations.

[27] http://wordnet.princeton.edu/

[28] http://gforge.inria.fr/projects/wolf/

[29] http://www.cfilt.iitb.ac.in/wordnet/webhwn/

1.3 ComLex

1.3.1 Entrée lexicales

ComLex est un dictionnaire électronique monolingue anglais, développé dans les années 90 à l'université de New York (Grishman et al., 1994). Une entrée de ComLex est présentée sous la forme d'une structure de traits, écrite dans le style du langage *Lisp*.

Chaque entrée de ComLex contient :

- la *partie du discours*[30] ;
- le trait `:orth` qui contient le *lemme* (si l'entrée a un pluriel, un féminin, une conjugaison irrégulière, l'entrée contiendra également des traits caractérisant ces irrégularités) ;
- la *sous-catégorisation* (ou *rection*) éventuelle, avec le trait `:subc` qui permet de décrire les valeurs possibles des compléments ;
- les autres *propriétés syntaxiques* éventuelles de l'entrée, avec le trait `:features`.

Par exemple, les entrées `abandon` de ComLex sont :

```
(verb     :orth "abandon"
          :subc ((np-pp :pval ("to")) (np)))

(noun     :orth "abandon"
          :features ((countable :pval ("with"))))
```

Cela indique que la première entrée est un verbe et que son complément est soit un syntagme nominal suivi d'un syntagme prépositionnel introduit par « to » (*np-pp :pval ("to")*), soit un syntagme nominal (*np*) ; quant à la seconde entrée, c'est un nom qui doit être précédé d'un déterminant au singulier (*countable*) à moins qu'il ne soit précédé par la préposition « with ».

1.3.2 Cadres verbaux

Les types des compléments apparaissant dans la sous-catégorisation (rection) sont définis à travers des *cadres verbaux*, qui contiennent les traits `:cs` pour la structure des constituants, `:gs` pour la structure grammaticale, `:features` pour d'autres caractéristiques, et `:ex` pour des exemples.

Voici par exemple le cadre verbal « s » de ComLex, qui peut être utilisé pour les propositions où un complément est introduit optionnellement par « that » :

```
(vp-frame s
     :cs ((s 2 :that-comp optional))
     :gs (:subject 1 :comp 2)
     :ex "they thought (that) he was always late")
```

La structure grammaticale ainsi définie indique que le sujet est le premier argument syntaxique du verbe, et que le complément est le deuxième.

[30] Classe morpho-syntaxique comme *nom*, *verbe*, *adjectif*, etc.

1.3.3 Pas de modélisation des collocations

ComLex ne contient pas d'informations concernant des relations entre lexies, et à plus forte raison, pas d'information sur les collocations.

1.4 EDR

Le dictionnaire électronique japonais-anglais *EDR* est issu d'un projet mené par le *Japan Electronic Dictionary Research Institute*. Plus qu'un dictionnaire, c'est une base lexicale faisant intervenir de nombreux « volumes » (Miyoshi et al., 1996).

1.4.1 Dictionnaires de mots

Les *dictionnaires de mots* sont des dictionnaires monolingues traditionnels. Il y a donc dans EDR deux dictionnaires de mots, un pour le japonais, et un pour l'anglais.

Une entrée d'un tel dictionnaire comporte :

- des informations sur le *vocable* : son nom, la partie du mot qui reste inchangée à la flexion, la division en syllabes (pour l'anglais) ou la représentation en *kanas* (pour le japonais), et sa prononciation ;
- des informations *grammaticales*, et notamment la partie du discours, l'« arbre syntaxique » (structure syntaxique entre le terme et ses composants) ;
- des informations *sémantiques* sur le concept exprimé par le mot : l'identifiant (bilingue), les vocables japonais et anglais qui lui sont associés dans les dictionnaires de concepts respectifs correspondants, et l'explication du concept en japonais et en anglais ;
- des informations sur l'usage et la fréquence d'occurrence du terme dans le corpus de référence (20 millions de phrases pour chaque langue).

Les dictionnaires de mots japonais et anglais contiennent respectivement environ 270 000 et environ 190 000 mots.

1.4.2 Dictionnaires bilingues

Il y a deux *dictionnaires bilingues* dans EDR, pour les deux sens de la traduction (japonais-anglais et anglais-japonais). Là encore, on reste dans des structures habituelles de dictionnaire ; une entrée bilingue comporte :

- le *vocable* (anglais ou japonais, selon la langue de départ du dictionnaire) ;
- une information grammaticale sur la *partie du discours* du vocable ;
- des informations sémantiques (comme celles qu'on peut trouver dans les dictionnaires de mots) ;
- et surtout, des *informations de correspondance* : le terme qu'on lui fait correspondre (et sa partie du discours), le type de correspondance (traduction, équivalence, paraphrase, transcription, explication).

Le dictionnaire bilingue japonais-anglais contient 230 000 mots. Le dictionnaire bilingue anglais-japonais contient 160 000 mots

1.4.3 Dictionnaires de concepts

Nous entrons, avec les *dictionnaires de concepts*, dans ce qui fait l'originalité d'EDR. Il s'agit de modéliser des concepts correspondant à la fois à l'anglais et au japonais. Le dictionnaire de concept contient 3 volumes :

- le *dictionnaire des concepts-clés* : chaque entrée modélise un concept, et contient un *identifiant*, un *mot-clé* anglais et un mot-clé japonais définissant le concept, et des explications du concept en anglais et en japonais ;
- le *dictionnaire de classification des concepts*, qui représente la hiérarchie des concepts ; les entrées de ce dictionnaire associent simplement, à chaque concept un *super-concept* (plus large) et un *sous-concept* (plus fin) ;
- le *dictionnaire de description des concepts*, qui représente les relations entre concepts sous la forme (concept 1, type de relation, concept 2, valeur de certitude associée à la relation).

Le dictionnaire de concepts contient environ 410 000 concepts.

1.4.4 Dictionnaires de co-occurrences

Une autre originalité de EDR vient des dictionnaires de co-occurrences [31] : ce sont des dictionnaires monolingues qui listent les co-occurrences les plus fréquentes du japonais et de l'anglais. La structure d'une entrée correspondant à une co-occurrence est la suivante :

- le syntagme ;
- la liste des constituants ;
- le *sous-arbre syntaxique* (la structure syntaxique du syntagme) ;
- des informations sur la *situation de co-occurrence* : l'effectif dans le corpus de référence, un exemple

Il manque évidemment un lien entre les co-occurrences et le reste du dictionnaire : pour optimiser l'utilisation de ces dictionnaires pour la traduction automatique, il faudrait que les co-occurrences monolingues soient associées soit à des concepts, soit à des co-occurrences ou des vocables de l'autre langue.

Les dictionnaires de co-occurrences japonais et anglais contiennent respectivement environ 900 000 et environ 460 000 co-occurrences.

1.4.5 Modélisation partielle des collocations

Les dictionnaires de co-occurrences de EDR contiennent un grand nombre de co-occurrences, dont beaucoup sont des collocations. Malheureusement, celles-ci ne sont pas typées : on ne peut pas distinguer les collocations des autres co-occurrences, ni entre différents types de collocations.

[31] Cette idée a été reprise du système *ATLAS II* (Uchida, 1989).

1.5 HowNet

HowNet est une base de connaissances[32] de l'anglais et du chinois, comportant des relations entre concepts, et des relations entre attributs de concepts[33] (Dong & Dong, 2003).

1.5.1 Graphe de concepts

La base HowNet peut être vue comme un graphe de concepts, dont les liens représentent des relations de synonymie, d'antonymie, de conversivité, entre un ensemble et une de ses parties, entre un attribut et le concept sur lequel il porte, etc.

Numéro du concept	NO.=000001	NO.=015492
Mot/syntagme chinois	W_C=打	W_C=打
Catégorie grammaticale du mot/syntagme chinois	G_C=V	G_C=V
Exemple d'emploi du mot/syntagme chinois	E_C=~□ 油，~□ 票，~□ ， 去~瓶酒，醋~来了	E_C=~毛衣，~毛□ ，□ ~双毛袜子，~草鞋，~一条□ 巾，□ ~麻□ ，~条□ 子
Mot/syntagme anglais	W_E=buy	W_E=knit
Catégorie grammaticale du mot/syntagme anglais	G_E=V	G_E=V
Exemple d'emploi du mot/syntagme anglais	E_E=	E_E=
Définition du concept	DEF=buy|□	DEF=weave|□ □

Tableau 2: Entrées du HowNet Knowledge Dictionary pour les deux sens du mot chinois « 打 »

Outre les données en elles-mêmes, HowNet contient des outils de gestion de la base, et de la documentation. Le cœur de la base est le « HowNet Knowledge Dictionary ». Chaque entrée de ce dictionnaire est composée de :

- un mot (ou un syntagme) chinois ;
  - sa catégorie grammaticale ;
  - un exemple d'emploi ;
- un mot (ou un syntagme) anglais ;
  - sa catégorie grammaticale ;
  - un exemple d'emploi ;

[32]Les créateurs de HowNet insistent sur le fait qu'il ne s'agit pas d'une base lexicale, d'un thesaurus ou d'un dictionnaire, mais bien d'une base de *concepts* qui utilise les mots parce qu'ils représentent des concepts.

[33]Elle peut être téléchargée sur le site http://www.keenage.com/, il existe également une version en ligne de la base à l'adresse http://hownet.kookge.com/.

- une définition (bilingue) du concept décrit dans l'entrée.

Le tableau 2 présente les entrées du HowNet Knowledge Dictionary correspondant aux deux sens d'un mot chinois polysémique.

Les concepts sont eux définis grâce à un ensemble de traits sémantiques. Par exemple, le concept « tug-of-war » (tir à la corde) est défini de manière à indiquer que c'est un fait, un exercice et un sport :

DEF=fact|事情,exercise|□ □,sport|体育

Les relations entre les différents traits sémantiques sont définies dans diverses taxonomies constituant la base.

La première version de HowNet a été rendue publique en 1999. La dernière version, datant de 2008, contient 93 247 entrées pour des mots (ou syntagmes) chinois et 107 698 entrées pour des mots (ou syntagmes) anglais ; elle contient également 86 846 entrées pour des concepts chinois et 107 511 entrées pour des concepts anglais.

HowNet a été utilisé comme base de connaissances pour la création de corpus annotés pour le chinois (Li et al., 2003).

1.5.2 Pas de modélisation des collocations

HowNet contient des relations de paradigmatique entre concepts. Malheureusement, elle ne permet pas de représenter des relations syntagmatiques comme les collocations.

2 Lexiques par regroupements

Décrire chaque élément du lexique, comme on le fait dans un modèle de lexique énumératif, a comme inconvénient de passer sous silence la plupart des relations entre termes. C'est pourquoi certains modèles de lexique, tels que ceux que nous allons présenter, tentent au contraire de regrouper les termes selon leurs propriétés communes.

2.1 Lexique génératif

On peut décrire un lexique de différentes manières. Dans le cas d'un lexique *énumératif*, (comme celui qui est proposé par la lexicologie explicative et combinatoire), toutes les lexies sont décrites. Une critique faite à ces lexiques énumératifs est leur traitement de la polysémie : en considérant comme deux entrées différentes des acceptions proches d'un même vocable, on perd généralement le lien entre les deux. Dans un *lexique génératif*, notion définie par James Pustejovsky (1993), ces acceptions proches sont représentées par une même entrée, auxquelles on peut ensuite appliquer diverses fonctions.

Il y a donc deux parties dans un lexique génératif :

- un *noyau* de lexies ;
- les opérations qui permettent de générer toutes les autres lexies à partir de ce noyau.

2.1.1 Noyau

La description des entrées du lexique génératif de Pustejovsky se fait à l'aide de quatre structures :

- la structure *argumentale* des prédicats représente :
  - le nombre d'arguments ;
  - le type argumental : vrai (nécessaire), par défaut (facultatif), fantôme (dont le sens est compris dans le sens de la lexie) ou modificateur ;
  - le type ontologique (humain, objet, etc.) ;
  - la « réalisation syntaxique » (ordre des arguments) ;
- la structure *événementielle*, indique le type des « événements » (*activité*, *état*, *transition*), qui sont parfois une composition de plusieurs sous-événements ;
- la structure de *qualia* spécifie les propriétés sémantiques, divisée en quatre *rôles* :
  - le rôle *constitutif* détermine la relation entre l'objet et ses constituants ;
  - le rôle *formel* indique ce qui permet de distinguer l'objet ;
  - le rôle *télique* est la fonction ou le but de l'objet ;
  - le rôle *agentif* décrit les facteurs qui interviennent lors de la création de l'objet ;
- la structure d'*héritage* lexical permet de spécifier les relations d'héritage entre les lexies.

2.1.2 Mécanismes génératifs

Trois mécanismes de génération sont décrits dans la théorie de Pustejovsky :

- la *cocomposition* permet d'utiliser la même entrée verbale pour des cas distincts selon son complément (« cuire les patates » induit un changement d'état de celles-ci, alors que « cuire un gâteau » crée le gâteau en question). Le mécanisme de cocomposition du verbe et de son argument produit une structure représentant le syntagme, où intervient une partie de la description de l'argument (pour l'exemple donné, la cocomposition exploite le fait que la structure qualia de « gâteau » indique qu'on le crée en le cuisant).
- le *liage sélectif* concerne les mots dont le sens est fixé par leur argument (par exemple, le nom sur lequel porte un adjectif) : le mot caractérise alors un événement exprimé dans la structure qualia de son argument, généralement dans le rôle télique, voire l'agentif (dans « bon couteau », « bon » fait référence à l'événement correspondant à la fonction télique de son argument « couteau », en l'occurrence « couper »).
- la coercion de types change le type d'un argument pour qu'il ait le type exigé par son prédicat : par exemple, on peut dire « commencer un livre », même si commencer attend un argument événement et que « livre » n'est pas un événement, parce que « livre » a dans sa structure qualia des événements (son rôle télique est « lire », son rôle agentif est « écrire ») ; l'expression peut donc signifier, selon le contexte, qu'on commence à écrire ou à lire un livre.

2.1.3 Modélisation partielle des collocations

Les seuls informations présentes dans un lexique génératif pouvant correspondre à des liens syntagmatiques sont les rôles *télique* et *agentif*, qui peuvent être utilisés pour décrire des verbes support exprimant la cause ou la réalisation.

Par exemple, on « fait » un « examen ».

Examen	
ARGSTR =	ARG1=**x:question**
EVENSTR =	E1=**e1:processus**
QUALIA =	Question.processus-lcp FORMAL = **demander(e1, z, x)** AGENT = **faire(e2, y, x)**

Figure 1: Entrée du lexique génératif pour « examen »

2.2 Sémantique des cadres – Framenet

2.2.1 Cadres sémantiques

La *sémantique des cadres* est une théorie de représentation du lexique proposée par Charles J. Fillmore (1982). L'idée de base est qu'il faut maîtriser les concepts présents dans la langue pour comprendre les mots, l'analyse du sens d'un mot nécessitant de connaître le sens des concepts auxquels il est relié.

Les *cadres sémantiques* sont des structures qui représentent des situations, des objets, ou des événements.

Chaque cadre comporte :

- des *éléments de cadre*, correspondant aux individus, objets ou concepts impliqués dans le concept décrit (les « actants » de Tesnière) ;
- des *unités lexicales*, les mots qui évoquent le concept décrit.

L'analyse du sens d'un mot se fait alors en recherchant quels cadres conceptuels il *évoque*, ou quels éléments de cadre il *instancie*.

2.2.2 Les cadres sémantiques de Framenet

Framenet est l'application concrète (menée par Fillmore lui-même) de la sémantique des cadres à la description du lexique, qui se fait à deux niveaux : les cadres sémantiques et les unités lexicales (Fillmore et al., 2003).

Dans Framenet, l'entrée correspondant à un cadre sémantique, ou *Frame*, comprend :

- une définition du concept ;
- les éléments de cadre (*frame element*, *FE*), classés en 2 groupes selon leur importance dans la réalisation du concept : les éléments noyaux (*core FEs*), nécessaires, et les éléments non noyaux (*non-core FEs*), facultatifs ;
- les relations du cadre avec d'autres cadres ;
- les unités lexicales (*lexical units*, *LU*) qui évoquent le concept.

Par exemple, l'entrée de Framenet correspondant au cadre « waking_up » (« réveil ») contient (au 16/07/2009) :

Définition :
« A sleeper transitions from a sleeping state to a wakeful state »
(« un dormeur passe d'un état de sommeil à un état d'éveil »)

Éléments de cadres:
noyaux :

Sleep_state [Sleep_state]	(état de sommeil)
Sleeper [Slpr]	(dormeur)

non noyaux :

Circumstances []	(circonstances du réveil)
Depictive [Dep]	(état dans lequel le dormeur se réveille)
Explanation []	(raison du réveil)
Frequency []	(fréquence de l'événement)
Manner[Man]	(manière de laquelle le dormeur se réveille)
Particular_iteration []	(état de sommeil)
Place [Place]	(lieu du réveil)
Time [Time]	(moment du réveil)
World_state [W-state]	(état du monde au réveil)

Héritage :

From: Event	(hérite de)
Subframe of: Sleep_wake_cycle	(sous-cadre de)
Precedes: Being_awake	(précède)
Is Preceded by: Sleep	(est précédé par)

Unités lexicales :
awake.v, come to.v, get up.v, revive.v, wake up.v, wake.v[34]

2.2.3 Les unités lexicales de Framenet

Les entrées de Framenet décrivant les unités lexicales contiennent :

- le cadre auquel appartient l'unité ;

[34]Les suffixes `.v` permettent de noter que ces unités sont des verbes.

- une définition ;
- la *réalisation syntaxique* des éléments du cadre, basée sur la fréquence de chaque réalisation constatée dans un corpus (une réalisation se décrit en deux parties : le type de syntagme[35] et les fonctions grammaticales[36]) ;
- les *patrons de valence* du cadre, basés sur le nombre d'occurrences de chaque patron.

Par exemple, l'entrée correspondant à l'unité lexicale « awake » (« réveiller », « se réveiller ») de Framenet contient (au 16/07/2009) :

Cadre :
`waking_up`

Définitions:
`COD: stop sleeping`

Réalisations syntaxiques:

Élément de cadres	Nombre d'éléments annotés	Réalisations	
`Sleep_state`	30	`INI.-- (27)` `PP[from].Dep (3)`	implicite[37] préposition « from » dépendant du verbe
`Sleeper`	30	`NP.Ext (30)`	syntagme nominal extérieur

Patrons de valence:

Élément de cadres	Patrons		
`30 TOTAL`	`Sleep_state`	Sleeper	
27	`INI.--`	`NP.Ext`	« John awakes »
3	`PP[from].Dep`	`NP.Ext`	« John awakes from hibernation »

2.2.4 Modélisation partielle des collocations

Framenet permet de représenter des liens de causalité entre concepts (ce qui peut correspondre à des collocations), mais ne permet pas de représenter des relations syntagmatiques en général.

[35]Par exemple : `NP` (syntagme nominal), `PP` (syntagme prépositionnel), `VPing` (syntagme verbal gérondif), etc.

[36]Par exemple : `Ext` (argument externe au syntagme), `Dep` (dépendant) pour tous les types de syntagmes, `Obj` (objet) pour les syntagmes verbaux, etc.

[37]Nous sommes dans le cas d'une « instanciation nulle indéfinie » (INI), c'est-à-dire que l'élément (ici l'état de sommeil) n'a pas de réalisation propre, et qu'il est l'objet absent d'un verbe pouvant être transitif : si « awake » est employé le plus souvent de manière intransitive (« se réveiller »), il peut l'être aussi de manière transitive, comme dans « I awoke her » (« je l'ai réveillée »). La valeur « `--` » correspond au fait que la fonction grammaticale n'est bien sûr pas définie dans le cas d'une réalisation implicite.

2.3 Classes de Levin

Il existe d'autres représentations du lexique basées sur le regroupement des mots qui partagent certaines propriétés. C'est le cas des *classes de Levin*. Notons tout d'abord qu'on s'intéresse ici à une partie seulement du lexique : les verbes.

2.3.1 Classes basées sur la syntaxe et la sémantique

Dans (Levin, 1993) sont d'abord définies des *alternances de diathèse* (« diathesis alternations »), qui listent les changements possibles d'organisation syntaxique des arguments sémantiques du verbe. Par exemple, le premier couple d'alternatives de diathèse données dans est « The butcher cuts the meat » (le boucher coupe la viande)/« The meat cuts easily » (la viande se coupe facilement)[38]. Cela se rapproche des études traditionnelles sur la construction syntaxique visant à décrire le régime d'emploi des termes.

Ensuite, Levin présente les classes de verbes « *pertinentes syntaxiquement, cohérentes sémantiquement* » : les mots d'un groupe ont un lien sémantique, mais sont d'abord regroupés selon leur comportement lors des modifications syntaxiques. Ce type de regroupement est une différence fondamentale avec Framenet, où les termes sont regroupés selon le concept qu'ils évoquent.

Par exemple, le groupe des « verbes poison » regroupe :

```
asphyxiate, crucify, drown, electrocute, garrotte, knife,
poison, shoot, smother, stab, strangle, suffocate.
```

Par comparaison, les cadres correspondant dans Framenet (qui ne contiennent pas que des verbes) sont :

- « Death » (mort) :

  ```
  asphyxiate.v, croak it.v, croak.v, death.n, decease.v,
  demise.n, die.v, drown.v, expire.v, kick the bucket.v,
  pass away.v, perish.v, starvation.n, starve.v,
  suffocate.v, suffocation.n, terminator.n ;
  ```

- « Killing » (tuer), qui contient 67 unités lexicales ;
- « Cause_harm » (faire du mal), qui contient 66 unités lexicales.

2.3.2 Pas de modélisation des collocations

Les classes de Levin ne permettent de relier des termes qu'à partir de leur appartenance à un même groupe, on ne peut donc en déduire que des liens paradigmatiques, et pas de liens syntagmatiques comme les collocations.

[38]Il s'agit d'un emploi *ergatif* : le sujet dans le cas intransitif correspond à l'objet direct du verbe dans le cas transitif ; par exemple en français « il casse la branche en deux » (cas transitif), «la branche casse facilement » (cas intransitif).

2.4 Lexique-Grammaire (tables du LADL)

Le terme de *lexique-grammaire* à été introduit par Maurice Gross dans les années 1970. Il consiste en la description des propriétés syntaxiques et sémantiques des prédicats de la langue (verbes, noms, adjectifs, adverbes).

2.4.1 Tables et construction syntaxique

Le lexique-grammaire (Gross, 1990) est d'abord divisé en groupes de tables associés à des catégories syntaxiques (noms, verbes support, etc.), eux-mêmes divisés en tables, correspondant à des constructions syntaxiques particulières. Chaque table comporte tous les mots possibles pour la construction associée, qui sera leur *cadre syntaxique de base.*

On appelle couramment ces structures « tables du LADL[39] ». Comme son nom le laisse supposer, une table décrit la correspondance entre les mots contenus (les lignes) et des propriétés syntaxiques et sémantiques de ceux-ci (les colonnes) : à l'intersection d'un mot et d'une propriété, se trouve un signe + ou − selon que la propriété est vraie ou non pour le nom. Ces propriétés peuvent concerner le prédicat ou bien ses arguments.

2.4.2 Modélisation partielle des collocations

Les seules données de lexique-grammaire pouvant modéliser des collocations sont celles concernant les verbes support ; il n'y a pas d'autre information concernant des liens syntagmatiques.

2.5 Lexical Conceptual Structures

Les *structures lexicales conceptuelles* (LCS), proposées par Bonnie Dorr, à partir des travaux de Jackendoff, cherchent à représenter les propriétés des termes du lexique, et surtout des prédicats, en les regroupant, comme dans les classes de Levin, selon leur propriétés syntaxiques et sémantiques.

2.5.1 Arbres issus des classes de Levin

Une LCS est un arbre (« un graphe orienté, avec une racine »). À chaque nœud sont associées plusieurs informations (Traum & Habash, 2000):

- un *type* (événement, état, chemin, manière, propriété, chose) ;
- une *primitive* : il peut s'agir d'une primitive structurelle (de base, comme `cause`, `go`, `be`, `to`, etc. pour l'anglais) ou non ;
- un *champ* , qui peut indiquer le lieu, la possession, l'identification, etc.

La description d'une LCS associée à un prédicat se réalise en introduisant des fils correspondant aux arguments du prédicat. Décrire un syntagme verbal se fait grâce à une *LCS complexe*, qui combine plusieurs LCS, les LCS associées aux arguments du verbe venant se placer dans la place qui leur est réservée dans la LCS racine associée au verbe.

[39] Laboratoire d'Automatique Documentaire et Linguistique (le laboratoire de Maurice Gross).

Par exemple, pour le verbe « close », la structure *45.4.b* (qui correspond à un cas d'emploi de la classe de Levin *45.4*) de la base renvoie à cette structure :

```
(cause    (* thing 1)
          (go ident (* thing 2)
                    (toward ident  (thing 2)
                                   (at ident (thing 2)
                                             (close+ed 9)))))
```

(les chiffres présents servent à coder le « rôle thématique », en l'occurrence : 1 pour l'agent, 2 pour le thème, 9 pour le prédicat).

Dans (Dorr & Kastova, 1998), est décrite la LCS correspond à la phrase « He closed the door » (il a fermé la porte) :

```
(cause    [Thing HE]
          (go ident [Thing DOOR]
                    (toward ident  [Thing DOOR]
                                   (at ident [Thing DOOR]
                                             [Property CLOSED]
                                                       ))))
```

2.5.2 Pas de modélisation des collocations

Comme pour les classes de Levin (dont elles sont issues), on ne peut déduire des LCS que des liens paradigmatiques entre éléments d'une même classe. Elles ne permettent pas de stocker des informations comme celles des collocations.

2.6 Verbnet

2.6.1 PropBank

PropBank est une ressource lexicale basée sur le corpus *Penn Treebank*, dont elle annote les propositions verbales selon leurs structures prédicats-arguments, en affectant des étiquettes de rôle sémantique aux constituants syntaxiques de ces propositions (Kingsbury & Palmer, 2002). Ce projet est mené par Martha Palmer à l'université du Colorado.

Par exemple, pour le verbe `count`, quatre cadres sont définis dans Propbank, correspondant à quatre sens différents du mot :

- pour le sens « énumérer » :

```
Arg0:counter
Arg1:thing counted
```

- pour le sens « inclure » :

```
Arg0:agent, entity causing some grouping
Arg1:theme, thing being included in some group
Arg2:group
```

- pour le sens « reposer sur » :

```
Arg0:depender
Arg1:depended on
Arg2:secondary predication on arg1
```

- pour le sens « importer » :

```
Arg1:thing
```

Les phrases extraites de PropBank sont annotées pour indiquer les relation et leurs arguments. Par exemple, la première phrase de l'entrée `count` (dans le sens « inclure ») de PropBank est annotée ainsi :

```
[ARG0 *-1] [ARGM-NEG Not] [rel counting] [ARG1 the extraordinary charge]
, the company said 0 it would have had a net loss of $ 3.1
million *U* , or seven cents a share.
```
[40]

2.6.2 Verbnet

La base *Verbnet* (Schuler, 2003) est présentée sur le site Web de Martha Palmer[41] comme « le plus grand des lexiques en ligne actuellement disponibles pour l'anglais ». Elle reprend notamment l'idée de PropBank de décrire les sens des verbes en utilisant des rôles, mais elle effectue une généralisation pour regrouper les verbes par classes.

Verbnet contient des classes de verbes anglais organisées, comme dans les classes de Levin, de manière hiérarchique (une classe peut avoir des sous-classes, qui peuvent elles-mêmes en avoir, etc.) ; à ces classes sont associées des informations syntaxiques et sémantiques.

L'organisation des classes de Verbnet a été construite à partir des classes de Levin, puis à l'aide d'autres études sur la classification des verbes anglais (Korhonen & Briscoe, 2004) (Kipper et al., 2006).

Une entrée de Verbnet représente une classe et contient :

- la liste des arguments du verbe (et leurs *restrictions de sélection*) : à chaque argument du verbe est associé un rôle sémantique (acteur, agent, instrument, récepteur, etc.) ; les restrictions peuvent indiquer par exemple qu'on ne peut sélectionner que quelque chose qui soit humain, abstrait, scalaire, solide, etc.) ;
- les *cadres syntaxiques* qui contiennent chacun :
  - les réalisations syntaxiques possibles du verbe, comme `Agent V Patient` ;
  - les *prédicats sémantiques*, qui modélisent le sens du verbe lorsqu'il est employé dans ce cadre ;
  - un exemple d'emploi pour la réalisation proposée dans le cadre.

Par exemple, l'entrée « focus-87.1 » de Verbnet contient :

[40] Un argument implicite est mis en évidence à l'annotation ; c'est le cas, dans cet exemple, de l'agent (argument zéro) en début de phrase.

[41] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html (consulté le 16/07/2009)

Membres :
Brood
Center
Concentrate
Converge
Focus
Marinate

Rôles :
Experiencer [+animate | +organization] (entre crochets : les restrictions)
Theme

Cadres
NP V PP.theme
Exemple : "We focused on it."
Réalisation syntaxique : Experiencer V {on} Theme <-sentential>
Prédicats Sémantiques :
concentrate(during(E), Experiencer, Theme) cause(E, Theme)

NP v PP.theme S_ING
Exemple : "We focused on reading the book."
Réalisation syntaxique : Experiencer V {on} Theme <+sc_ing>
Prédicats Sémantiques :
concentrate(during(E), Experiencer, Theme) cause(E, Theme)

NP v PP.theme what S
Exemple : "We focused on what he wanted."
Réalisation syntaxique : Experiencer V {on} Theme <+what_extract>
Prédicats Sémantiques :
concentrate(during(E), Experiencer, Theme) cause(E, Theme)

NP V PP.theme what S_INF
Exemple : "We focused on what to do."
Réalisation syntaxique : Experiencer V {on} Theme <+what_inf>
Prédicats Sémantiques :
concentrate(during(E), Experiencer, Theme) cause(E, Theme)

2.6.3 Pas de modélisation des collocations

Verbnet est encore un autre lexique où les informations concernant des liens entre lexies correspondent uniquement à des relations paradigmatiques.

3 Comparaison des représentations

Nous avons décrit plusieurs représentations du lexique, d'approches assez variées. Laquelle choisir comme cadre pour des travaux sur les collocations ?

Nous allons tout d'abord introduire les éléments de comparaison entre représentations qui nous semblent pertinents, divisés en deux groupes : les éléments généraux, et les éléments essentiels pour la gestion des collocations.

	Lexique	*Lexie*		*Liens entre lexies*		*Base de données*	
	Type	*Sémantique*	*Syntaxe*	*Paradigmatiques*	*Syntagmatiques*	*Taille*	*Langues*
DiCo (FLS)	Énumératif	Définition	Schéma de régime	FLS paradigmatiques	FLS syntagmatiques	1125 lexies, 26093 liens (version 11/08/08)	Français
WordNet	Énumératif	Regroupement par sens (synsets)		Dans les synsets + liens entre eux		*WordNet 3.0* : 117659 synsets, 206941 liens (paires sens-mots)	Anglais (*WordNet*) ; allemand, hollandais , italien, espagnol, français, tchèque, estonien (*EuroWordNet*)
ComLex	Énumératif		Propriétés syntaxiques			*COMLEX Syntax* :38000 entrées	Anglais
EDR	Énumératif	Définition et autres infos sémantiques			Co-occurrences non typées dans dict.	540 000 concepts . *Japonais* : 270 000 mots, 900 000 co-occurrences ; *Anglais* : 190 000 mots, 460 000 co-occurrences	Japonais, anglais
HowNet	Énumératif	Définition par traits sémantiques		Relations entre concepts		*Chinois* : 93 247 mots, 107 698 concepts ; *Anglais* : 86 846 mots, 107 511 concepts	Chinois, anglais
Lexique génératif	Génératif		Structure argumentale		Rôles télique et agentif ~ support		
Framenet	Regroupement par concepts	Appartenance à un cadre sémantique	Réalisation syntaxique, patron de valence	Liens entre éléments du groupe	Liens de causalité entre concepts	938 cadres définis, 5076 unités lexicales complètes	Anglais
Classes de Levin	Regroupement par syntaxe	Appartenance à une classe		Liens entre éléments du groupe		3000 verbes	Anglais
Lexique-Grammaire	Regroupement par syntaxe	Propriétés sémantiques (colonnes de la table)	Propriétés syntaxiques (colonnes de la table)	Liens entre éléments du groupe	verbes support	6000 verbes (31000 entrées dans 81 tables), 5000 noms, 6000 adjectifs	Français
LCS	Regroupement par syntaxe		Structure argumentale	Liens entre éléments du groupe		*LCS Database* : 11000 entrées verbales	Anglais
Verbnet	Regroupement par syntaxe	Prédicats sémantiques dans cadres syntaxiques	Cadres syntaxiques	Liens entre éléments du groupe		*Extended VerbNet* : 274 classes, 3769 lemmes, 5257 sens de verbes	Anglais

Tableau 3: Récapitulatif des différentes représentations existantes du lexique

3.1 Éléments de comparaison

La description d'un lexique se fait sur différents points :

- le *type* du lexique : est-il énumératif ? génératif ? avec des entrées regroupées d'après leurs propriétés communes ?
- La description des *lexies* :
  - *Syntaxe* : y a-t-il une représentation de la structure argumentale[42] des prédicats ?
  - *Sémantique* : y a-t-il une représentation de la signification de la lexie ?
- La description des *liens entre lexies* :
  - Liens paradigmatiques ?
  - Liens syntagmatiques ? (dont verbes support).
- Les *données existantes* : existe-t-il une mise en œuvre de la représentation ? Comment les données sont-elles construites ?

3.2 Récapitulatif

Le tableau 3 récapitule toutes les propriétés des différentes modélisations du lexique proposées, en fonction des différents éléments de comparaison que nous avons définis (on s'intéresse ici aux ressources informatisées). L'élément de comparaison déterminant pour nous est bien sûr celui qui concerne les liens syntagmatiques entre lexies (les collocations, les verbes support).

Nous pouvons constater que peu de modélisations existantes du lexique permettent de représenter les liens syntagmatiques entre les lexies. Lorsque cela est possible, généralement soit cela est limité à un sous-ensemble comme les verbes support (Lexique Génératif, Lexique-Grammaire), soit cela couvre une ensemble plus grand (le dictionnaire de co-occurrences de EDR).

La seule modélisation qui permette de représenter les collocations de manière simple et correcte est celle des Fonctions lexico-sémantiques. C'est donc celle que nous retenons pour nos travaux.

4 Fonctions lexico-sémantiques et théorie sens-texte

La lexicologie explicative et combinatoire est, à travers les FLS, la représentation du lexique de la *théorie sens-texte* (Žolkovskij & Mel'čuk, 1967) (Mel'čuk, 1997). Nous allons tout d'abord présenter plus en détail cette théorie, puis nous nous intéresserons à ses applications.

Développée par Igor Mel'čuk, la théorie sens-texte (TST) vise à établir la correspondance entre le sens et le texte, à décrire les mécanismes qui permettent de générer les textes correspondant à un sens donné, plus précisément de passer de représentations sémantiques à des représentations phoniques (paraphrases). Il y a 7 niveaux de modélisation dans la TST, basés sur les 4 niveaux habituels de description d'un énoncé :

[42]On rencontre aussi « structure argumentaire » dans la littérature.

- *représentation sémantique* ;
- *représentations syntaxiques* profonde et de surface ;
- *représentations morphologiques* profonde et de surface ;
- *représentations phonétiques* profonde et de surface.

Nous allons présenter ici les différentes étapes qui permettent de passer d'une représentation à une autre, en prenant pour exemple le sens « *Marc aime sa femme d'une manière très intense* ».

4.1 Représentation sémantique

La *représentation sémantique* est un réseau de sens, où les arcs représentent les relations prédicat-argument. Le sens de l'énoncé y est donc décomposé en unités de sens.

On y distingue le *thème* de la phrase (de quoi parle-t-on ?) et le *rhème* (qu'en dit-on ?), tel que cela est illustré par la figure 2[43].

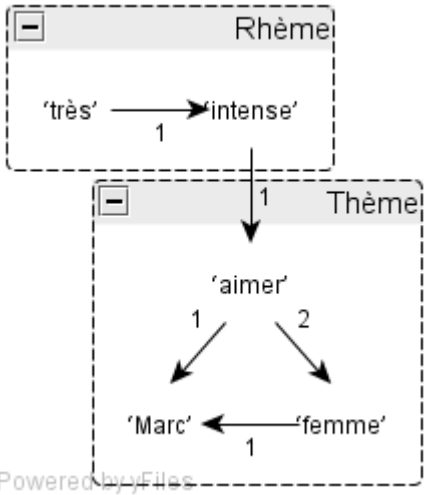


Figure 2: Représentation sémantique de « Marc aime sa femme d'une manière très intense » (Théorie Sens-Texte)

4.2 Représentations syntaxiques

Les deux niveaux de représentation syntaxique sont des *arbres* de *lexies*. La structure de l'arbre ne permet pas de déduire directement l'ordre final (dans la phrase) des lexèmes instanciant ces lexies.

[43]Sur la représentation sémantique, `'aimer'` dénote le prédicat sémantique qui a le sens de « aimer ».

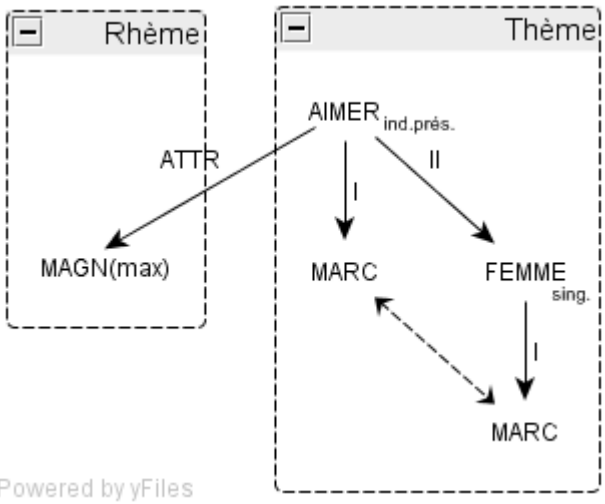


Figure 3: Représentation syntaxique profonde de « Marc aime sa femme d'une manière très intense » (Théorie Sens-Texte)

La *représentation syntaxique profonde* produit, à partir des sens présents dans la représentation sémantique :

- des lexies *pleines* (qui ont un sens) dans le thème ;
- des fonctions lexico-sémantiques, dans le rhème ;

Dans le processus qui permet de passer d'une représentation sémantique à une représentation syntaxique profonde, le passage d'un graphe à un arbre se fait en dédoublant certains nœuds (ainsi, dans l'exemple, le nœud `'marc'` a donné deux nœuds, l'un comme argument de `AIMER`, l'autre comme argument de `FEMME`); certains autres sont regroupés (ainsi les nœuds `'très'` et `'intense'` ont-ils donné un seul nœud $\texttt{Magn}_{\texttt{max}}$) .

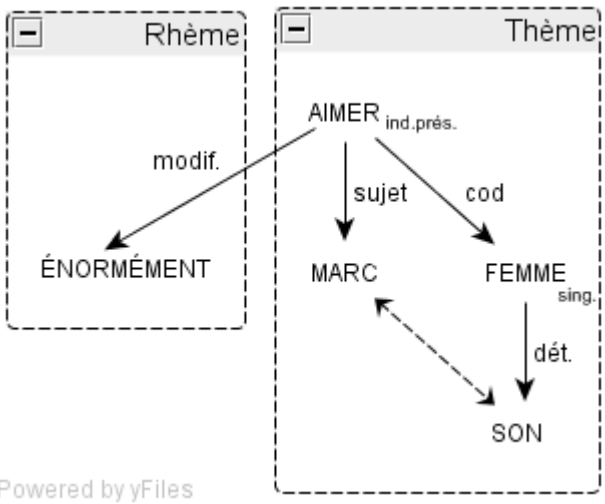


Figure 4: Représentation syntaxique de surface de « Marc aime sa femme d'une manière très intense » (Théorie Sens-Texte)

La représentation syntaxique de surface détermine les lexies qui apparaîtront dans la phrase finale, ainsi que le rôle grammatical qu'elles porteront. Les lexies vides y apparaissent, et notamment les pronoms et les adjectifs possessifs qui vont permettre d'éviter les répétitions en faisant référence à une lexie déjà présente.

C'est aussi là qu'apparaissent les valeurs des fonctions lexico-sémantiques. Dans l'exemple, le nœud Magn_{max} a été remplacé par une valeur $\text{Magn}_{\text{max}}(\text{aimer})$, « énormément », mais on aurait également pu choisir « à la folie » ou « de tout son cœur ».

4.3 Représentations morphologiques

La *représentation morphologique profonde* est une chaîne ordonnée de lexies marquées morphologiquement, construite à partir de la structure syntaxique de surface, et d'informations sur les lexies (par exemple que « femme » est féminin).

La *représentation morphologique de surface* est ensuite obtenue en transformant les marques morphologiques des lexies en morphèmes lexicaux et grammaticaux.

```
MARC AIMERind,pres,3,sing ÉNORMÉMENT SONfém,sing FEMMEsing
```
(profonde)

```
{MARC} {AIMER},{IND.PRÉS},{3SG} {ÉNORMÉMENT} {SON},{SG.FÉM}
{FEMME},{SG}
```
(de surface)

Figure 5: Représentations morphologiques de « Marc aime sa femme d'une manière très intense » (Théorie Sens-Texte)

Reste ensuite à traduire les morphèmes en phonèmes pour obtenir les représentations phonétiques ; nous ne développerons pas ce point ici.

4.4 Bilan sur les fonctions lexico-sémantiques dans la TST

Les fonctions lexico-sémantiques apparaissent donc dans la représentation syntaxique profonde. Elles sont générées à partir du sens présent dans la représentation sémantique, et leurs valeurs (pour la lexie avec lesquelles elles sont reliées) servent à obtenir les lexies de la représentation syntaxique de surface qui exprimeront ce sens.

En plus de permettre un traitement très satisfaisant du phénomène collocatif, les FLS s'inscrivent donc dans une théorie linguistique complète.

Plusieurs décennies de travail sur cette théorie ont permis d'affiner la théorie et d'en faire un outil puissant. En particulier, la réalisation des DEC a permis de lister les FLS standard et d'introduire les moyens de décrire des FLS irrégulières dans le cas où le sens de la collocation ne peut être décrit par une fonction standard. (FLS complexes, configuration de FLS, FLS non standard).

De même, la réalisation du DiCo, une version automatisée du DEC, puis de la base Papillon (Mangeot et al., 2003)[44] ont montré que, au prix de quelques simplifications, les FLS conviennent tout à fait à un traitement automatique dans des bases de données lexicales.

Cela valide par l'expérience l'hypothèse que nous avons faite lors du choix de la modélisation, c'est-à-dire que les FLS sont tout à fait adaptées à un traitement automatique des collocations.

Conclusion

De nombreux modèles de lexique existent et ils ont, pour la plupart, été mis en œuvre dans des bases de données lexicales. Ils décrivent les lexies à travers leurs propriétés syntaxiques ou sémantiques. La majorité des modèles lient les lexies dont les propriétés sont similaires ; seuls deux (fonctions lexico-sémantiques et EDR) s'intéressent précisément aux co-occurrences de termes.

Nous avons choisi, après étude, de placer nos travaux dans le cadre des fonctions lexico-sémantiques de la théorie sens-texte, qui constituent le seul modèle vraiment adapté à la description des collocations (en particulier de leur sens). Nous souhaitons donc construire un système permettant d'obtenir les valeurs des fonctions lexico-sémantiques syntagmatiques pour les entrées du lexique, c'est-à-dire des collocations typées. Il nous faut donc, avant de proposer notre propre approche, étudier les approches existantes visant à obtenir des occurrences de collocations (ou de phénomènes similaires).

[44] Ainsi que d'autres projets qui seront développés dans l'état de l'art au chapitre suivant.

Chapitre III - État de l'art de la collecte de collocations

Nous cherchons à obtenir des collocations en utilisant des processus automatiques, à partir de ressources existantes (informatiques ou humaines). La question principale est donc de savoir comment trouver des collocations. Et, avant tout, il faut savoir où les trouver.

Les corpus de documents comportent beaucoup de collocations (le phénomène est fréquent). On peut donc essayer d'en faire l'extraction, de manière statistique, ou, si on dispose d'une base d'exemples et de contre-exemples, d'appliquer des processus d'apprentissage automatique. Nous présenterons dans ce chapitre des travaux concernant l'extraction et l'apprentissage automatique de collocations, ou de phénomènes linguistiques proches.

D'autre part, tout locuteur natif d'une langue, connaît ses collocations ; nous nous intéresserons donc également aux moyens de faciliter la contribution des non-spécialistes à une base linguistique modélisant le phénomène collocatif.

1 Collocations et « expressions multi-mots »

La plupart des travaux qui traitent de l'extraction de collocations utilisent une définition plus large que la nôtre (expressions semi-figées) ; ils recherchent en fait plutôt des « *multi-word expressions* » (MWE), ou *expressions multi-mots*. Les MWE regroupent toutes les expressions qui comportent des restrictions sur la syntaxe ou sur la composition du sens : locutions, noms composés, noms propres, collocations lexicales, verbes support, collocations grammaticales (verbes à particules), etc. (Sag et al., 2002, Nerima et al., 2006). Les collocations sont un cas particulier de MWE, et il est donc intéressant de se pencher dans cet état de l'art sur les travaux concernant les MWE. En particulier, les collocations et les MWE partagent le fait d'apparaître plus souvent que par hasard, une propriété exploitée par la plupart des travaux présentés ici.

2 Concordances

Les tâches de lexicographie (création de dictionnaires) nécessitent de faire la description la plus exhaustive possible des différents sens d'un mot et des expressions dans lesquelles il intervient. Pour faciliter cette énumération, certains lexicographes utilisent des concordanciers.

Un *concordancier* est un outil qui permet d'extraire à partir d'un corpus et d'une expression donnée (simple chaîne de caractères ou expression régulière, avec éventuellement la

spécification de propriétés telles que la partie du discours d'un mot) l'ensemble des passages du corpus qui contiennent cette expression, ce qui permet d'obtenir un grand nombre d'emplois différents de l'entrée à définir.

```
   the affects of acid rain. But I think this
YOUR WINDOW: TROPICAL RAIN FORESTS, EMERALD
 Sunday night's heavy rain, was The Moody Blues.
near Lyndhurst. Heavy rain in early April
 CAPTIONS [/c] RECENT RAIN produced perfect
  starts and the real rain begins. Here we are
Dirty Mind and Purple Rain, and three from
happening to tropical rain forests because it
    hampered by heavy rain and low cloud cover
        by torrential rain during the last few
```

Tableau 4: 10 premiers résultats du concordancier du « Collins Wordbanks Online English » pour la requête « JJ+rain »

Collocat	Effectif	Effectif joint	T-score
the	2313407	1604	16.802643
acid	875	150	12.218696
and	1129483	740	10.492536
heavy	4414	113	10.463031
forest	2896	108	10.280152
in	765730	534	9.772361
wind	2598	78	8.713371
snow	1869	57	7.450203
rain	2327	52	7.081230
down	42206	75	6.698852

Tableau 5: 10 premiers termes en co-occurrence avec « rain » dans le « Collins WordBanks Online English » d'après le t-score

Le projet *COBUILD* mené par l'éditeur *Collins* et l'Université de Birmingham, a conduit à la création de *The Bank of English* (Järvinen, 1994), un corpus qui compte 524 millions de mots en juillet 2009[45], dont l'éditeur se sert pour la réalisation de ses dictionnaires. Une application de concordancier sur une partie réduite de cette base est disponible sur le site Internet de Collins[46], permettant notamment la restriction sur la partie du discours des éléments. Ainsi, dans l'exemple donné dans le tableau 4, sont renvoyés les extraits correspondant à l'expression « `JJ+rain` » qui correspond à un adjectif (`JJ`) suivi (+) du mot `rain`.
L'utilisation de l'informatique dans le traitement des langues a permis un traitement statistique des données, en analysant les corpus de manière à associer aux lexèmes qui y apparaissent un certain nombre de mesures statistiques, comme dans le logiciel DEREDEC (Plante, 1983).

[45]http://www.collins.co.uk/books.aspx?group=153
[46]http://www.collins.co.uk/Corpus/CorpusSearch.aspx

On peut également effectuer un traitement statistique des co-occurrences, pour évaluer si deux termes sont employés fréquemment ensemble ou non.
La page Web de Collins permet également d'obtenir les « *collocats* » d'un terme donné (c'est-à-dire les termes avec qui il apparaît), classés d'après des mesures sur les co-occurrences telles que l'*information mutuelle* ou le *t-score*. Un exemple est présenté dans le tableau 5.

Certains concordanciers sont bilingues : basés sur des corpus bilingues alignés par phrases (le plus souvent de manière automatique, il peut donc y avoir des erreurs), ils renvoient, pour chaque phrase dans laquelle l'expression recherchée apparaît, la (les) phrase(s) qui en sont la traduction. On peut citer notamment *TransSearch* (Macklovitch et al., 2000) développé au RALI (Université de Montréal), ou *ConcQuest*, développé à l'Université de Grenoble par Olivier Kraif (Kraif, 2008). Ils permettent de découvrir, pour une expression donnée, les traductions des phrases où elle apparaît, et donc de trouver, sinon une traduction, au moins une expression équivalente dans l'autre langue.

C'est également ainsi que fonctionnent les mémoires de traductions.

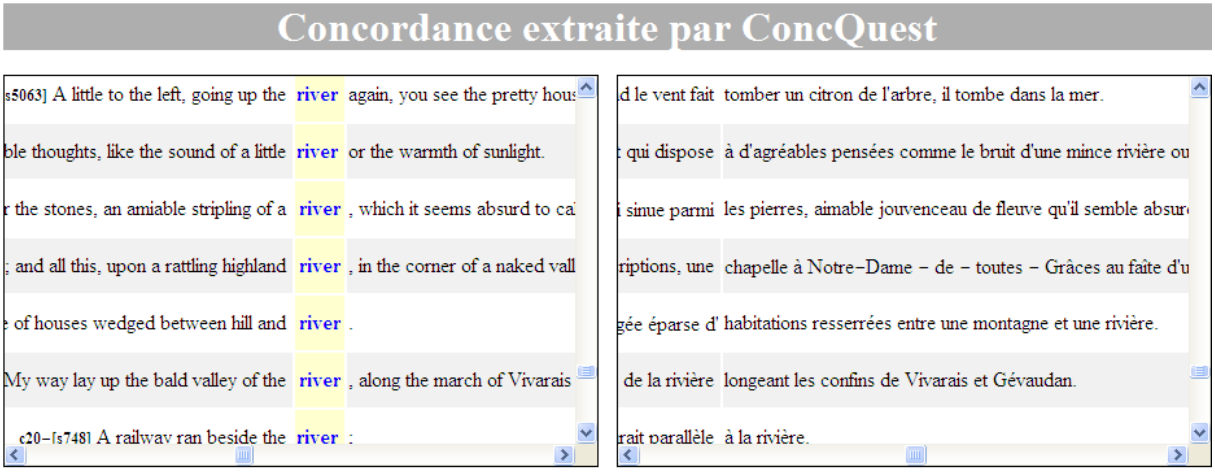


Figure 6: Sortie bilingue du concordancier ConcQuest pour la requête « river »

Si les concordanciers apportent une aide non négligeable aux lexicographes, celle-ci reste limitée, en particulier pour le problème des collocations. S'ils permettent de dénombrer les occurrences des termes, et avec elles les collocations dans lesquelles ils apparaissent, il est sans doute possible de faire mieux, en utilisant la puissance de calcul des processus automatiques pour tenter de faire émerger les collocations du reste des co-occurrences, en calculant, comme cela est fait dans le deuxième exemple de COBUILD présenté plus haut, les termes qui ont le plus de chance d'être des collocatifs d'une base donnée.

3 Extraction

La première piste pour la collecte de collocations est l'extraction à partir de données existantes, de corpus. Il faut pour cela disposer de grandes sources de documents, où l'on

extrait les occurrences des mots recherchés. L'idée sur laquelle repose l'extraction des collocations (et des MWE en général) est que les expressions de ce genre se rencontrent plus souvent que par hasard ; on peut donc envisager de calculer des mesures de collocabilité à partir de nombres d'occurrence et de co-occurrence de termes.

On peut trouver dans la thèse de Seretan (2008) une étude détaillée des travaux d'extraction de collocations.

3.1 Hybride statistique-linguistique

Il y a deux façons radicalement différentes de voir l'extraction de données à partir de corpus :

- l'utilisation de *patrons syntaxiques*, l'inconvénient étant qu'il faut définir ces patrons de manière exhaustive ;
- un calcul *statistique* pour classer les candidats selon la probabilité plus ou moins forte qu'ils aient la qualité recherchée ; ce calcul est basé sur la cooccurrence des termes dans un certain contexte.

Utiliser uniquement des patrons syntaxiques (sans calculs statistiques) ne suffit généralement pas : on ne dispose alors pas d'outils permettant de classer les candidats, de distinguer les bons des mauvais. En effet, le fait de satisfaire le patron n'est pas toujours suffisant, plus particulièrement pour les collocations où il y a une contrainte sur le figement sémantique de l'expression.

La statistique seule ne suffit pas non plus, les mots fréquents et vides de sens (déterminants, conjonctions) causant beaucoup de bruit dans les résultats.

C'est pourquoi les processus d'extraction sont généralement *hybrides* : on utilise des données linguistiques pour restreindre ou qualifier la liste de candidats. Souvent, l'intervention linguistique est faite sous la forme d'un pré-traitement qui permet de restreindre les candidats possibles. Dans d'autres cas, on produit des arbres d'analyse linguistique, sur lesquels portent ensuite les calculs statistiques.

3.2 Identifications des candidats

Les travaux d'extraction de collocations, de co-occurrences, de MWE, qui cherchent à classer les candidats, le font en se basant sur la fréquence de co-occurrence des termes. On peut distinguer deux axes dans l'identification des candidats :

- le *contexte* de co-occurrence (dans la même fenêtre, en relation, etc.)
- les *restrictions* sur les propriétés des candidats (par exemple la partie du discours).

3.2.1 Contexte

Quel type de contexte se prête le mieux à une extraction précise de collocations ? En l'absence d'informations sur le lien sémantique entre les termes d'une phrase, le plus intéressant semble l'exploitation des liens syntaxiques (pouvant impliquer un lien sémantique) : par exemple, une relation verbe-sujet ou verbe-objet peut cacher une collocation de type verbe support. De plus, une *résolution d'anaphores* peut être utile, la base et le collocatif n'étant pas nécessairement dans la même proposition ou dans la même phrase (comme dans cet exemple, trouvé sur

Internet : « *tout dépend comment on comprend l'*erreur. *Et tant qu'on n'a pas l'élève sous la main, on ne peut pas trop savoir ce qu'il avait dans le crâne au moment où il* l'*a* faite… »).

Toutefois, et malgré l'amélioration de la situation avec le temps, de telles ressources ne sont pas toujours disponibles, et il faut alors trouver d'autres astuces pour éviter d'avoir des résultats trop bruités.

3.2.2 Restriction

Toutes les relations syntaxiques, établies à la suite d'une analyse, ou présumées par la proximité des termes dans une même fenêtre, ne sont pas directement exploitables pour les collocations. Il faut donc filtrer les candidats possibles pour ne se concentrer que sur les termes pouvant réellement faire partie d'une collocation.

On peut pour cela utiliser un étiquetage des parties du discours des termes de corpus, pour ne considérer ensuite comme pertinentes que les collocations dont le squelette est d'un certain type, comme nom-adjectif, verbe-adverbe, verbe-nom (verbe support), etc.

3.3 Contextes

3.3.1 Fenêtre de mots

La base et son collocatif ne sont pas toujours juxtaposés ; se limiter à ce cas ne permet pas de détecter toutes les collocations existantes. La solution peut être alors de considérer que deux termes sont « en co-occurrence » s'ils apparaissent dans la même fenêtre de *n* mots, c'est-à-dire qu'ils sont à au plus *n* mots l'un de l'autre dans la phrase.

Le système SPIRIT, consacré à la recherche d'information, devait calculer une correspondance entre une requête en langue naturelle et les documents disponibles dans la base. Pour cela, il fallait procéder à l'indexation automatique de concepts (Andreewsky & Fluhr, 1975), en utilisant des informations syntaxiques (notamment sur la place dans la phrase, dans une certaine fenêtre) acquises par apprentissage automatique (Andreewsky et al., 1977). Ce système a d'abord été développé sur le français, adapté à l'espagnol (Andreewsky et al., 1983) et dans d'autres langues, puis à la recherche d'information multilingue (Fluhr et al., 1997) ; il est à la base des logiciels commerciaux édités par la société *Technologies*[47].

Le système proposé par Church & Hanks (1989) utilise à l'origine une fenêtre de 5 mots pour la co-occurrence et une mesure de classement ; les développements suivants (Church & Hanks, 1990) introduiront un pré-étiquetage grammatical (pour distinguer par exemple les occurrences de « to » comme préposition de celles comme marque de l'infinitif d'un verbe).

Le système Xtract (Smadja, 1993) procède à l'extraction. de « collocations » (pour lui, simplement des co-occurrences « arbitraires et récurrentes », de longueur variable, pas forcément contiguës), en utilisant une fenêtre de 5 mots. La première étape est l'extraction de bigrammes (tous les couples en co-occurrence dans la fenêtre de 5 mots), filtrés statistiquement avec une mesure d'association. Ces bigrammes sont ensuite étendus pour essayer d'obtenir des collocations plus longues. Les bigrammes sont aussi caractérisés à l'aide d'une analyse linguistique postérieure, qui leur associe un relation syntaxique de surface (telle

[47]http://www.spiritengine.com/

que sujet-verbe, verbe-objet, nom-nom). Smadja indique que l'utilisation de fenêtres de mots est un choix par défaut, en l'absence (à l'époque) d'analyseurs syntaxiques suffisamment robustes.

Malgré le développement des analyseurs syntaxiques, certains travaux sont encore menés sans eux, afin d'établir une méthode d'extraction qui demande le moins possible de ressources linguistiques (et qui soit donc davantage utilisable pour les langues peu dotées). C'est le cas de (Inkpen & Hirst, 2002), extrayant des collocations sous la forme de bigrammes (on peut considérer cela comme avoir une fenêtre de deux mots seulement) du *British National Corpus*, qui n'utilise un étiquetage grammatical que pour retirer les mots vides de sens (ce qui permet par exemple d'extraire un bigramme nom-nom dont les composants sont séparés dans le corpus par une préposition).

3.3.2 Patrons syntaxiques

Un autre type de contexte envisageable est de considérer les collocations comme des n-grammes (séquence de n éléments successifs, ici des mots), l'idée étant que la base et le collocatif sont souvent non détachés, d'autant plus que l'expression est figée. Par exemple, on ne cherche plus un verbe et un adverbe dans la même fenêtre, mais un verbe suivi d'un adverbe.

Ces travaux ont essentiellement été menés dans le domaine de la terminologie. LEXTER (Bourigault, 1994) utilise un corpus, qu'il étiquette grammaticalement, dont il identifie des syntagmes nominaux ensuite décomposés (en tête et expansion) et dont sont extraits des syntagmes plus petits ; les termes candidats sont ensuite organisés en réseau. TERMS (Justeson & Katz, 1995) se base sur les hypothèses relatives aux différences d'emploi entre syntagmes terminologiques et syntagmes non-terminologiques, et le fait que les premiers sont beaucoup plus fréquents dans un texte spécialisé ; il utilise un patron syntaxique complexe afin d'extraire les candidats.

Certains systèmes utilisent des mesures statistiques pour classer les candidats afin de distinguer les syntagmes qui sont effectivement terminologiques. C'est le cas d'ACABIT (Daille, 1994), qui réalise l'étiquetage grammatical d'un corpus, puis « guide linguistiquement » (en utilisant des automates à états finis permettant d'extraire les expressions correspondant aux patrons définis) les modèles statistiques pour les aider à distinguer les noms composés du reste des candidats. Neural (Frantzi & Ananiadou, 1995) utilise à la fois la linguistique (à travers des patrons morpho-syntaxiques) et la statistique (en utilisant des informations sur la fréquence mutuelle) dans un système de réseau de neurones avec rétro-propagation.

Une revue relativement complète (mais déjà assez datée) des systèmes d'extraction automatique de terminologie peut être trouvée dans (Cabré et al., 2001).

3.3.3 Relations de dépendances

Enfin, la solution qu'on peut espérer la meilleure, est de se baser sur une analyse dépendancielle des phrases de corpus, afin de ne conserver que les couples de termes qui ont entre eux une relation du bon type. Certains systèmes destinés à ce type d'extraction intègrent l'analyse de dépendances, d'autres prennent en entrée des corpus déjà analysés.

(Church & Hanks, 1990) fut précurseur dans cette voie, en proposant l'utilisation d'un analyseur syntaxique de surface pour extraire des triplets sujet-verbe-objet d'un corpus et le calcul d'une mesure d'association à partir du nombre d'apparition (des occurrences et des co-occurrences) dans ces triplets. (Lin, 1998) met l'importance sur le type de la relation syntaxique entre les termes, en définissant des triplets de dépendances (2 mots + le type de dépendance) et en adaptant la mesure d'information mutuelle à de tels triplets (celle-ci tient compte de toutes occurrences des termes, en relation ou non).

Ces travaux sur l'extraction de collocations peuvent avoir des applications à grande échelle. C'est le cas du dictionnaire de co-occurrences pour le logiciel *Antidote* de la société *Druide*, dont la réalisation est décrite dans (Charest et al., 2007) : un corpus de plus de 50 millions de mots a été analysé syntaxiquement, puis les relations produites ont été classées grâce à une mesure d'association (ce qui a permis d'éliminer les moins bons candidats) ; les co-occurrences restantes ont ensuite été validées manuellement par des linguistes, ce qui a permis d'obtenir un dictionnaire de 800 000 cooccurrences.

Les travaux sur l'extraction de co-occurrences basés sur l'analyse syntaxique de surface de la phrase ont été utilisés sur de grands corpus, dans le but d'offrir à la demande, de manière gratuite ou payante, des informations sur les co-occurrences dans le corpus d'un mot donné.

Prédicat			Argument			
Catégorie	Lemme	Relation	Catégorie	Lemme	IM ↑ ↓	Fréquence ↑ ↓
N	pluie	mod	A	verglaçant	10.663	31
N	pluie	mod	A	cru	10.663	5
N	pluie	mod	A	diluvien	10.663	226
N	pluie	mod	A	torrentiel	10.543	196
N	pluie	mod	A	battant	9.949	161
N	pluie	mod	A	cévenol	9.682	9
N	pluie	mod	A	bienfaisant	9.328	5
N	pluie	mod	A	acide	9.256	107
N	pluie	mod	A	givrant	9.083	7
N	pluie	mod	A	dru	8.682	12
N	pluie	mod	A	pénétrant	8.162	5
N	pluie	mod	A	glacial	8.162	31
N	pluie	mod	A	abondant	8.071	79
N	pluie	mod	A	intermittent	7.521	6
N	pluie	mod	A	orageux	7.4	7
N	pluie	mod	A	tropical	6.924	38
N	pluie	mod	A	tenace	6.843	10
N	pluie	mod	A	tomber	6.837	10
N	pluie	mod	A	glacer	6.826	18
N	pluie	mod	A	incessant	6.742	25

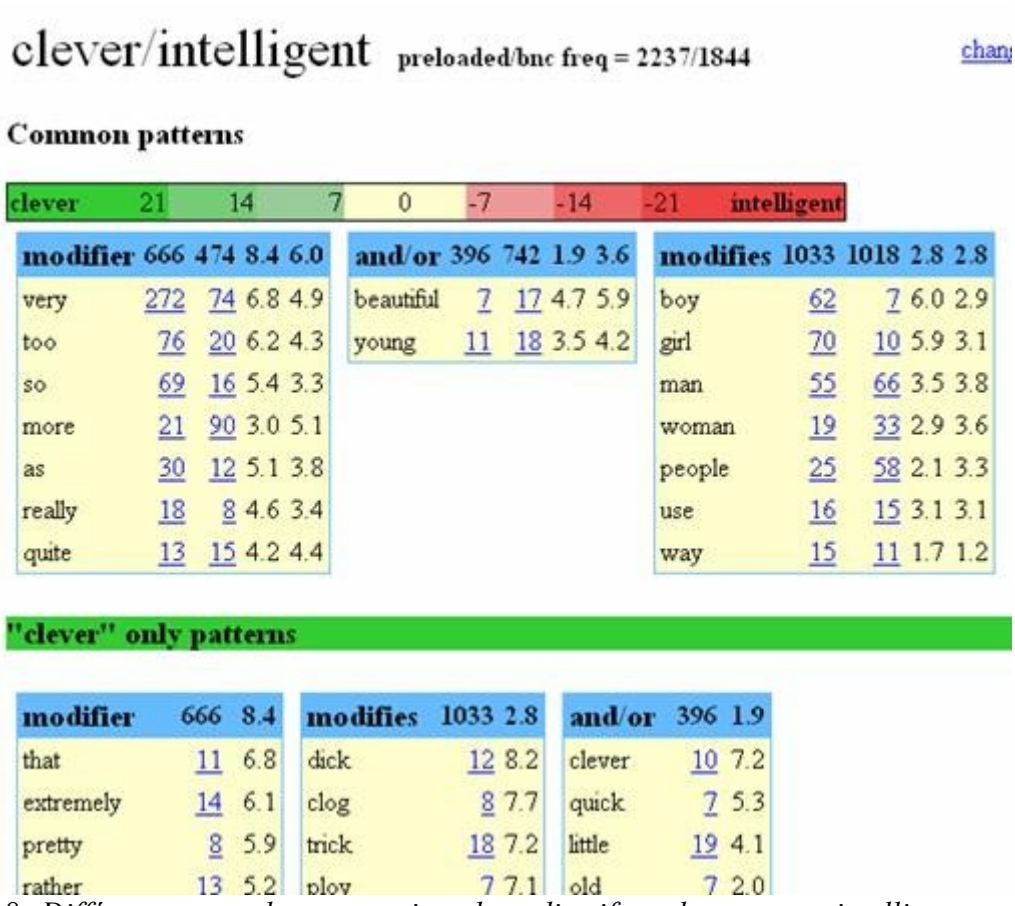
Figure 7: 20 premiers arguments du prédicat « pluie » dans « Les voisins de Le Monde »

Ainsi, pour le projet « *Les voisins de Le Monde* », ce sont 10 ans d'articles du quotidien *Le Monde* qui ont été étiquetés morpho-syntaxiquement par TreeTagger (Schmid, 1994) puis analysés syntaxiquement par Syntex (Bourigault & Fabre, 2000), les relations syntaxiques obtenues étant ensuite transformées en relations prédicat-argument ; enfin a été calculée l'information mutuelle des couples prédicat-argument. Le résultat est librement accessible sur le site Web du projet[48] par requête sur un mot (prédicat ou argument).

[48]http://www.irit.fr:8080/voisinsdelemonde/

La figure 7 présente un exemple de sortie obtenue sur le site de « Les voisins de Le Monde ». On peut observer la présence étonnante de l'argument « cru » pour le prédicat « pluie », sans doute due à une erreur d'analyse ; il se retrouve dans la tête du classement à cause d'un inconvénient connu de l'information mutuelle (IM), qui favorise les couples faisant intervenir des termes rares.

Le *Sketch Engine* (Kilgariff et al., 2004) est un site Internet payant qui, pour un corpus étiqueté grammaticalement passé en entrée (le système peut calculer des relations de dépendance si elles sont absentes), calcule les occurrences des mots et des triplets (mot1, relation, mot2) pour une mesure d'association pour chaque triplet. Le système permet alors de produire le « *word sketch* » (le « croquis » d'un mot) d'un mot passé en requête : sa description grâce aux termes avec lesquels il est en relation, groupés selon le type de la relation et classés d'après la valeur de la mesure d'association.



clever/intelligent preloaded/bnc freq = 2237/1844 chan

Common patterns

clever 21 14 7 0 -7 -14 -21 intelligent

modifier	666	474	8.4	6.0
very	272	74	6.8	4.9
too	76	20	6.2	4.3
so	69	16	5.4	3.3
more	21	90	3.0	5.1
as	30	12	5.1	3.8
really	18	8	4.6	3.4
quite	13	15	4.2	4.4

and/or	396	742	1.9	3.6
beautiful	7	17	4.7	5.9
young	11	18	3.5	4.2

modifies	1033	1018	2.8	2.8
boy	62	7	6.0	2.9
girl	70	10	5.9	3.1
man	55	66	3.5	3.8
woman	19	33	2.9	3.6
people	25	58	2.1	3.3
use	16	15	3.1	3.1
way	15	11	1.7	1.2

"clever" only patterns

modifier	666	8.4
that	11	6.8
extremely	14	6.1
pretty	8	5.9
rather	13	5.2

modifies	1033	2.8
dick	12	8.2
clog	8	7.7
trick	18	7.2
ploy	7	7.1

and/or	396	1.9
clever	10	7.2
quick	7	5.3
little	19	4.1
old	7	2.0

Figure 8: Différence entre les « croquis » des adjectifs « clever » et « intelligent » selon le Word Sketch Engine

Les principales différences avec *Les voisins de Le Monde* sont la possibilité pour l'utilisateur d'utiliser le système sur ses propres corpus, et la fonction « *sketch difference* » (différence entre croquis) qui permet de mettre en évidence les termes en co-occurrence communs à deux termes, et ceux qui n'apparaissent qu'avec l'un ou l'autre, ce qui peut être assez utile pour des termes dont le sens est proche.

La différence entre les croquis illustrée dans la figure 8 donne des informations intéressantes, notamment sur les termes qu'on rencontre avec « clever » mais pas avec « intelligent », comme « that » (surtout utlilisé pour « not that clever », « pas si intelligent »), « dick » ou « clog » (« clever dick » et « clever clogs » sont des locutions qu'on pourrait traduire par « monsieur je-sais-tout »).

Seretan & Wehrli (2006) présentent un système d'extraction de collocations fonctionnant sur différentes langues, et pouvant ainsi traiter (de manière monolingue) des corpus comparables de différentes langues (français, anglais, espagnol, italien). Pour cela, ce système se base sur l'analyseur syntaxique multilingue *Fips* (Wehrli, 2004). Il extrait des candidats vérifiant des patrons syntaxiques, puis les classe selon la mesure LLR. Des évaluations comparatives sont effectuées entre les résultats obtenus par cette méthode (basée sur l'analyse syntaxique) et ceux obtenus à l'aide d'une méthode basée sur une fenêtre de 5 mots, sur les proportions de candidats bien formés grammaticalement, de ceux qui sont des expressions multi-mots (mots composés, collocations, idiomes, entités nommées), et de ceux qui sont des collocations. Dans la quasi-totalité des tests réalisés, la qualité des résultats produits par la méthode basée sur l'analyse syntaxique de Fips est nettement meilleure (Seretan, 2008).

3.4 Étapes

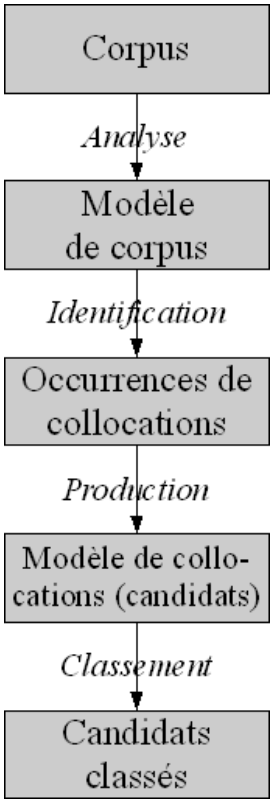


Figure 9: Étapes d'une extraction monolingue de collocations

On peut distinguer, dans l'extraction de collocations, les étapes suivantes :

- l'*identification* des informations nécessaires au processus (ici, les co-occurrences) ;

- l'extraction des informations de co-occurrence (à partir des contextes d'occurrence), qui permet la *production* de candidats ;
- le *classement* des candidats, grâce à une mesure basée sur la co-occurrence.

Il faut rajouter, avant tous ces traitements, une étape d'analyse de corpus pour, à partir de texte brut, obtenir un modèle de corpus (par exemple une analyse syntaxique et/ou de dépendance).

3.5 Choix de la mesure statistique

Dans une méthode d'extraction de collocations visant à classer les candidats, la mesure d'association a un rôle crucial : c'est elle qui doit faire ressortir les bons candidats et écarter les mauvais.

Ces mesures sont basées sur plusieurs types d'occurrences (on se place ici dans le cas où les co-occurrences correspondent à des relations « orientées »[49]) :

- le nombre d'occurrences des termes ;
- pour les co-occurrences de termes (a,b) [50], on peut calculer différentes valeurs, récapitulées dans le tableau 6 (« tableau de contingence »):
  - différents nombres de co-occurrences observés :
    - $O_{xy}=|\{(x,y)|x\in A\wedge y\in B\}|$ (entre a et b) ;
    - $O_{x\bar{y}}=|\{(x,y)|x\in A\wedge y\notin B\}|$ (entre a et un autre terme que b) ;
    - $O_{\bar{x}y}=|\{(x,y)|x\notin A\wedge y\in B\}|$ (entre un autre terme que a et b) ;
    - $O_{\bar{x}\bar{y}}=|\{(x,y)|x\notin A\wedge y\notin B\}|$ (entre un autre terme que a et un autre terme que b) ;
  - d'autres nombres de co-occurrences, liés aux précédents (voir aussi tableau 6) :
    - $T1_x=O_{xy}+O_{x\bar{y}}$ (entre a et un terme quelconque : le nombre d'occurrences de a comme première composante) ;
    - $T1_{\bar{x}}=O_{\bar{x}y}+O_{\bar{x}\bar{y}}$ (entre un autre terme que a et un terme quelconque) ;
    - $T2_Y=O_{xy}+O_{\bar{x}y}$ (entre un terme quelconque et b : le nombre d'occurrences de b comme seconde composante) ;
    - $T2_{\bar{y}}=O_{x\bar{y}}+O_{\bar{x}\bar{y}}$ (entre un terme quelconque et un autre terme que b) ;
    - $N=T1_x+T1_{\bar{x}}=T2_y+T2_{\bar{y}}$
- les nombres attendus de co-occurrences pour les couples de termes (récapitulés dans le tableau 7) :
  - E_{xy} (entre a et b) ; $E_{x\bar{y}}$ (entre a et un autre terme que b) ; $E_{\bar{x}y}$ (entre un autre terme que a et b) ; $E_{\bar{x}\bar{y}}$ (entre un autre terme que a et un autre terme que b) ;

[49]C'est-à-dire qu'on distingue la relation entre a et b de celle entre b et a. C'est le cas notamment des co-occurrences correspondant à une relation de dépendance (verbe-sujet, modifié-modificateur, etc.).

[50]Avec A et B les ensembles respectifs d'occurrences de a et de b, et N le nombre total de co-occurrences.

- ces nombres peuvent être calculés à partir des nombres de co-occurrences observées (voir tableau 6) :

$$E_{ij} = \frac{T1_i \cdot T2_j}{N}$$
$$i \in \{x, \bar{x}\};\ j \in \{y, \bar{y}\}$$

	$y \in B$	$y \notin B$	*total*
$x \in A$	$\boldsymbol{O_{xy}}$	$\boldsymbol{O_{x\bar{y}}}$	$T1_x$
$x \notin A$	$\boldsymbol{O_{\bar{x}y}}$	$\boldsymbol{O_{\bar{x}\bar{y}}}$	$T1_{\bar{x}}$
total	$T2_y$	$T2_{\bar{y}}$	N

Tableau 6: Tableau de contingence pour un couple de termes (a,b)

	$y \in B$	$y \notin B$
$x \in A$	$\boldsymbol{E_{xy}} = \frac{T1_x \cdot T2_y}{N}$	$\boldsymbol{E_{x\bar{y}}} = \frac{T1_x \cdot T2_{\bar{y}}}{N}$
$x \notin A$	$\boldsymbol{E_{\bar{x}y}} = \frac{T1_{\bar{x}} \cdot T2_y}{N}$	$\boldsymbol{E_{\bar{x}\bar{y}}} = \frac{T1_{\bar{x}} \cdot T2_{\bar{x}}}{N}$

Tableau 7: Nombres d'occurrences attendues pour un couple de termes (a,b)

Les mesures statistiques cherchent généralement à mettre en rapport le nombre des co-occurrences observées avec celui qu'on pouvait attendre (au hasard, à partir des nombres d'occurrences).

Un chapitre entier de la thèse de Evert (2004) est consacré aux mesures d'association (pour le calcul de collocations)[51], qu'il divise en deux groupes, selon que la mesure cherche à évaluer :

- l'*importance de l'association* (« signifiance »), parmi lesquelles[52] :
  - le z-score $zscore = \frac{O_{xy} - E_{xy}}{\sqrt{E_{xy}}}$
  - le t-score $tscore = \frac{O_{xy} - E_{xy}}{\sqrt{O_{xy}}}$
  - le χ^2 (chi deux) $\chi^2 = \sum_{i \in \{x, \bar{x}\},\ j \in \{y, \bar{y}\}} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$ [53]

[51] Une page Web, issue de ce chapitre, est maintenue à jour par Evert : http://collocations.de/AM/.

[52] Les mesures ne sont définies que si la co-occurrence est observée : $O_{xy} > 0 \Rightarrow T1_x > 0 \wedge T2_y > 0 \Rightarrow E_{xy} > 0$.

[53] Le chi deux n'est pas défini si l'un des termes est la composante de toutes les collocations (ce qui provoquerait des E_{ii} non nuls).

- le LLR (« log-likelihood ratio », logarithme du rapport de vraisemblance) :

$$G^2 = 2 \sum_{i \in \{x, \bar{x}\},\ j \in \{y, \bar{y}\}} O_{ij} \cdot \log\left(\frac{O_{ij}}{E_{ij}}\right)$$ [54]

- le *degré d'association*, on peut notamment citer :
  - le nombre de co-occurrences[55] : $freq = O_{xy}$
  - l'information mutuelle $MI = \log\left(\frac{O_{xy}}{E_{xy}}\right)$

Les justifications théoriques de ces mesures sont développées longuement dans (Evert, 2004). D'après McEnery et al. (2005) :

- l'information mutuelle « mesure la force des collocations » :
  - plus grande est l'information mutuelle, plus fort est le lien entre les deux objets ;
    - si la valeur est proche de 0, la co-occurrence est sans doute due au hasard ;
    - si la valeur est négative, les deux objets ont tendance à s'éviter.
  - elle a tendance à surévaluer les couples de faible effectif.
- t-score « mesure la confiance avec laquelle on peut affirmer qu'il y a une association » :
  - le t-score est considéré comme significatif s'il est plus grand que 2 ;
  - il dépend de la taille du corpus, et fait ressortir les co-occurrences se produisant le plus fréquemment (alors que MI a tendance à surévaluer les paires contenant des mots peu fréquents).

Pour éviter cette surestimation lors de l'utilisation de l'information mutuelle, certains travaux proposent une pondération à l'aide du nombre de co-occurrences :

- $W = \frac{O_{xy}}{N} \times MI = \frac{O_{xy}}{N} \times \log\left(\frac{O_{xy}}{E_{xy}}\right)$ (Fung & McKeown, 1997).
- $wMI = \log(O_{xy}) \times MI = \log(O_{xy}) \times \log\left(\frac{O_{xy}}{E_{xy}}\right)$ (Seo et al., 2003).

Peut-on prétendre savoir quelle mesure est la « meilleure » ? Plusieurs travaux ont pour but de comparer les différentes mesures d'association, et d'essayer de trouver la meilleure, celle qui sera la plus efficace pour distinguer les candidats corrects des autres. Pour cela, ils évaluent la qualité des résultats produits à partir des mêmes données avec des mesures différentes. On peut citer notamment (Daille, 1994) et (Pecina & Schlesinger, 2006), ce dernier comparant 82 mesures. Cependant, comme le note Seretan (2008), ces évaluations dépendent souvent du corpus utilisé, et sont difficilement généralisables.

[54] Le LLR n'est pas défini si tous les O_{ij} ne sont pas non nuls.

[55] Appelé improprement « fréquence ».

3.6 Filtrage des résultats

3.6.1 Information de substitution

Nous avons décrit les principaux modes d'extraction de collocations, basés le plus souvent sur les nombres de co-occurrences. Intéressons nous maintenant à d'autres moyens de filtrer les résultats, en tentant de distinguer parmi eux :

- ceux qui sont *non-compositionnels* (si le sens de l'expression est compositionnel, c'est-à-dire s'il est la composition du sens de ses composants, ce n'est pas une collocation, ou alors c'est une collocation transparente qui ne pose pas de problème pour les processus de traitement des langues) ;
- ceux dont les composants sont *non substituables* (le sens change si on remplace un composant par un de ses synonymes, ce qui est un signe de figement).

Pearce (2001) utilise un analyseur syntaxique pour extraire les couples de termes qui ont une relation particulière entre eux, puis tente de distinguer les collocations des autres couples, grâce au fait que le figement partiel des collocations rend moins probable la substitution des termes, en utilisant pour cela les relations de synonymie de Wordnet.

3.6.2 Analyse sémantique latente (LSA)

On peut aussi tenter de distinguer les collocations par le calcul de similarités entre celles-ci et leurs composants, en se basant sur des méthodes comme LSA pour obtenir des vecteurs qu'on suppose être reliés aux sens des unités.

3.6.2.1 Décomposition en valeurs singulières (SVD)

L'*analyse sémantique latente* (« latent semantic analysis », *LSA*) est une méthode introduite par Deerwester et al. (1991), basée sur la *décomposition en valeurs singulières* (SVD)[56] qui permet de décomposer une matrice en un produit de 3 matrices :

$$M_{m\times n} = U_{m\times m} . \Sigma_{m\times n} . V^*_{n\times n}$$

où :

- U est une matrice carrée *unitaire* ($U.U^* = U^* . U = I$, U^* étant la matrice adjointe[57] de U et I étant la matrice identité) ;
- $\Sigma = (\sigma_{i,j})_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n}$ est une matrice *diagonale* rectangulaire :
  $\forall (i,j), i\neq j \Rightarrow \sigma_{i,j} = 0$;
  - par convention, les valeurs « diagonales » $\sigma_{i,i}$ sont rangées par ordre décroissant ;
- V est une matrice carrée *unitaire*.

[56] À ne pas confondre avec la *décomposition en vecteurs propres*, qui ne peut être appliquée que sur une matrice carrée.

[57] La matrice adjointe (notée M^*) d'une matrice M est la matrice conjuguée de la matrice transposée de M : $M^* = conjug({}^tM)$. On obtient la matrice conjuguée d'une matrice en remplaçant ses éléments par leurs conjugués (même partie réelle, partie imaginaire opposée).

Les valeurs diagonales de Σ sont appelées *valeurs singulières* de *M*. Elles sont associées à des *vecteurs singuliers* qui sont les colonnes de *U* (vecteurs singuliers *à gauche*) et de *V* (vecteurs singuliers *à droite*).

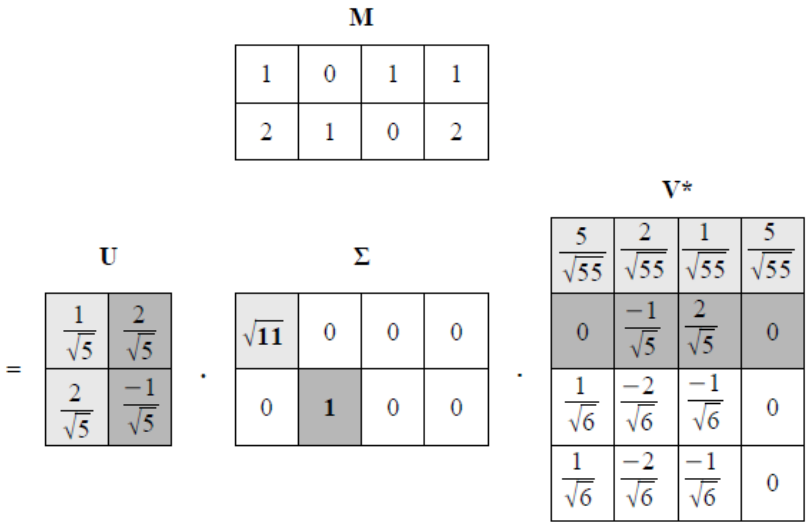


Figure 10: Exemple de décomposition en valeurs singulières d'une matrice

À partir de la décomposition $M = U.\Sigma.V^*$, on peut déduire :

- $M^*.M = (V.\Sigma^*.U^*).(U.\Sigma.V^*) = V.(\Sigma^*.\Sigma).V^*$, donc les colonnes de *V* (vecteurs singuliers à droite) sont les vecteurs propres de $M^*.M$;
- $M.M^* = (U.\Sigma^*.V^*)(V.\Sigma.U^*) = U.(\Sigma^*.\Sigma).U^*$, donc les colonnes de *U* (vecteurs singuliers à gauche) sont les vecteurs propres de $M.M^*$;
- les valeurs diagonales de Σ (valeurs singulières) sont donc les racines carrées des valeurs propres de $M^*.M$ (ou de $M.M^*$).

3.6.2.2 LSA

LSA est généralement utilisé sur des matrices termes-documents, notamment en recherche d'information, pour faire émerger, à un niveau intermédiaire entre termes et documents, un espace de concepts :

- les vecteurs singuliers à gauche (colonnes de U) sont alors les vecteurs décrivant les liens entre les termes et ces concepts;
- les vecteurs singuliers à droite (colonnes de V) sont alors les vecteurs décrivant les liens entre ces concepts et les documents.

On peut ensuite mesurer la proximité des vecteurs, pour comparer entre eux les documents ou les termes.

Il est possible, dans SVD, de faire une approximation sur le produit de matrices en ne conservant que les *k* plus grandes valeurs diagonales de Σ (et en adaptant les dimensions de U et V, retirant les vecteurs singuliers associées à une valeur singulière disparue). Cela permet de réduire le nombre de dimensions des données, et de favoriser l'utilisation d'algorithmes

manipulant des matrices. On peut alors appliquer, entre autres, des algorithmes de *partitionnement* (clustering).

3.6.2.3 Applications

Il y a plusieurs travaux relatifs à l'utilisation de LSA pour l'extraction d'expressions multi-mots (MWE).

Schone & Jurafski (2001) calculent, à l'aide de LSA, des vecteurs sémantiques pour chaque n-gramme du corpus. Ces vecteurs sont ensuite utilisés pour tenter de vérifier la non-compositionnalité (le vecteur sémantique de l'expression doit être corrélé avec la somme des vecteurs sémantiques de ses composants) et la non-substituabilité (le vecteur sémantique de l'expression doit être corrélé avec le vecteur sémantique de la même expression où on a remplacé un composant par un autre terme proche de ce composant). Pour le premier cas, cela « n'aide pas » ; pour le second, cela produit des « gains modestes en performance ».

Baldwin et al. (2003) utilisent LSA sur une matrice de 50 000 mots (les plus fréquents) par 1 000 mots (les mots pleins les plus fréquents) pour réduire la seconde dimension à 100 ; un calcul de similarité basé sur cette sortie est ensuite réalisé entre les MWE et leurs composants pour évaluer la compositionnalité des MWE, mais l'évaluation peu claire ne permet pas de prouver que la méthode est utile.

(Katz & Giesbrecht, 2006) contient une évaluation de similarité basée sur LSA (à partir d'une matrice croisant les 20 000 mots pleins les plus fréquents avec les 1 000 mots les plus fréquents, réduisant la seconde dimension à 100) : l'expérimentation de comparaison entre les vecteurs des MWE et la somme des vecteurs de leurs composants montre une différence de comportement entre les MWE compositionnels et les autres. Une autre expérimentation montre qu'on peut, à partir du vecteur d'une MWE, prédire de manière relativement bonne s'il est ou non compositionnel (en le comparant à la somme de ses composants).

3.7 Travaux bilingues

Les travaux d'extraction bilingue de collocations sont plus rares, sans doute parce qu'ils demandent plus de ressources (corpus multilingues, généralement parallèles, dont les documents d'une langue sont les traductions des documents de l'autre langue).

(Kupiec, 1993) utilise des corpus parallèles (*Hansard*, débats du parlement canadien en français et en anglais), alignés par phrases, pour obtenir des traductions entre syntagmes nominaux français et anglais. Après avoir utilisé un logiciel pour étiqueter les parties du discours, il extrait des syntagmes nominaux, puis calcule les correspondances entre syntagmes anglais et français selon la fréquence à laquelle ils apparaissent dans des phrases alignées. Ce travail ne porte pas sur les collocations (certains syntagmes se réduisent d'ailleurs à un seul mot) ; il montre cependant comment doit se dérouler la partie bilingue d'un processus d'extraction de bicollocations si l'on utilise des corpus parallèles.

Champollion (Smadja et al., 1996) a pour but de permettre la traduction de collocations à l'aide de corpus alignés (*Hansard*). Il ne s'agit pas à proprement parler d'extraction bilingue, mais d'une extraction monolingue de collocations en anglais puis, pour chaque collocation anglaise, de l'identification de la chaîne qui a le plus de chance d'être la traduction française de celle-ci.

Les corpus bilingues (ou multilingues) parallèles facilement accessibles étant rares, certains travaux portent au contraire sur l'extraction de connaissances à partir de corpus monolingues indépendants et de dictionnaires bilingues (Rapp, 1999), (Koehn & Knight, 2000). Lu & Zhou (2004) extraient des traductions anglais-chinois de collocations à partir de corpus dans les deux langues et de dictionnaires bilingues ; bien qu'ils n'obtiennent que des couples de traductions se traduisant mot à mot (ce qui n'est pas forcément très utile : les collocations les plus intéressantes à trouver sont celles qui posent problème aux traducteurs automatiques), ils mettent le doigt sur ce à quoi doit ressembler une extraction bilingue (voire multilingue) de collocations : des processus monolingues permettant de repérer, pour chaque langue, ce qu'on espère être des collocations, et un processus multilingue qui établit les correspondances (de traduction) entre les collocations des différentes langues.

Nous avons présenté plus haut un système d'extraction de collocations monolingues sur des corpus multilingues, basé sur l'analyseur syntaxique Fips (Seretan & Wehrli, 2006). Celui-ci a ensuite été complété par l'utilisation d'un aligneur de phrases de corpus parallèles et d'une procédure d'association entre collocations monolingues pour produire des bicollocations, tenant (optionnellement) compte du fait que les collocatifs des composantes d'une bicollocation ne sont pas nécessairement traduction l'un de l'autre (Seretan & Wehrli, 2007).

4 Apprentissage automatique

L'extraction n'est cependant pas le seul moyen d'obtenir des collocations. Puisqu'il s'agit de co-occurrences particulières (elles apparaissent plus souvent que par hasard, elles sont basées sur des relations syntaxiques), on peut envisager d'effectuer un apprentissage automatique des caractéristiques des collocations, puis de retrouver les co-occurrences ayant ces propriétés. Il faut cependant pour cela disposer de données de référence, ce qui rend ces démarches moins accessibles et donc moins répandues.

Les travaux présentés dans (Sébillot, 2002) effectuent un apprentissage de « relations lexicales sémantiques » qui ont la forme nom-verbe ; il s'agit plus précisément de produire des « couples qualia » (un nom et un verbe liés par un des rôles de la structure qualia du Lexique Génératif). Pour cela, une évaluation manuelle est d'abord effectuée sur des phrases contenant des relations extraites à l'aide d'analyse syntaxique, qui permet d'avoir des exemples et des contre-exemples, auxquels sont associés des prédicats pouvant contenir, selon les expériences, l'étiquette grammaticale ou sémantique des termes entourant le nom et le verbe, la distance entre le nom et le verbe, leur position (lequel est avant l'autre ?). Une implémentation de l'algorithme de Muggleton permet alors de générer des clauses, qui sont ensuite utilisées comme règles dans un processus d'extraction. Même si les résultats sont satisfaisants, le coût en travail humain pour l'étiquetage des phrases rend ce genre d'expérimentations peu accessible.

(Yang, 2003) utilise d'abord l'algorithme d'apprentissage *C4.5* pour réaliser des arbres de décision (permettant d'établir si un couple est ou non une collocation) sur les mesures d'association ; il améliore ensuite la qualité des résultats (peu satisfaisants après la première étape) en utilisant une classification probabiliste pour son *Decision Tree Learner for Retrieval* (DTLR).

Le système présenté dans (Zinsmeister & Heid, 2003) utilise lui aussi un algorithme d'arbre de décision, mais sur des classes de triplets adjectif-nom-verbe, selon la valeur LLR sur le triplet et sur les couples adjectif-nom et nom-verbe, ce qui semble efficace pour distinguer les triplets correspondant[58] à des collocations de ceux correspondant à des locutions.

(Wanner et al., 2006) décrit 3 méthodes d'apprentissage automatique qui ont pour but la classification de collocations en fonctions lexico-sémantiques. La première effectue un classement à partir de prototypes de collocations avec la méthode des k plus proches voisins ; la seconde, à partir de caractéristiques sémantiques présumées des exemples de chaque type (= fonction lexico-sémantique), utilise un réseau bayésien naïf ; enfin, la dernière emploie un réseau bayésien naïf augmenté par un arbre, à partir des corrélations supposées entre les propriétés sémantiques des différents exemples, pour chaque type. La qualité de la classification produite avec la première méthode (plus proches voisins), bien qu'étant assez souvent dépassée par l'une des deux autres, est jugée plus « stable », et donc plus intéressante à utiliser.

5 Contribution humaine

Pour obtenir des collocations, il n'est pas absolument nécessaire d'avoir de grands corpus de documents d'où les extraire. En effet, elles font partie des connaissances de tout locuteur natif de la langue : il n'est pas besoin d'être linguiste pour savoir qu'une « peur bleue » est intense, que « commettre un crime » est le réaliser, ou que « infliger une défaite » signifie « vaincre » alors que « subir une défaite » signifie « être vaincu ».

Une autre façon d'utiliser l'outil informatique pour obtenir des collocations peut donc être la réalisation d'interfaces homme-machine permettant de remplir facilement une base lexicale.

5.1 Wiktionnaire

Wiktionary[59], en français *Wiktionnaire*[60], est l'un des nombreux projets lancés par la *Wikimedia Fundation* à la suite du succès de l'encyclopédie collaborative *Wikipedia*, en anglais en décembre 2002, puis en français en mars 2004 (Wikimedia, 2008). Les contributions y sont cependant bien moins nombreuses que chez sa grande sœur : fin 2008, il y avait environ 80 contributeurs « actifs » (dépassant les 5 modifications par mois) sur le wiktionnaire francophone (Wikimedia, 2009a), contre environ 5000 contributeurs dépassant le même seuil sur la version anglophone de Wikipédia (Wikimedia, 2009b).

On peut y trouver beaucoup de définitions et de traductions d'entrées, mais peu d'informations sur les collocations, même indirectement. Ainsi (au 16/07/2009), l'expression « pluie battante » n'est pas citée dans l'entrée « pluie », mais dans celle de l'adjectif « battant », auquel est associé la traduction « beating » ; de même, « soleil de plomb » n'est pas référencé dans l'entrée « soleil » mais dans celle de plomb ; on ne peut y trouver ni « café serré » ni « bêtise insondable ». Typiquement, une entrée de wiktionnaire comporte le même type

[58]Cela correspond au cas où le collocatif n'est pas un terme mais une expression.

[59]http://en.wiktionary.org

[60]http://fr.wiktionary.org

d'informations que celles présentes dans les ressources pré-existantes qui ont aidé à sa conception : des traductions de l'entrée seule, et des expressions (non traduites) dans lesquelles l'entrée apparaît, parmi lesquelles on peut trouver des collocations. Les informations détaillées sur les collocations qui n'existaient pas dans les ressources pré-existantes sont donc absentes du wiktionnaire.

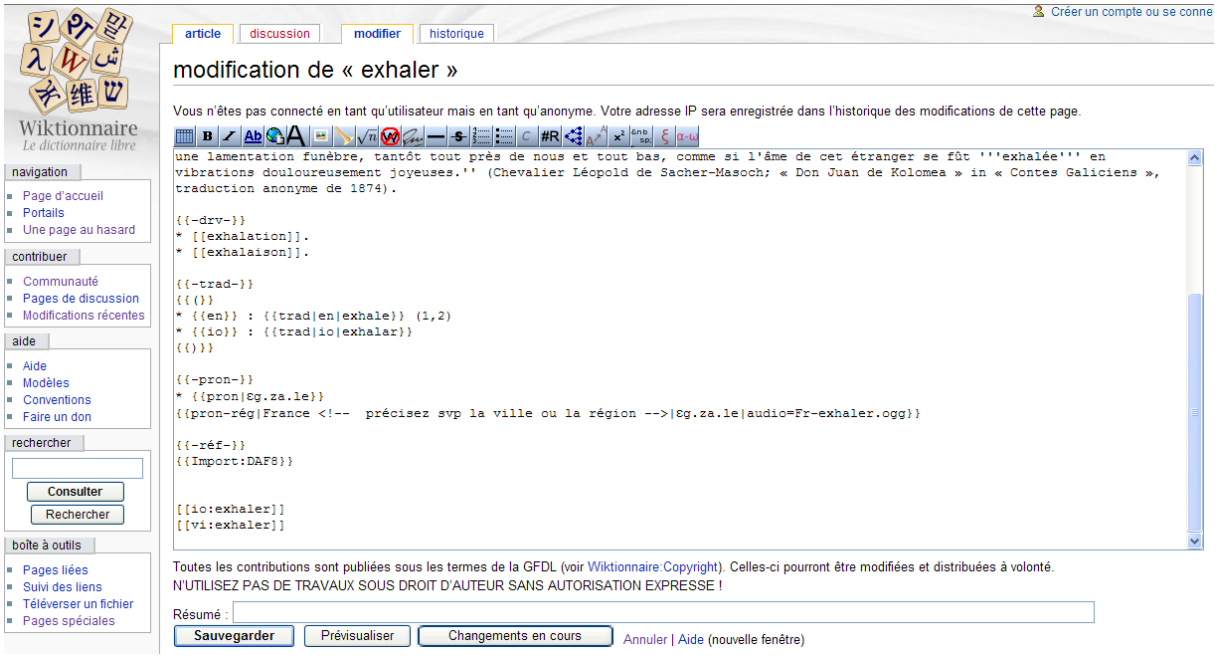


Figure 11: Interface wiki d'édition d'une entrée du wiktionnaire

En réalité, le wiktionnaire est essentiellement, de par le problème même qu'il veut résoudre (la lexicographie), réservé à des contributeurs passionnés. Au contraire de Wikipedia où tout le monde peut contribuer avec sa propre expertise (certains sur des domaines scientifiques pointus, d'autres sur leur chanteuse préférée), il est difficile de motiver le grand public pour modifier des entrées de dictionnaires dont le contenu peut provenir de dictionnaires réputés ou de passionnés de lexicographie.

Sur le problème précis des collocations, même si le wiktionnaire n'utilise pas une représentation de leur sens (comme les fonctions lexico-sémantiques), on pourrait néanmoins envisager que les contributeurs puissent ajouter leurs traductions. Cependant, le problème des collocations nécessitant au niveau monolingue une maîtrise élevée de la langue en question, les éventuels contributeurs devraient maîtriser les deux langues (l'originale et celle de la traduction) pour s'assurer de la correction des informations qu'ils apporteraient, ce qui réduit grandement la cible possible.

Cela est dû justement à l'absence de représentation du sens : si des contributeurs peuvent indiquer que « fumeur » est intensifié par « gros » et « Raucher » par « starker », il n'est alors nécessaire, pour connaître la traduction de « gros fumeur », que de savoir comment traduire « fumeur », ce qui demande moins de connaissances bilingues que les collocations. Ne pas passer par une représentation du sens réduit donc grandement la cible possible.

Le désordre dans lequel sont présentées les informations utiles pour la traduction rend difficile leur emploi dans des systèmes de traduction. Le site Web *Word Reference*[61] illustre bien ce problème, en enrichissant les entrées des dictionnaires par des forums de discussions sur la manière de traduire des expressions correspondantes ; si cela peut être très utile pour un internaute, ce n'est pas utilisable (en l'état) par un programme de traduction, faute de structuration de l'information.

5.2 Papillon

5.2.1 Principe

Le projet *Papillon*, lancé par le *GETA* à Grenoble et le *National Institute of Informatics* à Tokyo, avait pour but la création d'une base lexicale multilingue (Mangeot et al., 2003), utilisant une représentation du sens directement inspirée par la lexicologie explicative et combinatoire (Mel'čuk et al., 1995).

Cette base peut être vue comme un dictionnaire composé de plusieurs volumes : un par langue, et un pour la structure de pivot interlingue. Les entrées monolingues comportent des informations sur les collocations.

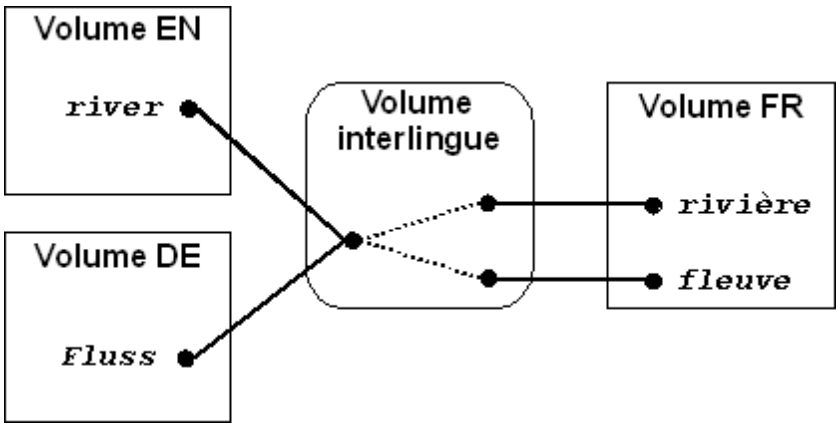


Figure 12: Représentation du raffinement sémantique dans la base lexicale Papillon

L'architecture du dictionnaire s'organise selon deux niveaux :

- la *macro-structure* (liens entre les volumes) : les volumes monolingues de la base sont organisés en étoile, autour d'un pivot abstrait (le volume interlingue) fait d'*acceptions interlingues* (Sérasset, 1994), nommées *axies* dans le cadre du projet Papillon (Boitet et al., 2002) pour faire pendant au terme « lexie » ;
- la *micro-structure* (contenu des volumes) :
  - les entrées des volumes monolingues (les *lexies*) reprennent la structure interne du dictionnaire DiCo, avec quelques adaptations ;
  - les entrées du volume interlingue (les *axies*) sont plus simples : une axie relie des lexies des volumes monolingues jugées synonymes (ou au moins équivalentes en

[61] http://www.wordreference.com/

traduction)[62] , chaque lexie étant nécessairement reliée à une axie ; le volume interlingue modélise les différences de raffinement sémantique selon les langues, ainsi l'axie associée à « river » se raffine-t-elle en deux axies distinctes liées à « rivière » et « fleuve » (voir figure 12).

La base, amorcée avec des dictionnaires existants, peut ensuite recevoir les contributions d'utilisateurs (ajout, suppression, modification), qui doivent ensuite être validées par des spécialistes pour y être intégrées.

5.2.2 Mise en œuvre

5.2.2.1 Données

La base lexicale Papillon a deux parties :

- *Papillon-CDM*, qui intègre des dictionnaires existants dans une format standardisé, contient plus de 2 millions d'entrées dans 9 langues (dont certains « idiomes » pouvant contenir des collocations), et est relativement stable ;
- *Papillon-NADIA*, qui contient 1170 lexies dans 4 langues (français, anglais, japonais, malais) et 42 axies (Mangeot, 2005), est en phase de collecte d'informations (une idée majeure étant d'utiliser les données de Papillon-CDM pour générer les lexies et les axies de Papillon-NADIA).

Elle peut être consultée et éditée sur le site Web du projet : http://www.papillon-dictionary.org.

5.2.2.2 Interface d'édition

L'interface d'édition en ligne permet de renseigner les différentes informations nécessaires à la définition d'une entrée monolingue, notamment les valeurs des différentes fonctions lexico-sémantiques pour cette entrée (voir figure 13).[63]

La plateforme développée lors du projet Papillon, et nommée *Jibiki*, convient tout à fait pour la réalisation de données organisées par des spécialistes. Elle a d'ailleurs été réutilisée avec succès pour d'autres projets de construction de dictionnaires multilingues, comme *LexAlp*, une base terminologique européenne multilingue pour la terminologie de la Convention Alpine (Brunet-Manquat & Sérasset, 2006), le *Grand Dictionnaire Estonien-Français* (Mangeot & Chalvin, 2006), ou la base PIVAX (Nguyen et al., 2007) destinée aux systèmes de traductions automatiques utilisant un « pivot lexical ».

Au contraire, cette plateforme est peu accessible pour une contribution du grand public, celui-ci ne connaissant pas les fonctions lexico-sémantiques. La base Papillon-NADIA, basée contrairement aux autres projets sur la contribution bénévole, a souffert de cela pour se développer, d'autant plus que les efforts de recherche du projet *Papillon* ont été surtout orientés vers le développement de la plateforme *Jibiki*.

[62]Si une axie relie des lexies à arguments (le plus souvent, des lexies prédicatives), elle porte l'indication du nombre d'arguments (avec un ordre standard) et les liens vers les lexies portent l'ordre des arguments pour cette lexie (s'il est différent de l'ordre standard).

[63]Fournir la possibilité de compléter les ressources existantes est quelque chose de classique dans les bases de données lexicales en ligne, comme par exemple le dictionnaire allemand-japonais *Wadoku Jiten* d'Ulrich Apel : http://www.wadoku.de/

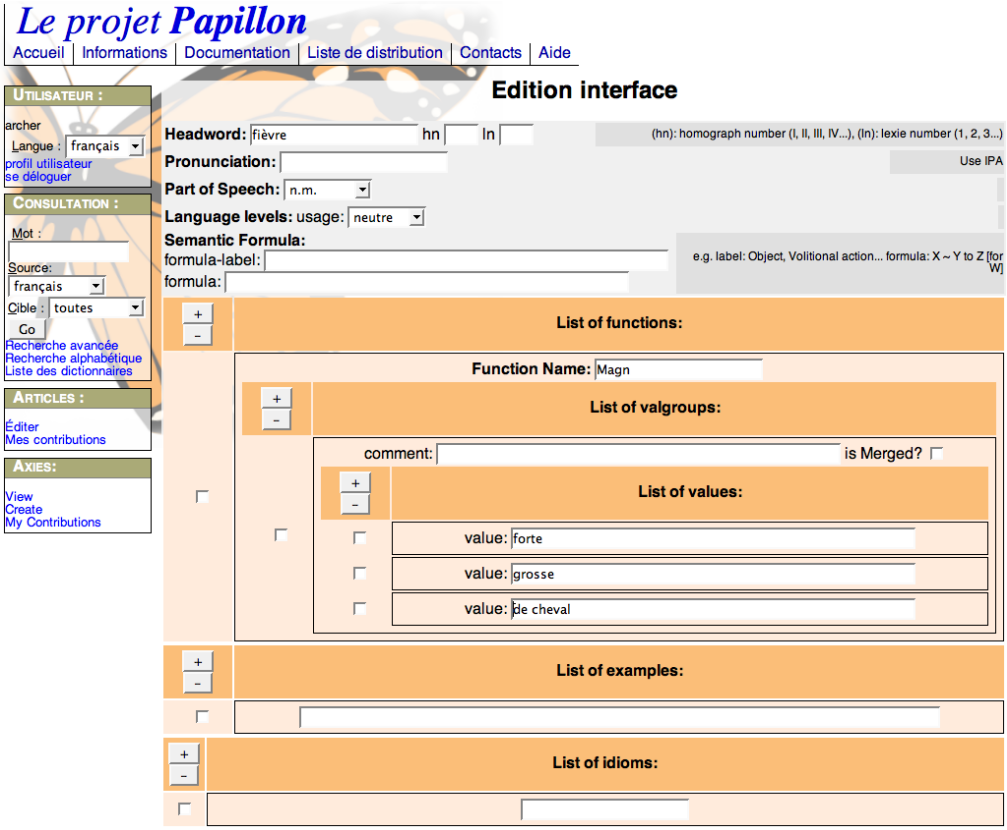


Figure 13: Interface de la base lexicale multilingue Papillon

5.3 Jeuxdemots

Les deux problèmes principaux pour la collecte d'informations sur les collocations (et plus généralement sur le lexique) sont :

- Comment faire « accoucher » les contributeurs de leurs connaissances de la langue, sans rentrer dans les détails linguistiques ?
- Comment motiver des bénévoles non spécialistes à contribuer ?

Pour obtenir des informations linguistiques auprès de non spécialistes, il suffit souvent de présenter ce qu'on souhaite obtenir de manière simple, à l'aide d'exemples (« on dit *une peur bleue*, un *gros fumeur*, comment dit-on pour *une bagarre* ? »)[64].

La motivation peut passer par un aspect ludique de la contribution : un jeu sur les connaissances linguistiques, où les réponses des joueurs permettent d'établir des relations entre les termes de la langue. C'est le principe de base du projet *Jeuxdemots*, qui vise à obtenir les fonctions lexico-sémantiques qui manquent à Papillon-NADIA.

[64]Ce principe est d'ailleurs également utilisé pour la désambiguïsation interactive des énoncés (Blanchon et al., 2006).

5.3.1 Jeu collaboratif

Une partie de Jeuxdemots, à l'adresse *http://www.lirmm.fr/jeuxdemots*, se déroule en deux phases :

- le joueur a une minute pour donner le plus de mots possible correspondant au terme et à la relation proposés ;
- à la fin de la partie :
  - si la partie a déjà été jouée par un autre : les deux joueurs gagnent des points en fonction de leurs propositions communes (et en perdent s'ils n'en ont pas de communes) ;
  - si la partie n'a pas déjà été jouée : les propositions sont gardées en mémoire en attendant qu'un autre joue la partie (et rapporte éventuellement des points).

L'essence même d'un jeu basé sur les connaissances impliquant la possibilité qu'on puisse se tromper, comment savoir s'il faut garder ou non les informations produites par les joueurs ?

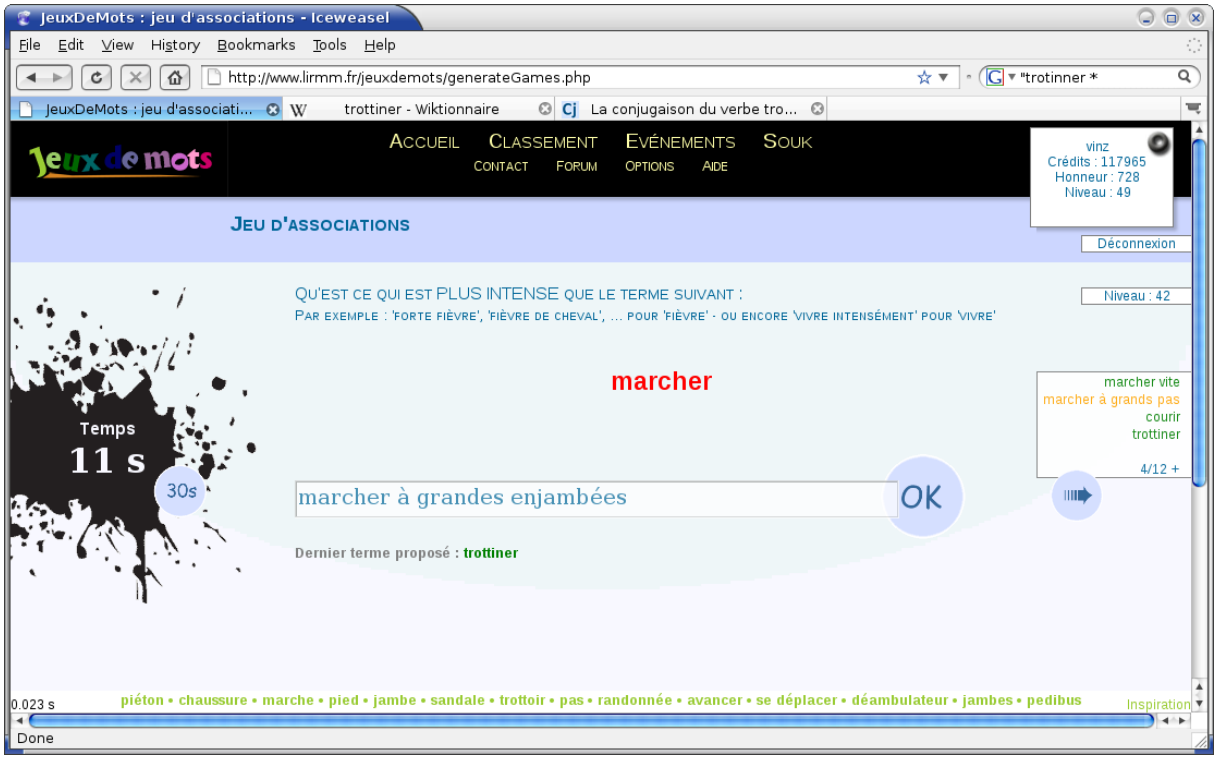


Figure 14: Exemple de partie de Jeuxdemots pour la relation « Magn »

La solution adoptée par Jeuxdemots (Lafourcade & Joubert, 2008) est inspirée de *The ESP Game*, un jeu pour la description d'images (Von Ahn & Dabbish, 2004) utilisé sous licence par Google pour son *Image Labeler*, dont le principe est de faire jouer les participants en duo indépendant[65], en leur indiquant qu'ils n'auront de points que pour les réponses communes aux

[65]Un joueur ne sait pas avec qui il joue, ce qui évite les collusions qui pourraient fausser les résultats.

deux joueurs. Le système ne garde alors que ces réponses communes, l'idée étant qu'une même réponse donnée par deux joueurs indépendants a de fortes chances d'être correcte.

Le premier type de relation (entre le terme affiché et les mots attendus) proposé au joueur est « *idées associées* », mais le joueur peut ensuite, s'il atteint un score suffisant, acheter d'autres relations, telles que « *synonyme* », « *contraire* », « *Magn* », etc.

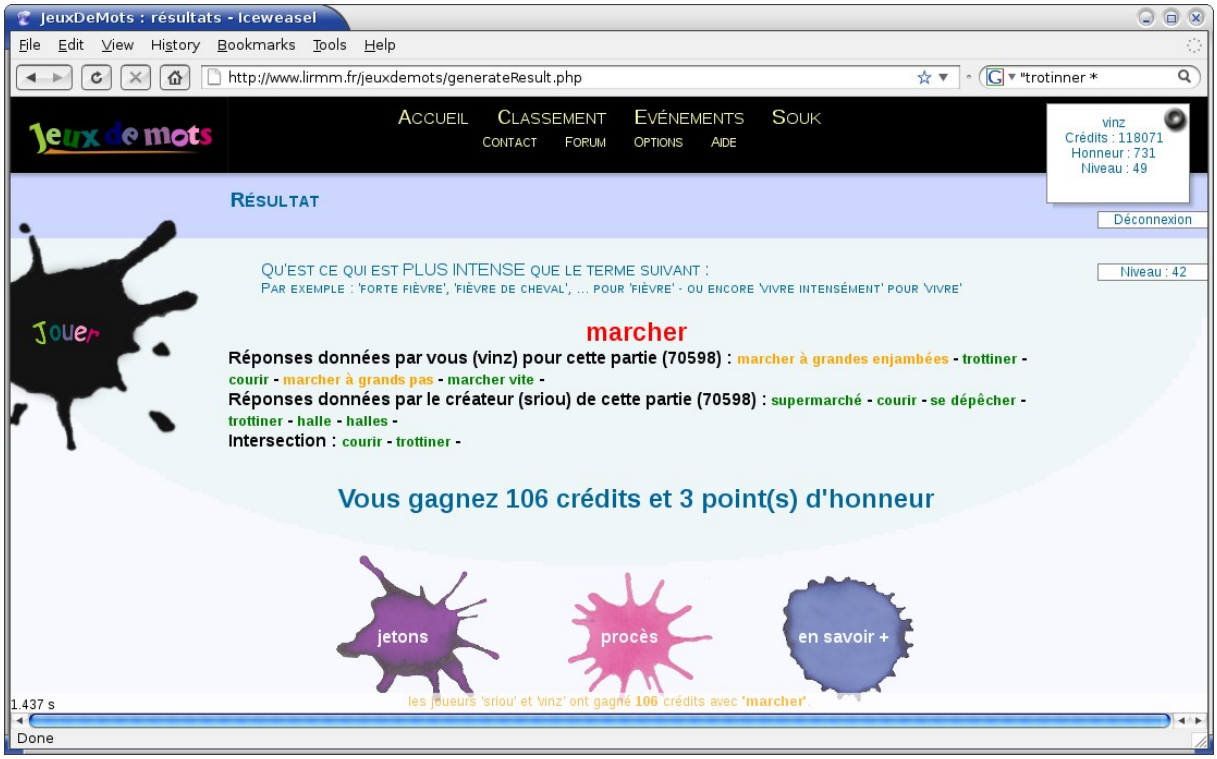


Figure 15: Exemple de résultat de partie de Jeuxdemots pour la relation de base « idées associées »

Lancé à l'été 2007, Jeuxdemots dispose, en juillet 2009, d'un nombre important de membres ayant joué des centaines de parties, voire pour certains des milliers, qui lui ont déjà permis la production d'environ 325 000 relations entre termes (pour environ 190 000 termes).

5.3.2 Bilinguisme

Le principe de Jeuxdemots, déjà décliné dans d'autres langues, est-il utilisable pour produire des relations bilingues, en particulier de traduction ? Plusieurs problèmes peuvent en faire douter :

- produire des relations de traduction entre termes n'est pas vraiment utile (les résultats obtenus ne vaudront pas le contenu d'un dictionnaire) ; l'intérêt serait d'obtenir des traductions d'expressions comme les collocations.
- un jeu bilingue nécessite des joueurs ayant une connaissance suffisante des deux langues pour ne pas « passer » trop souvent, ceux-ci sont peu nombreux ; la cible du

> jeu se réduirait encore plus s'il fallait traduire non plus des mots simples, mais des collocations.[66]

La mise en relation multilingue des réseaux lexicaux produits avec les versions de Jeuxdemots en différentes langues ne passe sans doute pas par un jeu, mais bien par le travail d'un expert.

6 Raffinement du problème

6.1 Choix de l'extraction

Il y a, dans chaque piste de collecte que nous avons présentée (extraction, apprentissage, contribution), des travaux intéressants qui permettent effectivement d'améliorer la collecte de collocations. Sur quelle piste nous engager ?

L'apprentissage automatique, s'il est parfois difficile à mettre en œuvre, peut apporter réellement un plus à la collecte. Cependant, les méthodes supervisées peuvent nécessiter un travail préalable important pour la réalisation de la base d'apprentissage. D'autres utilisent la valeur des mesures d'association. Dans tout les cas, il faut être capable de produire en nombre des candidats pour pouvoir utiliser l'algorithme. L'apprentissage doit donc, d'un point de vue stratégique, passer après d'autres modes d'extraction.

La contribution de locuteurs est également un domaine prometteur (que ces contributeurs soient spécialistes ou non), mais prend du temps avant de donner des résultats en quantité satisfaisante.

L'extraction de collocations est donc à la fois le domaine le plus stratégique, et celui où le défi nous semble le plus intéressant. C'est pourquoi les travaux présentés dans la suite de cette thèse seront relatifs à l'extraction de collocations à partir de corpus.

6.2 Redéfinition du problème

Le problème que nous souhaitons désormais résoudre est celui de la collecte de collocations par extraction à partir de corpus, en utilisant des informations linguistiques (pour mieux identifier les candidats) et des mesures statistiques basées sur leurs effectifs (pour mieux les classer).

Si une telle extraction est surtout réalisée au niveau monolingue, il faut aussi fournir des efforts sur l'aspect bilingue afin d'extraire des bicollocations.

Conclusion

Notre but étant d'améliorer la collecte de collocations, nous avons étudié dans ce chapitre les différents moyens d'y parvenir (y compris pour les aspects bilingues), que ce soit par

[66]On peut néanmoins envisager de cibler la communauté des spécialistes de la traduction, en les « gratifiant » de nouvelles connaissances (par exemple, de nouvelles expressions dans la langue cible)

extraction, apprentissage automatique, ou contribution humaine. La voie qui nous a semblé la plus intéressante à suivre, notamment parce qu'elle permet d'obtenir des résultats rapides via des processus automatiques sans besoin de références préalables, est celle de l'extraction à partir de corpus.

Pour être efficace (et produire des candidats corrects), un processus d'extraction de connaissances doit être hybride, c'est-à-dire utiliser à la fois des informations linguistiques (obtenues par, entre autres, des analyseurs automatiques) et des mesures statistiques. En effet, la collocation est un phénomène linguistique qui se prête particulièrement bien aux calculs statistiques, puisqu'il s'agit d'une co-occurrence qui se déroule plus souvent que par hasard. L'intervention humaine dans le processus (pour le guider) permettrait sans doute l'amélioration des résultats.

Ils nous faut donc maintenant faire des expérimentations sur les extractions, pour valider ces hypothèses, et mieux cerner quels sont les problèmes informatiques posés par l'extraction et comment les résoudre.

Partie II - Besoins et solutions de l'extraction de connaissances linguistiques

Chapitre IV - Besoins de l'extraction

Après étude des différents travaux existants, nous avons décidé de nous concentrer sur l'extraction à partir de corpus, hybride (utilisant des informations linguistiques et statistiques) et semi-automatique (avec intervention humaine pour guider le processus). Ce chapitre traitera donc de l'extraction de collocations, mais plus généralement de l'extraction de connaissances. Nous cherchons en effet à spécifier des démarches qui soient les plus génériques possibles, c'est-à-dire qui ne soient pas limitées à un type particulier d'information.

Nous effectuons d'abord des expérimentations d'extraction monolingue utilisant des informations sémantiques présentes dans les vecteurs conceptuels et classant les candidats en utilisant différentes mesures basées sur l'information mutuelle. Nous réalisons ensuite des expérimentations bilingues en exploitant le fait que, si deux collocations sont traduction l'une de l'autre, ce n'est pas nécessairement le cas des collocatifs qui les composent (« pluie battante » se traduit par « driving rain » bien que « battante » ne se traduise pas par « driving »).

Bien que les démarches d'extraction monolingue et bilingue que nous décrirons présentent des innovations, l'objet principal de ce chapitre est d'étudier ce que ces expérimentations nous apprennent en général sur le processus d'extraction de connaissances, et en particulier sur les besoins les plus importants à résoudre.

1 Extraction monolingue

Nous présentons tout d'abord un processus d'extraction monolingue de collocations concernant une fonction lexico-sémantique particulière.

1.1 Méthode hybride et semi-automatique

Nous choisissons ici de réaliser une extraction de collocations de la forme *base verbale-collocatif adverbial*. Une des raisons de ce choix est le fait que les adverbes sont moins variables sémantiquement que les adjectifs : par exemple, « bleu » peut être employé de manière collocative (« peur bleue ») mais pas toujours (« voiture bleue »).

1.1.1 Corpus analysés

La démarche que nous présentons ne contient pas d'analyse linguistique, celle-ci étant considérée comme une tâche extérieure, antérieure à celle-ci.

Plusieurs contextes d'extraction sont utilisés, reposant sur l'analyse linguistique (en pratique une analyse de dépendances) ; il s'agit de différents moyens pour essayer de trouver des couples où l'adverbe modifie le verbe :

- un contexte dit « linéaire », où le verbe est suivi immédiatement de l'adverbe ;
- un contexte dit « de dépendance », basé sur les sorties d'un analyseur de dépendances.

L'adverbe ne suivant pas forcément le verbe qu'il modifie, l'utilisation des dépendances produites par analyse permet d'obtenir plus de couples (moins de silence) mais on risque, en s'éloignant du verbe, d'associer incorrectement des adverbes avec le verbe (plus de bruit).

Notre approche peut être critiquée sur le fait qu'elle ne prenne pas directement en compte (pour estimer la collocabilité) la séparation possible entre base ou collocatif, ou l'ordre dans lequel ils apparaissent. Mais il faut rappeler que les expérimentations menées ici ont pour but premier de mieux connaître le problème de l'extraction, plus que de fournir la meilleure approche possible.

Notre méthode d'extraction présuppose le traitement linguistique permettant d'obtenir des contexte de co-occurrence pertinents.

1.1.2 Mesure statistique

Nous employons ici une mesure basée sur l'information mutuelle, avec une pondération pour éviter un des inconvénients de MI (le fait de favoriser des couples rares), décrite dans le chapitre précédent, et telle qu'introduite par (Fung & McKeown, 1997) :

$$W(w_{1,}w_2) = P(w_{1,}w_2) \times \log \frac{P(w_{1,}w_2)}{P(w_1)P(w_2)} \quad [67]$$

Plus précisément, nous utilisons l'adaptation de cette mesure aux triplets <*terme1, relation, terme2*>, décrite dans (Wu & Zhou, 2003) :

$$WMI(w_{1,}r,w_2) = P(w_{1,}r,w_2) \times \log \frac{P(w_{1,}r,w_2)}{P_g(w_1|r)P_d(w_2|r)P(r)} \quad [68]$$

1.1.3 Processus semi-automatique

Les travaux que nous présentons ne sont pas manuels, mais ils ne sont pas pour autant entièrement automatiques. En effet, les données automatiques que nous utilisons sont, par essence, susceptibles d'être bruitées. Léon & Millot (2005) font passer la précision finale de leur extraction de relations lexicales bilingues de 7,5% à 83% avec une simple validation manuelle des relations anglaises extraites, ce qui montre tout l'intérêt de l'intervention humaine dans un processus automatique d'extraction.

Il s'agit de combiner :

- la *précision* du travail humain, trop lent pour faire un traitement rapide à grande échelle ;
- la *rapidité*, la puissance du calcul automatique, qui, seul, ne permet pas une qualité suffisante.

La démarche la plus efficace doit donc combiner les deux, et cela de manière optimale : si le processus doit évidemment rester globalement automatique, il faut y introduire une intervention manuelle, pour lui permettre d'utiliser les informations linguistiques qu'il n'est pas capable de générer lui-même.

---

[67] Ne pas confondre les probabilités *P( )* utilisées dans les mesures, et les nombres d'occurrences observées *O( )* ou espérées *E( )* utilisées précédemment.

[68] avec $P_g(w_1|r)$ et $P_d(w_2|r)$ les probabilités d'avoir, respectivement, w_1 comme premier élément d'une relation, et w_2 comme second élément d'une relation, sachant que cette relation est r.

1.2 Filtrage sémantique

Nous avons défini la collocation comme un objet à trois composantes : la base, le collocatif, et le sens exprimé. Les collocations que nous cherchons à extraire devront donc avoir un sens, nous choisissons l'*intensification* (fonction lexico-sémantique `Magn`) pour nos expériences.

1.2.1 Ressources sémantiques pour l'extraction

Comme nous avons pu le voir dans l'état de l'art, la grande majorité des travaux cherchent simplement à extraire des collocations, sans tenir compte de leur sens ; on considère généralement que c'est le spécialiste chargé d'utiliser les données produites qui devra résoudre le problème d'identification du sens. Cette identification est essentielle, surtout dans le cadre dans lequel nous nous plaçons, celui des fonctions lexico-sémantiques : trouver des collocatifs n'est pas suffisant, il faut savoir de quelles fonctions lexico-sémantiques ils sont les valeurs.

Si la question est généralement évitée dans les processus automatiques, c'est que les problèmes d'ordre sémantique sont peu faciles à résoudre, faute de connaissances linguistiques suffisantes de ce type.

Quelles sont les ressources sémantiques exploitables pour le traitement du sens des collocations lors de leur extraction ? De quelle manière peut-on les utiliser ?

On s'intéresse ici de préférence aux données dont la production, automatique ou semi-automatique, peut être reproduite dans un temps raisonnable ; l'idée étant d'inscrire notre démarche dans un processus globalement automatique plus large.

(Wanner et al., 2006), dont nous avons présenté les travaux dans la section 4 du chapitre précédent, effectue un apprentissage automatique supervisé, pour essayer de trouver les caractéristiques discriminantes qui permettent d'identifier dans quelle fonction lexico-sémantique ranger la collocation. Une grande limitation de ces travaux est qu'ils portent sur des fonctions lexico-sémantiques représentant des verbes support (vides sémantiquement), qui présentent par essence de grandes différences de syntaxe entre elles : il serait sans doute bien moins efficace pour distinguer les collocations correspondant respectivement aux fonctions lexico-sémantiques `Magn` (intensification) et `Bon` (appréciation positive).

Une idée qui nous semble plus pertinente pour un filtrage sémantique est l'utilisation d'une représentation du sens qui permettrait de calculer des *distances sémantiques*. Cela pourrait permettre un filtrage sémantique intéressant parmi tous les candidats.

Une solution pourrait être la production de vecteurs sémantiques associés aux termes à l'aide de LSA, comme ce qui se fait dans les travaux présentés dans la section 3.6 du chapitre précédent. Comme LSA, nous avons choisi d'utiliser des ressources où le sens des mots est représenté de manière vectorielle, mais nos vecteurs sont produits d'une autre manière, en utilisant davantage de ressources linguistiques.

1.2.2 Vecteurs conceptuels

L'idée que nous avons retenue est d'utiliser les vecteurs conceptuels, qui « représentent les idées associées aux mots, aux expressions, et de façon générale à tout segment textuel » (Schwab et al., 2002).

1.2.2.1 Principe

Une première utilisation des vecteurs pour des problèmes liés à la linguistique a été présentée dans le modèle vectoriel de Salton (1975) (Salton & McGill, 1986). La *théorie des vecteurs conceptuels* combine cette approche vectorielle avec les travaux de Chauché (1990), qui proposaient de représenter le sens à partir d'un jeu de concepts prédéterminés[69] : il s'agit de représenter le sens en projetant le champ sémantique dans un espace vectoriel.

On peut trouver une définition précise des vecteurs conceptuels dans (Schwab, 2001) :

> Soit C un ensemble fini de n concepts, un vecteur conceptuel V est une combinaison linéaire des éléments c_i de C. Pour un sens A, le vecteur V_a est la description, en extension, des activations des concepts de C.

Les termes sont donc représentés par des vecteurs dans un espace dont les dimensions correspondent à des concepts de base de la langue ; contrairement aux travaux utilisant LSA, ces concepts ne sont pas produits automatiquement mais choisis préalablement.

1.2.2.2 Distance

L'utilisation d'une représentation vectorielle a le grand avantage de permettre d'évaluer la *proximité thématique* de deux termes en comparant les deux vecteurs les représentant. La mesure de *similarité* entre vecteurs conceptuels retenue est le *cosinus* :

$$Simil(\vec{X},\vec{Y}) = \cos(\widehat{\vec{X},\vec{Y}}) = \frac{\vec{X}\cdot\vec{Y}}{\|\vec{X}\|\times\|\vec{Y}\|}$$

(où $\vec{X}\cdot\vec{Y}$ est le produit scalaire des vecteurs).

On peut, à partir de cette similarité, définir une *distance angulaire*, qui correspond, en pratique, à l'angle entre deux vecteurs :

$$D_A(\vec{X},\vec{Y}) = \arccos(Simil(\vec{X},\vec{Y}))$$

La distance obtenue entre deux vecteurs permet de qualifier le lien sémantique entre les termes qu'ils représentent (Schwab et al., 2004) :

- si leur distance est inférieure à $\pi/4$ (45°), les termes sont *thématiquement proches*, ils partagent plusieurs concepts ;
- si leur distance est supérieure à $\pi/4$, la proximité thématique est faible ;
- si leur distance est proche de $\pi/2$ (90°) : les termes n'ont aucune relation thématique.

1.2.2.3 Construction d'une base de vecteurs conceptuels

Comment choisir les concepts qui serviront de dimensions à l'espace vectoriel ? L'option la plus raisonnable semble être d'utiliser des ressources linguistiques existantes, comme le thésaurus *Roget* (1852) pour l'anglais, qui contient 1000 concepts (en 6 niveaux), ou le thésaurus *Larousse* (1992) pour le français, qui contient 873 concepts (organisés en 4 niveaux).

[69] À partir de thésaurus.

Une expérimentation a été menée au sein du laboratoire *LIRMM* à Montpellier sur les vecteurs conceptuels en français, en utilisant les concepts du thésaurus Larousse. Les différentes étapes de cette réalisation sont les suivantes :

- les vecteurs associés aux concepts de base sont créés : ceux-ci ne sont pas indépendants (par exemple « guerre » et « paix ») ; cette interdépendance est modélisée par la projection de certains autres concepts de base sur ces vecteurs, en fonction de leur distance dans l'arbre des concepts ou de choix manuels ;
- on effectue un *amorçage manuel* avec des vecteurs précalculés pour obtenir le noyau de la base ;
- les vecteurs sont ensuite raffinés : de nouveaux vecteurs sont créés, à partir de définitions (analysées en des arbres linguistiques lors de ce processus en utilisant l'analyseur *Sygfran*) de différentes sources (dictionnaires, listes de synonymes, indexations manuelles) et des vecteurs existants ; le raffinement est fait en boucle, en espérant que les vecteurs tendent ainsi vers la meilleure modélisation possible.

Par exemple, dans l'expérimentation décrite, le vecteur correspondant aux projections des différents sens du terme « existence », tel que présenté dans (Schwab et al., 2004), est :

$V_{existence}$ =
{EXISTENCE[0.82], VIE[0.44], IDENTITÉ[0.38], ÉTAT[0.33], …}

1.2.2.4 Applications

Les vecteurs conceptuels ont déjà été utilisés pour trouver des valeurs correspondant à des fonctions lexico-sémantiques paradigmatiques.

On s'en est ainsi servi pour extraire des *synonymes* (Lafourcade & Prince, 2001) :

- *relatifs* (à un troisième terme, par exemple « trier » et « ranger » sont synonymes relativement à « ordonner ») ;
- *subjectifs* (les termes peuvent être confondus dans certains domaines, tout en devant être différenciés dans d'autre, cela dépend du *point de vue*).

Ils peuvent également être utilisés pour obtenir des *antonymes* (Schwab et al., 2002) :

- *complémentaires* (« l'affirmation de l'un entraîne la négation de l'autre », comme « vie »/« mort ») ;
- *scalaires* (deux extrémités d'un système échelonné qui peuvent être non vérifiées en même temps, comme « chaud »/« froid », avec « tiède » au milieu de l'échelle) ;
- *duals*[70] (comme « vendre »/« acheter » ou « question »/« réponse »).

1.2.2.5 Utilisation comme filtre sémantique

Nous proposons ici l'utilisation des vecteurs conceptuels pour d'autres fonctions lexico-sémantiques : les FLS *syntagmatiques* qui ont un sens (ce qui exclut les verbes support).

Le problème que nous devons résoudre est celui du filtrage des candidats que nous produisons, notre but précis pour l'expérimentation étant de reconnaître et distinguer les collocations représentant la fonction lexico-sémantique `Magn` (l'*intensification*).

[70]Orthographe utilisée dans la thèse de Schwab (2005) et dans ses autres travaux.

Une solution pourrait être de définir une classe de collocatifs (puisque ce sont eux qui portent la modification du sens) qui expriment l'intensification.

Les vecteurs conceptuels pourraient nous servir à obtenir une telle classe. En effet, nous l'avons vu, la distance angulaire entre deux vecteurs est censée permettre d'établir la proximité thématique éventuelle entre les termes correspondants : si nous sélectionnons les vecteurs les plus proches de celui d'*intensité* (un des 873 concepts de base), nous obtiendrons une liste de termes censés être les plus proches du thème « intensité », et donc, nous l'espérons, les plus aptes à représenter une intensification.

Un tel filtre a l'avantage d'augmenter la précision de l'extraction. Il faut toutefois reconnaître qu'il permettra de trouver surtout des collocations assez transparentes, et fera manquer beaucoup de collocations plus opaques ; mais cela a néanmoins un intérêt certain, à cause de la non-prédictibilité des collocations.

En pratique, nous avons formé notre classe de collocatifs en prenant les adverbes dont les vecteurs étaient parmi les 500 plus proches du vecteur « intensité ».

1.2.2.6 Raffinement manuel

Parmi les adverbes figurant dans la liste ainsi produite, certains ne permettront manifestement jamais d'exprimer l'intensification, soit parce que le concept d'intensité y est bien présent, mais ne correspondant pas à l'intensification d'un verbe (« trop », « en partie », « très », « par trop », « peu », « de plus », « d'autant », « tant », « partiellement », « tellement »), soit parce que le sens est plus éloigné (« à gauche »). Nous proposons donc d'ajouter une étape supplémentaire au filtrage, qui consiste en la validation manuelle de la liste produite à partir des vecteurs conceptuels.

1.2.3 Verbes d'action

Le filtrage proposé jusqu'ici concerne le sens de la collocation (intensification ? appréciation positive ? etc.), et se concentre donc sur les collocatifs (adverbiaux) puisque ce sont eux qui expriment ce sens.

Mais il existe également une source d'erreur, qui se situe au niveau de la base (verbale). En effet, la modification exprimée par un adverbe ne peut pas porter :

- sur un *auxiliaire* (« être », « avoir »), dont le sens est vide ;
- sur des *verbes d'état* (« être », « rester », « demeurer », « sembler », « devenir », « paraître », etc.) ou *de changement d'état* (« mourir », « naître », « tomber », etc.)

Nous souhaitons donc limiter la possibilité d'être une collocation aux candidats dont la base est un verbe d'action. Pour ce faire, nous utilisons une liste d'exceptions (« stoplist ») contenant les auxiliaires et les verbes d'état et de changement d'état, dont la présence comme base supposée d'un candidat entraîne l'élimination de celui-ci.

1.3 Déroulement de l'extraction

La démarche d'expérimentation monolingue que nous présentons se fait dans l'ordre suivant, à travers des modules (ad hoc) écrits en C :

- *import des sorties de l'analyseur* (qui peuvent être de formes très diverses) dans un format qui nous sert de standard ;
- *identification des couples* de termes qui co-apparaissent :
  - en *contexte de dépendance* : d'après l'arbre de dépendances produit par l'analyseur, on conserve les relations entre un verbe et un adverbe correspondant à une modification du verbe ;
  - en *contexte linéaire* (séquentiel) : d'après les informations syntaxiques produites par l'analyseur, on parcourt les données à la recherche d'un verbe suivi d'un adverbe, et on génère donc une relation si le terme lu est un adverbe et si le précédent est un verbe ;
- *production des candidats* et *classement des candidats* (dans un même module pour éviter de répéter inutilement les parcours des données) :
  - parcours de tous les termes (identifiés par leur couple <lemme, partie du discours>) et calcul de leurs nombres d'occurrence ;
  - parcours de tous les couples de termes qui co-apparaissent (identifiés par leur couple <lemme du verbe, lemme de l'adverbe>) et calcul du nombre des co-occurrences ;
  - calcul de la mesure WMI associée à chaque couple, à partir des effectifs (nombres d'occurrence et de co-occurrence) précédemment calculés ;
- filtrage (éventuel)
  - sortie des candidats dans l'ordre décroissant de leur mesure WMI.

1.4 Évaluation

Les expérimentations ont été effectuées sur le corpus *LeMonde95*, qui contient tous les articles publiés par le journal *Le Monde* au cours de l'année 1995. Nous avons utilisés les résultats produit par l'analyseur de dépendances XIP (de Xerox) sur ce corpus.

	Langue	*Expérimentations*	*Documents*	*Phrases*	*Mots*
LeMonde95	FR	monolingues	47 646	1 016 876	24 730 579

Tableau 8: Caractéristiques du corpus utilisé pour les expérimentations monolingues

1.4.1 Expérimentations

Nous avons mené quatre expérimentations différentes d'extraction, respectant le déroulement décrit plus haut :

1. contexte de *dépendance, aucun filtrage.*
2. contexte de *dépendance*, filtrage :
   - des verbes d'action ;
   - des adverbes : par *vecteurs conceptuels*.
3. contexte *linéaire*, filtrage :

   - des verbes d'action ;
   - des adverbes : par *vecteurs conceptuels*.
4. contexte *linéaire*, filtrage :
   - des verbes d'action ;
   - des adverbes : par *vecteurs conceptuels* raffinés manuellement.

Ces expérimentations avaient pour but de mener 3 comparaisons pour évaluer les efficacités respectives de 3 facteurs :

- le *filtrage* par vecteurs conceptuels (et verbe d'états) : 1 et 2 utilisent le même contexte d'extraction, seul le filtrage diffère ;
- le *contexte d'extraction* : 2 et 3 utilisent le même filtrage, seul le contexte diffère ;
- le *raffinement manuel* : 3 et 4 utilisent le même contexte, et en partie le même filtrage; seul diffère la présence ou non d'un raffinement manuel des vecteurs conceptuels.

1.4.2 Méthode d'évaluation

Pour évaluer la qualité de l'extraction, nous calculons une mesure de *précision* pour chaque expérimentation, c'est-à-dire la proportion de candidats corrects parmi ceux extraits.

Contrairement à ce qui se fait classiquement en recherche d'information, nous ne pouvons pas calculer de rappel (la proportion d'objets de référence retrouvés par le processus), en l'absence d'une base de référence : on manque de ressources pour cela, et il est bien plus facile de connaître, en recherche d'information, les documents pertinents pour une requête, le nombre total des candidats (documents) étant un ensemble connu et fini, ce qui n'est pas le cas des collocatifs éventuels d'une base. L'évaluation est nécessairement subjective (en évaluant la qualité des candidats produits), il n'y a pas non plus d'*étalon-or* (« gold standard ») dans l'extraction de collocations.

Par ailleurs, nous ne comparons pas les résultats des expérimentations de la démarche proposée à des travaux existants, par défaut : les « collocations » que cherchent à extraire ces travaux ne sont le plus souvent pas définies de la manière dont nous l'avons fait (il est vain de comparer des travaux ne visant pas à extraire la même chose).

De plus, la restriction que nous effectuons sur le sens porté par la collocation rend nos travaux sans comparaison possible à l'heure actuelle.

1.4.3 Résultats et observations

Le tableau 9 présente la précision calculée pour chaque expérimentation, qui correspond à la proportion de candidats corrects (nous considérons qu'un candidat est correct s'il s'agit bien d'une collocation exprimant l'intensification) parmi *les 1 000 premiers candidats* (couples de termes) classés selon la mesure WMI (présentée page 80).

	1	*2*	*3*	*4*
Contexte	Dépendance	Dépendance	Linéaire	Linéaire
Filtrage des verbes	Aucun	Verbes d'action	Verbes d'action	Verbes d'action
Filtrage des adverbes	Aucun	Vecteurs conceptuels	Vecteurs conceptuels	Vecteurs conceptuels+raffinement
Précision	**17%**	**41%**	**44%**	**83%**

Tableau 9: Précision des expérimentations monolingues

1	**2**	**3**	**4**
mettre fin[71]	expliquer en partie	juger trop	*régner sans partage*
valoir mieux	savoir trop	expliquer en partie	*changer radicalement*
coûter cher	*régner sans partage*	*régner sans partage*	*réduire considérablement*
vouloir pas	juger trop	montrer très	*devoir beaucoup*
mettre en avant	savoir très	savoir trop	*ignorer superbement*
entrer en vigueur	*changer radicalement*	ancrer à gauche	*assumer pleinement*
manquer pas	ancrer à gauche	*changer radicalement*	*parler beaucoup*
hésiter pas	*réduire considérablement*	*réduire considérablement*	*aimer beaucoup*
pouvoir pas	*aimer beaucoup*	réduire d'autant	*exercer pleinement*
lire ci-dessous	importer peu	financer en partie	*montrer particulièrement*
empêcher pas	financer en partie	*devoir beaucoup*	*varier considérablement*
suffire pas	*devoir beaucoup*	coûter très	*consacrer entièrement*
lire ci-contre	coûter très	*ignorer superbement*	*augmenter considérablement*
devoir pas	réduire d'autant	*assumer pleinement*	*jouer pleinement*
savoir pas	*assumer pleinement*	voter à gauche	*faire beaucoup*
agir là	*ignorer superbement*	importer peu	*profiter pleinement*
payer cher	*parler beaucoup*	progresser de plus	*ressembler beaucoup*
cacher pas	voter à gauche	augmenter de plus	*compter beaucoup*
peser lourd	*consacrer entièrement*	*parler beaucoup*	*modifier de fond en comble*
devoir donc	*exercer pleinement*	*aimer beaucoup*	*affectionner particulièrement*

Tableau 10: Vingt premiers candidats produits pour chaque expérience monolingue

Le tableau 10 présente les premiers candidats produits par chacune des expérimentations monolingue (les candidats corrects, c'est-à-dire qui sont bien des collocations d'intensification, sont en italique).

[71]Est bien une collocation (« mettre » est un verbe support de « fin ») mais pas du type recherché (intensification).

1.4.3.1 Filtrage sémantique

Le filtrage sémantique simple sur les verbes et les adverbes permet de multiplier la précision par 2,5. Il faut toutefois remarquer que ce filtrage ne fait pas qu'éliminer des candidats incorrects, il élimine aussi également beaucoup de vraies collocations (près de 80%, ce qui n'est bien sûr pas satisfaisant : le filtrage est trop restrictif).

Parmi les collocations retrouvées lors de la 1ère expérimentation qui disparaissent lors de ce filtrage, il y a non seulement des collocations assez opaques, le plus souvent basées sur la métaphore (« subir de plein fouet », « reprendre de plus belle », « mener tambour battant », « défendre bec et ongles », « manquer cruellement »), mais aussi un nombre non négligeable de collocations relativement décodables (« refuser obstinément », « respecter scrupuleusement », « fréquenter assidûment »).

Si le filtrage permet d'augmenter la précision, il fait donc également baisser le rappel. Il faudrait, en jouant sur les seuils, trouver un compromis acceptable ; cependant, il serait difficile d'évaluer le meilleur compromis, en l'absence de calcul de rappel, et par conséquent de *F-mesure*.

1.4.3.2 Contexte

Contexte	*Co-occurrences*	*Couples distincts*	*Couple distincts après filtrage sémantique*
Linéaire	202 731	64 894	3 085
Dépendance	535 510	140 049	6 661

Tableau 11: Couples de termes en co-occurrence selon le contexte

L'idée qui sous-tendait l'utilisation de contextes de dépendance est que cette information (partielle) sur la syntaxe de la phrase permettrait peut-être d'améliorer la qualité des candidats produits par rapport au contexte linéaire seul. Ce n'est pas le cas, et l'expérimentation utilisant le contexte linéaire a même une qualité légèrement meilleure.

Pourquoi l'utilisation d'informations de dépendance donne-t-elle des résultats mitigés ?

Il génère certes plus de couples verbes-adverbes prétendument en relation de modification que le contexte linéaire, mais la validité des couples générés est moins bonne[72] (même s'il y a, au total, plus de candidats corrects produits avec le contexte de dépendances). En effet, si l'on peut avoir une confiance assez forte dans le fait qu'un adverbe qui suit immédiatement un verbe d'action (contexte linéaire) le modifie, cette confiance diminue au fur et à mesure qu'on s'éloigne du verbe.

Les erreurs possibles sont :

- au niveau de la *partie du discours* : reconnaissance erronée d'un terme comme un adverbe (ce qui est rare quand il s'agit du terme suivant immédiatement le verbe) ;

[72]La qualité de l'analyseur de dépendances employé y joue un rôle important.

- au niveau des *dépendances* : attachement à tort d'un adverbe à un verbe (par exemple, l'adverbe peut être en réalité le modificateur au sein d'un syntagme nominal complément du verbe).

Si la proportion de couples valides baisse quand on utilise le contexte de dépendance, leur quantité (absolue) augmente néanmoins de manière significative, ce qui correspond pour ce problème précis à une baisse de la précision et à une augmentation du rappel, et mène sans doute à la même chose pour le problème global de l'extraction : les candidats corrects produits avec le contexte de dépendance (expérimentation 2) seraient plus nombreux que ceux produits avec le contexte linéaire (expérimentation 3), mais se retrouveraient dans un bruit plus fort.

1.4.3.3 Raffinement manuel

Le but du raffinement manuel est de guider le processus pour corriger les sorties du filtrage par les vecteurs conceptuels qui servent de filtre sémantique, afin d'améliorer ce filtre. L'intervention manuelle est à la fois simple et rapide : on lit la liste des adverbes proches thématiquement d'*intensité* d'après les vecteurs conceptuels, et on en retire ceux ne pouvant jamais exprimer l'intensification. Cette simple modification permet d'éliminer d'un seul coup 47% des candidats produits, et la précision passe alors à 83%.

Cependant, malgré un filtre sémantique relativement efficace après son ajustement manuel, il reste encore environ un candidat sur six incorrect. Cela est dû à la diversité des sens exprimés par un même terme suivant le contexte : s'il y a des adverbes qui servent presque à coup sûr à intensifier le verbe dont ils dépendent (« considérablement », « beaucoup », « extrêmement ») et d'autres qui ne servent jamais à cela (adverbes de lieu, de temps, de négation, de doute, etc., ainsi que des adverbes d'intensité ne pouvant exprimer l'intensification, comme « trop »), il y a également une troisième catégorie, celle des adverbes qui expriment ou non l'intensification selon les verbes qu'ils modifient : ainsi, « dessiner superbement » est une appréciation positive, mais « ignorer superbement » est une intensification. Cette troisième catégorie est sans doute la plus intéressante d'un point de vue linguistique, puisqu'elle provoque des collocations plus difficiles à identifier.

Cette considération sur les adverbes serait sans doute plus difficilement applicable aux adjectifs qui modifient les noms : les métaphores y apparaissant davantage, et il est beaucoup plus difficile de prétendre les séparer correctement en trois catégories comme précédemment (« noir » n'est-il jamais collocatif d'intensification ? Si, dans « colère noire »).

1.5 Bilan des expérimentations

1.5.1 Utilité

Nous avons présenté une extraction monolingue de collocations, produisant automatiquement une liste de couples de termes candidats classés d'après une mesure statistique basée sur les nombres d'occurrence et de co-occurrence, puis filtrée sémantiquement grâce à des données liées à la théorie des vecteurs conceptuels et à une intervention manuelle.

Les résultats prouvent l'intérêt des *vecteurs conceptuels* comme outil et ressource dans l'extraction de collocations. Nous avons choisi les vecteurs conceptuels parce qu'il existait

déjà une expérimentation faite avec une ressource de ce type, et dont on pouvait tirer facilement les informations dont nous avions besoin.

Il faudrait néanmoins pouvoir, dans le cadre d'un système plus large d'extraction de collocations (et sans doute pour d'autres informations linguistiques), y intégrer des vecteurs sémantiques[73] ; pour cela, la production de tels vecteurs à l'aide de LSA semble convenir tout à fait.

1.5.2 Enseignements sur le processus d'extraction

Cette première tentative d'extraction de collocations nous montre que le processus est relativement séquentiel et peut être divisé facilement en plusieurs opérations indépendantes : import, identification, production, classement, filtrage. La production des candidats et leur classement pourraient être séparés, mais on gagne en rapidité en les regroupant, car on élimine ainsi un parcours superflu : en même temps qu'on découvre les candidats, on calcule les effectifs, ce qui est nécessaire pour leur classement.

Tout ce qui concerne la production des filtres utilisés est secondaire : si cela a sa place dans un système plus large d'extraction de collocation, c'est comme outil complémentaire, pas au cœur de la démarche. Nous souhaitons justement, dans cette thèse, nous concentrer sur ce qui est au cœur de la démarche d'extraction.

2 Extractions bilingues contrastives

Après avoir étudié l'extraction monolingue, nous nous intéressons dans cette section aux problèmes bilingues : comment extraire des bicollocations à partir de corpus ?

2.1 Entités de référence associées

L'extraction de bicollocations est faite à partir de corpus des deux langues, en repérant tout d'abord les collocations monolingues. Le problème qui se pose ensuite est la manière d'associer entre elles les collocations pour former des bicollocations : par exemple, si on a extrait « petit fumeur » et « fumeur invétéré » en français, ainsi que « light smoker » et « hardened smoker », comment savoir quelle collocation française correspond à quelle collocation anglaise, en l'absence de relations de traduction entre ces collocatifs dans les dictionnaires ?[74]

Le meilleur moyen pour cela est de comparer les contextes d'emploi respectifs des collocations monolingues extraites : si un document français parle à peu près de la même chose qu'un document anglais, il est probable qu'on trouve, en les observant, des correspondances entre collocations, donc des bicollocations.

On peut distinguer deux types de corpus bilingues :

[73]Les vecteurs sémantiques, au contraire des vecteurs conceptuels, sont créés dans une base à k dimensions, k étant fixé avant le calcul, et les k dimensions ne correspondant *a priori* à aucun concept identifié.

[74]On peut certes avoir, dans certains cas (essentiellement des collocations opaques) la traduction de la tournure complète dans le dictionnaire, mais cela n'est pas généralisable.

- les *corpus parallèles*, dont les contenus sont traductions l'un de l'autre, et qui peuvent être alignés par documents, ou par phrases ;
- les *corpus comparables*, qui ne sont pas en relation de traduction, mais concernent le même thème.

2.1.1 Corpus parallèles

Les corpus parallèles semblent être le type de ressources privilégié pour l'extraction de bicollocations, lorsqu'ils sont alignés par phrases, voire au niveau sous-phrastique.

	Type	*Langues*	*Alignement français-anglais*		
			Unités d'alignement	*Mots français (mots par unité)*	*Mots anglais (mots par unité)*
Aligned Hansards of the 36th Parliament of Canada[75]	Débats du parlement canadien (1997-2000)	2 (anglais, français)	1 278 000	21 242 000 (17,4)	19 830 000 (11,5)
Europarl3	Débats du parlement européen 1996-2006	11[76]	1 333 785	43 988 004 (33,0)	39 431 862 (29,6)
JRC-AC	Lois de l'Union Européenne	22 (langues de l'UE[77] sauf l'irlandais)	1 266 712	39 100 499 (30,9)	34 588 383 (27,3)
Emea	Documents de l'agence européenne des médicaments	22 (langues de l'UE sauf l'irlandais)	891 091	14 513 025 (16,3)	30 580 774 (34,3)
EUConst	Projet de constitution européenne (2005)	21 (langues de l'UE sauf le bulgare et le roumain)	10 286	177 162 (17,2)	164 697 (16,0)
OpenOffice.org	Documentation de OpenOffice.org	6 (anglais, français, allemand, espagnol, suédois, japonais)	38 014	496 780 (13,1)	478 654 (12,6)
KDE4	Fichers de localisation de KDE4	86	166 384	1 913 313 (11,5)	2 007 971 (12,0)
PHP	Documentation de PHP	22	45 484	382 407 (8,4)	522 603 (11,5)

Tableau 12: Principaux corpus parallèles disponibles gratuitement

[75] Ce corpus ne doit pas être confondu avec celui habituellement désigné dans la littérature comme le « Hansard », contenant les débats au parlement canadien du milieu des années 1970 à 1988, qui est payant et commercialisé par le *Linguistic Data Consortium*.

[76] Anglais, français, allemand, espagnol, italien, portugais, néerlandais, grec, danois, suédois, et finnois.

[77] Allemand, bulgare, anglais, danois, espagnol, estonien, finnois, français, irlandais, grec, hongrois, italien, letton, lituanien, maltais, néerlandais, polonais, portugais, roumain, slovaque, slovène, suédois, et tchèque.

Au départ, les corpus parallèles sont généralement alignés seulement par documents. L'alignement par phrases est plus compliqué, car la liste des phrases d'un document traduit n'est pas la liste des traductions des phrases du document de départ. En effet, si la traduction d'une phrase par une autre est bien le cas le plus courant, ce n'est pas pour autant toujours le cas : une phrase du premier document peut être traduite par plusieurs phrases dans le second, plusieurs phrases du premier peuvent être traduites en une seule du second, d'autres phrases peuvent même apparaître ou être supprimées lors de la traduction.

Les méthodes pour résoudre le problème d'alignement par phrases sont le plus souvent basées (de manière combinée) sur :

- les *points d'ancrage* (mots identiques, mots en relation de traduction) trouvés dans les deux documents (Kay, 1988) ;
- la corrélation entre les tailles des phrases (Gale & Church, 1993).

On peut trouver une bonne revue des travaux d'alignement dans (Véronis, 2000). Parmi les outils existants d'alignement de phrases, on peut notamment citer *Champollion* (Ma, 2006) ou *Alinéa* (Duchet & Kraif, 2006). Cette tâche peut également être incluse dans un outil plus général comme *Giza++* (Och & Ney, 2003).

La nature même des corpus parallèles fait qu'il en existe peu de grande taille (à l'échelle du million de phrases). On peut cependant citer le corpus *Hansard* (débats du parlement canadien, en français et anglais), ou le projet « open source » OPUS[78], visant à collecter les corpus parallèles multilingues disponibles sur Internet comme les débats du Parlement Européen (Europarl), le projet de constitution européenne, ou les documentations de logiciels « open source ». Le tableau 12 présente les principaux corpus parallèles téléchargeables gratuitement, et leurs caractéristiques pour la paire de langues français-anglais.

2.1.2 Corpus de documents comparables

Si on peut espérer obtenir des bicollocations assez précises grâce aux corpus alignés, on peut aussi juger la restriction adoptée trop forte : deux documents peuvent très bien parler du même thème sans être pour autant traduction l'un de l'autre. L'idée est de considérer par exemple, que si « heavy smoker » et « gros fumeur » apparaissent dans deux documents ayant le même thème (le tabagisme), alors (compte tenu de la relation de traduction entre « smoker » et « fumeur ») il y a des chances pour qu'il s'agisse de deux collocations traductions l'une de l'autre.

2.1.2.1 Principe

On dit que des documents sont *comparables* lorsqu'ils parlent du même thème.

L'utilisation, pour l'extraction de bicollocations, de documents comparables plutôt que de documents alignés, permet d'espérer un meilleur rappel (mais sans doute une moins bonne précision).

Le moyen le plus facile pour obtenir des documents comparables est l'utilisation de corpus spécialisés dans un domaine particulier ; cependant, on risque ainsi d'extraire essentiellement des « *combinaisons lexicales spécialisées* » (L'Homme, 2003), ce qui ne nous intéresse pas vraiment, non seulement parce que nous voulons proposer une démarche qui puisse

[78] http://urd.let.rug.nl/tiedeman/OPUS/

s'appliquer également sur des corpus non spécialisés, mais aussi parce que le sous-langage d'un corpus technique est généralement très standardisé, et que les constructions métaphoriques pouvant rendre une traduction automatique incompréhensible y sont relativement rares.

Ici, nous souhaitons partir de deux corpus dont les contenus sont partiellement comparables, nous utiliserons pour cela dans nos expérimentations des corpus d'articles de journaux couvrant la même période : une partie des événements décrits y sera donc commune, et une partie seulement (une grande partie des articles d'un journal ou d'une agence de presse est en effet consacrée à des événements locaux, dont l'importance ne dépasse pas les frontières).

2.1.2.2 Établir la comparabilité

D'après Shinyama & Sekine (2003), les propriétés permettant de reconnaître des documents comparables de langues différentes sont les suivantes :

- on y retrouve les mêmes *entités nommées* (noms propres, noms de marques, etc.), les mêmes nombres.
- on retrouve la traduction littérale d'une partie des termes d'un document dans l'autre document : c'est parmi ces termes traduits mot à mot qu'on peut s'attendre à trouver les concepts définissant le thème (ils sont moins sensibles à la traduction que le reste de la phrase). Cette hypothèse est notamment utilisée en recherche d'information multilingue et est généralement suffisante.
- les articles décrivant un même événement sont souvent publiés dans un intervalle de temps court (cette propriété est loin d'être toujours vérifiée, mais elle présente l'avantage de réduire énormément le nombre d'associations éventuelles entre documents).

Le thème d'un document étant essentiellement contenu dans les noms (et les titres), on se restreindra ici à comparer les noms qu'on retrouve, directement ou après traduction, d'un document à l'autre.

Les seules informations que nous gardons dans les modèles de documents réalisés pour cette étape sont :

- leur date de publication ;
- les syntagmes nominaux qu'ils contiennent.

Pour évaluer la similarité entre deux ensembles de syntagmes nominaux, en tenant compte des différences de longueur des documents (rien n'empêche un document court et un document long d'être comparables s'ils parlent du même thème, il faut donc ne pas pénaliser ce cas), nous utilisons la mesure *overlap* (avec, ici, X et Y deux ensembles de syntagmes nominaux issus de deux documents différents), répandue en recherche d'information[79] :

$$overlap(X, Y) = \frac{|X \cap Y|}{min(|X|, |Y|)}$$

Nous introduisons une mesure de similarité thématique (et donc de comparabilité), entre deux documents, à partir d'*overlap* :

[79] Cette mesure est aussi connue sous le nom de *coefficient de Simpson* (1943)

$$Compar(D_{1,}D_2)=\frac{overlap(N_{1,}N_2)+overlap(trad(N_1),N_2)}{2}$$

où N_1 et N_2 sont les ensembles de syntagmes nominaux issus respectivement de D_1 et D_2, et $trad(N_1)$ l'ensemble des traductions des membres de N_1.

On conserve alors seulement les associations de documents pour lesquelles la valeur de comparabilité est supérieur à un seuil (fixé empiriquement à 0,2 dans nos expérimentations).

Le calcul des associations entre documents s'effectue en parcourant tous les documents du premier corpus. Pour chaque document D_1 rencontré :

- on traduit, à l'aide d'un dictionnaire, tous les syntagmes nominaux de D_1 (quand il y a plusieurs traductions possibles, on les conserve toutes) ;
- on parcourt tous les documents du second corpus publiés de deux jours avant à deux jours après la publication de D_1 et, pour chaque document D_2 de cette liste :
  - on calcule $Compar(D_{1,}D_2)$;
  - on crée une association entre D_1 et D_2 si la valeur de comparabilité obtenue est supérieure au seuil fixé.

La mesure utilisée pour la comparabilité des documents est certes relativement naïve et évidemment critiquable et améliorable. Mais il faut rappeler que la démarche présentée dans ce chapitre vise à décrire l'extraction de bicollocations à partir d'entités bilingues associées, et suppose que ces associations ont été faites préalablement à l'extraction ; l'association de documents de corpus comparables telle que nous l'avons faite avait pour but de combler de manière simple l'information qui nous manquait dans les ressources dont nous disposions.

2.2 Associations de termes et de collocations

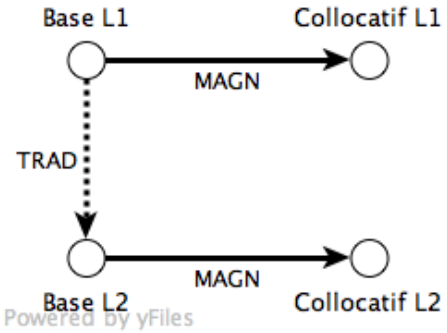


Figure 16: Liens entre les composantes d'une bicollocation d'intensification

2.2.1 Extraction contrastive

Nous cherchons ici à extraire des bicollocations (couples de collocations exprimant le même sens et dont les bases sont traductions l'une de l'autre). Si l'on met de côté la contrainte sur le sens en considérant que le problème sera réglé par ailleurs, il reste qu'il doit y avoir une relation de traduction entre les deux bases. Au contraire, les deux collocatifs sont

indépendants : il est possible que les choix arbitraires des collocatifs au niveau monolingue se portent sur des collocatifs dont le sens habituel est différent.

Nous parlons d'*extraction contrastive* pour désigner ce type d'extraction qui tient compte du contraste possible entre les sens habituels des collocatifs.

2.2.2 Apparitions conjointes dans des entités associées

Nous voulons utiliser les corpus bilingues pour établir des liens bilingues entre éléments de phrases (typiquement des mots) français et anglais. Le problème est donc de passer d'associations entre documents, voire entre phrases, à des associations bilingues entre mots. Une solution pourrait être d'utiliser l'alignement sous-phrastique (dans le cas où les phrases sont en relation de traduction) pour obtenir directement des relations de traduction entre mots ou syntagmes de la phrase. Mais cela n'est applicable qu'avec des corpus parallèles, et nous souhaitons proposer une approche qui s'applique aussi aux corpus comparables.

La méthode de recherche d'associations bilingues entre termes que nous utilisons, plus naïve, se base simplement sur :

- des *associations entre entités de référence* (documents *alignés* ou *comparables*, phrases *alignées*) ;
- un *dictionnaire*, contenant des relations de traduction entre termes.

On considère alors qu'il existe une relation d'association bilingue entre les termes *t1* et *t2* s'ils appartiennent respectivement à des entités de référence *e1* et *e2* associées, et si *t1* et *t2* sont traductions d'après le dictionnaire.

2.2.3 Mesure d'associabilité des collocations

Nous souhaitons que notre mesure de bicollocabilité tienne non seulement compte des collocabilités respectives de ses composantes monolingues, mais aussi (et surtout) de la fréquence avec laquelle celles-ci se retrouvent conjointement dans des documents associés. Par exemple, plus que de savoir quelle confiance on peut avoir dans l'association entre les bases « rain » et « pluie », c'est savoir celle qu'on peut avoir dans l'association entre « driving rain » et « pluie battante » qui nous intéresse.

Nous dirons par définition que *les occurrences de deux collocations sont en association bilingue s'il est établi que leurs bases respectives sont en association*, c'est-à-dire si elles sont traductions l'une de l'autre d'après un dictionnaire, et si elles apparaissent souvent conjointement dans des entités associées.

Pour tenter d'évaluer l'associabilité entre deux collocations, nous nous sommes inspirés de la mesure cosinus, souvent utilisée en recherche d'information (X et Y étant les ensembles respectifs de documents où apparaissent x et y) :

$$\cos(x, y) = \frac{|X \cap Y|}{\sqrt{|X| \times |Y|}}$$

Cette mesure ne peut correspondre à notre problème, car les objets que nous voulons comparer ne co-apparaissent pas directement dans des entités textuelles, mais apparaissent dans des entités associées.

Nous définissons donc une adaptation de cette mesure au problème d'association bilingue, que nous nommons « cosinus bilingue » ; cette mesure vise à modéliser l'associabilité entre deux collocations (de langue différente) selon les apparitions conjointes de ces collocations dans des entités associées :

$$\cos_{bilingue}(c_{1,}c_2)=\frac{|AE(c_{1,}c_2)|}{\sqrt{|C_1|\times|C_2|}}$$

où C_1 et C_2 sont les ensembles respectifs des entités où apparaissent c_1 et c_2, et AE(c_1,c_2) l'ensemble des associations d'entités $\langle e_{1,}e_2\rangle$ où e_1 et e_2 contiennent respectivement c_1 et c_2.

Le comptage des occurrences d'associations entre collocations est fait en parcourant toutes les associations d'entités.

Pour chaque couple d'entités $\langle E,F\rangle$ associées, on teste tous les couples de collocations $\langle e_i,f_j\rangle\in E\times F$ pouvant être liées. Si l'on peut craindre légitimement une explosion combinatoire, le fait de se restreindre aux seules co-occurrences nous intéressant (avec une certaine partie du discours pour chaque composant, plus éventuellement une certaine relation entre eux) permet de limiter grandement le phénomène[80].

2.2.4 Mesure de bicollocabilité

La mesure qui doit évaluer la bicollocabilité d'un candidat doit :

- modéliser avant tout l'associabilité des deux composantes monolingues ;
- favoriser les candidats dont les *deux* composantes monolingues sont probablement des collocations, d'après la mesure de collocabilité monolingue : nous utilisons la somme des deux mesures monolingues (cette somme ayant les résultats les plus élevés lorsque les deux mesures monolingues sont positives).

La mesure de bicollocabilité que nous proposons est la suivante (c_1 et c_2 étant les deux composantes monolingues du candidat bicollocation) :

$$Bicoll(c_{1,}c_2)=[WMI(c_1)+WMI(c_2)]\times\cos_{bilingue}(c_{1,}c_2)$$

2.3 Déroulement de l'extraction

Le processus d'extraction de bicollocations est assez complexe, dans le sens où on peut le décomposer en trois traitements distincts, correspondants aux trois liens existant dans une bicollocation : les 2 monolingues (entre bases et collocatifs) et la traduction entre bases.

Ces traitements reprennent chacun les phases décrites pour le traitement monolingue (voir figure 9 page 59 : analyse, identification, production, classement), et se croisent parfois, pour se rejoindre finalement (voir figure 17).

Toutes les opérations composant la démarche bilingue ont été réalisées à travers des modules (ad hoc) écrits en C :

- *Import monolingue* des données (conversion dans nos formats standard de stockage)
  - des *sorties de l'analyseur* « langue 1 » ;
  - des *sorties de l'analyseur* « langue 2 ».

[80]On se limitera par exemple, pour chaque langue, aux relations de modifications entre un verbe et un adverbe. Lors de nos expérimentations, nous avons pu constater qu'il y avait rarement plus de 4 couples de part et d'autre.

- *Import bilingue* des données (conversion dans nos formats standard de stockage) :
  - du fichier contenant des *correspondances bilingues* entre entités (l'alignement de phrases pour les corpus parallèles, l'association de documents comparables pour les corpus comparables).
- *Identification monolingue* des *couples d'occurrences* de termes qui co-apparaissent, pour le contexte défini, au niveau monolingue (pour les deux langues) ;
- *Identification bilingue* des relations de traduction entre occurrences de bases (dans le même module que la production des associations bilingues entre collocations, pour éviter de répéter les parcours) ;

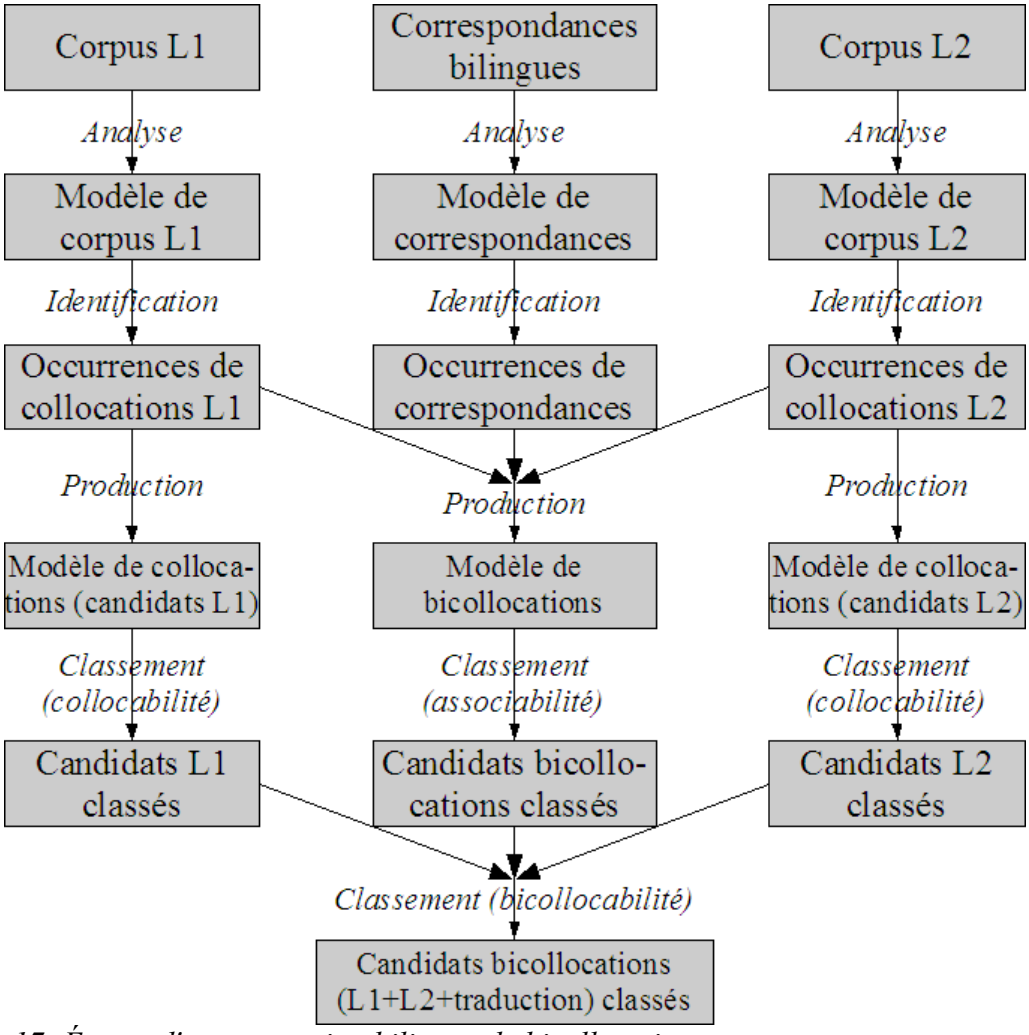


Figure 17: Étapes d'une extraction bilingue de bicollocations

- *Production* et *classement* monolingues (pour chacune des deux langues), comme pour l'extraction monolingue :
  - parcours des termes, pour calculer leur nombre d'occurrences ;
    - parcours des couples de termes qui co-apparaissent, pour calculer leur nombre de co-occurrences ;

        - calcul de la mesure de collocabilité (*WMI*) pour chacun de ces couples.
- *Production* et *classement* bilingues :
    - création de liens entre les bases qui apparaissent dans des documents associés et qui sont traductions l'un de l'autre : pour reconnaître les bases, il faut savoir quelles sont les collocations. On a donc besoin ici de l'information monolingue.
    - calcul, au fil du parcours, des nombres d'occurrence des collocations monolingues.
    - calcul, au fil du parcours, des nombres de co-occurrence des bicollocations (le nombre d'associations rencontrées entre deux collocations monolingues).
    - calcul du *cosinus bilingue* pour chacun de ces couples de collocations.
- *Classement final bilingue* :
    - parcours de tous les candidats bicollocations :
        - calcul de *Bicoll*, notre mesure de bicollocabilité.

2.4 Évaluation

Les expérimentations que nous avons effectuées ont concerné :

- les corpus comparables *LeMonde95* et *GH95* (articles publiés respectivement par *Le Monde* et le *Glasgow Herald* en 1995), analysés syntaxiquement avec XIP ;
- les composantes française et anglaise de la version 2 du corpus parallèle *EuroParl* (débats au parlement européen d'avril 1996 à septembre 2003), déjà étiquetées grammaticalement[81].

	Langue	*Expérimentations*	*Documents*	*Phrases*	*Mots*
LeMonde95	FR	comparable	47 646	1 016 876	24 730 579
GH95	EN	comparable	56 472	1 321 323	28 122 780
EuroParl-fr v2	FR	parallèle	495	1 089 670	31 115 677
EuroParl-en v2	EN	parallèle	491	1 064 462	25 089 232

Tableau 13: Caractéristiques des corpus utilisés pour les expérimentations bilingues

2.4.1 Expérimentations

Nous avons mené deux expérimentations d'extraction de bicollocations sur des corpus bilingues français-anglais :

1. corpus de *documents comparables*, avec association par documents (par nos soins, d'après la méthode que nous avons décrite) ;
2. corpus *parallèle*, contenant des associations par phrases (déjà calculées dans la version que nous utilisons).

[81]Corpus téléchargeable sur le site du projet OPUS : http://urd.let.rug.nl/tiedeman/OPUS/Europarl.php

Ces expérimentations utilisent le contexte linéaire (l'adverbe suit immédiatement le verbe), en partie parce qu'il donnait des résultats légèrement meilleurs au niveau monolingue, mais aussi parce que le corpus parallèle que nous utilisons pour notre expérimentation ne contient pas d'informations de dépendance que nous puissions exploiter.

Le filtrage « de base » que les processus effectuent est le suivant :

- filtrage des verbes : élimination des auxiliaires et des verbes d'état en français et en anglais ;
- filtrage des adverbes :
  - par *vecteurs conceptuels* en français ;
  - élimination des adverbes de temps, lieu, doute, etc. en anglais en utilisant une liste d'exceptions (« stoplist »), par manque de ressources immédiatement exploitables de type « vecteur sémantique ».

Pour le français, le raffinement manuel est celui décrit au chapitre précédent. Pour l'anglais, il s'agit surtout d'écarter des termes qui ne sont pas à proprement parler des adverbes indépendants qui pourraient modifier un verbe, mais font partie de la structure de verbes composés (« phrasal verbs ») comme par exemple « up » (« get up ») ou « apart » (« keep apart »), ou de syntagmes adverbiaux plus grands, comme « too », souvent reconnu seul quand il fait partie de « too much » ou « too many ».

2.4.2 Méthode d'évaluation

Comme pour l'extraction monolingue, et pour les mêmes raisons (absence de base de référence), nous calculons seulement une mesure de précision.

Il en va de même pour l'impossibilité de comparer les résultats de nos expérimentations à des travaux existants, rares, ne cherchant pas à extraire des bicollocations, et sans restriction sur le sens exprimé.

2.4.3 Résultats et observations

Expérimentations	*Nombre de candidats positifs selon le filtrage*		
	Sans	*Filtrage de base*	*Raffinement manuel*
comparable	80 298	3 973	201
parallèle	15 583	1 995	43

Tableau 14: Nombre de candidats produits par les expérimentations bilingues

Le tableau 15 présente les 20 premiers candidats produits par chacune des expérimentations bilingues. Les composantes monolingues correctes (celles qui sont bien des collocations) sont en italique ; les candidats corrects (ceux qui sont bien des bicollocations d'intensification) sont en gras.

Comparable		Parallèle	
mettre beaucoup	take seriously	*soutenir pleinement*	support wholly
jouer pleinement	*work hard*	*jouer pleinement*	work together
vouloir particulièrement	***want really***	***changer radicalement***	***change radically***
accomplir particulièrement	*perform strongly*	***modifier radicalement***	***modify radically***
regretter énormément	***regret deeply***	*jouer pleinement*	play right
mettre beaucoup	take long	*travailler intensivement*	*work hard*
jouer pleinement	work regularly	***contribuer grandement***	***contribute significantly***
exercer pleinement	*train specially*	***changer radicalement***	***change dramatically***
jouer pleinement	act unlawfully	***contribuer grandement***	***contribute considerably***
jouer pleinement	act responsibly	accepter partiellement	*accept fully*
prendre particulièrement	take seriously	***varier grandement***	***vary considerably***
jouer pleinement	*gamble compusively*	***augmenter prodigieusement***	***increase dramatically***
graver beaucoup	cut cleanly	***varier grandement***	***vary widely***
concerner énormément	relate directly	***augmenter radicalement***	***increase dramatically***
appeler carrément	grow steadily	***investir abondamment***	***invest heavily***
exercer pleinement	***practise widely***	*tenir pleinement*	keep constantly
jouer pleinement	gamble undoubtedly	***condamner radicalement***	***condemn outright***
jouer pleinement	*act harshly*	***opposer radicalement***	***oppose totally***
jouer pleinement	work endlessly	***souffrir terriblement***	***suffer terribly***
jouer pleinement	work manfully	*transformer radicalement*	change however

Tableau 15: Vingt premiers candidats produits pour chaque expérience bilingue

La première remarque que l'on peut faire, avant même d'analyser la précision des extractions, concerne le nombre de candidats positifs produits (on ne s'intéresse qu'aux candidats positifs, les seuls susceptibles d'être réellement des bicollocations), assez décevant.

La combinaison des filtrages en français et en anglais réduit si considérablement le nombre de candidats susceptibles d'être des bicollocations qu'on peut difficilement envisager l'utilisation de cette démarche en l'état. Cela peut sans doute être amélioré en utilisant des filtres moins sélectifs, mais on risque alors d'obtenir une qualité beaucoup moins bonne, tant le problème se complique par rapport à l'extraction monolingue. En effet, deux collocations d'intensification dont les bases sont traductions mutuelles n'expriment pas forcément le même sens : cela peut venir des bases (qui, quand on observe le contexte, n'ont en fait pas le même sens) ou des collocatifs (exprimant des modifications légèrement différentes), ces problèmes seront détaillés dans la suite.

2.4.3.1 Comparaisons des résultats

Les tableaux 16 et 17 présentent l'évaluation des résultats produits lors des expérimentations sur les corpus comparables et parallèles.

Les « composants » français et anglais de la bicollocation sont les deux couples de termes monolingues (base-collocatif) formant la bicollocation.

Un composant monolingue du candidat est considéré comme valide s'il est bien une collocation.

La traduction est considérée comme valide si les composants français et anglais du candidat sont bien traductions l'un de l'autre.

Enfin, le candidat est valide (c'est-à-dire qu'il est bien une bicollocation) si ses composants monolingues et sa traduction sont valides.

Composant français		*Valide*		*Non valide*	
Composant anglais		*Valide*	*Non valide*	*Valide*	*Non valide*
100 premiers résultats	**Traduction valide**	**15**	1	0	0
	Traduction non valide	25	45	5	9
100 résultats suivants	**Traduction valide**	**5**	1	0	0
	Traduction non valide	22	61	2	9

Tableau 16: Précision pour l'expérimentation sur les corpus comparables (200 premiers résultats)

Composant français	*Valide*		*Non valide*	
Composant anglais	*Valide*	*Non valide*	*Valide*	*Non valide*
Traduction valide	**19 (36,5%)**	0	1	6
Traduction non valide	2	9	2	4

Tableau 17: Précision pour l'expérimentation sur les corpus parallèles (43 premiers résultats)

On peut constater que l'extraction utilisant des corpus parallèles donne, comme attendu, une meilleure précision, avec des réserves dues au fait que l'évaluation porte sur un nombre relativement faible de candidats positifs produits pour chaque expérimentation.

Parmi les candidats incorrects au niveau du composant anglais, on retrouve des adverbes comme « rather », pouvant parfois exprimer l'intensification, mais on peut aussi avoir quelque chose d'incompréhensible sans connaître le reste de la phrase (« sell rather », « make rather »). Une autre source d'erreur vient de locutions non reconnues comme un seul élément par l'analyseur syntaxique, comme « naturally speaking » ou « take (something) further ».

2.4.3.2 Considérations sur les bicollocations

Ces expérimentations mettent en avant un fait qui peut surprendre : il n'est pas suffisant d'avoir deux relations exprimant l'intensification, avec une relation de traduction entre leurs bases, pour avoir une bicollocation. Il y a plusieurs explications possibles :

- les *bases* ne sont pas aussi synonymes qu'on pouvait l'espérer. Le coupable est ici la *polysémie* : la base française se traduit dans certains cas vers la base anglaise… mais pas dans le cas où elle est employée avec son collocatif. Un exemple rencontré est « exercer pleinement »/« train specially », où « pleinement » correspond à un emploi de « exercer » dans un sens précis (mettre en œuvre, utiliser), alors que « train » est une traduction pour un autre sens (dresser, entraîner).
- les bases sont bien employées dans le même sens, mais la modification peut *ne pas porter sur la même composante du sens*. Un exemple rencontré est « accomplir particulièrement »/« perform strongly » : l'intensification porte, dans le premier cas, sur l'accomplissement lui-même, et dans le second cas, sur la manière dont l'accomplissement est réalisé.

2.5 Bilan des expérimentations

2.5.1 Utilité

Les expérimentations présentées dans cette section montrent qu'on peut extraire des bicollocations à partir de corpus. Une réserve est à apporter sur le nombre de résultats produits, conséquence du filtrage sémantique : ce filtrage n'a pas la même importance selon qu'on utilise des corpus parallèles ou comparables.

En effet, dans le cas des corpus parallèles, avec des co-occurrences basées sur des phrases alignées, on peut avoir une certaine confiance dans le lien de traduction entre les deux composantes monolingues des candidats, une partie du rôle du filtrage sémantique étant de séparer les candidats exprimant l'intensification des autres. Dans le cas des corpus comparables, les co-occurrences sont basées sur des documents dont le thème est similaire, et produit des candidats (au rôle de bicollocation) dont les composantes ont beaucoup moins de chance d'être traductions mutuelles, ce qui rend l'étape de filtrage très importante.

Si l'on peut réduire la sévérité du filtrage et garder une qualité honorable lorsqu'on utilise des corpus parallèles, cela ne peut être le cas avec des corpus de documents comparables.

Il semble donc que les corpus parallèles (ou toute sorte de donnée sémantique permettant d'obtenir des relations de traduction assez sûres entre éléments du texte) sont la ressource privilégiée pour l'extraction de bicollocations, mais les problèmes liés à la taille, à la disponibilité, et au coût éventuel de ces corpus en restreignent l'usage.

2.5.2 Enseignements sur le processus d'extraction

Les travaux présentés dans ce chapitre montrent la complexité de l'extraction de bicollocations. Néanmoins, on peut séparer facilement ce qui concerne les collocations au niveau monolingue (pour chaque langue) et les relations de traduction au niveau bilingue. Les calculs relatifs aux collocations monolingues peuvent être effectuées indépendamment du reste, comme s'il s'agissait d'une extraction monolingue.

La complexité de la démarche vient en fait de la partie proprement bilingue, qui dépend des calculs monolingues, pour passer de relations entre bases à des relations entre collocations, qui sont des occurrences de bicollocations. On peut néanmoins décomposer aussi facilement le processus d'extraction en étapes (import, identification, production, classement).

Si cette extraction fait intervenir beaucoup de calculs différents, ceux-ci peuvent être facilement isolés et exécutés distinctement, ce qui encourage la création d'outils permettant d'effectuer cette extraction non pas comme un bloc, mais comme un enchaînement de calculs simples.

Conclusion

Nous avons présenté dans ce chapitre plusieurs expérimentations, monolingues et bilingues, visant à l'extraction de connaissances (des collocations) à partir de corpus. Ces expérimentations ont notamment montré l'intérêt d'utiliser des connaissances sémantiques telles que les vecteurs sémantiques, puis les vecteurs conceptuels pour filtrer les résultats, ainsi que celui d'une intervention humaine dans le processus afin de guider les résultats.

Le déroulement de l'extraction peut se décomposer en plusieurs phases :

1. l'*analyse*, qui permet d'obtenir un modèle des données traitées ;
2. l'*identification*, qui sélectionne, parmi toutes les informations du modèle, celles qui sont pertinentes pour ce qu'on souhaite extraire ;
3. la *production*, qui utilise les informations sélectionnées pour générer des candidats ;
4. le *classement*, qui range les candidats dans l'ordre décroissant de leur capacité à être du type (de connaissances) recherché.

Dans le cas bilingue (facilement généralisable à un cas multilingue), le processus peut être décomposé en plusieurs parties (pour les différentes langues, et pour les informations purement interlingues), reprenant chacune le déroulement décrit précédemment, et pouvant interagir.

On peut donc voir l'extraction de collocations, et plus généralement (ce sera une hypothèse pour la suite) pour l'extraction de connaissances linguistiques variées, comme une succession d'étapes, dont les résultats peuvent éventuellement se combiner. Cela doit permettre la spécification d'outils pour l'extraction qui soient à la fois simples (afin d'être facilement utilisable par des linguistes non informaticiens), et génériques, en décomposant le processus pour le rendre plus modulaire.

Chapitre V - Un problème d'outils et de modélisation

Au centre de cette thèse se trouve l'extraction de connaissances linguistiques, et plus précisément son amélioration. Pour mieux cerner les problèmes posés, nous avons réalisé diverses expérimentations d'extraction (monolingue et bilingue) de collocations dans le chapitre précédent, en définissant des algorithmes dont nous avons réalisé une implémentations *ad hoc* (que nous n'avons pas décrite en détail car elle ne possède pas d'intérêt particulier). Ce faisant, nous nous sommes rendus compte qu'il devrait être possible de définir les méthodes et algorithmes d'extraction dans un cadre plus générique, et aussi de les implémenter dans un cadre informatique.

Nous allons ici étudier comment on peut améliorer concrètement les travaux d'extraction de connaissances linguistiques. Nous verrons d'abord les différentes façons de le faire, et pourquoi nous avons choisi de nous concentrer sur les outils, et plus particulièrement sur la définition d'un modèle pour l'extraction de connaissances à partir de données telles que des corpus. Nous présenterons ensuite le « cahier des charges » que doit respecter, selon nous, un tel modèle, et pourquoi il doit reposer sur une structure de graphes. Enfin, nous étudierons les diverses utilisations des graphes en traitement des langues (comme outils du processus, comme ressources ou comme résultats), ainsi que les différentes structures existantes, pour préciser les contraintes que devra respecter notre modèle (en ce qui concerne sa structure et sa façon de représenter les différentes étapes de l'extraction).

1 Améliorer l'extraction

Nous avons procédé à plusieurs expérimentations visant à reconnaître un phénomène linguistique particulier, celui des collocations, (utilisant pour cela des mesures d'association) et les avons décrites dans le chapitre précédent. Les observations faites à propos de ces expérimentations ouvrent de nombreuses perspectives, sur l'augmentation de la qualité et sur les outils à adapter ou à construire.

1.1 Qualité de l'extraction

Nous pourrions axer nos recherches sur l'extraction de collocations, en cherchant à obtenir le processus qui donne les meilleurs résultats, tant au niveau de la qualité que de la quantité. En effet, chaque module composant un processus d'extraction peut être amélioré (et doit donc être étudié dans ce but). Ce sont les suivants :

- le pré-traitement des données ;
- la reconnaissance des candidats :
  - les informations à utiliser pour guider le processus

        - informations sémantiques complémentaires (comme les vecteurs sémantiques) : le processus étant dépendant de ces informations, choisir des données disponibles seulement pour l'anglais rendrait le processus inapplicable dans les autres langues ;
        - informations linguistiques (patrons) ;
        - informations statistiques (co-occurrences)
    - l'utilisation de ces informations ;
- le classement des candidats : le choix de la mesure évaluant la capacité recherchée.

Il serait également très intéressant d'exploiter la généricité de certaines opérations composant le processus d'extraction de collocations, de manière à les réutiliser pour réaliser d'autres tâches d'extraction plus ou moins proches.

1.2 Outils pour l'extraction

La définition d'un processus d'extraction n'est pas le seul axe de recherche possible ; on peut également se consacrer aux outils (informatiques) permettant d'assurer la mise en œuvre d'une extraction d'après la démarche spécifiée. Un bon outil doit être facilement disponible ; il doit également poser le moins possible de restrictions sur son utilisation : il doit gérer divers types d'extraction, sur différents types de ressources.

1.2.1 Un manque d'outils

1.2.1.1 Problème de disponibilité

Lorsque nous avons voulu réaliser les expérimentations présentées dans le chapitre précédent, nous avons été confronté au manque d'outils permettant l'extraction de collocations.

En effet, si un certain nombre de travaux ont déjà été consacrés à l'extraction de collocations, la mise en œuvre y est vue comme un problème secondaire, ne servant qu'à permettre la validation des hypothèses par l'expérience. L'implémentation de l'extraction décrite est donc le plus souvent faite par des programmes *ad hoc*, ce qui pose plusieurs problèmes :

- ils ne sont pas rendus disponibles à la communauté ;
- leur réalisation nécessite d'avoir une double compétence en linguistique (pour décrire le processus) et en informatique (pour l'implémenter).

Parfois, également, la mise en œuvre de l'extraction n'est qu'une partie de systèmes plus gros, non accessibles librement.

1.2.1.2 Des outils trop spécifiques

On peut citer, parmi les logiciels existants destinés à traiter facilement des données linguistiques, INTEX/NooJ (Silberztein, 2005), ou LinguaStream (Widlöcher & Bilhaut, 2007) qui est fondé sur l'enchaînement de modules d'analyse paramétrables. Ces logiciels sont limités à des tâches d'analyse (lexicale, syntaxique, etc.) de documents et, bien que disposant d'une modélisation performante des données, ne permettent pas de mener à bien des tâches d'extraction. D'ailleurs, même si des programmes génériques pour l'extraction de collocation

existaient et étaient utilisables librement, ils seraient difficilement réutilisables pour un autre type d'extraction.

Les programmes existants sont écrits dans le but de mettre en œuvre une solution particulière. Ils sont donc trop centrés sur la vision spécifique à cette solution, ne mettant en œuvre qu'un processus précis faisant des hypothèses sur les données en entrée, sur les informations calculées, etc.

Cela a plusieurs conséquences sur la mise en œuvre d'un processus par un outil existant d'extraction :

- le processus à tester n'est pas nécessairement compatible avec l'outil, il peut comporter des opérations non prévues par celui-ci, comme l'utilisation, dans nos expérimentations, de vecteurs sémantiques (nous nous sommes en fait limité à des vecteurs conceptuels) comme filtres ;
- les diverses hypothèses faites sur les données et les mesures peuvent s'avérer trop restrictives lorsqu'on veut réaliser une extraction avec un autre type de données (sous un autre format), qu'on veut obtenir le contexte de co-occurrence d'une autre manière, ou qu'on veut calculer la collocabilité avec une autre mesure, etc.

Nous n'avons donc pas pu adapter des outils existants pour réaliser l'extraction voulue, ni pu adapter notre problème pour qu'il soit soluble en les utilisant.

La trop grande spécificité des outils développés a une autre conséquence, qui découle des précédentes : l'implémentation du processus est généralement faite d'un bloc, ne permettant pas d'isoler les opérations. La mise en œuvre de ces dernières n'est alors pas réutilisable pour des processus d'extraction d'autres types de connaissances linguistiques.

Ces inconvénients concernent les processus d'extraction en général, et donc également ceux visant à obtenir autre chose que des collocations.

1.2.2 La place des dictionnaires (et des autres ressources linguistiques)

L'extraction de connaissances lexicales est souvent centrée sur le corpus, dans lequel se trouve l'information qu'on veut mettre en évidence. Un dictionnaire utilisé lors d'un processus d'extraction n'est vu que comme un outil : l'ajout d'une nouvelle ressource implique souvent l'écriture d'un nouveau script (pour y lire correctement l'information), voire de nouvelles expérimentations.

Certes, la distinction entre corpus et dictionnaires peut être motivée par une différence de forme (les premiers relient des termes entre eux, alors que l'analyse des seconds produit des liens entre les occurrences de termes). Mais les uns comme les autres sont des ensembles de connaissances linguistiques reliées entre elles, et peuvent donc être vus comme des graphes.

Ainsi, plutôt que de réserver des traitements spécifiques à chaque type de ressources (dictionnaire et corpus), il nous semble plus pertinent de définir un traitement général des ressources linguistiques, qu'on adapterait selon le type. Dans ce nouveau cadre, l'extraction ne serait plus centrée uniquement sur le corpus, et on peut imaginer de nouveaux processus combinant plus efficacement les données du corpus et celles de dictionnaires.

1.3 Centrage sur un outil informatique générique

La définition d'un processus d'extraction (visant à améliorer les résultats actuels), tout comme la spécification d'outils pour ce genre de processus (puis leur réalisation), présente un intérêt certain et de nombreuses perspectives.

Notre pourrions donc étudier en détail l'extraction de collocations dans le but d'améliorer les différentes étapes du processus. Mais nous menons des recherches en informatique et nous considérons que le rôle des informaticiens dans le traitement des langues est avant tout de fournir aux spécialistes (linguistes) des outils leur permettant de mener leur tâche à bien, au mieux et le plus facilement possible.

C'est pourquoi nous avons choisi d'axer notre travail sur la spécification d'un outil destiné à rendre les tâches d'extraction (et leur guidage par les linguistes) plus facilement réalisables, en insistant notamment sur la possibilité de paramétrer le plus possible le déroulement de ces tâches.

2 Cahier des charges d'un outil pour l'extraction

La spécification d'outils pour l'extraction passe en premier lieu par la définition des contraintes que cet outil devra respecter. Elle concerne l'expression des processus de calcul, la représentation des données et plus particulièrement celle des « graphes linguistiques », structures ici fondamentales.

2.1 Processus

2.1.1 Simplicité des opérations

La qualité d'un outil repose en premier lieu sur sa simplicité d'utilisation. Cela est d'autant plus vrai dans notre cas, que les utilisateurs visés sont des linguistes avec des connaissances limitées en informatique.

Il faut donc que le modèle (sur lequel se base l'outil) fasse abstraction du type de connaissance, afin que sa mise en œuvre semble la plus « naturelle » possible et requière le moins possible d'explications préalables. Cela doit se traduire par l'utilisation d'un langage de haut niveau, permettant de manipuler (et de paramétrer) facilement les opérations du processus.

Nous devons certainement descendre à une plus faible granularité des opérations que ce que nous avons rencontré dans nos expérimentations précédentes (identification-production-classement) : il ne s'agit pas de convertir ces opérations-là en opérations de manipulation de données, mais de voir quelles opérations de base on peut enchaîner pour obtenir le même résultat.

2.1.2 Généricité des opérations

Un outil d'extraction doit être compatible avec le plus grand nombre possible de tâches d'extraction. Il faut pour cela qu'il soit relativement complet, et donc qu'il permette de réaliser toutes les opérations essentielles à de telles démarches.

Ces opérations doivent, pour la plupart, être *génériques*, et facilement *paramétrables*, afin d'être utilisables dans diverses tâches indépendantes. Cela passe par la dissociation des différentes composantes du processus, notamment de ce qui concerne l'extraction de connaissances en général, et de ce qui dépend du type de connaissances à extraire.

2.2 Données

2.2.1 Simplicité de la représentation

La *simplicité* est également requise pour la représentation des données. Quel que soit le type des données, leur représentation doit pouvoir être facilement interprétée : il faut qu'un humain puisse comprendre facilement comment écrire ou lire des informations dans la structure choisie, ce qui nous commande d'éviter les structures de données trop complexes.

Cette simplicité ne doit toutefois pas se faire au détriment de l'ampleur de la couverture des ressources, en éliminant les cas qui sembleraient trop complexes à modéliser : il faut tenter de représenter le plus clairement possible toutes les données, y compris les plus complexes. De plus, si la manipulation des données doit sembler simple aux yeux de l'utilisateur, elle doit également le rester pour un processus automatique.

2.2.2 Variété des ressources

Si les opérations définies dans le modèle doivent être utilisables dans plusieurs processus indépendants, elles doivent aussi l'être avec plusieurs types de données (corpus monolingues, corpus bilingues, dictionnaires, etc.). Elles doivent même permettre d'utiliser *différentes ressources de façon conjointe*, en combinant les connaissances qu'elles contiennent.

Le dictionnaire doit être vu comme une ressource d'informations comme une autre, telle que le corpus. Ainsi, le problème à modéliser n'est pas l'extraction à partir de corpus, mais bien l'extraction à partir de ressources linguistiques variées, pouvant être des corpus.

2.3 Graphes

Il nous faut donc mettre l'accent sur la modélisation des données, qui doivent être représentées de manière à être à la fois :

- rapidement compréhensibles par un humain ;
- facilement utilisables par un processus automatique, au niveau du *parcours*, mais aussi de la *modification* (ajout ou suppression d'informations) ;
- assez souples pour permettre de modéliser le plus possible de données (corpus monolingues, bilingues, dictionnaires, etc.).

Puisque les ressources linguistiques utilisées par l'extraction sont des relations entre objets linguistiques, la solution la plus immédiate et la plus simple est de représenter les données sous forme de graphes, dont les nœuds seront les objets linguistiques en question et les arcs les relations (binaires) entre ces objets.

Nous devons donc réaliser des outils permettant la gestion (de manière générique) de graphes représentant des données linguistiques. Il faut en particulier pouvoir effectuer diverses tâches (identification, production de candidats, classement, etc.) de manière indépendante. Le défi est donc de fournir un environnement de gestion de données linguistiques, basé sur la réalisation d'opérations simples de manipulation de graphes.

Les relations n'étant pas toujours binaires, il nous faudra également étudier la manière de représenter les relations n-aires.

3 Manipulation de graphes

Nous avons choisi de considérer l'extraction de connaissances comme un problème de manipulation de graphes. Nous devons donc définir un modèle de graphes adapté.

Commençons par étudier comment les graphes sont généralement utilisés pour le traitement des langues.

3.1 Extraction basée sur les graphes

De nombreux travaux de divers domaines liés au traitement des langues utilisent les graphes ; nous en donnons ici quelques exemples.

3.1.1 Extraction de connaissances

En *recherche d'information*, les graphes représentent fréquemment des distributions probabilistes, et sont le plus souvent des réseaux (pouvant impliquer les documents, les termes du texte, et les concepts) sur lesquels on peut réaliser de l'inférence (Turtle, 1991) ou de la propagation d'activation (Preece, 1981).

Dans *l'extraction d'information*, les graphes représentent souvent les relations basées sur la dépendance syntaxique ou la co-occurrence entre termes, et on leur applique des algorithmes de partitionnement pour obtenir des termes sémantiquement proches (Widdows & Dorrow, 2002) ou regrouper des n-uplets similaires (Hassan et al., 2006).

Ces outils n'utilisent les graphes que pour un problème précis, et supposent une interprétation préalable du corpus étudié pour produire les données constituant le graphe. On peut citer à l'inverse les travaux utilisant des graphes seulement au niveau de la phrase, comme *Intex* (Silberztein, 1993, 1999), son clone libre *Unitex* (Paumier, 2003), ou son évolution *NooJ* (Silberztein, 2005 ; Koeva et al., 2007).

Jijkoun & De Rijke (2007) utilisent des graphes orientés pour représenter divers types de structures linguistiques, afin de voir certaines tâches de traitement des langues comme des

transformations de graphes (vers des graphes d'un autre type) : leur méthode est basée sur l'apprentissage automatique de ces transformations.

3.1.2 Dictionnaires comme graphes

On peut également être intéressé, non pas par la liste des informations extraites, mais par le réseau que celles-ci forment. Pour produire de telles relations entre informations, il faut le faire à partir ressources existantes, comme les dictionnaires.

À partir de 1992, l'équipe de traitement des langues de *Microsoft Research* a créé un graphe sémantique orienté à partir du *Longman Dictionary of Contemporary English* (Dolan et al., 1993), pour utiliser ensuite le résultat comme outil d'analyse (Richardson et al., 1993). Ce travail, et le même effectué sur le *American Heritage Dictionary (3rd Edition)*, ont permis la réalisation de la base de connaissances *MindNet* (Richardson et al., 1998). Plus récemment, le système *Prox* (Gaume et al., 2006) calcule un graphe de proximité des entrées d'un dictionnaire à partir d'un premier graphe reliant un terme à un autre si le second est présent dans la définition du premier.

3.2 Modification du graphe

Y a-t-il un intérêt à appliquer directement des modifications sur la structure d'un graphe ? Nous prenons ici l'exemple des systèmes-Q, définis comme « des ensembles de règles de réécriture portant sur des suites de symboles » (Colmerauer, 1970). Les systèmes-Q, qui furent utilisés pour écrire le système METEO permettant de traduire les bulletins météorologiques canadiens de l'anglais vers le français et inversement, sont à l'origine du langage de programmation logique *Prolog* (Colmerauer & Roussel, 1992).

Les suites de symboles sont représentées par des graphes orientés. Chaque arc porte un arbre orienté ordonné simplement étiqueté. Le rôle des systèmes-Q est donc de transformer un graphe en un autre, selon les règles qu'il contient.

```
A + B + C      == S.
A + S + X + C  == S.
X + C          == C + X.
B + B          == B + X.
```

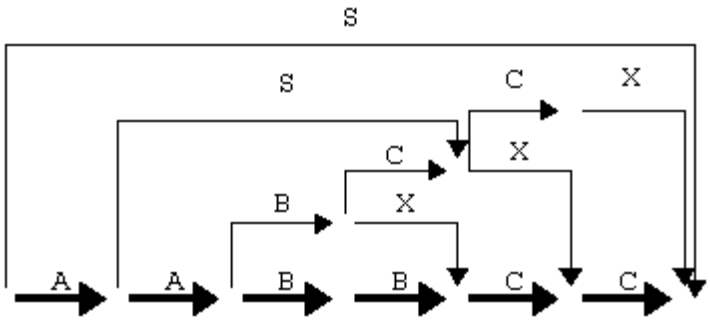


Figure 18: Application des règles d'un système-Q

La figure 18 montre le fonctionnement du système-Q formé des règles d'ajout indiquées, sur le graphe de départ (qui n'est que la transformation d'une liste de symboles en une chaîne d'arcs étiquetés)[82] : on teste toutes les instanciations possibles des règles (et en fonction des nouveaux arcs qu'on peut ajouter au graphe courant par « réécriture » des anciens). On conserve finalement les chemins qui vont du point d'entrée au point de sortie, et qui ne contiennent pas d'arc ayant servi pour une réécriture (dans l'exemple, uniquement l'arc `S` du sommet).

On voit ici l'utilité d'une structure qui évolue pour passer d'une représentation de l'information à une autre. Il faut, selon nous, permettre de modifier le graphe, lors des étapes d'extraction de connaissances : c'est en effectuant ces modifications qu'on passe d'un type de connaissances à un autre.

3.3 Structure des graphes linguistiques

Plus que l'utilisation des graphes, c'est la manière dont on les représente qui nous intéresse. Si de nombreux travaux en traitement des langues concernent les graphes, la plupart les utilisent dans une forme élémentaire. C'est pourquoi nous étudions maintenant des modèles de graphes (ou de réseaux) qui utilisent une structure plus détaillée pour représenter le contenu.

3.3.1 Réseaux sémantiques

Les réseaux sémantiques sont des graphes cherchant à associer entre eux des sens, à partir des liens qu'ils entretiennent dans la langue.

3.3.1.1 Graphes existentiels (Peirce)

L'une des premières tentatives de modéliser les relations sémantiques sous forme de graphes fut l'introduction des *graphes existentiels* de Peirce (1931-1935) au début du XX^{ème} siècle. Selon lui, les graphes existentiels, qu'il a introduits comme étant des formes graphiques de la logique, visent à donner « une image en mouvement de la pensée ».

La théorie des graphes existentiels se divise en trois systèmes :

- *alpha* pour la logique propositionnelle, faisant intervenir des ensembles ;
- *beta* pour la logique du premier ordre (c'est à ce niveau qu'apparaissent les « lignes d'identité » reliant les instances du même sens) ;
- *gamma* pour la logique modale.

Ces graphes servent donc à représenter des énoncés logiques.

La figure 19 présente le graphe existentiel correspondant à l'énoncé « Si le parent d'un bébé est de sexe féminin, alors c'est la mère de ce bébé », ce qui se traduirait en logique par $(\forall x, \forall y)\ \ estParentde(x, y) \wedge estDeSexeFeminin(x) \wedge estBebe(y) \Rightarrow estMereDe(x, y)$.

[82] On présente ici un exemple théorique de système-Q (d'après Colmerauer & Roussel, 1992), relativement simple. En pratique, les systèmes-Q sont plus complexes : ils utilisent notamment des variables permettant de représenter un ensemble de valeurs possibles (par exemple, l'ensemble des étiquettes, l'ensemble des arbres, l'ensemble des forêts) ; les règles utilisant ces variables sont alors applicables pour n'importe quelle valeur possible de la variable (Colmerauer, 1970).

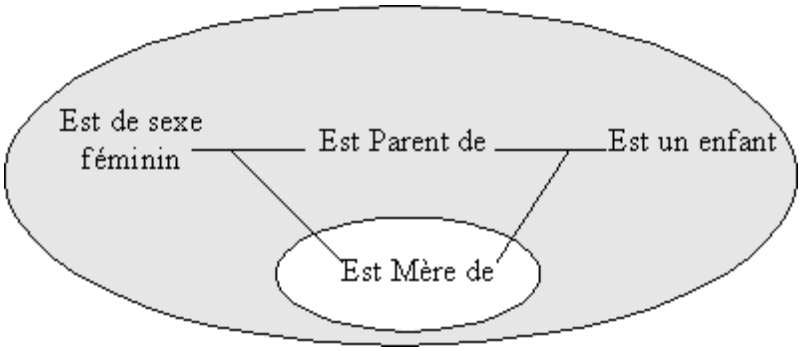


Figure 19: Exemple de graphe existentiel de Peirce

L'ellipse grisée représente la première proposition (la condition), l'ellipse blanche contenue dans la première représente la seconde proposition (l'impliquée)[83].

La structure des graphes existentiels diffère de ce qu'on peut rencontrer habituellement :

- les nœuds du graphes sont les relations ;
- les liens entre eux représentent les objets (les « lignes d'identité » reliant en effet les différentes instances d'un même objet) ; ainsi l'argument de $estDeSexeFeminin(x)$, le premier argument de $estParentDe(x, y)$ et celui de $estMereDe(x, y)$, qui désignent une seule et même personne, sont reliés entre eux.

Le principal inconvénient de cette théorie est qu'il faut une bonne connaissance de la logique formelle pour pouvoir interpréter ces graphes qui, de plus, ne sont pas « classiques » (ils contiennent des arcs à plus de deux extrémités).

3.3.1.2 Réseaux sémantiques hiérarchiques

La première modélisation des graphes sémantiques destinée à l'enregistrement du sens sur un support infomatique fut la *mémoire sémantique* de Quillian (1968) ; chaque mot y était stocké avec des liens vers d'autres. Très vite il fut supposé que ce réseau de mots était organisé de manière hiérarchique (Collins & Quillian, 1969), du plus général au plus particulier[84].

On peut distinguer, sur la structure de mémoire présentée dans la figure 20 :

- des arcs représentent des liens d'appartenance ou d'inclusion (« autruche » est inclus dans « oiseau ») ;
- d'autres arcs expriment des propriétés spécifiques (« avoir des ailes » est une propriété de « oiseau » parce que les oiseaux ont généralement des ailes[85], mais que les animaux – catégorie dans laquelle « oiseau » est incluse – n'en ont pas tous).

[83] Nous ne rentrerons pas ici dans une description détaillée des graphes existentiels ; on peut trouver une bonne explication de cette théorie, du point de vue mathématique et logique, dans la thèse de Zeman (1965).

[84]Bien que les relations dans un réseau sémantique soient dans l'autre sens (du plus particulier au plus général).

[85]Ainsi, la propriété « peut voler » est attachée à la catégorie « oiseau », même s'il existe des oiseaux qui ne peuvent voler, parce qu'elle est générale. Les exceptions sont traitées en attachant la propriété « ne peut pas voler » à « autruche » et aux autres oiseaux ne pouvant voler.

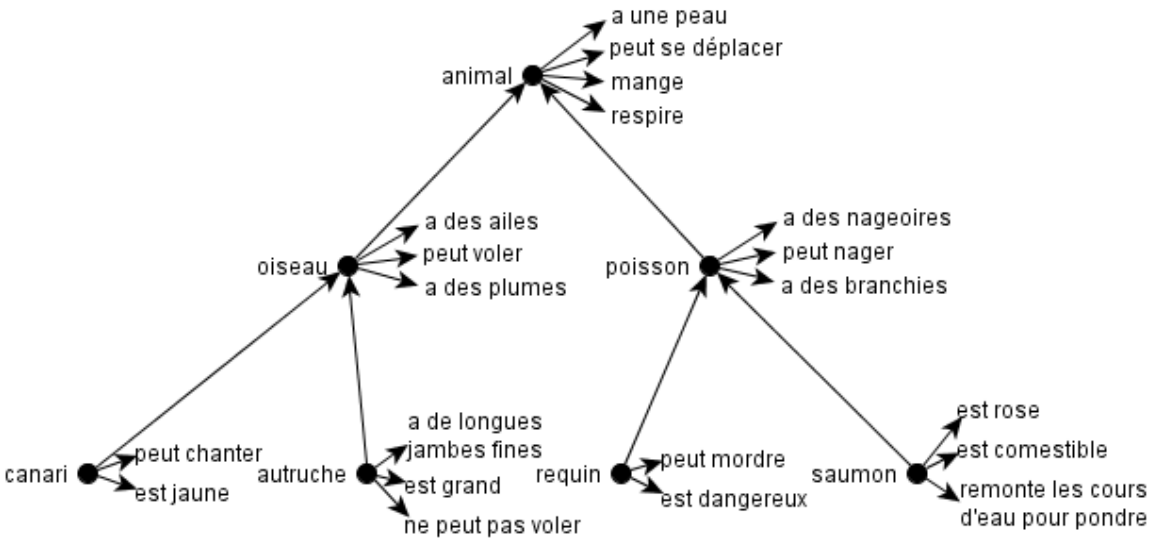


Figure 20: Exemple de structure de mémoire sémantique (Collins & Quillian, 1969)

Dans ce modèle, l'association entre concepts est faite par inférence, ou par propagation d'activation (l'activité d'un nœud peut être propagée sur les liens qui l'entourent pour activer d'autres nœuds, qui peuvent eux-mêmes la propager, etc.).

L'hypothèse de ces travaux, validée sur quelques tests, est que la rapidité avec laquelle le cerveau est capable d'associer deux concepts dépend de la distance qui les sépare dans le graphe (et donc du nombre d'opérations à faire pour les associer). La suite des travaux, notamment (Collins & Quillian, 1970), relativisera cette hypothèse, en tout cas en ce qui concerne la hiérarchie de la structure : bien que *chien* soit plus proche dans la hiérarchie supposée de *mammifère* que d'*animal*, les testeurs répondent correctement plus vite à « un chien est un animal » qu'à « un chien est un mammifère ».

Ces réseaux sémantiques seront ensuite reformulés par Minsky (1975) sous la forme de *cadres* (« frames »), renfermant des propriétés individuelles et liés entre eux par des relations (notamment d'inclusion). Les systèmes de cadres de Minsky inspireront eux-même plusieurs langages de description des connaissances, comme *KRL* (Bobrow & Winograd, 1977) ou *Representation Language Language* (Lenat & Grenier, 1980).

3.3.1.3 Graphes conceptuels

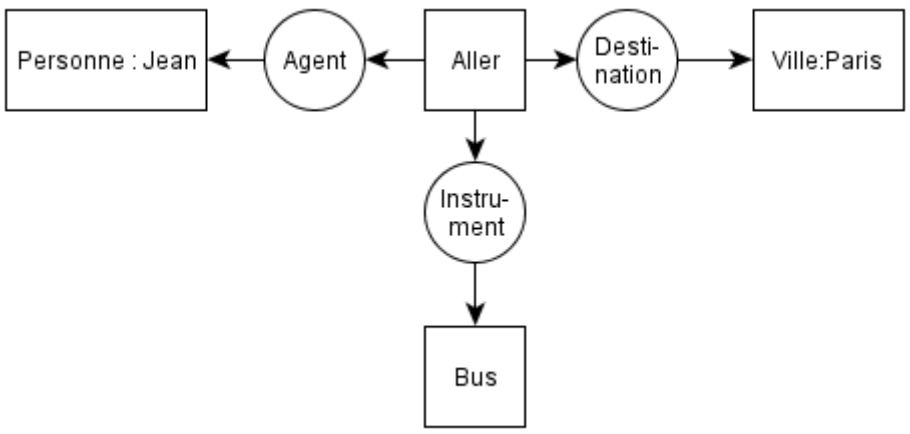


Figure 21: Graphe Conceptuel de « Jean va à Paris en bus »

Influencés à la fois par les graphes existentiels (quant à la représentation des relations sous formes de nœuds) et les réseaux sémantiques (quant à la hiérarchie de concepts), les graphes conceptuels ont été introduits par Sowa à la fin des années 1970 (Sowa, 1976, 1984, 2000).

Un *graphe conceptuel* est un graphe bipartite comportant deux types de nœuds :

- les concepts, qui ont un type et un référent ;
- les relations conceptuelles, qui ont un type et une valence.

Chaque arc doit relier une relation conceptuelle à un concept ; on dit alors que l'arc appartient à cette relation conceptuelle, et qu'il est attaché à ce concept.

Si des concepts peuvent n'être liés à aucune relation conceptuelle, tous les arcs (appartenant à n'importe quelle relation conceptuelle) doivent être attachés à exactement un concept. Les types des concepts, ainsi que ceux des relations conceptuelles, sont organisés en hiérarchie.

3.3.1.4 Graphes UNL

Le langage UNL (*Universal Networking Language*) est issu d'un projet, supporté par l'*UNDL Foundation*[86], puis par le *Consortium U++*, visant à faciliter la traduction, la communication, la recherche d'information multilingue, etc. (Boguslavsky et al., 2000, 2005). C'est un *interlingua* formel.

Pour traduire une phrase de l'anglais vers le français, un système de traduction basé sur UNL « enconvertit » un énoncé anglais en UNL (sous la forme d'un graphe), puis « déconvertit » le graphe obtenu vers le français (Sérasset & Boitet, 1999). L'ajout d'une nouvelle langue au système ne nécessite que l'écriture de l'« enconvertisseur » et du « déconvertisseur » correspondants.

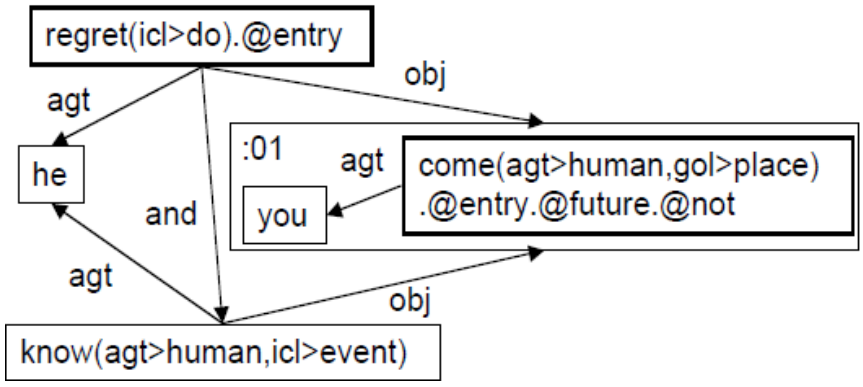


Figure 22: Graphe UNL de la phrase « il sait que tu ne viendras pas et il le regrette » (Tsai, 2004)

La représentation UNL d'une phrase est faite de concepts munis d'attributs sémantiques et de relations entre eux (Uchida, 2001) (UNL, 2005), et peut donc être vue comme un graphe.

Plus précisément, le graphe UNL d'une phrase est un hypergraphe, dont les nœuds sont :

- des *UWs* (*universal words*), qui représentent un concept, chaque UW étant formé de :

[86] *Universal Networking Digital Language Foundation*.

  - un en-tête pour décrire le concept (un mot anglais) ;
  - une liste de contraintes sur les relations qu'il peut entretenir avec d'autres UWs ;
- des *scopes*, qui sont des sous-graphes connexes représentant une partie de la phrase :
  - un scope, qui peut prendre les caractéristiques d'un nœud normal, peut être vu comme un graphe replié (Tsai, 2004) ;
- chaque scope a un unique nœud d'entrée.

Chaque nœud, UW ou scope, peut porter des attributs permettant de décrire « l'information subjective » (du point de vue du locuteur) comme le temps, l'aspect, les références, le centre de l'énoncé, les attitudes, la perspective, ainsi que les conventions d'écriture (UNL, 2005).

La figure 22, tirée de (Tsai, 2004) montre le graphe UNL correspondant à la phrase « il sait que tu ne viendras pas et il le regrette ».

Le nœud `regret(icl>do).@entry` contient :

- l'UW `regret(icl>do)`, qui représente le concept verbal « regretter » ;
- l'attribut `@entry` indique qu'il s'agit du nœud d'entrée du graphe.

Il a comme agent `he` et comme objet un scope (`:01`) correspondant à la partie « que tu ne viendras pas ».

Ce scope a lui-même un nœud d'entrée qui comporte :

- l'UW `come(agt>human, gol>place)`, qui correspond à l'acception de `come` (« venir ») dont l'agent est un humain et le but un endroit ;
- les attributs sémantiques `@future` et `@not`, exprimant respectivement l'idée de futur et la négation.

3.3.2 Systèmes lexicaux

Les *systèmes lexicaux* sont un type de structure lexicale introduit par Polguère (2006). Ils ont notamment pour but d'éviter la séparation habituelle des ressources au niveau de leur structure de représentation, entre bases lexicales ayant la forme de dictionnaires (qu'on peut voir globalement comme du texte) et celles ayant la forme de réseaux (qu'on peut voir comme des graphes). Ils ont été développés dans le but de permettre une représentation graphique du *DiCo* (Polguère, 2000), un dictionnaire informatisé basé sur la lexicographie explicative et combinatoire et surtout sur les fonctions lexico-sémantiques de la théorie sens-texte (Mel'čuk et al., 1995).

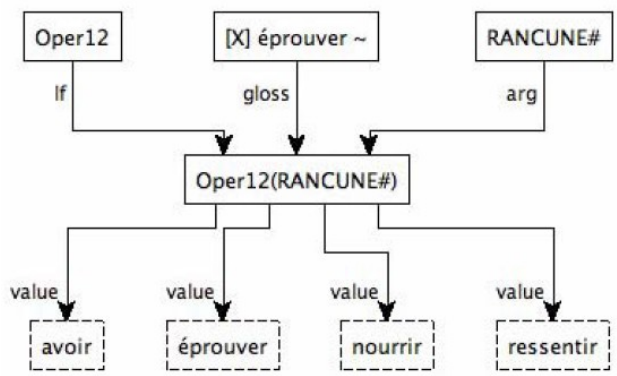


Figure 23: Système lexical correspondant à une fonction lexico-sémantique d'une entrée du DiCo (Polguère, 2006)

Les systèmes lexicaux de Polguère sont définis comme étant :

- (totalement) orientés ;
- non hiérarchiques ;
- hétérogènes : les nœuds peuvent représenter des termes, des collocations, des fonctions lexico-sémantiques, des sens, etc ;
- flous : les arcs et les nœuds portent une « valeur de vérité », qui mesure leur validité ; la vérité de l'information n'est pas évaluée de manière binaire (information correcte ou incorrecte), mais plutôt scalaire[87].

La figure 23 est la représentation d'une relation contenue dans une entrée du DiCo sous forme de système lexical. Il est à noter que la relation (ici $\texttt{Oper}_{12}\texttt{(RANCUNE\#)}$) est représentée par un nœud, et qu'elle est reliée à la fonction lexico-sémantique correspondante, et à ses arguments (l'argument de la fonction, ses valeurs).

Ce qui nous intéresse ici est la démonstration qu'on peut utiliser la même représentation pour tous les types de données et tous les types de relations manipulées, ce que Polguère appelle *hétérogénéité*.

3.3.3 Graphes multiniveau

Nous étudions maintenant l'utilisation des graphes multiniveau pour le traitement des langues. Ils sont essentiellement utilisés en recherche d'information, où les niveaux servent à séparer différents types d'objets linguistiques (termes, documents, etc.)[88].

[87]Une valeur de vérité proche de 0 signifie que l'information a été évaluée comme probablement incorrecte, une valeur proche de 1 correspond a une information évaluée comme probablement correcte ; cette évaluation peut être réalisée de manière automatique.

[88]On peut d'ailleurs voir la représentation de la Théorie Sens-Texte comme un réseau multiniveau.

3.3.3.1 Hypertextes

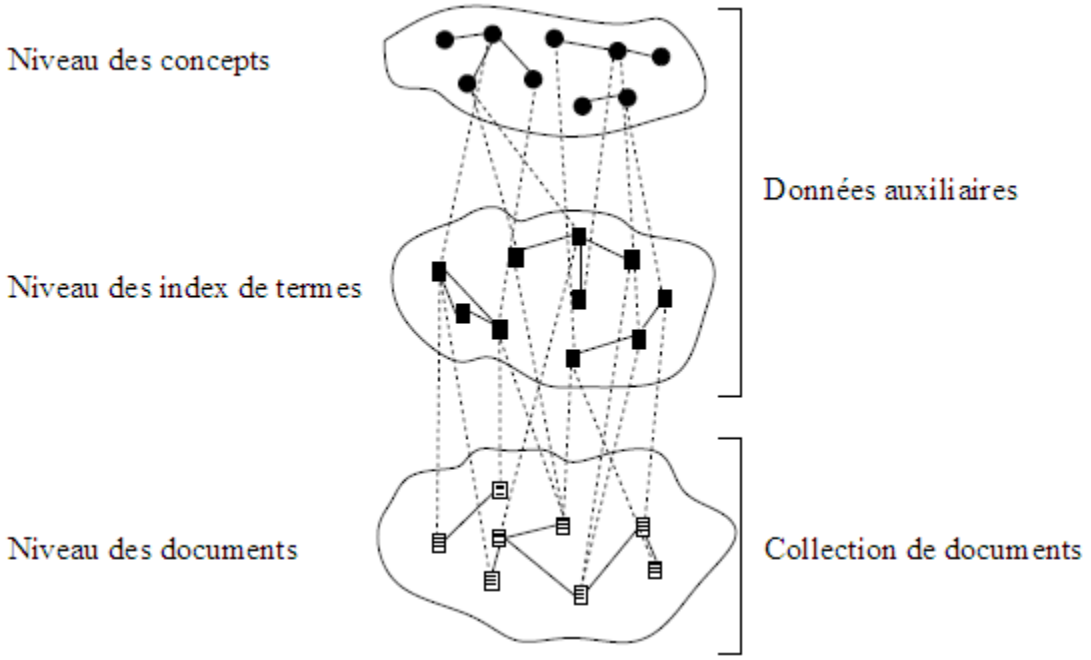


Figure 24: Structure à 3 niveaux d'un hypertexte (Agosti & Crestani, 1993)

Les *hypertextes*[89] d'Agosti & Crestani (1993) proviennent du domaine de la recherche d'information, et servent à modéliser non seulement les liens entre documents et termes (ce qui est très classique en recherche d'information), mais aussi entre termes et concepts.[90]

Les hypertextes sont des graphes non orientés à trois niveaux, comme illustré sur la figure 24 :

- chacun de ces trois niveaux contient des nœuds correspondant à un type d'information distinct (respectivement documents, termes, et concepts) ;
- les liens (non-orientés) sont possibles seulement :
  - à l'intérieur d'un niveau (entre documents, entre termes, ou entre concepts) ;
  - entre deux niveaux adjacents (entre documents et termes, entre termes et concepts).

Les hypertextes ont été adaptés, sous le nom d'*hypertextes multiniveau* (« multi-level hypertext »), dans le domaine des bibliothèques numériques (Fischer & Fuhr, 2002).

3.3.3.2 MLAG

Les MLAG (*Multi-Level Association Graphs*, graphes d'association multiniveau) proposés par Witschel (2007, 2008) ont une structure similaire aux hypertextes, sans la restriction sur le

[89]Sans rapport avec le sens aujourd'hui le plus répandu (textes contenant des liens vers d'autres textes).

[90]On peut comparer ces « hypertextes » au logiciel *HyperCard* (Apple), basé sur une pile de « cartes » virtuelles (contenant un fond et des objets graphiques) dans laquelle on navigait grâce à des liaisons programmées entre les piles.

nombre de niveaux, et avec des poids sur les arcs. Ils sont typiquement utilisés pour modéliser les relations entre documents et termes.

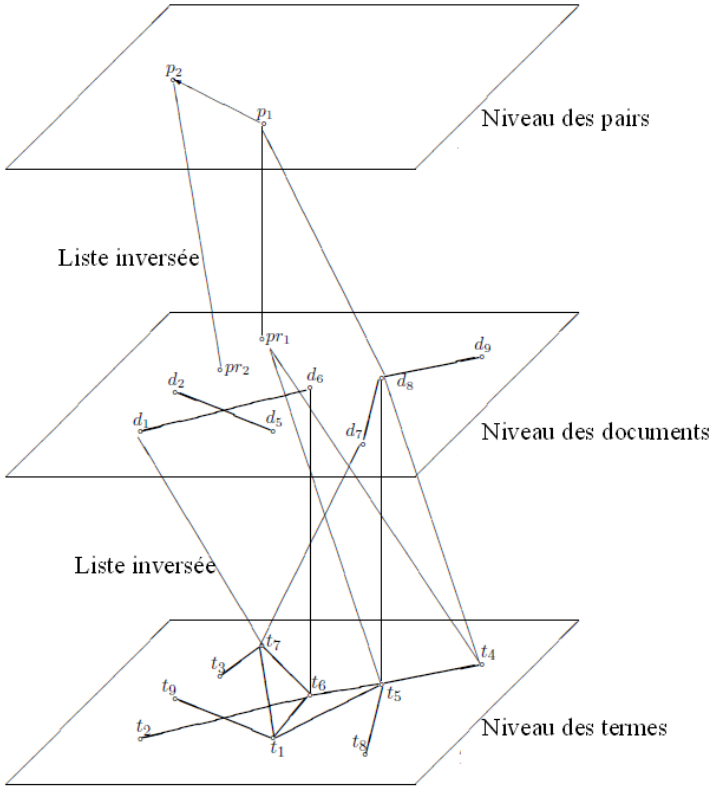


Figure 25: Structure à 3 niveaux d'un MLAG (Witschel, 2007)

Un MLAG est défini comme :

- l'union de *n* graphes orientés $L_1, \ldots, L_n$, chaque graphe $L_i = G(VL_i, EL_i, WL_i)$ comprenant :
  - un ensemble de nœuds VL_i ;
  - un ensemble d'arcs $EL_i \subseteq VL_i \times VL_i$;
  - une fonction $WL_i : EL_i \rightarrow \mathbb{R}$ qui retourne les poids associés aux arcs ;
- *n-1* graphes bipartites (ou « *listes inversées* ») entre deux niveaux successifs, chaque graphe $I_{j,j+1} = G(VI_{j,j+1}, EI_{j,j+1}, WI_{j,j+1})$ comportant :
  - un ensemble de nœuds $VI_{j,j+1} = VL_j \cup VL_{j+1}$;
  - un ensemble d'arcs reliant les niveaux $EI_{j,j+1} \subseteq (VL_j \times VL_{j+1}) \cup (VL_{j+1} \times VL_j)$;
  - une fonction de poids $WI_{j,j+1} : EI_{j,j+1} \rightarrow \mathbb{R}$.

La figure 25 présente l'exemple d'un MLAG à trois niveaux : les termes, les documents, et les pairs[91].

[91]MLAG a été défini dans le cadre de la recherche d'information sur les réseaux pair-à-pair.

Si l'on peut critiquer les MLAG sur les attributs (pourquoi se limiter à un seul ?), ils nous semblent néanmoins très utiles pour dissocier divers niveaux de connaissances (reliés entre eux). Cela peut nous être assez utile lors d'une extraction, si l'on doit changer de niveau d'abstraction pour aller vers une information plus générale, plus résumée.

3.3.4 Observations

Tous les exemples de modèles de graphes que nous avons étudiés ne cherchent pas à représenter les mêmes types de connaissances que ce que nous visons. Cela est toutefois secondaire, puisque nous nous intéressons à la modélisation de l'information, indépendamment de son type.

Les réseaux sémantiques confirment l'intérêt de la représentation de relations sémantiques sous forme de graphes. Certains semblent toutefois pécher par la difficulté qu'on peut avoir à les interpréter sans connaissance particulière (c'est le cas des graphes existentiels), d'autres peuvent paraître trop simples.

Les graphes conceptuels (tout comme les graphes existentiels) nous proposent une représentation où chaque relation ou instance de relation est représentée par un nœud (et où ses arguments sont les autres nœuds reliés au nœud représentant la relation) ; ce choix complique le graphe et, selon nous, ne doit être fait que lorsque des contraintes précises le motivent.

Les systèmes lexicaux nous intéressent essentiellement pour leur hétérogénéité. Ils prouvent l'intérêt d'utiliser la même représentation pour tous les types de données (comme nous le suggérions dans notre cahier des charges), et montrent que la structure de graphe se prête très bien à cela.

Enfin, les graphes multiniveau séparent, entre leurs différents niveaux, les types d'objets linguistiques. Nous pensons que l'utilisation des niveaux peut nous permettre de séparer différentes visions des connaissances, par exemple entre termes et occurrences de termes.

3.4 Représentation de relations n-aires

3.4.1 Matérialisation des relations

Comme nous l'avons vu, les graphes existentiels de Peirce et les graphes conceptuels de Sowa représentent les relations par des nœuds. Si cela n'est pas forcément le plus intuitif (on a tendance a représenter les liens entre objets par des liens au niveau graphique entre les nœuds correspondant à ces objets), cela peut pourtant s'avérer très utile.

En effet, les liens qu'on doit représenter ne sont pas toujours entre objets élémentaires. La matérialisation des relations sous forme de nœuds peut notamment permettre d'associer les relations correspondantes de deux langues (Boitet & Zaharin, 1988 ; Al-Adhaileh & Tang, 2002)[92].

[92]On peut par exemple associer la relation représentant un fragment d'une phrase française, avec la relation représentant le fragment correspondant d'une phrase anglaise.

3.4.2 Relations n-aires

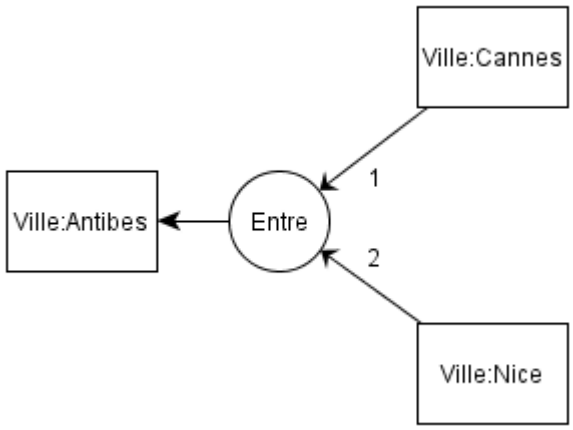


Figure 26: Graphe Conceptuel de « Antibes est entre Cannes et Nice »

La matérialisation des relations sous forme de nœuds est également une bonne technique pour résoudre la représentation d'une relation entre 3 arguments ou plus, en évitant les hyperarcs : il suffit alors de relier 3 nœuds au nœud matérialisant la relation.

Certains modèles de graphes, comme TELA, treillis pour la traduction fondée sur la mémoire (Planas, 1998), ou les graphes conceptuels, prévoient d'ailleurs explicitement ce cas, mais en faisant toujours la distinction entre les arguments « sources » et les arguments « cibles » de la relation. La figure 26 présente un graphe conceptuel représentant une relation à 3 arguments.

Il sera certainement nécessaire d'adopter dans notre modèle, au moins dans certains cas, une telle représentation des relations, en s'affranchissant de la distinction entre arguments « sources » et « cibles ».

4 Cahier des charges d'un modèle de graphe pour l'extraction de connaissances linguistiques

En fonction des contraintes dégagées ci-dessus, nous sommes maintenant en mesure de préciser notre modèle de graphe pour l'extraction de connaissances linguistiques, au niveau des données et des processus.

4.1 Données

4.1.1 Simplicité de la représentation

La structure générale de notre modèle de graphes est la plus simple possible : les nœuds et les arcs représentent respectivement les objets linguistiques et les relations entre eux. Cela assure une interprétation facile par l'homme comme par la machine.

Dans certains cas (relations n-aires, liens entre relations), il faut matérialiser les relations, pour éviter les hyperarcs. Notre modèle devra donc le permettre.

Nous adoptons une structure *multiniveau*, pour distinguer différents niveaux d'abstraction de l'information (par exemple celui des occurrences de termes dans le corpus, et celui des termes), que nous préférons aux « scopes » d'UNL ; ces derniers pourraient en effet sembler trop complexes aux non-initiés.

4.1.2 Variété des ressources

Les graphes de notre modèle n'ont *pas de structure prédéfinie*. Ils peuvent ainsi représenter tous les types de données considérés (analyses de corpus monolingues ou bilingues, dictionnaires, etc.).

De même, aucune supposition n'est faite sur les nœuds et les arcs, qui peuvent porter *n'importe quel attribut* : le modèle ne fixe pas d'attributs (ni, par conséquent, les propriétés correspondantes) ni de liste d'attributs possibles.

4.2 Processus

4.2.1 Simplicité des opérations

Les opérations de notre modèle font abstraction de l'information portée par les nœuds et les arcs sur lesquels elles agissent. Ce sont des opérations sur les graphes, et elles doivent être vues ainsi : il ne faut pas chercher à ce qu'on puisse toujours traduire une opération du processus d'extraction en une seule opération sur les graphes.

Les opérations *peuvent modifier la structure* du graphe, soit dans le même niveau, soit dans un autre niveau ; nous définirons des opérations qui produiront le contenu d'un niveau à partir du contenu d'un autre niveau.

Les modifications apportées au graphe par les opérations définies doivent être simples et compréhensibles aisément. Mieux vaut enchaîner deux modifications simples d'un graphe que vouloir absolument les regrouper pour « coller » à l'opération souhaitée du processus d'extraction.

4.2.2 Généricité des opérations

Les opérations doivent être génériques et paramétrables. La volonté de généricité doit d'ailleurs entraîner la simplicité des opérations (en leur commandant d'enlever toute

spécificité qui pourrait limiter leur réutilisation) : une même opération effectuant une modification précise du graphe pourra alors être utilisée dans divers processus.

Conclusion

Dans ce chapitre, nous avons étudié ce que nous avons appris des expérimentations d'extraction de collocations. Nous avons choisi de consacrer nos recherches à l'élaboration d'outils destinés à faciliter la réalisation de tâches d'extraction par des linguistes, plutôt qu'au processus lui-même.

Après avoir établi les contraintes que devait respecter le modèle sous-jacent à de tels outils (simplicité et généricité, tant au niveau des données qu'à celui des opérateurs), nous avons proposé que celui-ci soit basé sur les graphes. Nous avons ensuite procédé à une étude de différentes structures existantes de graphes, et conclu que notre modèle devrait être très ouvert (aucune contrainte sur les attributs, sur la structure des graphes) et multiniveau (pour représenter différentes vues des connaissances).

Nous allons maintenant définir formellement un modèle de graphe compatible avec ces objectifs.

Chapitre VI - Modèle de graphe pour un traitement de ressources à haut niveau

Souhaitant améliorer la collecte de données lexicales, et après l'étude de cas pratiques, nous avons redéfini notre problème comme celui de la spécification d'un nouveau modèle pour l'extraction de connaissances. Cela nous a conduit à établir au chapitre précédent le « cahier des charges » que ce modèle doit suivre (il doit être générique et de haut niveau, de manière à s'adapter au plus de problèmes différents possibles), puis à étudier les modèles de graphes existants pour les tâches d'extraction de connaissances linguistiques.

Dans ce chapitre, nous introduisons *MuLLinG*, notre propre modèle de graphe. La première version est un modèle simple, mais déjà multiniveau, pour permettre de résoudre au mieux les contraintes fixées dans le cahier des charges. Nous présenterons ensuite les différents opérateurs liés à ce modèle. Nous décrirons enfin la seconde version de ce modèle, plus complexe, car destinée à représenter les relations n-aires (entre plus de deux nœuds du graphe).

1 MuLLinG, un modèle de graphe multiniveau

1.1 Réponses au « cahier des charges »

1.1.1 Un modèle de graphe

La première contrainte à satisfaire dans la spécification de notre modèle est qu'il soit de « haut niveau », ne dépende pas de son implémentation informatique, et qu'il soit ainsi compréhensible par le plus grand nombre, en particulier par des linguistes.

La simplicité s'imposant, nous avons choisi de représenter les connaissances linguistiques (objets linguistiques et relations entre eux) sous la forme de *graphes*.

1.1.2 Peu contraignant

Les autres contraintes à satisfaire sont de proposer un outil générique (qu'on puisse utiliser pour différents types d'opérations) et pouvant accepter facilement divers types de données. Notre modèle de graphe doit être peu contraignant, tant sur le contenu du graphe que sur la manière dont on peut l'utiliser.

Cela se traduit par une totale liberté sur les attributs des nœuds et des arcs du graphe. Notre modèle ne pose *pas de contrainte sur ces attributs*.

Les opérateurs associés à notre modèle de graphe devront donc tenir compte de cette liberté, et la garantir, en évitant de supposer l'existence de certains attributs des graphes. Les attributs

ne devront pas être vus comme des variables globales dont l'existence est un préalable au fonctionnement des opérateurs, mais devront intervenir comme paramètres de ces derniers.

1.1.3 Plusieurs niveaux

1.1.3.1 Différentes vues

De plus, notre modèle de graphe doit avoir une structure évolutive, permettant en particulier de présenter différentes « vues » du contenu (avant et après une opération visant à changer la granularité avec laquelle on voit les choses).

Nous pourrions définir notre modèle et les opérateurs associés de telle manière qu'un tel changement se traduirait par la création d'un nouveau graphe à partir de l'ancien.

Mais nous souhaitions pouvoir conserver, au sein d'un même graphe, ces différentes vues, pour qu'elles puissent être utilisées en même temps dans une opération de modification du graphe. C'est pourquoi, plutôt que de créer de nouveaux graphes lors des changements de vue, nous avons préféré créer de nouveaux niveaux dans le même graphe.

Notre modèle de graphe doit donc être *multiniveau*, chaque niveau représentant une vue différente du contenu.

1.1.3.2 Liens entre les niveaux

Une fois décidé que les graphes sont multiniveau, nous devons préciser comment l'information doit y être représentée et hiérarchisée.

La première contrainte que nous posons est d'ordre pratique : pour éviter une trop grande complexité de notre structure à niveaux, notre modèle de graphe doit, comme les MLAG présentés au chapitre précédent, n'autoriser les liens qu'entre nœuds du même niveau ou de niveaux adjacents. Cela étant fixé, nous posons une restriction importante sur notre modèle :

> Le passage à un niveau supérieur doit correspondre à une représentation plus restreinte du contenu[93].

Cela permet de définir un *ordre des niveaux* du graphe, de la vue la plus détaillée (le niveau le plus « bas ») vers la vue la plus générale, la plus résumée (le niveau le plus « haut »). Nous traduisons cela dans notre modèle, en limitant le sens des liens entre nœuds de niveau adjacents :

> Un arc interniveau doit nécessairement être orienté et relier un nœud d'un niveau donné et un nœud du niveau immédiatement supérieur.

Enfin, puisque la généralisation a pour but de réduire l'information visible et de n'en garder que ce qui est jugé le plus important, nous posons une contrainte sur celle-ci :

> Un objet du niveau inférieur ne peut être associé qu'à un seul objet plus général au niveau immédiatement supérieur (qui, lui, peut être également la généralisation d'autres objets du niveau inférieur).

[93]Créer un nouveau niveau dans un tel graphe, c'est voir le problème sous un angle moins détaillé, plus abstrait (en passant par exemple des relations entre occurrences de termes à celles entre termes).

Cette contrainte est restrictive : elle empêche de décomposer un nœud en plusieurs au niveau supérieur (par exemple, produire « de » et « le » à partir de l'article « du », ou produire les différents sens d'un mot ambigu), imposant de faire une telle modification au niveau courant. Cela permet toutefois de définir les niveaux comme allant du plus spécifique (niveau inférieur) au plus général (niveau supérieur).

Pour résumer, les liens entre objets des niveaux de notre modèle ne peuvent exister que d'un niveau inférieur vers le niveau immédiatement supérieur, et sont uniques pour un nœud donné du niveau inférieur.

1.2 Une structure multiniveau

Nous introduisons ici notre modèle de graphe linguistique multiniveau, appelé *MuLLinG*[94] (Multi-Level Linguistic Graph).

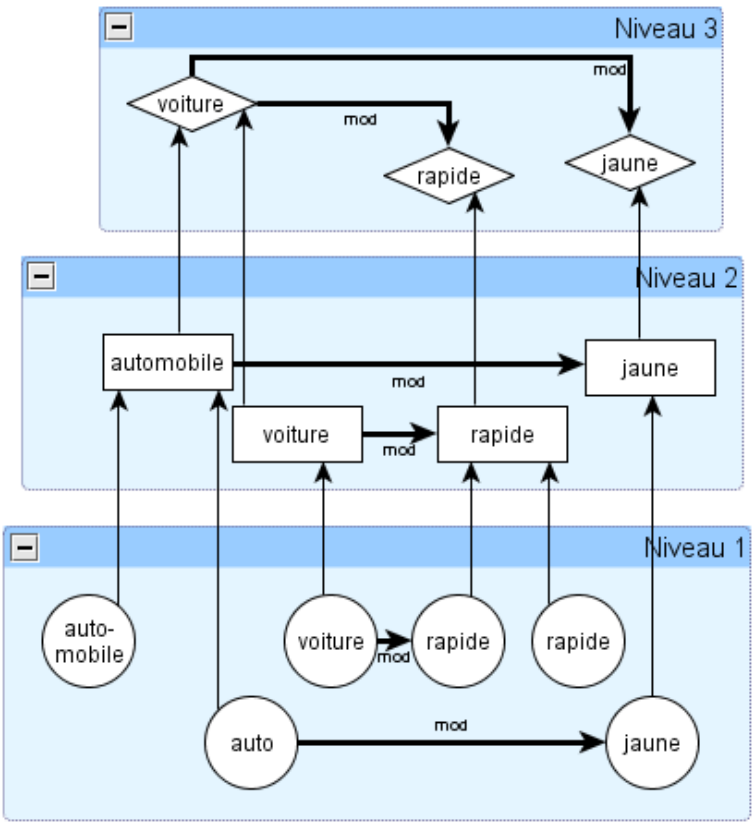


Figure 27: Exemple de graphe MuLLinG à trois niveaux

Les graphes MuLLinG (tels que celui présenté sur la figure 27) ont les caractéristiques suivantes :

- le graphe est découpé en plusieurs niveaux ordonnés (représentant des vues différentes du contenu) ;

[94]En anglais, les « *mulling spices* » sont les épices utilisées pour la cuisson du vin chaud (« *mulled wine* »), telles que l'anis étoilé, le clou de girofle ou la noix de muscade.

- chaque nœud du graphe appartient à un (et un seul) niveau ;
- il y a deux types d'arcs :
  - arcs intraniveau (entre nœuds du même niveau) : il peut y avoir plusieurs arcs intraniveau différents entre deux nœuds (s'ils ont un attribut pour les différencier) ;
  - arcs interniveau (obligatoirement d'un nœud d'un niveau *n* vers un nœud du niveau supérieur *n+1*) : ils font le lien entre un objet du niveau inférieur et celui qui le représente d'une manière plus globale (lui et d'autres) au niveau supérieur ; un même nœud ne peut donc être la source que d'un seul arc interniveau ;
- tous les nœuds et tous les arcs intraniveau peuvent porter n'importe quel attribut[95].

On peut constater que le modèle MuLLinG est assez proche des MLAG présentés au chapitre précédent, en particulier parce qu'il s'agit de graphes multiniveau, portant des attributs, et n'autorisant les arcs qu'à l'intérieur des niveaux ou entre niveaux adjacents. Il y a cependant deux différences majeures d'un graphe MuLLiNG par rapport à un MLAG :

- l'absence de restrictions sur les attributs (MuLLinG permet n'importe quel attribut sur les nœuds et les arcs, là où MLAG se limite aux poids des arcs), due à notre volonté de proposer un modèle générique, notamment en ce qui concerne les informations portées par les données à traiter ;
- la restriction sur le sens des arcs interniveau (là où MLAG autorise tout), afin de définir l'ordre des niveaux du graphe :
  - MuLLinG se limite aux arcs interniveau allant du niveau inférieur vers un niveau supérieur (alors que MLAG permet des arcs dans les deux sens) ;
  - un nœud du niveau inférieur ne peut être la source que d'un seul arc.

1.3 Définition précise

Nous définissons les *graphes linguistiques multiniveau* comme des multigraphes orientés $G^n=(V, E, F, A, \Phi, a_V, a_E)$ (pour *n* niveaux) avec :

- les nœuds :
  - V : l'ensemble des nœuds, formé de *n* sous-ensembles disjoints $V_1,\ldots,V_n$ correspondant aux *n* niveaux;
- les arcs intraniveau :
  - E : l'ensemble des arcs intraniveau, formé de *n* ensembles disjoints $E_1,\ldots,E_n$ ($i\in\{1,\ldots,n\}$) ;
  - A : l'ensemble des fonctions $\alpha_i : E_i \rightarrow V_i \times V_i$ associant à un arc intraniveau ses deux extrémités (cette façon de représenter ces arcs permet d'en avoir plusieurs entre deux nœuds);
- les arcs interniveau :

[95]Si, par souci de simplification, les objets de la figure 27 ne portent qu'un seul attribut (le nom pour les nœuds, le type de relation pour les arcs intraniveau), cela ne doit pas laisser entendre que c'est toujours le cas : ces objets peuvent porter plusieurs attributs (par exemple, pour un nœud représentant un terme : la partie du discours, le lemme, etc.).

- F : l'ensemble des arcs interniveau, dans *n-1* ensembles disjoints $F_{1,},\ldots,F_{n-1}$ ($i\in\{1,\ldots,n-1\}$) définis comme $F_i=\{\langle x,y\rangle\in V_i\times V_{i+1} \mid y=\varphi(x)\}$;
- Φ : l'ensemble des fonctions $\varphi_i : V_i \rightarrow V_{i+1}$, associant à un nœud d'un certain niveau un nœud du niveau supérieur qui le représente d'une manière plus globale[96] ;

- les attributs :
  - $a_V=\{f : V\rightarrow\Sigma_V\}$ et $a_E=\{f : E\rightarrow\Sigma_E\}$ (Σ_V,Σ_E étant les alphabets correspondant respectivement aux attributs des objets de *E* et de *V*) modélisent les attributs.

Dans la suite, la valeur de la fonction correspondant à l'attribut *att* pour l'arc ou le nœud *X* sera notée *X.att*.

Pour correspondre à ce qu'on rencontre habituellement dans les modélisations informatiques des graphes, nous parlerons, dans les opérations que nous présentons, de la *source* et de la *cible* d'un arc ; elles seront toutes les deux considérées comme *extrémités* de l'arc.

2 Opérateurs génériques

Nous présentons ici des opérateurs pour l'extraction de connaissances linguistiques. Volontairement, ces opérateurs ne sont pas liés à un processus (même si nous avons gardé en tête la manière dont de tels processus se déroulent) : ils ont pour but d'être génériques.

Il s'agit d'éviter la tentation de convertir une opération d'extraction en un opérateur complexe de manipulation de graphes qui ne serait pas utilisable pour d'autres tâches, là où une succession d'opérateurs simples (et génériques) est plus facile à comprendre et aussi efficace.

2.1 Ce qui relève du modèle

Dans la spécification des opérateurs, nous avons plusieurs fois dû faire des choix : comment présenter des opérateurs de manière détaillée, sans rentrer dans ce qui relève plus des problèmes d'implémentation ?

Lorsqu'on manipule des objets porteurs d'information (sur leurs attributs), il est souvent nécessaire de lire ou de modifier cette information.

Nous avons considéré que cela ne relevait pas directement des opérateurs de notre modèle et que, bien qu'ils puissent facilement accéder aux propriétés d'un objet ou les modifier (en manipulant les valeurs des attributs correspondants), ils ne devaient pas être utilisés pour cela. Au contraire, ils doivent être strictement limités à l'opération à effectuer, la résolution de ces problèmes secondaires devant alors être spécifiée ailleurs et n'intervenir que comme un paramètre de l'opération.

C'est pourquoi, à chaque fois qu'un opérateur doit évaluer, comparer, ou modifier les propriétés d'objets, il prend *en paramètre* la fonction nécessaire à la tâche.

[96]On le verra dans la suite, le nœud du niveau supérieur représente la *classe d'équivalence* du nœud du niveau inférieur.

Cela rend les processus effectués par les opérateurs plus simples à comprendre. Ils concernent uniquement la résolution du problème essentiel, sans la mélanger avec les problèmes secondaires.

2.2 Opérateurs : accès et modification du graphe

Les premières opérations que nous définissons (par rapport à la représentation de graphe que nous avons choisie) correspondent à l'ajout ou à la suppression d'un nœud ou d'un arc. Elles n'ont pas vocation *a priori* à être utilisées seules, mais plutôt à servir dans des opérateurs enchaînant ce genre de modifications.

2.2.1 Ajout d'objets linguistiques

AjouteNœud(nœud *v*, entier *i*) :
{ajoute le nœud v au i-ème niveau}
$V_i \leftarrow V_i \cup \{v\}$.

AjouteArcIntra(arc *e*, entier *i*, nœud *s*, nœud *c*) :
{ajoute un arc e entre s et c au i-ème niveau ; présupposé : $s, c \in V_i$, $e \in E_i$ *}*
$E_i \leftarrow E_i \cup \{e\}$; $\alpha_i(e) \leftarrow \langle s, c \rangle$.

AjouteArcInter(arc *e*, entier *i*, nœud *s*, nœud *c*) :
{ajoute l'arc e entre s (niveau i) et c (niveau i+1) ; présupposé : $s, c \in V_i$, $e \in F_i$ *}*
$\varphi_i(s) \leftarrow c$; $e \leftarrow \langle s, c \rangle$

2.2.2 Suppression d'objets linguistiques

SupprimeArcIntra(arc *e*) :
{supprime l'arc e ; présupposé : $e \in E$ *}*
$e \in E_i$: $E_i \leftarrow E_i - \{e\}$ *{*$\alpha_i(e)$ *n'est plus défini}*

SupprimeArcInter(arc *e*) :
{supprime l'arc e ; présupposé : $e \in F$ *}*
$e \in F_i$: $F_i \leftarrow F_i - \{e\}$

NettoieNœud(nœud v) :
{supprime tous les arcs intraniveau ou interniveau dont v est une extrémité ; destiné à rendre v isolé avant suppression}
$v \in V_i$:
$\forall e \in E_i \mid v = source(e) \vee v = cible(e)$: SupprimeArcIntra(e);
si ($i < n-1$) : $\forall e \in F_i \mid v = source(e)$: SupprimeArcInter(e);
si ($i > 1$) : $\forall e \in F_{i-1} \mid v = cible(e)$: SupprimeArcInter(e);

SupprimeNœud(nœud v) :
{supprime le nœud v, après avoir supprimé tous les arcs dont il est une extrémité}
NettoieNœud(v);
$v \in V_i$: $V_i \leftarrow V_i - \{e\}$.

On introduit une fonction permettant de supprimer, non seulement un nœud, mais aussi sa descendance, c'est-à-dire :

- les nœuds du niveau inférieur auquel il est relié (les éléments de la classe d'équivalence qu'il représente) ;
- les arcs reliant ces nœuds au nœud de départ ;
- la descendance propre de chacun de ces nœuds.

SupprimeNœudEtDescendance(nœud v) :
{supprime le nœud v, après avoir supprimé sa « descendance », ainsi que tous les arcs dont il est une extrémité}
$v \in V_i$: si (i>0) :
$\forall v' \mid (v, v') \in F_i$:
SupprimeNœudEtDescendance(v');
SupprimeNœud(v).

Nous définissons aussi les fonctions qui permettent d'accéder facilement aux deux extrémités des arcs (qu'ils soient intraniveau ou interniveau).

Source (arc *e*) → nœud :
si $e \in E_i$: $\rightarrow \alpha_i(e).premier$; si $e \in F_i$: $\rightarrow e.premier$;

Cible (arc *e*) → nœud :
si $e \in E_i$: $\rightarrow \alpha_i(e).second$; si $e \in F_i$: $\rightarrow e.second$;

2.3 Application – Filtrage

Nous introduisons des opérations qui permettent d'appliquer une certaine modification à un ensemble d'arcs ou de nœuds du graphe.

On peut par exemple, pour tous les arcs dont la valeur d'un attribut correspondant à une mesure est supérieure à un seuil fixé, modifier en conséquence un attribut donné sur chaque extrémité de cet arc.

On peut également envisager d'appliquer des opérations plus compliquées, modifiant la structure du graphe.

AppliqueSurNœuds (fonction($V \rightarrow bool$) *filtre*, fonction(V) *modif*)[97] :
{Applique, sur tous les nœuds vérifiant filtre*, la fonction* modif*}*
$\forall v \in V$: si filtre(v) : Modif(v) ;

AppliqueSurArcs (fonction($E \rightarrow bool$) *filtre*, fonction(E) *modif*) :
{Applique, sur tous les arcs vérifiant filtre*, la fonction* modif*}*
$\forall e \in E$: si filtre(e) : Modif(e)

L'application la plus immédiate de telles fonctions est le filtrage (pour supprimer, par exemple, tous les arcs dont la valeur de la mesure est inférieure à un seuil fixé) :

SupprimeNœudsSi (fonction($V \rightarrow bool$) *filtre*) :
{Supprime tous les nœuds vérifiant filtre*}*
AppliquerSurNœuds(filtre, SupprimeNœud)

SupprimeArcsSi (fonction($V \rightarrow bool$) *filtre*) :
{Supprime tous les arcs vérifiant filtre*}*
AppliquerSurArcs(filtre, SupprimeArcIntra)

2.4 Émergence

2.4.1 Principe

Dans le modèle MuLLinG, on passe d'un niveau à l'autre en réduisant la vue : le niveau supérieur doit présenter une version résumée, moins détaillée, des connaissances linguistiques présentes au niveau inférieur.

Nous appelons **émergence** l'étape consistant à produire un nouveau niveau du graphe ; cette production comprend deux parties :

- émergence des nœuds : création de nœuds au niveau supérieur (liés avec ceux du niveau inférieur) ;
- émergence des arcs : création d'arcs entre les nœuds (nouvellement créés) au niveau supérieur.

[97]L'utilisateur doit définir la fonction de filtrage (et contrôler éventuellement le niveau) et la fonction de modification, cette dernière devant, à partir d'un objet du graphe (arc ou nœud selon les cas), modifier le graphe au niveau des attributs ou de la structure en elle-même.

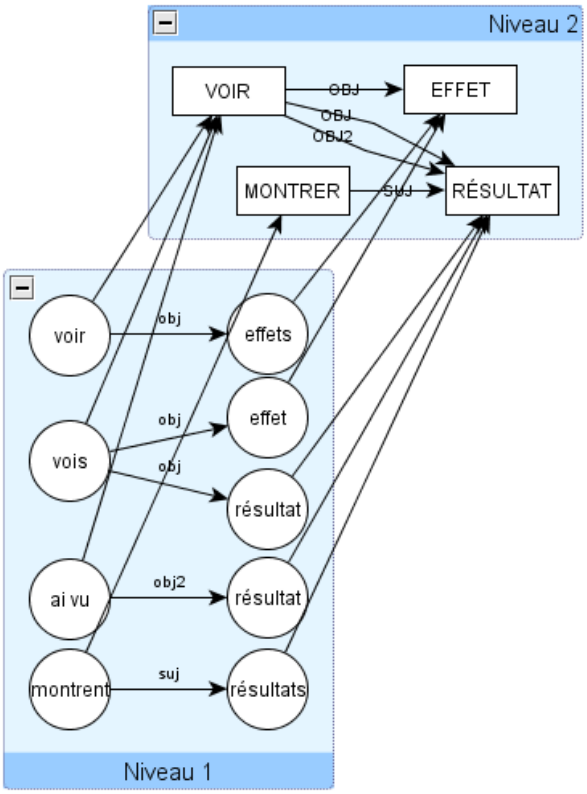


Figure 28: Exemple de regroupement par classes d'équivalence

2.4.2 Classes d'équivalence

Comment passer de manière simple d'une information dense (au niveau inférieur) à une information moins dense (au niveau supérieur) ? Il faut regrouper les contenus informationnels proches, de telle sorte qu'on perde en granularité de l'information.

Pour cela, notre modèle utilise les *classes d'équivalence* : chaque nœud (respectivement arc) du niveau supérieur représente une classe d'équivalence de nœuds (respectivement d'arcs) du niveau inférieur.

Considérons l'exemple d'un graphe représentant les occurrences de termes et les dépendances entre eux, et obtenu à partir des résultats d'un analyseur de dépendances, comme celui de la figure 28. Lorsqu'on cherche à extraire des connaissances à partir de données collectées dans un corpus, il faut passer du stade des occurrences de termes à celui des termes eux-mêmes.

Typiquement, l'émergence des nœuds permet de regrouper toutes les occurrences équivalentes (par exemple celles qui ont le même lemme et la même partie du discours), alors que l'émergence d'arcs permet de produire les relations correspondantes entre termes (les classes représentant toutes les occurrences), en associant ou dissociant ces relations selon qu'on les juge équivalentes ou non.

Nous introduisons une fonction permettant d'obtenir facilement, à partir d'un nœud donné, le nœud (du niveau supérieur) qui représente la classe d'équivalence à laquelle il appartient :

ClasseDuNœud (nœud v) $\rightarrow$ nœud :
$v \in V_i : \rightarrow \varphi_i(v)$

2.4.3 Arguments des opérateurs d'émergence

Quels arguments les opérateurs d'émergence (de nœuds, d'arcs) doivent-ils prendre en compte ?

2.4.3.1 Niveau

L'émergence étant le passage d'un niveau à un autre, il faut nécessairement, pour la réaliser, connaître le *niveau de départ*.

2.4.3.2 Filtre

Tous les objets d'un niveau ne portent pas nécessairement une information utile pour la création de l'information du nouveau niveau. Il faut pouvoir filtrer facilement les objets qu'on juge inutiles. Les paramètres de l'émergence doivent donc contenir une *fonction de filtrage*.

2.4.3.3 Classe d'équivalence

Dans le cadre de l'émergence, il faut connaître la classe d'équivalence d'un nœud ou d'un arc, mais il n'est pas nécessaire de savoir quels autres nœuds ou quels autres arcs lui sont équivalents. Cela nous permet d'éviter une recherche d'équivalents qui serait très coûteuse en temps de calcul.

Nous considérons que le calcul de la classe d'équivalence d'un arc ou d'un nœud ne relève pas de l'opération d'émergence, et qu'il doit donc n'en être qu'un paramètre.

La manière de définir la fonction d'équivalence (les attributs concernés et l'équivalence entre certaines valeurs) et la partition correspondante des arcs ou des nœuds concerne alors l'implémentation du modèle.

Nos opérateurs d'émergence ont donc pour paramètre une *fonction* associant, à chaque objet (arc ou nœud selon le cas) considéré, l'*identifiant de sa classe d'équivalence*.

2.4.3.4 Calcul des valeurs des attributs des objets créés

Pour être utiles dans la suite d'un processus, les objets (nœuds et arcs) créés lors de l'émergence doivent porter de l'information. Cette nouvelle information, à produire, doit être déduite des informations portées par les objets appartenant à la classe d'équivalence que l'objet modélise, et en donner une version *unifiée*.

L'information étant représentée sous la forme d'attributs des objets du graphe, il faut donc établir la valeur des attributs des arcs et des nœuds lorsqu'on réalise l'émergence. Pour cela, il faut passer en paramètres :

- les attributs dont on va modifier (voire créer) la valeur ;
- pour chaque attribut, la façon dont on calcule cette nouvelle valeur, qui dépend :

- de l'attribut concerné ;
- des propriétés de l'objet de départ (celui qu'on ajoute à la classe d'équivalence) ;
- des propriétés de l'objet représentant la classe d'équivalence.

Nous présentons maintenant quelques exemples utiles de telles fonctions[98] :

Copie(attribut *a*, nœud v_{source} , nœud v_{classe}) → *valeur*[99] :
{si l'attribut a de v_{classe} *n'a pas de valeur : copie sa valeur pour* v_{source} *, sinon : ne fait rien}*
si ($v_{classe}.att$ non défini) : → $v_{source}.att$

Incrémente(attribut *a*, nœud v_{source} , nœud v_{classe}) → *valeur* :
{ si l'attribut a de v_{classe} *n'a pas de valeur : 1, sinon : ajoute 1}*
si ($v_{classe}.att$ non défini) : → 1
sinon : → $v_{classe}.att$ +1

Prend_X(attribut *a*, nœud v_{source} , nœud v_{classe}) → *valeur* :
{si l'attribut a de v_{classe} *n'a pas de valeur : X, sinon : ne fait rien}*
si ($v_{classe}.att$ non défini) : → X

2.4.4 Création d'objets matérialisant la classe

L'émergence consiste à associer un objet du niveau inférieur à un objet du niveau supérieur représentant la classe d'équivalence à laquelle il appartient.

Il est nécessaire de pouvoir manipuler facilement la correspondance entre identifiants de classe et les objets correspondants. C'est pourquoi on suppose l'existence des fonctions suivantes[100] :

Classe_V(chainecars *c*) → nœud :
{renvoie, s'il existe, le nœud correspondant à la matérialisation de la classe de nœuds dont l'identifiant est la chaîne c*}*
Classe_E(chainecars *c*,<nœud *s*, nœud *t*>) → arc :
{renvoie, s'il existe, l'arc correspondant à la matérialisation de la classe des arcs dont l'identifiiant est la chaîne c*, entre le nœud* s *et le nœud* t*}*

[98]Les fonctions présentées calculent les valeurs des attributs des nœuds, on peut déduire trivialement les fonctions similaires de calcul de valeurs des attributs d'arcs.

[99]Les problèmes liés au type de la valeur (chaîne de caractères, entier, réel, etc.) doivent être résolus à l'implémentation du modèle.

[100]On ne précise pas comment doivent être implémentées les fonctions ; la façon la plus simple de le faire est sans doute d'utiliser des tables de hachage lors de l'émergence pour conserver les informations nécessaires.

Les deux fonctions que nous introduisons maintenant ont un but similaire, l'une pour les classes de nœuds, l'autre pour les classes d'arcs : retourner, selon le cas, le nœud ou l'arc représentant la classe souhaitée, en le créant s'il n'existe pas déjà.

Le passage en paramètre du niveau permet de faciliter l'ajout éventuel du nœud ou de l'arc.

NœudClasse (chainecars *c*, entier *Niv*) → nœud :

{renvoie le nœud (situé au niveau Niv*) correspondant à la classe d'équivalence de nœuds identifiée par* c *; s'il n'existe pas, il est créé}*

Variables : nœud v_{classe} .

Corps :

si (Classe_V(c) est défini) :

→ Classe_V(c)

sinon :

AjouteNœud(v_{classe} , Niv) ;

Classe_V(c) ← v_{classe} ;

→ v_{classe}

ArcClasse (chainecars *c*, nœud *s*, nœud *t*, entier *Niv*) → arc :

{renvoie l'arc (situé au niveau Niv*) correspondant à la classe d'équivalence d'arcs identifiée par* c *et les 2 extrémités* s *et* t*; s'il n'existe pas, il est créé}*

Variables : arc e_{classe} .

Corps :

si (Classe_E(c,<s,t>) est défini) :

→ Classe_E(c,<s,t>)

sinon :

AjouteArcIntra(e_{classe} , Niv, s, t) ;

Classe_E(c,<s,t>) ← e_{classe} ;

→ e_{classe}

2.4.5 Opérateur : émergence de nœuds

Nous définissons l'*émergence de nœuds* comme l'opération qui, à partir d'une fonction donnée d'équivalence entre nœuds[101] :

- calcule, au niveau inférieur, les classes d'équivalence (entre nœuds) induites ;
- produit, au niveau supérieur, les nœuds matérialisant ces classes
  - crée et modifie des valeurs des attributs de ces nœuds;
- ajoute des arcs interniveau entre chaque nœud du niveau inférieur et le nœud (au niveau supérieur) modélisant la classe d'équivalence à laquelle il appartient.

[101]Par exemple, dans le cas d'un graphe d'occurrences de termes, considérer comme équivalents les nœuds portant le même lemme et la même partie du discours.

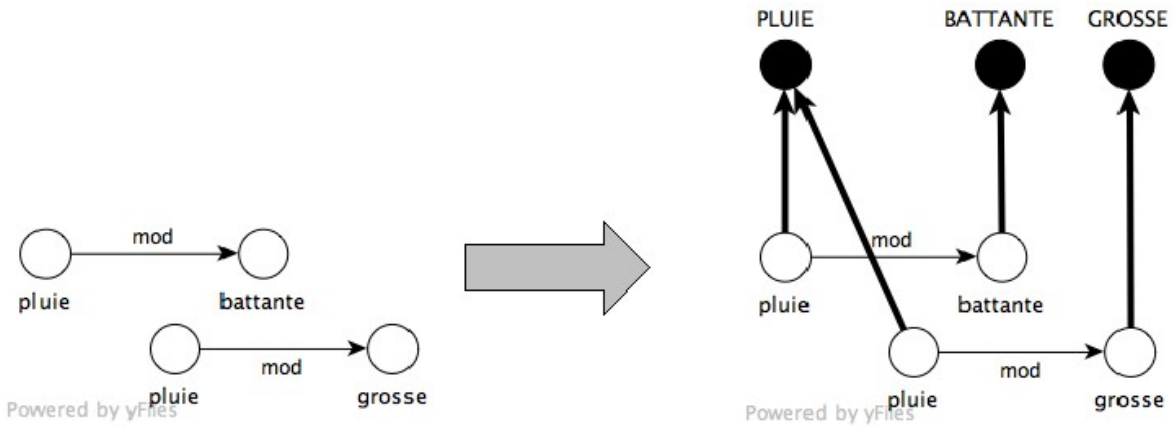


Figure 29: Création de classes d'équivalence de nœuds au niveau supérieur

ÉmergenceDeNœuds

Paramètres d'entrée :

entier *Niv*,
fonction($V \rightarrow bool$) *filtre*,
fonction($V_{Niv} \rightarrow chainecars$) *ClasseEquiv*,
liste(<attribut,fonction($attribut \times V \times V \rightarrow valeur$ >) *CalculAttributs.*

Définition :

Fait émerger, pour les nœuds du niveau Niv *vérifiant* filtre *et dont la classe d'équivalence est calculée par* ClassEquiv*, les nœuds au niveau* Niv+1 *correspondant à ces classes (auxquels ils sont reliés), dont la valeur des attributs est calculée grâce à* CalculAttributs

Variables : nœud v_{classe} , arc e.

Corps :

$\forall v \in V_{Niv}$:
si filtre(v) :
$v_{classe} \leftarrow$ NœudClasse(ClasseEquiv(v), Niv+1)
AjouteArcInter(*e*, Niv, v, v_{classe});
$\forall \langle att, f \rangle \in CalculAttributs$:
$v_{classe}.att \leftarrow f(att, v, v_{classe})$

2.4.6 Opérateur : émergence d'arcs

Nous définissons également l'*émergence d'arcs* comme l'opération qui, à partir d'une fonction donnée d'équivalence entre arcs[102] :

- calcule, au niveau inférieur, les classes d'équivalence (entre arcs) induites ;

[102]On peut par exemple considérer comme équivalents les arcs représentant des relations de dépendance dont le sens est proche et dont on juge qu'elles peuvent être confondues dans la tâche à réaliser.

- produit, au niveau supérieur, des arcs entre nœuds modélisant des classes d'équivalence : un tel arc regroupe tous les arcs (du niveau inférieur) *équivalents* allant d'un nœud appartenant à la classe modélisée par la source (de l'arc du niveau supérieur) vers un nœud appartenant à la classe modélisée par la cible (de l'arc du niveau supérieur).
  - crée et modifie des valeurs des attributs de ces arcs et de leurs extrémités[103].

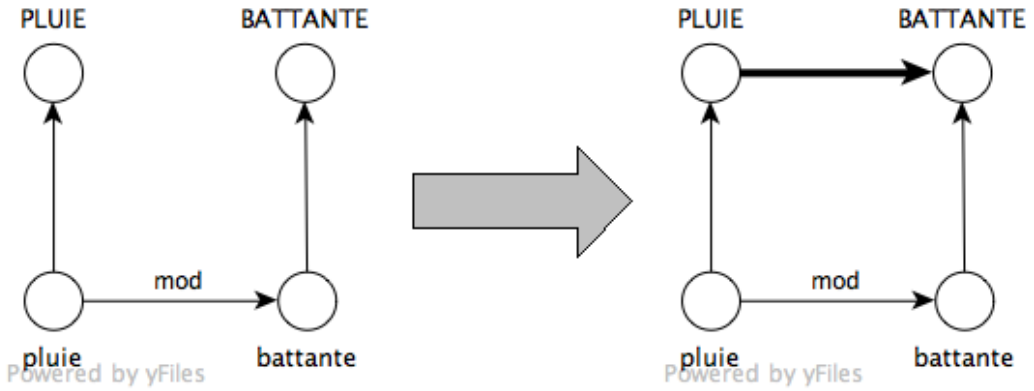


Figure 30: Émergence de liens au niveau supérieur (représentation simple)

ÉmergenceDArcs

Paramètres d'entrée :

entier *Niv*,
fonction($E \rightarrow bool$) *filtre*,
fonction($E_{Niv} \rightarrow chainecars$) *ClasseEquiv*,
liste(<attribut,fonction($attribut \times E \times E \rightarrow valeur$ >) *CalculAttributsArc, CalculAttributsSource, CalculAttributsCible.*

Définition :

Fait émerger, pour les arcs du niveau Niv *vérifiant* filtre *et dont la classe d'équivalence est calculée par* ClassEquiv*, les arcs au niveau* Niv+1*, correspondant à ces classes et aux classes de leurs extrémités, dont la valeur des attributs est calculée grâce à* CalculAttributsArc *(alors que la valeur des attributs de leurs extrémités est calculée grâce à* CalculAttributsSource *et* CalculAttributsCible*).*

[103]Si, pour chaque arc du niveau inférieur, lorsqu'on considère l'arc correspondant au niveau supérieur, on incrémente un compteur de relations à ses extrémités, on obtient, pour chaque nœud représentant une classe (au niveau supérieur) le nombre de fois où les nœuds formant cette classe sont concernés par cette relation, ce qui peut être très utile pour calculer des mesures d'association.

Variables : arc e_{classe}

Corps :

$\forall e \in E_{Niv}$:

si filtre(e) :

$e_{classe} \leftarrow$ ArcClasse(ClasseEquiv(e), ClasseDuNœud(source(e)), ClasseDuNœud(cible(e)), Niv+1) ;

$\forall \langle att, f \rangle \in CalculAttributsArc$:

$e_{classe}.att \leftarrow f(att, e, e_{classe})$

$\forall \langle att, f \rangle \in CalculAttributsSource$:

$source(e_{classe}).att \leftarrow f(att, e, e_{classe})$

$\forall \langle att, f \rangle \in CalculAttributsCible$:

$cible(e_{classe}).att \leftarrow f(att, e, e_{classe})$

2.5 Calcul de mesures

Les méthodes visant à extraire des connaissances à partir de données linguistiques utilisent souvent des mesures pour classer les candidats qu'elles produisent selon leur capacité (estimée) à correspondre à l'information recherchée.

2.5.1 Mesures classiques

Nous considérons tout d'abord une première catégorie de mesures, réalisées simplement à partir de la valeur des attributs de l'objet évalué (la valeur de cette mesure pouvant être stockée sur ses attributs).

Dans notre modèle, cela peut être effectué avec les fonctions AppliquerSurNœuds et AppliquerSurArcs, en passant en paramètre la fonction qui :

- calcule la mesure à partir de la valeur d'attributs de l'objet ;
- stocke cette valeur sur un autre attribut de l'objet.

Supposons que nous ayons une mesure calculant la valeur de l'attribut *capacité* d'un arc à partir de celle des attributs *nbdoccurrences* et *confiance*.

Pour obtenir les valeurs correspondantes sur tous les arcs vérifiant un filtre *f* donné, on appellera :

AppliquerSurNœuds(*f,CalculeCapacite*)

avec CalculeCapacite(nœud *v*) : {v.capacité=v.nboccurrences * v.confiance}

2.5.2 Mesures d'association

L'extraction de connaissances à partir de relations entre objets linguistiques vise souvent à obtenir des associations d'objets, et si possible à évaluer une propriété de cette association (sa force, son importance, etc.). Cela correspond, au niveau des graphes, à des mesures basées sur le nombre d'occurrences des nœuds (objets linguistiques) et des arcs (relations entre ces objets linguistiques).

2.5.2.1 Valeurs à obtenir

Considérons le cas pratique d'un graphe issu d'un corpus (dont le premier niveau représente toutes les occurrences, et le second le regroupement par termes), dans lequel nous voudrions calculer une mesure d'association entre deux termes donnés.

Pour cela, on doit se baser sur :

- le nombre d'entités de référence (généralement les phrases) ;
- les nombres d'apparitions (dans des entités distinctes) des termes :
  - le nombre d'occurrences du premier terme ;
  - le nombre d'occurrences du second terme ;
- les nombres d'apparitions (dans des entités distinctes) des relations entre termes :
  - le nombre de relations où interviennent les occurrences du premier terme ;
  - le nombre de relations où interviennent les occurrences du second terme ;
  - le nombre de relations entre les occurrences du premier terme et celles du second ;
  - éventuellement le nombre total de relations .

Pour mesurer l'association entre termes (regroupant toutes les occurrences, donc au niveau supérieur), il faut revenir aux occurrences (au niveau inférieur) pour obtenir les informations nécessaires.

Il serait fastidieux de parcourir ainsi le graphe, d'autant plus que cela peut être fait lors de l'émergence, en utilisant les fonctions de calcul d'attributs. En effet, pour chaque arc du niveau inférieur étudié, on peut, sur l'arc (au niveau supérieur) représentant la classe à laquelle il appartient :

- sur l'arc lui-même : incrémenter le compteur d'occurrences ;
  - CalculAttributsArc : (<nboccurrences, incr_arc>)
  - avec incr_arc(att, e_{source} , e_{classe}) : e_{classe} .att+1
- sur sa source : incrémenter le compteur d'occurrences de relations dont il est le premier argument
  - CalculAttributsSource : (<nboccurrences, incr_s>)
  - avec incr_s(att, e_{source} , e_{classe}) : source(e_{classe}).att+1
- sur sa cible : incrémenter le compteur d'occurrences de relations dont il est le second argument.
  - CalculAttributsCible : (<nboccurrences, incr_c>)
  - avec incr_c(att, e_{source} , e_{classe}) : cible(e_{classe}).att+1

Dans cet exemple, on a considéré qu'il n'y avait qu'une seule relation étudiée, ce qui n'est pas toujours le cas en pratique.

Les différentes occurrences qui peuvent être utiles dépendent des contraintes sur les 3 composantes d'un lien (les 2 objets reliés – les nœuds – et la relation caractérisant le lien). Elles sont présentées dans le tableau 18, où :

- A et B sont les nœuds représentant deux classes d'équivalence, dont on qualifiera les nœuds les composant respectivement de « type *a* » et de « type *b* »;

- X est l'arc représentant les occurrences de relations de type *R*, entre nœuds de type *a* et de type *b*.

Un exemple illustrant ce choix de variables est présentée dans la figure 31.

Nombre	*Concerne*	*Alias*[104]	*Sur le graphe : on compte (au niveau inférieur à celui de A, B, X)…*
Entités textuelles			
entités	*global*	*nbentites*	toutes les entités
Nœuds			
a	*A*	*A.nboccs*	les nœuds de type *a*
b	*B*	*B.nboccs*	les nœuds de type *b*
*	*global*	*totaloccs*	tous les nœuds
Arcs (relations)			
(a,R,b)	X	$X.nboccs[R]$	les relations de type *R* entre nœuds de type *a* et *b*
(a,*,b)		$X.nboccs_{toutesrel}$	les relations de type quelconque entre nœuds de type *a* et *b*
(a,R,*)	A	$A.d^{+}[R]$	les relations de type *R* partant de nœuds de type *a*
(a,*,*)		$A.d^{+}_{toutesrel}$	les relations de type quelconque partant de nœuds de type *a*
(*,R,b)	B	$B.d^{-}[R]$	les relations de type *R* arrivant à des nœuds de type *b*
(*,*,b)		$B.d^{-}_{toutesrel}$	les relations de type quelconque arrivant à des nœuds de type *b*
(*,R,*)	*global*	$totaloccs[R]$	les relations de type R
(*,*,*)		$totaloccs_{toutesrel}$	toutes les relations

Tableau 18: Occurrences utilisées dans le calcul de mesures

Nous définissons alors des fonctions à 12 arguments, qui attendent comme paramètres les nombres d'occurrences calculés pour le lien et les deux nœuds (dans l'ordre : *nbentités*, *nbocc(a)*, *nbocc(b)*, *nbocc(*)*, *nbocc(a,R,b)*, *nbocc(a,*,b)*, *nbocc(a,R,*)*, *nbocc(a,*,*)*, *nbocc(*,R,b)*, *nbocc(*,*,b)*, *nbocc(*,R,*)*, *nbocc(*,*,*)*) ;

Définition de type:
F12 : fonction($\mathbb{R}^{12} \rightarrow \mathbb{R}$) ;[105]

[104]Les noms donnés dans cette colonne ne sont destinés qu'à permettre de désigner facilement l'attribut correspondant par la suite ; ils ne préjugent en rien des noms que ces derniers peuvent avoir dans l'implémentation.

[105]Pour l'implémentation, on peut bien sûr le limiter aux nombres décimaux (« float »).

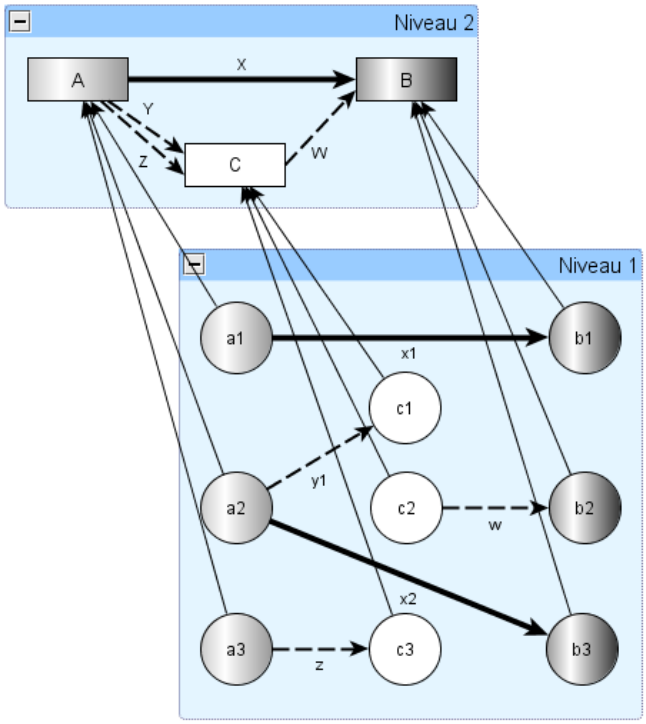


Figure 31: Graphe multiniveau mettant en évidence les classes d'équivalence utilisées pour le calcul des mesures

On peut par exemple définir ainsi les mesures habituelles dans l'extraction basée sur la co-occurrence, telles que l'information mutuelle ou WMI :

MI ($\mathbb{R}$ *nent*, *na*, *nb*, *nX*, *arb*, *aXb*, *arX*, *aXX*, *Xrb*, *XXb*, *XrX*, *XXX*) → $\mathbb{R}$:
{renvoie l'information mutuelle à partir des valeurs d'occurrence passées en paramètre}

$$\rightarrow \log\left(\frac{aXb \cdot nent}{na \cdot nb}\right)$$

WMI ($\mathbb{R}$ *nent*, *na*, *nb*, *nX*, *arb*, *aXb*, *arX*, *aXX*, *Xrb*, *XXb*, *XrX*, *XXX*) → $\mathbb{R}$:
{renvoie WMI à partir des valeurs d'occurrence passées en paramètre}

$$\rightarrow \left(\frac{arb}{nent}\right) \cdot \log\left(\frac{arb \cdot XrX}{arX \cdot Xrb}\right)$$

2.5.2.2 Opérateur : calcul de mesure d'association

On peut ainsi calculer, pour tous les liens d'un certain type (c'est-à-dire tous les arcs vérifiant une certaine propriété), des mesures d'association. On introduit à cet effet la fonction

CalculeF12, qui doit être normalement utilisée après l'émergence (qui permet de calculer les nombre d'occurrences et de co-occurrences dont on a besoin).

CalculeF12
Paramètres d'entrée :
attribut *att*,
entier *Niv*,
fonction($E \rightarrow bool$) *filtre*,
relation *Rel*[106],
F12 *m*;
Définition :
Pour tous les arcs du niveau Niv *vérifiant* filtre*, calcule la mesure* m *relative à la relation* Rel*, et stocke la valeur sur l'attribut* att.

Variables : arc e.
Corps :
$\forall e \in E_{Niv}$:
si (filtre(e)) :
e.att ← CalculeF12SurArc(Rel,e,m)

avec

CalculeF12SurArc (relation *R*, arc *X*, F12 *f*) → valeur :
{Calcule la valeur de la mesure représentée par f *, basée sur la relation* R *sur l'arc X}*

Variables : nœuds *A,B*;
corps :
A ← source(X); B ← cible(X);
→ f(*nbentites* , *A.nboccs* , *B.nboccs* , *totaloccs* , $X.nboccs[R]$, $X.nboccs_{toutesrel}$, $A.d^{+}[R]$, $A.d^{+}_{toutesrel}$, $B.d^{-}[R]$, $B.d^{-}_{toutesrel}$, $totaloccs[R]$, $totaloccs_{toutesrel}$)

Cette dernière fonction n'a pas pour but d'être utilisée seule, mais sert pour le calcul global des valeurs des mesures.

2.5.2.3 Opérateurs : calcul de mesure combinée

Lorsqu'on a calculé des mesures pour certains arcs, ainsi que pour chaque extrémité, on peut alors souhaiter pouvoir calculer une mesure pour la relation principale, à partir des valeurs de ces mesures secondaires.

[106]Si la mesure utilisée n'utilise pas le type de la relation, on peut passer en paramètre un type de relation quelconque.

Nous définissons pour cela des fonctions à 3 arguments (mesure pour l'arc, mesure pour la source, mesure pour la cible) :

Définition de type :
F3 : $\mathbb{R}^3 \rightarrow \mathbb{R}$;

L'opérateur correspondant au calcul d'une nouvelle mesure sur les arcs, à partir de mesures déjà calculées sur les arcs et leurs extrémités, est :

CalculeF3
Paramètres d'entrée :
attribut *att*,
entier *Niv*,
fonction($E \rightarrow bool$) *filtre*,
attribut *attarc*, *attsource*, *attcible*.
F3 *m*;
Définition :
Pour tous les arcs du niveau Niv *vérifiant* filtre*, calcule la mesure* m *à partir des 3 valeurs (l'attribut* attarc *de l'arc, l'attribut* attsource *de la source de l'arc, l'attribut* attcible *de la cible de l'arc,) et stocke le résultat sur l'attribut* att.
Variables : arc e.

Corps :
AppliqueSurArcs()
$\forall e \in E_{Niv}$:
si (filtre(e)) :
e.att=m(e.attarc,source(e).attsource, cible(e).attcible).

2.6 Union, intersection, différence

L'un des principaux buts de la représentation des connaissances linguistiques sous forme de graphes est de permettre le mélange d'informations en fusionnant les graphes. On peut ainsi imaginer fusionner des graphes issus de corpus, de dictionnaires, etc.

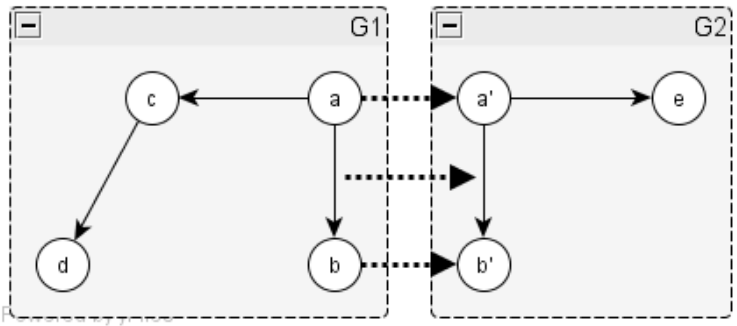
2.6.1 Principe

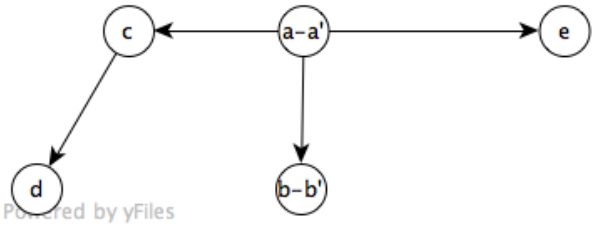
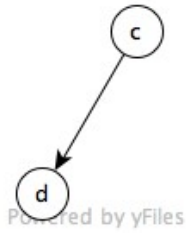
Il y a de nombreuses manières de mélanger les informations de deux graphes ; nous n'étudierons que les cas les plus simples :

- intersection : le graphe résultat contient les objets que les deux graphes ont en commun ;

- union : le graphe résultat contient tous les objets des deux graphes, qu'ils soient ou non en commun ;
- différence : le graphe résultat contient les objets qui appartiennent au premier graphe et pas au second.

Ces opérations sont définies sur l'ensemble des niveaux des graphes opérandes.

<table>
<tr><td>Avant…</td><td> 

(les nœuds <code>a</code> et <code>b</code> sont respectivement identifiés avec <code>a'</code> et <code>b'</code>,
l'arc entre <code>a</code> et <code>b</code> est identifié à celui entre <code>a'</code> et <code>b'</code>)</td></tr>
<tr><td>Après union
(G1∪G2)</td><td> 
</td></tr>
<tr><td>Après intersection
(G1∩G2)</td><td> 
</td></tr>
<tr><td>Après complément (ou différence)</td><td> 

G2\G1</td><td> 

G2\G1</td></tr>
</table>

Tableau 19: Exemples d'union, d'intersection et de différence entre graphes

2.6.1.1 Identité entre nœuds et entre arcs

Il est primordial, lorsqu'on cherche à mélanger les informations de deux graphes, de savoir quels objets ils ont en commun.

Pour cela, il faut pouvoir identifier (associer de manière unique) les nœuds et les arcs du premier et du second graphe (vus comme *identiques*[107]) : deux nœuds ou deux arcs ainsi associés ne donnent respectivement qu'un seul nœud ou un seul arc dans le graphe résultant de la fusion.

C'est à partir d'une telle identification préalable qu'on peut ensuite réaliser des opérations entre ensembles (union, intersection, différence), tel que cela est illustré par le tableau 19.

2.6.1.2 Graphes résultats

Nous allons étudier maintenant, pour chaque composante du graphe, la manière dont les opérations d'union, d'intersection ou de différence doivent se comporter.

Soit *H* le graphe résultat d'une telle opération entre *G1* et *G2*.

- les nœuds, les arcs intraniveau, et les arcs interniveau étant des ensembles, on peut appliquer les opérations ensemblistes classiques sur chacune de ces composantes :
  - union ($H=G1\cup G2$) : $H.V=G1.V\cup G2.V$, $H.E\,G1.E\cup G2.E$, $H.F=G1.F\cup G2.F$;
  - intersection ($H=G1\cap G2$) : $H.V=G1.V\cap G2.V$, $H.E=G1.E\cap G2.E$, $H.F=G1.F\cap G2.F$;
  - différence (complémentation) : ($H=G1\setminus G2$) : $H.V=G1.V\setminus G2.V$, $H.E=G1.E\setminus G2.E$, $H.F=G1.F\cup G2.F$;
- les arcs copiés ou fusionnés conservent bien sûr leurs extrémités, donc les fonctions qui leur sont associées conservent leurs valeurs :
  - $H.\alpha_i(e) = G1.\alpha_i(e)$ si $e\in G1.E$, $G2.\alpha_i(e)$ si $e\in G2.E$;
  - $H.\varphi_i(f) = G1.\varphi_i(f)$ si $f\in G1.F$, $G2.\varphi_i(f)$ si $f\in G2.F$;
- les attributs des objets (nœuds et arcs) conservés ont un comportement qui dépend de l'opération réalisée et du cas dans lequel il se trouve :
  - si l'objet n'appartient qu'à un seul graphe, il est copié dans le graphe résultat : les attributs de la copie sont copiés de l'original (on utilise cependant une *fonction de copie* pour permettre de modifier la représentation de l'information lors de la copie[108]);
  - si l'objet est commun aux deux graphes, il faut fusionner les deux instances, qui n'ont pas nécessairement les mêmes attributs : on utilise alors une *fonction de fusion* de l'information portée par les attributs.

[107]L'utilisateur doit définir pour cela des fonctions d'identité (associant de manière unique les objets des deux graphes) : deux nœuds (ou deux arcs) peuvent alors être considérés comme identiques même si les valeurs de leurs attributs diffèrent. On unifiera les objets reconnus comme « identiques » (d'après ces fonctions) lors du traitement.

[108]Cela est très utile dans le cas de l'union, où les arcs et les nœuds du résultat peuvent provenir des deux graphes de départ, qui n'ont pas nécessairement les mêmes attributs.

2.6.2 Copie et fusion d'objets

Lorsqu'il faut créer des arcs (qu'on soit en mode de copie ou de fusion), cela se fait bien sûr à partir de nœuds déjà créés. Pour cela, il est nécessaire de pouvoir manipuler facilement la correspondance entre les nœuds des graphes de départ et les nœuds correspondants qu'on a créé dans le graphe résultat. C'est pourquoi on suppose l'existence de la fonction suivante[109] :

NouveauNœud(nœud *v*) → nœud :
{renvoie, s'il existe, le nœud du graphe résultat créé à partir de v *(par copie ou fusion de nœuds)}*

Les opérateurs que nous présentons maintenant ne sont pas destinés à être utilisés directement, mais à être appelés par les opérateurs d'intersection, d'union, ou de différence que nous présenterons plus tard. Ils permettent d'effectuer :

- la copie de nœuds, d'arcs intraniveau d'un graphe vers un autre (avec copie des attributs correspondants) ;
- la fusion de nœuds, d'arcs intraniveau de deux graphes vers un troisième (avec fusion des attributs correspondants) .

Pour cela, ces opérateurs utilisent des fonctions de copie ou de fusion définies par l'utilisateur.

2.6.2.1 Copie de nœuds et d'arcs

AjouteCopieNœud(nœud *cop*, nœud *v*, entier *Niv*,
fonction($V * V$) *copieattsnœud*) :
{ajoute dans le graphe courant cop*, nœud représentant la copie du nœud* v*, au* Niv*-ième niveau, en copiant les attributs selon* copieattsnœud*}*
AjouteNœud(*cop*,Niv);
NouveauNœud(v) ← cop;
copieeattsnœud(cop, v).

AjouteCopieArcIntra(arc *cop*, arc *e*, entier *Niv*, fonction($E * E$) *copieattsarc*) :
{ajoute dans le graphe courant cop*, arc représentant la copie de l'arc* e*, au* Niv*-ième niveau, en copiant les attributs selon* copieattsarc*}*
AjouteArcIntra(*cop*,Niv, NouveauNœud(source(e)),
NouveauNœud(cible(e)));
copieattsarc(cop, e).

[109]On ne précise pas comment les valeurs doivent être implémentées ; la façon la plus simple de le faire est sans doute d'utiliser des tables de hachage lors des opérations d'intersection, union, ou différence, pour conserver les informations nécessaires.

2.6.2.2 Fusion de nœuds et d'arcs

AjouteFusionNœud(nœud *fus*, nœud *v1*, nœud *v2*, entier *Niv*,
fonction($V * V * V$) *fusionneattsnœud*) :
{ajoute dans le graphe courant fus*, nœud représentant la fusion des nœuds* v1 *et* v2*, au* Niv*-ième niveau, en fusionnant les attributs selon* fusionneattsnœud*}*
AjouteNœud(*fus*,Niv);
NouveauNœud(v1) ← vh;
NouveauNœud(v2) ← vh;
fusionneattsnœud(fus, v1, v2).

AjouteFusionArcIntra(arc *fus*, arc *e1*, arc *e2*, entier *Niv*,
fonction($E * E * E$) *fusionneattsarc*) :
{ajoute dans le graphe courant fus*, arc représentant la fusion des arcs* e1 *et* e2*, au* Niv*-ième niveau, en fusionnant les attributs selon* fusionneattsarc*}*
AjouteArcIntra(*fus*,Niv, NouveauNœud(source(e1)),
NouveauNœud(cible(e1)));
fusionneattsarc(fus, e1, e2).

2.6.2.3 Gestion des arcs interniveau

Un arc interniveau n'ayant pas d'attributs et étant nécessairement unique à partir d'une source et d'une cible données, sa création dans le graphe résultant est la même selon qu'il soit issu d'un arc interniveau présent dans un seul graphe ou dans les deux.

AjouteNouvelArcInter(arc *nouvf*, arc *f*, entier *Niv*) :
{ajoute dans le graphe courant nouvf*, arc interniveau entre la source et la cible correspondant aux extrémités de* f*, au* Niv*-ième niveau}*
AjouteArcInter(*nouvf*, Niv, NouveauNœud(source(f)),
NouveauNœud(cible(f))).

2.6.3 Opérateur : union de graphes

Le premier opérateur que nous présentons est l'*union*. C'est le plus compliqué des trois, le résultat produit n'étant pas sous-graphe d'un des graphes de départ.

Pour chaque type d'objet, l'opérateur calcule l'union des ensembles concernés, en étudiant d'abord les nœuds, puis les arcs intraniveau, et enfin les arcs interniveau. Selon qu'un objet est commun aux deux graphes (il a deux instances jugées identiques, l'une dans le premier, l'autre dans le second) ou non, on procède alors à la fusion ou à la copie.

Union

Paramètres d'entrée :

graphe *G1*, fonction($V*V*V$) *fus-v*,
graphe *G2*, fonction($E*E*E$) *fus-e*
fonction($G1.V*G2.V \rightarrow bool$) *Vident*, fonction($V*V$) *cop-v*,
fonction($G1.E*G2.E \rightarrow bool$) *Eident*, fonction($E*E$) *cop-e*

Renvoie :

graphe

Définition :

Réalise (dans le graphe courant) l'union entre les graphes G1 *et* G2*, l'identité entre nœuds, et entre arcs intraniveau, étant établie respectivement à l'aide de* VIdent *et* EIdent*; la valeur des attributs des arcs et nœuds créés est établie à l'aide des fonctions* fus-v *et* fus-e *(pour la partie fusionnée, à l'intersection* $G1 \cap G2$ *) et* cop-v *et* cop-e *(pour la partie copiée, le reste)*

Variables : graphe *H*, entier *Niv*, nœud *vh*, arc *eh*, *fh*.

Corps :

{Union des nœuds}

$\forall v1 \in G1.V$:
  si ($\exists v2 \in G2.V \mid VIdent(v1, v2)$) :
    $v1 \in G1.V_{Niv}$: H.AjouteFusionNœud(*vh*, v1,v2, Niv, fus-v)
  sinon :
    $v1 \in G1.V_{Niv}$: H.AjouteCopieNœud(*vh*, v1, Niv, cop-v)
$\forall v2 \in G2.V$:
  si ($\neg\exists v1 \in G1.V \mid VIdent(v1, v2)$) :
    $v2 \in G2.V_{Niv}$: H.AjouteCopieNœud(*vh*, v2, Niv, cop-v)

{Union des arcs intraniveau}

$\forall e1 \in G1.E$:
  si ($\exists e2 \in G2.E \mid EIdent(e1, e2)$) :
    $e1 \in G1.E_{Niv}$: H.AjouteFusionArcIntra(*eh*,e1,e2, Niv,fus-e)
  sinon :
    $e1 \in G1.E_{Niv}$: H.AjouteCopieArcIntra(*eh*, e1, Niv, cop-e)
$\forall e2 \in G2.E$:
  si ($\neg\exists e1 \in G1.E \mid EIdent(e1, e2)$) :
    $e2 \in G2.E_{Niv}$: H.AjouteCopieArcIntra(*eh*, e2, Niv, cop-e)

{Union des arcs interniveau}

$\forall f1 \in G1.F$:
  $f1 \in G1.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f1, Niv)
$\forall f2 \in G2.F$:
  si ($\neg\exists f1 \in G1.F \mid VIdent(source(f1), source(f2)) \wedge VIdent(cible(f1), cible(f2))$) :
    $f2 \in G2.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f2, Niv)

$\rightarrow$ H

Il convient de remarquer (et cela est valable aussi bien pour l'union que pour l'intersection et la différence) que l'opérateur ne peut bien fonctionner que si les paramètres qu'on lui fait utiliser respectent les contraintes liées à leur rôle :

- les fonctions d'identité doivent associer de manière unique les objets des deux graphes :
  - elles doivent définir une relation d'équivalence (symétrie, transitivité, réflexion) ;
  - deux nœuds d'un même graphe ne peuvent être identiques ;
- on ne peut identifier des arcs si leurs extrémités ne sont pas identifiées entre elles.

2.6.4 Opérateur : intersection de graphes

Intersection

Paramètres d'entrée :

graphe *G1*,
graphe *G2*,
fonction($G1.V * G2.V \rightarrow bool$) *VIdent*,
fonction($G1.E * G2.E \rightarrow bool$) *EIdent*,
fonction($V * V * V$) *fus-v*,
fonction($E * E * E$) *fus-e*

Renvoie :

graphe

Définition :

Réalise (dans le graphe courant) l'intersection entre les graphes G1 *et* G2*, l'identité entre nœuds, et entre arcs intraniveau, étant établie respectivement à l'aide de* Vident *et* EIdent *; la valeur des attributs des arcs et nœuds créés est établie à l'aide des fonctions* fus-v *et* fus-e*.*

Variables : graphe *H*, entier *Niv*, nœud *vh*, arc *eh*, *fh*.

Corps :

$\forall v1 \in G1.V$:
  si ($\exists v2 \in G2.V \mid VIdent(v1, v2)$) :
    $v1 \in G1.V_{Niv}$: H.AjouteFusionNœud(*vh*, v1,v2, Niv, fus-v)
$\forall e1 \in G1.E$:
  si ($\exists e2 \in G2.E \mid EIdent(e1, e2)$) :
    $e1 \in G1.E_{Niv}$: H.AjouteFusionArcIntra(*eh*,e1,e2, Niv,fus-e)
$\forall f1 \in G1.F$:
  si ($\exists f2 \in G2.F \mid VIdent(source(f1), source(f2)) \wedge VIdent(cible(f1), cible(f2))$) :
    $f1 \in G1.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f1, Niv)
$\rightarrow$ H

L'intersection ne garde que les nœuds et les arcs qui sont présents dans les deux graphes.

L'opérateur effectue la fusion des nœuds, puis celle des arcs intraniveau et interniveau. Pour cela, elle parcourt le premier graphe et ne conserve (pour la fusion) que les objets dont on a trouvé un « identique » dans le second graphe.

2.6.5 Opérateur : différence de graphes

La différence conserve les objets qui sont présents dans le premier graphe mais pas dans le second. Le processus de l'opérateur est relativement similaire à celui de l'intersection, puisqu'il parcourt lui aussi les objets du premier graphe, la différence étant que l'intersection conserve seulement ceux qu'on trouve en commun dans l'autre graphe (procédant alors à une fusion) alors que la différence conserve ceux qui n'y sont pas (procédant alors à leur copie).

Différence

Paramètres d'entrée :

graphe *G1*,
graphe *G2*,
fonction($G1.V * G2.V \rightarrow bool$) *VIdent*,
fonction($G1.E * G2.E \rightarrow bool$) *EIdent*,
fonction($V * V$) *cop-v*,
fonction($E * E$) *cop-e*

Renvoie :

graphe

Définition :

Réalise (dans le graphe courant) la différence entre les graphes G1 *et* G2*, l'identité entre nœuds, et entre arcs intraniveau, étant établie respectivement à l'aide de* Vident *et* EIdent *; la valeur des attributs des arcs et nœuds créés est établie à l'aide des fonctions* cop-v *et* cop-e*.*

Variables : graphe *H*, entier *Niv*, nœud *vh*, arc *eh*, *fh*.

Corps :

$\forall v1 \in G1.V$:
si ($\neg \exists v2 \in G2.V \mid VIdent(v1, v2)$) :
$v1 \in G1.V_{Niv}$: H.AjouteCopieNœud(*vh*, v1, Niv, cop-v)
$\forall e1 \in G1.E$:
si ($\neg \exists e2 \in G2.E \mid EIdent(e1, e2)$) :
$e1 \in G1.E_{Niv}$: H.AjouteCopieArcIntra(*eh*, e1, Niv, cop-e)
$\forall f1 \in G1.F$:
si ($\neg \exists f2 \in G2.F \mid VIdent(source(f1), source(f2)) \wedge VIdent(cible(f1), cible(f2))$) :
$f1 \in G1.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f1, Niv)
$\rightarrow$ H

3 Représentation complexe

Le modèle et les opérateurs associés que nous avons présentés jusqu'ici permettent de couvrir une grande partie des données linguistiques disponibles. Cependant, il existe des données dont la structure n'est pas prévue dans notre modèle, en particulier les relations à 3 arguments ou plus (pouvant par exemple provenir d'analyses de dépendance). Comment, dans notre modèle, représenter de tels cas, plus complexes, de manière simple ?

3.1 Arcs à plus de deux extrémités

3.1.1 Relations n-aires

Si l'extraction de connaissances linguistiques concerne le plus souvent des données pouvant être représentées respectivement par un nœud et un arc entre deux nœuds (comme les termes et les relations binaires entre eux), les choses se compliquent si on souhaite représenter :

- des arcs n-aires ($n>2$) ;
- des relations entre arcs.

Plusieurs cas peuvent causer ce problème :

- un *hyper-arc* reliant 3, 4, voire plus de nœuds (par exemple, ceux associés à « pomme », « de », et « terre »)[110] ;
- une relation entre deux objets linguistiques qui ne sont pas représentés par des nœuds (par exemple, dans les bicollocations, entre des couples monolingues de termes) ;

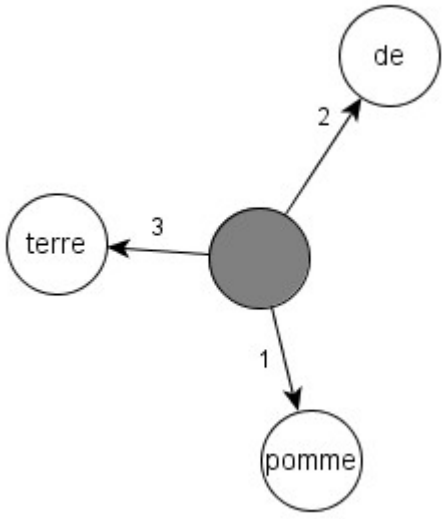


Figure 32: Représentation d'un hyperarc dans la représentation complexe de MuLLinG

[110]Comme dans les hypergraphes définis par Berge (1987).

Nous proposons donc d'ajouter à notre modèle une représentation particulière à ces objets, en reprenant ce qui se fait classiquement : présenter la relation non plus comme un arc, mais comme un nœud relié à ses arguments. Cela constitue ce que nous appelons la *représentation complexe* de notre modèle. Dans celle-ci :

- la relation est *matérialisée* par un nœud ;
- les nœuds représentant les arguments de la relation sont reliés (par des arcs spécifiques) au nœud matérialisant la relation.

Plus précisément, une relation *R* entre *n* nœuds du même niveau *Niv* ($(x_1, x_2, \ldots, x_n)$ avec $\forall i, x_i \in V_{Niv}$) aura pour représentation complexe :

- un nœud $v_R \in V_{niv}$, matérialisation de la relation ;
- des arcs entre la matérialisation de la relation (v_R) et chacun de ses arguments (x_i).

Ces arcs, qui ne peuvent exister *qu'entre nœuds de même niveau*[111], ne portent qu'un numéro d'ordre[112]. Nous les nommons *arcs arguments*.

Ils ne font pas partie de E_{Niv} , car ils ont des contraintes supplémentaires : un seul numéro d'ordre possible par couple (matérialisation de lien, argument), des numéros d'ordre tous distincts pour chaque argument d'une même matérialisation.

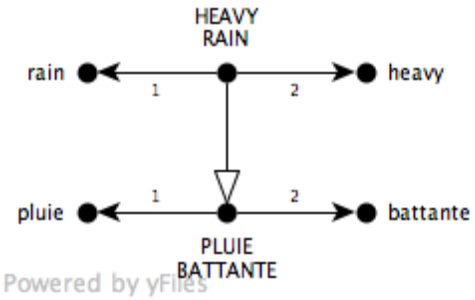


Figure 33: Utilisation de la représentation complexe pour les bicollocations

3.1.2 Définition précise

Nous définissons donc une nouvelle version de nos graphes linguistiques multiniveau, quand ils contiennent de telles représentations complexes, en ajoutant à la structure déjà présentée dans la section 1.3 de ce chapitre (page 128):

- C : l'ensemble des arcs arguments, formé de *n* ensembles disjoints $C_1, \ldots, C_n$ ($i \in \{1, \ldots, n\}$)
  - chaque C_i (correspondant à un niveau) est lui même formé de *m* ensembles disjoints $C_{i,1}, \ldots, C_{i,m}$ ($j \in \{1, \ldots, m\}$), définis comme $C_{i,j} = \{\langle x, y \rangle \in V_i \times V_i \mid \rho_i(x, j) = y\}$;

[111] Ils relient un nœud matérialisant une relation au niveau *Niv*, à l'un des arguments de cette relation, également au niveau *Niv*.

[112] Le « numéro d'ordre » dit s'il s'agit du premier argument, du second, du troisième, etc.

- P : l'ensemble des fonctions arguments $\rho_i : V_i \times \mathbb{N} \rightarrow V_i$ qui, pour un nœud v et un entier j, donnent le j-ème argument de v.

Il n'y a pas de distinction, au niveau de la structure du graphe, entre les nœuds matérialisant les relations et les nœuds « normaux ». Cela permet notamment de considérer les arcs représentant des relations faisant intervenir un nœud lui-même matérialisation d'une relation comme n'importe quel autre arc, sur lequel on peut utiliser les opérateurs habituels.

Par exemple, la figure 33 montre comment on peut représenter simplement une bicollocation grâce à cette représentation complexe, en reliant deux nœuds matérialisant deux collocations monolingues (dont les arguments sont leurs bases et leurs collocatifs).

Par opposition à cette représentation complexe, nous appellerons, par la suite, *représentation simple* la structure habituellement utilisée, pour un objet seul (un nœud), ou pour deux objets (un arc entre deux nœuds).

Dans la suite, afin de simplifier le discours, le nœud v_c qui est la cible de l'arc portant une relation de type « i-ème argument » dont la source est v_s sera simplement désigné comme « i-ème argument » de v_s.

Le tableau 20 présente les arcs et les nœuds impliqués dans la représentation complexe d'une relation, selon son nombre d'arguments ; à titre de comparaison, on indique aussi la représentation simple des relations unaires et binaires.

Nombre d'arguments	Représentation complexe		Représentation simple	
	Nœuds	Arcs	Nœuds	Arcs
1	v_R, v_1	(v_R, v_1)	v_1	
2	v_R, v_1, v_2	(v_R, v_1), (v_R, v_2)	v_1, v_2	(v_1, v_2)
3	v_R, v_1, v_2, v_3	(v_R, v_1), (v_R, v_2), (v_R, v_3)		
...				
N	v_R, v_1, v_2, v_3, …, v_{N-1}, v_N	(v_R, v_1), (v_R, v_2), (v_R, v_3), …, (v_R, v_N)		

Tableau 20: Représentations simples et complexes d'une relation R entre v_1, v_2, … et v_N

3.2 Nouveaux opérateurs de la représentation complexe

Les opérateurs présentés jusqu'ici correspondaient à la représentation simple, et ne sont donc pas utilisables pour la représentation complexe. Ils nous faut donc de nouveaux opérateurs. Nous allons pour cela adapter ceux existants. Mais il y a d'abord des opérateurs nouveaux à introduire, car leur existence est directement liée à la représentation complexe.

3.2.1 Accès et modification du graphe

Aux opérateurs d'accès et de modification de graphe présentés pour la représentation simple, nous en ajoutons un nouveau, afin de traiter les objets du graphes qui apparaissent avec la représentation complexe, en l'occurrence les arcs correspondant aux arguments de relation :

AjouteArcArgument(arc *e*, entier *i*, nœud *s*, nœud *c,* entier *numarg*) :
{ajoute l'arc argument numéro numarg *entre* s *et* c *(niveau* i*); présupposé :* $s, c \in V_i,\ e \in C_i\}$

$\rho_i(s, numarg) \leftarrow c\ ;\ e \leftarrow \langle s, c \rangle$

Nous définissons également les fonctions permettant d'accéder facilement aux arguments d'un nœud qui matérialise une relation :

Argument(nœud *v*, entier *numarg*) :
{renvoie le numarg*-ième argument du nœud* v *; présupposé :* v *est la matérialisation d'une relation}*

$v \in V_i : \rightarrow \rho_i(v, numarg)$

Arg1(nœud *v*) = Argument(v,1).
Arg2(nœud *v*) = Argument(v,2).
etc.

3.2.2 Changement de représentation

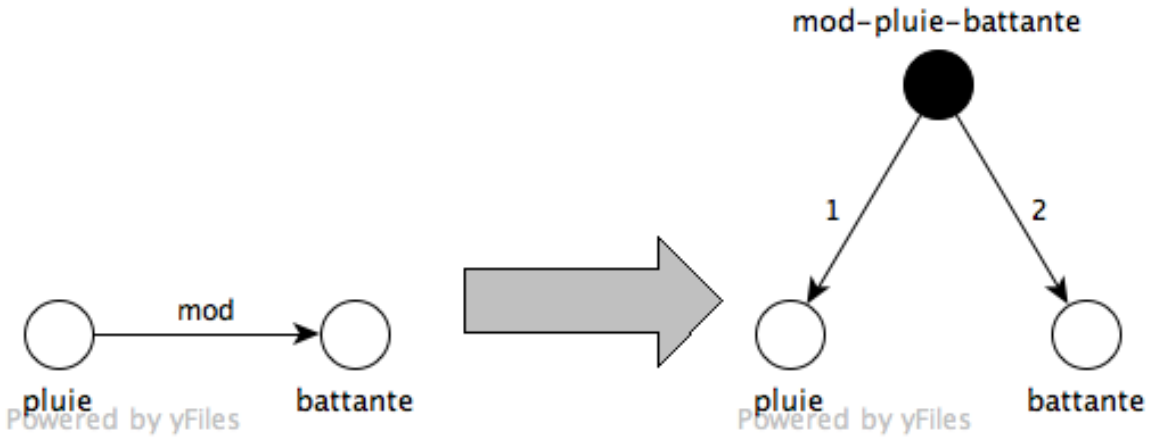


Figure 34: Matéralisation d'une relation en nœuds (représentation complexe)

Il peut être utile, même lorsqu'on utilise uniquement des relations binaires, de matérialiser les relations sous forme de nœuds (comme dans l'exemple des bicollocations présenté dans la figure 33)[113].

[113]On pourrait, au niveau de l'implémentation, choisir de « matérialiser » tous les arcs sous forme de noeuds, pour obtenir un graphe bipartite (arcs arguments entre relations et arguments des relations) pouvant être vu comme plus simple à manipuler. Il nous semble falloir éviter ce choix, qui serait plus coûteux en terme d'espace de stockage en mémoire.

À cet effet, nous présentons la manière de réaliser la matérialisation d'un arc (représentation simple) en nœud avec deux arguments (représentation complexe).

Le processus de cet opérateur, illustré par la figure 34, est relativement simple. Pour chaque arc à matérialiser sous forme de nœud, il faut :

- ajouter un nœud matérialisant la relation portée par l'arc ;
- ajouter les arcs arguments :
  - le premier entre la matérialisation de la relation, et la source de l'arc original ;
  - le second entre la matérialisation de la relation, et la cible de l'arc original ;
- supprimer l'arc original.

Nous introduisons l'opérateur réalisant cela ; nous utilisons ici encore une fonction pour calculer la valeur des attributs du nœud matérialisant la relation à partir des valeurs des attributs de l'arc qui la représentait.

MatérialisationArcs

Paramètres d'entrée :

entier *Niv*,
fonction($E \rightarrow bool$) *filtre*,
liste(<attribut,fonction($attribut \times V \rightarrow valeur$ >)
CalculAttributsMaterialisation.

Définition :

Matérialise les arcs du niveau Niv *vérifiant* filtre *; la valeur des attributs des nœuds ainsi créés est calculée grâce à* CalculAttributs

Variables : ensemble(arcs) $ArcsMater \leftarrow \emptyset$, nœud v_{mat} , arcs e_1 , e_2 .

Corps :

$\forall e \in E_{Niv}$:
  si filtre(e) :
    $ArcsMater \leftarrow ArcsMater \cup \{e\}$
$\forall e \in ArcsMater$:
  AjouteNœud(v_{mat} , Niv);
  AjouteArcArgument(e_1 , Niv, v_{mat} , source(e), 1);
  AjouteArcArgument(e_2 , Niv, v_{mat} , cible(e), 2);
  $\forall \langle att, f \rangle \in CalculAttributsMaterialisation$:
    $v_{mat}.att \leftarrow f(att, v)$
  SupprimeArcIntra(e).

3.3 Opérateurs adaptés à la représentation complexe

Les opérateurs présentés jusqu'ici étaient faits pour une représentation simple de l'information ; il faut les adapter à la représentation complexe. Dans ceux-ci, la source et la

cible d'un arc recevant (en général) un traitement identique, il suffit de généraliser ce traitement à tous les arguments de la relation matérialisée.

3.3.1 Émergence

Nous étudions tout d'abord l'adaptation des opérateurs d'émergence.

Il n'est pas nécessaire de modifier l'émergence de nœuds, qui ne concerne pas les arcs intraniveau. Au contraire, ce que nous avons appelé *émergence d'arcs* doit être adapté au cas complexe où les relations ne sont plus représentées par des arcs mais par des nœuds les matérialisant (ci-après nommés « nœuds-matérialisation »).

3.3.1.1 Création d'objets matérialisant la classe

Il faut pouvoir manipuler facilement la correspondance entre les identifiants de classes et les objets correspondants. C'est pourquoi on suppose l'existence de la fonction suivante :

Classe_VMater(chainecars *c*, tableau(nœuds) *t*>) → nœud :
{renvoie, s'il existe, le nœud-matérialisation correspondant à la matérialisation de la classe des nœuds-matérialisation identifiés par la chaîne c*, entre les arguments listés dans* t*}*

La fonction suivante doit retourner le nœud(-matérialisation) représentant la classe souhaitée, en le créant s'il n'existe pas déjà.

NœudMatérialisationClasse (chainecars *c*, tableau(nœuds) *t*, entier *Niv*) → nœud :
{renvoie le nœud-matérialisation (situé au niveau Niv*) correspondant à la classe d'équivalence d'arcs identifiée par* c *et les arguments listés dans* t*; s'il n'existe pas, il est créé}*
Variables : nœud v_{classe} , arc *e*.
Corps :
si (Classe_VMater(c,t) est défini) :
→ Classe_VMater (c,t)
sinon :
AjouteNœud(v_{classe} , Niv) ;
pour *i* de 1 à taille(t) :
AjouteArcArgument(*e*, Niv, v_{classe} , t[i], i) ;
Classe_VMater(c,t) ← v_{classe} ;
→ v_{classe}

3.3.1.2 Opérateur : émergence de relations matérialisées

Dans la représentation simple, on faisait émerger les arcs. Les choses sont plus compliquées dans la représentation complexe : faire émerger les classes de nœuds-matérialisation nécessite de créer, au niveau supérieur, de nouveaux nœuds représentant ces classes, comme le montre la figure 35.

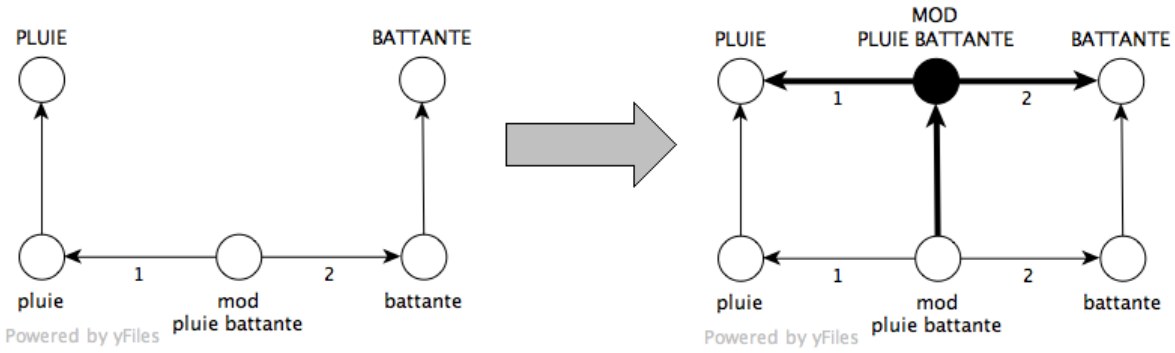


Figure 35: Émergence de liens au niveau supérieur (représentation complexe)

L'émergence de relations complexes (matérialisées), à partir d'une fonction donnée d'équivalence entre relations matérialisées :

- calcule, au niveau inférieur, les classes d'équivalence induites (non plus des classes d'arcs, mais des classes de nœuds-matérialisation) ;
- produit, au niveau supérieur :
  - les nœuds matérialisant ces classes, ainsi que les arcs arguments ; un tel nœud-matérialisation regroupe tous les nœuds-matérialisation (au niveau inférieur) *équivalents*, correspondant à des relations dont le premier argument appartient à la classe modélisée par le premier argument (du nœud-matérialisation au niveau supérieur), dont le deuxième argument appartient à la classe modélisée par le deuxième argument (du nœud-matérialisation au niveau supérieur), etc. ;
    - crée et modifie les valeurs des attributs de ces nœuds-matérialisations et de leurs arguments ;
- ajoute des arcs interniveau entre chaque nœud du niveau inférieur et le nœud (au niveau supérieur) modélisant la classe d'équivalence à laquelle il appartient .

ÉmergenceDeRelationsMaterialisées
Paramètres d'entrée :
entier *Niv*,
fonction($V \rightarrow bool$) *filtre*,
fonction($V_{Niv} \rightarrow chainecars$) *ClasseEquiv*,
liste(<attribut,fonction($attribut \times V \times V \rightarrow valeur$ >)
CalculAttributsMaterialisation,
Tableau[liste(<attribut,fonction($attribut \times V \times V \rightarrow valeur$ >)]
CalculAttributsArguments.

Définition :

Fait émerger, pour les nœuds-matérialisation du niveau Niv *vérifiant* filtre *et dont la classe d'équivalence est calculée par* ClassEquiv*, les nœuds-matérialisation au niveau* Niv+1*, correspondant à ces classes et aux classes de leurs arguments, dont la valeur des attributs est calculée grâce à* CalculAttributsMaterialisation *(alors que la valeur des attributs de leurs arguments est calculée grâce à* CalculAttributsArguments*).*

Variables : entier i, nœud v_{classe} , Tableau[nœuds] *ClassArgs*=[ ];

Corps :

$\forall v \in V_{Niv}$:
    si filtre(v) :
        pour *i* de 1 à nbarguments(v) :
            ClassArgs[i]=ClasseDuNœud(Argument(v,i));
        $v_{classe} \leftarrow$ NœudMaterialisationClasse(ClasseEquiv(e), ClassArgs, Niv+1) ;
        $\forall \langle att, f \rangle \in CalculAttributsMaterialisation$:
            $v_{classe}.att \leftarrow f(att, v, v_{classe})$
        pour i de 1 à taille(CalculAttributsArguments)
            $\forall \langle att, f \rangle \in CalculAttributsArguments[i]$:
                $Argument(v_{classe}, i).att \leftarrow f(att, v, v_{classe})$

3.3.2 Mesures d'association binaire

Pour le calcul de mesures, il nous faut adapter les opérateurs concernant les arcs ; cela touche donc les mesures d'association.

3.3.2.1 Opérateur : calcul de mesure d'association binaire

Lorsque nous sommes en présence de nœuds matérialisant des relations binaires, la seule différence avec le cas de la représentation simple est que les relations ne sont plus représentées par des arcs, mais matérialisées par des nœuds.

Nous pouvons donc utiliser les mêmes occurrences que celles présentées dans le tableau 18 (p.141), à la différence que *X*, qui représente les occurrences de relations de type *R* entre nœuds de type *a* et de type *b*, n'est plus un arc mais un nœud.

Un exemple illustrant ce choix de variables est présenté dans la figure 36.

Après application à l'opérateur CalculeF12 (défini dans la représentation simple) des légères modifications que cela implique, nous obtenons l'opérateur permettant de calculer une mesure d'association pour relations binaires matérialisées par des nœuds.

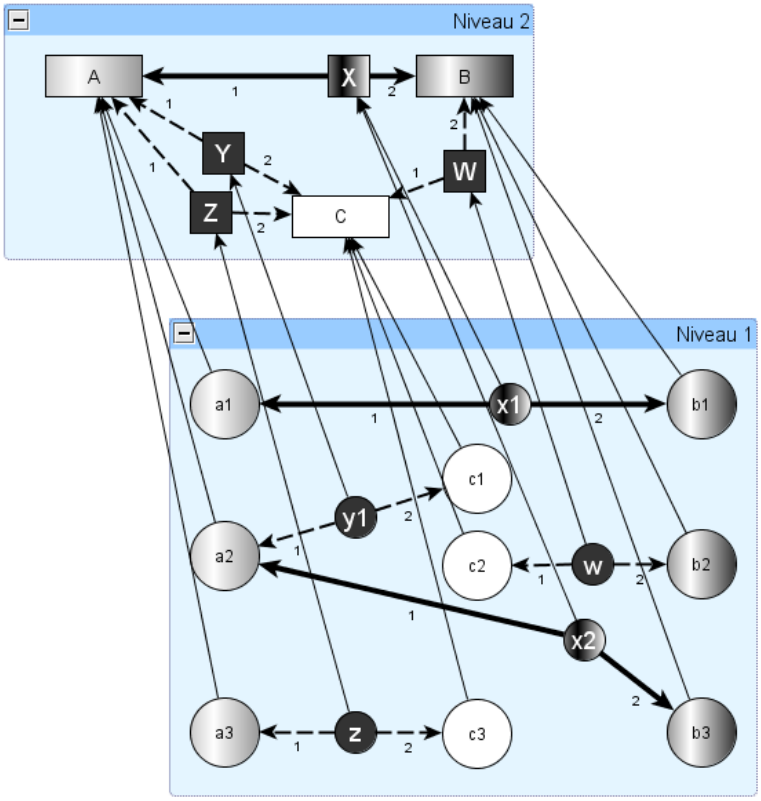


Figure 36: Graphe multiniveau mettant en évidence les classes d'équivalence (dans la représentation complexe), utilisées pour le calcul des mesures

CalculeF12Complexe

Paramètres d'entrée :

attribut *att*,
entier *Niv*,
fonction($V \rightarrow bool$) *filtre*,
relation *Rel*,
F12 *m*;

Définition :

Pour tous les nœuds-matérialisation du niveau Niv *vérifiant* filtre*, calcule la mesure* m *relative à la relation* Rel*, et stocke la valeur sur l'attribut* att.

Variables : arc *e*.

Corps :

$\forall v \in V_{Niv}$:
si (filtre(v)) :
v.att ← CalculeF12SurNœudMaterialisation(Rel,v,m)

avec

CalculeF12SurNœudMatérialisation (relation *R*, nœud *X*, F12 *f*) → valeur :
{Calcule la valeur de la mesure représentée par f *, basé sur la relation* R *sur le nœud-matérialisation X}*
variables : nœuds *A, B*;
corps :
A ← Arg1(X); B ← Arg2(X);
→ f(*nbentites* , *A.nboccs* , *B.nboccs* , *totaloccs* , $X.nboccs[R]$, $X.nboccs_{toutesrel}$, $A.d^{+}[R]$, $A.d^{+}_{toutesrel}$, $B.d^{-}[R]$, $B.d^{-}_{toutesrel}$, $totaloccs[R]$, $totaloccs_{toutesrel}$)

Cette dernière fonction n'a pas pour but d'être utilisée seule, mais sert pour le calcul global des valeurs des mesures.

Il est possible d'étendre le calcul des mesures d'association à des relations ayant plus de 2 arguments. Les relations n-aires peuvent varier sur *n+1* composantes (la relation elle-même et les *n* arguments), et font donc intervenir (si on étend ce qui est fait dans le cas *n=2*) $2^{n+1}+(n+1)+1$ occurrences. Nous ne détaillerons pas cela ici[114], mais on pourrait donc calculer facilement des mesures d'association à 21 arguments pour les relations ternaires, à 38 arguments pour les relations quaternaires, etc.

3.3.2.2 Mesure combinée

Le calcul possible de mesures d'association sur les nœuds dans la représentation complexe renforce l'intérêt de la mesure combinée (présentée dans la représentation simple).

Considérons l'exemple des bicollocations, représentées comme un arc entre deux nœuds matérialisant des relations binaires (voir la figure 33 p.153). Avec les opérateurs que nous avons présentés jusqu'ici, il est relativement facile de calculer, pour chaque bicollocation, une mesure d'association pour chaque extrémité (nœud-matérialisation de couple monolingue), ainsi qu'une autre mesure pour l'arc en lui-même (entre ces deux nœuds-matérialisation), avant de les combiner pour produire une mesure unique évaluant la capacité recherchée[115].

3.3.3 Union, intersection et différence

Dans la représentation complexe, les opérations d'union, d'intersection ou de différence entre graphes se passent de manière identique en ce qui concerne les nœuds et les arcs intraniveau et interniveau. Nous ne présenterons donc ici que ce qui diffère de la représentation simple, c'est-à-dire ce qui concerne les *arcs arguments* (qui doivent être, comme les autres objets du graphe, identifiés entre eux, pour permettre le mélange de l'information).

[114]Dans ce chapitre, nous nous concentrons sur les opérateurs de MuLLinG qui nous paraissent être génériques et essentiels (à la compréhension et à l'utilisation) pour extraire de l'information linguistique, plutôt que de lister tous les cas particuliers.

[115]Comme nous l'avons fait dans notre expérimentation (voir figure 17 p.97).

3.3.3.1 Graphes résultats

Nous allons étudier maintenant, pour chaque composante du graphe, la manière dont les opérations d'union, d'intersection ou de différence doivent se comporter.

Soit *H* le graphe résultat d'une telle opération entre *G1* et *G2*, graphes utilisant la représentation complexe :

- tous les objets correspondant aux nœuds, aux arcs intraniveau, ou aux arcs interniveau ont le même comportement qu'avec la représentation simple, de même pour les attributs correspondants.
- les arcs arguments sont des ensembles, on peut donc appliquer les opérations ensemblistes :
  - union ($H = G1 \cup G2$) : $H.C = G1.C \cup G2.C$,
  - intersection ($H = G1 \cap G2$) : $H.C = G1.C \cap G2.C$,
  - différence (complémentation) : ($H = G1 \setminus G2$) : $H.C = G1.C \setminus G2.C$,
- les arcs arguments copiés ou fusionnés conservent bien sûr leurs extrémités, donc les fonctions qui leur sont associées conservent leurs valeurs :
  - $H.\rho_i(v, j) = G1.\rho_i(v, j)$ si $v \in G1.V$, $G2.\rho_i(v, j)$ si $v \in G2.V$;

3.3.3.2 Gestion des arcs argument

Le traitement des arcs argument est similaire à celui des arcs interniveau : un arc argument n'ayant pas d'attribut (autre que le numéro d'ordre) et étant nécessairement unique étant donné une source et une cible, sa création dans le graphe résultant est la même, qu'il soit issu d'un arc argument présent dans un seul graphe, ou qu'il soit présent dans les deux.

AjouteNouvelArcArgument(arc *nouvc*, arc *c*, entier *Niv,* entier *numarg*) :
{ajoute dans le graphe courant nouvc*, arc argument (*numarg-*ième argument) entre la source et la cible correspondant aux extrémités de* c*, au* Niv-*ième niveau}*
AjouteArcArgument(*nouvc*, Niv, NouveauNœud(source(c)),
NouveauNœud(cible(c)), numarg).

3.3.3.3 Opérateur : union de graphes

Nous présentons d'abord l'opérateur d'*union*, le plus compliqué des trois. Pour chaque ensemble, l'opérateur calcule l'union des ensembles concernés, en étudiant d'abord par les nœuds, puis les arcs intraniveau, puis les arcs interniveau, puis (c'est ce qui diffère de la représentation simple) les arcs arguments. Selon les cas, on copie ou fusionne les objets.

Union

Paramètres d'entrée :

graphe *G1*,
graphe *G2*,
fonction($G1.V * G2.V \rightarrow bool$) *VIdent*,
fonction($G1.E * G2.E \rightarrow bool$) *EIdent*,
fonction($V * V * V$) *fus-v*,
fonction($E * E * E$) *fus-e*
fonction($V * V$) *cop-v*,
fonction($E * E$) *cop-e*

Renvoie :

graphe

Définition :

Réalise (dans le graphe courant) l'union entre les graphes G1 *et* G2*, l'identité entre nœuds, et entre arcs intraniveau, étant établie respectivement à l'aide de* Vident *et* Eldent *; la valeur des attributs des arcs et nœuds créés est établie à l'aide des fonctions* fus-v *et* fus-e.*(pour la partie fusionnée, à l'intersection* $G1 \cap G2$ *) et* cop-v *et* cop-e.*(pour la partie copiée, le reste)*

Variables : graphe *H*, entier *Niv*, nœud *vh*, arc *eh*,*fh*,*ch*, entier *numarg*.

Corps :

{Union des nœuds}
$\forall v1 \in G1.V$:
si ($\exists v2 \in G2.V \mid VIdent(v1, v2)$) :
$v1 \in G1.V_{Niv}$: H.AjouteFusionNœud(*vh*, v1,v2, Niv, fus-v)
sinon :
$v1 \in G1.V_{Niv}$: H.AjouteCopieNœud(*vh*, v1, Niv, cop-v)
$\forall v2 \in G2.V$:
si ($\neg \exists v1 \in G1.V \mid VIdent(v1, v2)$) :
$v2 \in G2.V_{Niv}$: H.AjouteCopieNœud(*vh*, v2, Niv, cop-v)
{Union des arcs intraniveau}
$\forall e1 \in G1.E$:
si ($\exists e2 \in G2.E \mid EIdent(e1, e2)$) :
$e1 \in G1.E_{Niv}$: H.AjouteFusionArcIntra(*eh*, e1,e2, Niv,fus-e)
sinon :
$e1 \in G1.E_{Niv}$: H.AjouteCopieArcIntra(*eh*, e1, Niv, cop-e)
$\forall e2 \in G2.E$:
si ($\neg \exists e1 \in G1.E \mid EIdent(e1, e2)$) :
$e2 \in G2.E_{Niv}$: H.AjouteCopieArcIntra(*eh*, e2, Niv, cop-e)

{Union des arcs interniveau}
$\forall f1 \in G1.F$:
  $f1 \in G1.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f1, Niv)
$\forall f2 \in G2.F$:
  si ($\neg \exists f1 \in G1.F \mid VIdent(source(f1), source(f2)) \wedge VIdent(cible(f1), cible(f2))$) :
    $f2 \in G2.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f2, Niv)
{Union des arcs argument}
$\forall c1 \in G1.C$:
  $c1 \in G1.C_{Niv,numarg}$: H.AjouteNouvelArcArgument(*ch*, c1, Niv, numarg)
$\forall c2 \in G2.C$:
  $c2 \in G2.C_{Niv,numarg}$: si ($\neg \exists c1 \in G1.C_{Niv,numarg} \mid VIdent(source(c1), source(c2)) \wedge VIdent(cible(c1), cible(c2))$) :
    H.AjouteNouvelArcArgument(*ch*, c2, Niv, numarg)
$\rightarrow$ H

Comme cela a été dit pour les opérateurs (union, intersection, différence) dans le cas de la représentation simple, il faut respecter les contraintes sur l'identification des nœuds et des arcs dans ces opérations, en particulier ici pour que les extrémités de leurs arcs arguments soient également correctement identifiées entre elles. En effet, pour qu'un nœud-matérialisation copié ou fusionné conserve un sens, il faut que tous les nœuds arguments de la relation soient copiés ou fusionnés et que tous les arcs arguments correspondants soient restitués.

3.3.3.4 Opérateur : intersection de graphes

L'opérateur d'intersection effectue la fusion des nœuds, puis celle des arcs intraniveau et interniveau, puis (c'est ce qui diffère de la représentation simple) les arcs arguments.

Intersection
Paramètres d'entrée :
  graphe *G1*,
  graphe *G2*,
  fonction($G1.V * G2.V \rightarrow bool$) *VIdent*,
  fonction($G1.E * G2.E \rightarrow bool$) *EIdent*,
  fonction($V * V * V$) *fus-v*,
  fonction($E * E * E$) *fus-e*
Renvoie :
  graphe

Définition :
  Réalise (dans le graphe courant) l'intersection entre les graphes G1 *et* G2*, l'identité entre nœuds, et entre arcs intraniveau, étant établie respectivement à l'aide de* Vident *et* EIdent *; la valeur des attributs des arcs et nœuds créés est établie à l'aide des fonctions* fus-v *et* fus-e*.*

Variables : graphe *H*, entier *Niv*, nœud *vh*, arc *eh*,*fh*,*ch*, entier *numarg*.
Corps :
$\forall v1 \in G1.V$:
si ($\exists v2 \in G2.V \mid VIdent(v1, v2)$) :
$v1 \in G1.V_{Niv}$: H.AjouteFusionNœud(*vh*, v1,v2, Niv, fus-v)
$\forall e1 \in G1.E$:
si ($\exists e2 \in G2.E \mid EIdent(e1, e2)$) :
$e1 \in G1.E_{Niv}$: H.AjouteFusionArcIntra(*eh*, e1,e2, Niv,fus-e)
$\forall f1 \in G1.F$:
si ($\exists f2 \in G2.F \mid VIdent(source(f1), source(f2)) \wedge VIdent(cible(f1), cible(f2))$) :
$f1 \in G1.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f1, Niv)
$\forall c1 \in G1.C$:
$c1 \in G1.C_{Niv, numarg}$: si ($\exists c2 \in G2.C_{Niv, numarg} \mid VIdent(source(c1), source(c2)) \wedge VIdent(cible(c1), cible(c2))$) :
H.AjouteNouvelArcArgument(*ch*, c1, Niv, numarg)
$\rightarrow$ H

3.3.3.5 Opérateur : différence de graphes

La différence de graphes conserve les objets qui sont présents dans le premier graphe mais pas dans le second. On effectue la copie des objets du premier graphe qui ne sont pas identifiés avec des objets du second (d'abord les nœuds, puis les arcs intraniveau, interniveau, et arguments).

Différence
Paramètres d'entrée :
graphe *G1*,
graphe *G2*,
fonction($G1.V * G2.V \rightarrow bool$) *VIdent*,
fonction($G1.E * G2.E \rightarrow bool$) *EIdent*,
fonction($V * V$) *cop-v*,
fonction($E * E$) *cop-e*
Renvoie :
graphe
Définition :
Réalise (dans le graphe courant) la différence entre les graphes G1 *et* G2*, l'identité entre nœuds, et entre arcs intraniveau, étant établie respectivement à l'aide de* VIdent*,* Eident *; la valeur des attributs des arcs et nœuds créés est établie à l'aide des fonctions* cop-v *et* cop-e*.*

Variables : graphe *H*, entier *Niv*, nœud *vh*, arc *eh*,*fh*,*ch*, entier *numarg*.
Corps :
$\forall v1 \in G1.V$:
si ($\neg \exists v2 \in G2.V \mid VIdent(v1, v2)$) :
$v1 \in G1.V_{Niv}$: H.AjouteCopieNœud(*vh*, v1, Niv, cop-v)
$\forall e1 \in G1.E$:
si ($\neg \exists e2 \in G2.E \mid EIdent(e1, e2)$) :
$e1 \in G1.E_{Niv}$: H.AjouteCopieArcIntra(*eh*, e1, Niv, cop-e)
$\forall f1 \in G1.F$:
si ($\neg \exists f2 \in G2.F \mid VIdent(source(f1), source(f2)) \wedge VIdent(cible(f1), cible(f2))$) :
$f1 \in G1.F_{Niv}$: H.AjouteNouvelArcInter(*fh*, f1, Niv)
$\forall c1 \in G1.C$:
$c1 \in G1.C_{Niv, numarg}$: si ($\neg \exists c2 \in G2.C_{Niv, numarg} \mid VIdent(source(c1), source(c2)) \wedge VIdent(cible(c1), cible(c2))$) :
H.AjouteNouvelArcArgument(*ch*, c1, Niv, numarg)

$\rightarrow$ H

Conclusion

Nous avons présenté dans ce chapitre MuLLinG, notre modèle de graphe pour le traitement d'ensembles de données linguistiques. Les graphes de ce modèle sont multiniveau, chaque niveau supérieur représentant une vue moins détaillée (plus résumée) de l'information que ne le fait le niveau inférieur. Le modèle MuLLinG répond aux contraintes fixées (haut niveau, généricité tant au niveau des problèmes qu'à celui des données). En effet, les opérations ont été définies indépendamment des problèmes précis d'extraction, pour être utiles au plus grand spectre possible de problèmes linguistiques.

Nous avons présenté les opérateurs en nous concentrant sur leur fonctionnement propre, et en nous détachant de tout ce qui concerne l'implémentation (calcul des classes d'équivalence, des nouveaux attributs, fonctions d'identité, etc.), considéré comme un ensemble de paramètres. Il reste maintenant à montrer que notre modèle est utile pour différents problèmes linguistiques, c'est-à-dire à l'implémenter et à l'expérimenter.

Partie III - Implémentation et applications

Chapitre VII - Implémentation

Après avoir effectué diverses expérimentations d'extraction de connaissances lexicales (collocations), nous avons choisi de consacrer notre travail à la spécification d'outils pour l'extraction, génériques et simples d'utilisation. Nous avons ensuite défini un modèle de graphes de connaissances linguistiques, conforme aux contraintes que nous avions préalablement fixées.

Afin de prouver l'utilité de cette spécification, nous l'avons mise en œuvre sous la forme d'une bibliothèque implémentant les opérateurs décrits, et permettant ainsi la gestion de graphes linguistiques multiniveau. Cette bibliothèque nous permettra de vérifier l'intérêt de notre modèle, dans diverses applications.

Nous justifierons le choix, guidé par les bibliothèques existantes, que nous avons effectué pour le langage de programmation. Nous étudierons également les différents langages de stockage de graphes, et expliquerons ce qui nous a conduit à préférer GraphML. Enfin, nous décrirons les principaux problèmes posés par la programmation d'une bibliothèque implémentant le modèle défini.

1 Langage de programmation

Afin de manipuler des graphes, des nœuds, des arcs, voire des structures encore plus compliquées, nous utiliserons un langage permettant la programmation par objets. Nous souhaitions en premier lieu que notre implémentation puisse se reposer sur une bonne *gestion de la structure interne* des graphes, en utilisant des bibliothèques déjà existantes, afin de nous concentrer uniquement sur les modifications de la structure (et pas sur la manière dont celles-ci sont réalisées).

Les trois principales bibliothèques permettant la gestion de structures de graphes sont *LEDA* (Melhorn & Näher, 1999) et *Boost Graph Library*[116], toutes deux écrites en *C++,* et *Ocamlgraph* (Conchon et al., 2005) écrite en *Objective Caml*. Nos principaux critères de choix furent :

- la *licence* de la bibliothèque : nous souhaitions réaliser un outil utilisable facilement, qui se base sur un outil disponible gratuitement ; cela exclut LEDA, bibliothèque propriétaire (relativement chère)[117] ;
- la *couverture* de la bibliothèque : nous souhaitions pouvoir programmer toutes les opérations sur les graphes correspondant à la spécification que nous avons faite, sans être trop bloqué par les contraintes de la bibliothèque ; Ocamlgraph nous semble trop limité.

[116]http://www.boost.org/

[117]L'éditeur de LEDA propose désormais une édition gratuite (mais très limitée) du logiciel. Cette version (lancée en janvier 2008) n'existait pas lors de notre choix et n'a donc pas pu entrer en ligne de compte dans celui-ci.

Nous avons donc choisi d'utiliser Boost Graph Library (*BGL*), qui a l'avantage de faire partie du projet de bibliothèques C++ « open-source » *Boost* fédérant une large communauté. Nous avons d'ailleurs utilisé également d'autres bibliothèques de Boost dans notre implémentation. Un autre avantage de BGL est qu'elle permet la lecture et l'écriture de graphes aux formats répandus DOT et GraphML.

Notre propre bibliothèque a été réalisée en C++, un choix cohérent avec la programmation par objets, ce qui facilite bien sûr beaucoup notre utilisation de Boost.

2 Choix d'un langage de stockage

Un outil pour la gestion de graphes (linguistiques) doit bien sûr permettre leur sauvegarde et leur chargement.

Le traitement de données sous la forme de graphes étant un problème très fréquent, de nombreux langages existent pour leur sauvegarde.

Nous souhaitons que les fichiers représentant les graphes soient facilement interprétables par un humain. Cela exclut donc les formats binaires, et nous restreint aux formats de fichier basés sur du texte « brut ».

Nous allons étudier ici les plus répandus d'entre eux. La description que nous en faisons est partielle ; en effet, la plupart de ces langages peuvent supporter des structures complexes (avec hypernœuds, hyperarcs, ports) qui ne nous intéressent pas, puisque nous avons choisi d'avoir une structure de graphe simple (nœuds, arcs, et attributs).

2.1 DOT

Graphviz est un ensemble de programmes destiné au dessin de graphes (Gansner & North, 2000). Chacun de ces programmes correspond à un algorithme différent de dessin, le plus connu étant *dot*, produisant des dessins de graphes organisés en hiérarchie (Gansner et al., 2006).

Graphviz a son propre langage pour le stockage des graphes à dessiner : *DOT* (Graphviz). Celui-ci permet aux nœuds et aux arcs de porter des attributs concernant leur apparence (forme, couleur, etc.), qui seront interprétés par le programme de dessin pour obtenir la présentation voulue.

2.1.1 Déclaration du graphe

Dans DOT, un graphe se déclare en spécifiant son type :

- *graph* : non orienté

```
graph nomdugraphe {
}
```

- *digraph* : orienté

```
digraph nomdugraphe {
}
```

2.1.2 Déclaration des nœuds et arcs

Les nœuds et les arcs sont déclarés à l'intérieur des accolades qui correspondent au graphe auquel ils appartiennent.
Un nœud se déclare simplement en donnant le nom de son identifiant :

```
digraph exemple {
      a,b;
}
```

Un arc se déclare en donnant les identifiants de sa source et de sa cible (il n'est pas nécessaire que celles-ci aient été préalablement déclarées), séparées par un opérateur :

- `==` si le graphe n'est pas orienté :

```
graph exemple {
      a==b;
}
```

- `->` si le graphe est orienté.

```
digraph exemple {
 a->b;
}
```

2.1.3 Attributs

Le langage DOT définit une liste finie d'attributs pour les nœuds, les arcs, et le graphe, relatifs à la forme du graphe ou de ses composantes (taille, couleur, forme, label, etc.). Ces attributs sont destinés à être interprétés par les programmes de dessin de graphes. On ne peut pas en introduire d'autres[118].
L'attribut d'un graphe se déclare entre les accolades lui correspondant :

```
digraph exemple {
      bgcolor=gray;
      a->b;
}
```

Pour les arcs et les nœuds, les attributs doivent être mis entre crochets après la déclaration de l'objet auxquels il se rapportent :

```
digraph exemple {
      bgcolor=gray;
      a [shape=house];
      a->b [label="Truc"];
}
```

[118]Les attributs n'étant pas dans la liste spécifiée sont invalides. Si un fichier DOT contient des attributs invalides, ceux-ci sont simplement ignorés par les programmes de dessin.

La figure 37 présente le dessin produit par *dot* pour le code précédent. On peut notamment remarquer que la forme de *b* n'ayant pas été spécifiée, il adopte par défaut celle d'une ellipse.

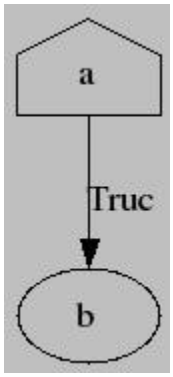


Figure 37: Graphe produit avec dot à partir de l'exemple d'un fichier écrit en langage DOT

Si le langage DOT a l'avantage d'être assez simple et d'être supporté par des logiciels répandus, l'inconvénient vient du fait qu'il est orienté vers le dessin de graphes, alors que nous nous intéressons à la manipulation et à la modification des graphes. En particulier, le fait de ne pas pouvoir introduire nos propres attributs nous semble très contraignant alors que nous voulons poser le moins de contraintes possibles sur les données.

2.2 GML

Graph Modelling Language (GML) est un format de graphes défini par Himsolt (1997), créé dans un but de portabilité (proposer un standard pour la description de graphes) et de souplesse.

Si la syntaxe de DOT définissait le contenu structures entre accolades, celle de GML le fait entre crochets.

2.2.1 Déclaration du graphe

La déclaration d'un graphe ne nécessite pas de lui donner un identifiant.

```
graph [
]
```

On utilise un attribut pour déclarer si le graphe est orienté ou non (voir plus bas).

2.2.2 Déclaration des nœuds et arcs

La déclaration d'un nœud nécessite un identifiant.

```
graph [
      node [
            id 2
      ]
]
```

Pour déclarer un arc, orienté ou non, il faut indiquer les identifiants de sa source et de sa cible. Si le graphe n'est pas orienté, le choix de la « source » et de la « cible » peut se faire de manière arbitraire, cela n'ayant pas de conséquence sur l'interprétation de l'arc.

```
graph [
      node [id 2]
      node [id 4]
      edge [
            source 2
            target 4
      ]
]
```

2.2.3 Attributs

La description d'un graphe en GML met en évidence la structure des données. Pour déclarer un attribut, on le place à l'intérieur de la structure correspondant à l'objet auquel il se rapporte. Il s'agit de « structures de traits » typées.

Par exemple, on utilise l'attribut de graphe *directed* pour indiquer si le graphe en question est orienté ou non (il est non orienté par défaut) :

```
graph [
      directed 1
      node [id 2]
]
```

De la même manière, on ajouter des attributs aux nœuds et aux arcs :

```
graph [
      directed 1
      node [
            id 2
            label "II"
      ]
      node [
            id 4
            label "IV"
      ]
      edge [
            source 2
            target 4
            label "2-4"
      ]
]
```

La valeur des attributs peut elle-même être une structure (entre crochets), c'est notamment le cas de l'attribut *graphics* des nœuds et des arcs. GML permet d'utiliser n'importe quel attribut.

L'exemple donné permet de représenter le graphe présenté dans la figure 38.

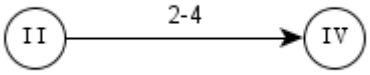


Figure 38: Graphe correspondant à l'exemple de fichier GML

2.3 XGMML

Dans les années 1990, le développement des espaces de stockage a rendu beaucoup moins critique la question de la taille des fichiers stockés. Cela a permis l'apparition et l'essor de langages plus verbeux. C'est le cas de XML (Bray et al., 1998), qui s'est rapidement imposé, grâce à sa structure d'arbre, comme un standard pour la description de données structurées, entraînant la réalisation de nombreux outils pour la manipulation de structures XML.

Cela a conduit à l'apparition de nouveaux langages de description de graphes, basés sur XML. Le premier fut XGMML (*eXtensible Graph Markup and Modeling Language*), simple transposition de la structure de GML en XML (Punin et al., 1999, 2001).[119]

2.3.1 En-tête

Le passage à XML implique des ajouts, afin de se conformer à cette syntaxe. En premier lieu, un fichier écrit dans un langage basé sur XML doit toujours se déclarer comme tel :

```
<?xml version="1.0"?>
<!DOCTYPE graph PUBLIC "-//John Punin//DTD graph description//EN"
"http://www.cs.rpi.edu/~puninj/XGMML/xgmml.dtd">
```

2.3.2 Déclaration du graphe

Pour déclarer un graphe XGMML, on doit utiliser un élément <*graph*> ; celui-ci ne peut contenir, hors attributs[120], que des éléments <*node*> ou <*edge*>.

```
<?xml version="1.0"?>
<!DOCTYPE graph PUBLIC "-//John Punin//DTD graph description//EN"
"http://www.cs.rpi.edu/~puninj/XGMML/xgmml.dtd">
<graph>
</graph>
```

Comme en GML, on utilise un attribut pour déclarer si le graphe est orienté ou pas (voir plus bas).

[119]XGMML donnera lui-même naissance à LOGML, un langage de description des *logs* de serveurs web (Punin et al., 2001).

[120]Au sens XML du terme, c'est à dire une chaîne `nomattribut="valeurattribut"` dans la balise ouvrante du nœud.

2.3.3 Déclaration des nœuds et arcs

On déclare un nœud du graphe comme un élément <*node*> de la structure XML. L'identification d'un nœud se fait avec l'attribut *id* de l'élément <*node*> correspondant[121].

Un arc du graphe se déclare comme un élément <*edge*> de la structure XML ; celui-ci doit obligatoirement comporter les attributs *source* et *cible* pour identifier les extrémités de l'arc (là encore, si le graphe n'est pas orienté, le choix de la « source » et de la « cible » peut se faire de façon arbitraire, cela n'ayant pas de conséquence sur l'interprétation de l'arc).

```
<?xml version="1.0"?>
<!DOCTYPE graph PUBLIC "-//John Punin//DTD graph description//EN"
"http://www.cs.rpi.edu/~puninj/XGMML/xgmml.dtd">
<graph>
      <node id="2"/>
      <node id="4"/>
      <edge source="2" target="4"/>
</graph>
```

2.3.4 Attributs

XGMML permet de mettre des attributs sur le graphe, les nœuds, ou les arcs. Comme GML, il peut accepter n'importe quel attribut.

Selon les cas, l'attribut d'un objet est représenté comme :

- un attribut de l'élément XML correspondant (à l'intérieur de la balise ouvrante) : cas des attributs standard (définis dans la spécification) :

```
<?xml version="1.0"?>
<!DOCTYPE graph PUBLIC "-//John Punin//DTD graph description//EN"
"http://www.cs.rpi.edu/~puninj/XGMML/xgmml.dtd">
<graph directed="1">
 <node id="2" label="II"/>
 <node id="4" label="IV"/>
 <edge source="2" target="4" label="2-4"/>
</graph>
```

- un fils de l'élément XML correspondant :
  - attribut *graphics* : on insère un nœud <*graphics*>, comme fils de l'élément XML ;
  - autres attributs : on insère un nœud <*att*> comme fils de l'élément XML, en indiquant le type et la valeur de l'attribut :

```
<node id="3" label="III"/>
      <att name="poids" type="int" value="12"/>
      <att name="vote" type="string" value="oui"/>
</node>
```

[121] L'attribut *id* n'est pas obligatoire d'après la spécification de XGMML, mais il l'est en pratique pour qu'on puisse définir ce nœud comme une extrémité d'un arc.

2.4 GXL

Les langages présentés jusqu'ici visaient à représenter des graphes généraux, sans supposition sur leur utilisation. Au contraire, GXL (*Graph eXchange Language*) a été développé dans le but de permettre l'échange de données sous forme de graphes, dans les processus (Holt et al., 2000).

2.4.1 En-tête

Un fichier GXL doit tout d'abord déclarer qu'il est basé sur XML et qu'il est en GXL.

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
```

2.4.2 Déclaration du graphe

Pour déclarer un graphe GXL, on doit utiliser un élément <*gxl*> contenant lui-même un élément <*graph*>. Par défaut, le graphe est orienté.

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl>
      <graph id="exemple">
      </graph>
</gxl>
```

2.4.3 Déclaration des nœuds et arcs

Un nœud du graphe représenté se déclare à l'aide d'un élément <*node*> de la structure XML, et doit nécessairement être identifié (attribut *id*). Un arc se déclare à l'aide d'un élément <*edge*>, et on doit indiquer obligatoirement sa source et sa cible (attributs *from* et *to*).

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl>
      <graph id="exemple">
            <node id="p" />
            <node id="q" />
            <edge id="c" to="q" from="p" />
      </graph>
</gxl>
```

La déclaration des nœuds et des arcs est donc assez similaire (sur le principe) à celle de XGMML. Outre les nœuds et les arcs, GXL permet également de représenter des relations n-aires, avec l'élément <*rel*>.

2.4.4 Attributs

Comme pour XGMML, il y a plusieurs façons de représenter un attribut (de graphe, de nœud, d'arc) :

- attribut de l'élément XML correspondant (à l'intérieur de la balise) : cas de quelques attributs de base présentés dans la spécification (ils sont cependant beaucoup moins nombreux que dans la spécification de XGMML) :
- élément *<attr>* fils de l'élément XML correspondant, devant obligatoirement avoir un attribut *name* (nom de l'attribut), et un fils indiquant la valeur.

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl>
      <graph id="exemple" edgemode="directed">
            <node id="p">
                  <attr name="file">
                        <string> main.c</string>
                  </attr>
            </node>
            <node id="q">
                  <attr name="file">
                        <string> test.c</string>
                  </attr>
            </node>

            <edge id="c" to="q" from="p">
                  <attr name=" line">
                        <int> 42</int>
                  </attr>
            </edge>
      </graph>
</gxl>
```

2.5 GraphML

GraphML est un autre langage de description de graphes basé sur XML, créé dans le but d'être à la fois simple, général[122], extensible et robuste (Brandes et al., 2001). Inspiré par les précédents modèles, il a été conçu pour devenir le standard de la représentation des graphes.

2.5.1 En-tête

Comme pour les autres formats basés sur XML, un fichier GraphML doit commencer par une en-tête déclarant le type du contenu. Alors que les formats XML précédents utilisaient les *DTD* (déclaration de type de document), GraphML est spécifié par un *schéma XML* (un format plus récent et plus puissant, qui a l'avantage d'être lui-même en XML).

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd"/>
```

[122]GraphML peut modéliser facilement des hyperarcs, des graphes hiérarchiques (avec hypernœuds), ainsi que des « ports » sur les nœuds (les ports d'un nœud sont des endroits de connexion des arcs sur le nœud).

2.5.2 Déclaration du graphe

La déclaration du graphe principal se fait à l'aide d'un élément XML <*graph*>, à l'intérieur de l'élément <*graphml*>. Par défaut, un graphe est orienté.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
      <graph>
      </graph>
</graphml>
```

2.5.3 Déclaration des nœuds et des arcs

Les nœuds et les arcs se déclarent respectivement avec des éléments XML <*node*> et <*edge*>. Un nœud doit nécessairement être identifié (attribut *id* de l'élément <*node*>) ; un arc doit obligatoirement indiquer sa source et sa cible (attribut *source* et *target* de l'élément <*edge*>).

```
      <graph>
            <node id="n0"/>
            <node id="n1"/>
            <edge source="n1" target="n0"/>
      </graph>
```

Comme GXL, GraphML dispose d'un élément spécial pour représenter les relations n-aires, sous la forme d'hyperarcs : <*hyperedge*>.

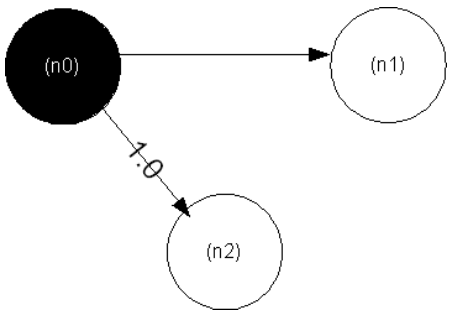


Figure 39: Exemple de graphe orienté à attributs

2.5.4 Attributs

La principale différence entre GraphML et les autres formats vient de la gestion des attributs.

En effet, dans les autres formats, on doit donner à chaque fois le nom de l'attribut souhaité avant de pouvoir donner sa valeur. Dans GraphML, la déclaration des attributs est séparée de leur utilisation :

- *déclaration dans l'en-tête* du graphe à l'aide de nœuds <*key*> : pour chaque attribut, on indique respectivement son identifiant, le type d'objet sur lequel il porte, son nom, et son type; avec les attributs XML *id*, *for*, *attr.name*, et *attr.type* [123];
  - on peut fixer une valeur par défaut à un attribut en ajoutant un élément <*default*> comme fils de l'élément <*key*> concerné.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
      <key id="d0" for="node" attr.name="color" attr.type=
"string">
            <default>white</default>
      </key>
      <key id="d1" for="edge" attr.name="weight" attr.type=
"double"/>
```

- *utilisation au sein du graphe* à l'aide des éléments <*data*> fils de l'élément correspondant :
  - les différents attributs de l'objet sont différenciés grâce à leurs identifiants (attribut *key* de l'élément en question).[124]

```
      <graph id="G" edgedefault="directed">
            <node id="n0">
                  <data key="d0">black</data>
            </node>
            <node id="n1"/>
            <node id="n2"/>
            <edge id="e0" source="n0" target="n2">
                  <data key="d1">1.0</data>
            </edge>
            <edge id="e1" source="n0" target="n1/>
      </graph>
</graphml>
```

L'exemple donné permet de réaliser la figure 39.

La séparation entre la déclaration des attributs et leur utilisation permet de changer facilement le nom d'un attribut dans le graphe : il suffit de modifier son nom à la déclaration, alors qu'on aurait dû, avec les autres modèles, modifier l'attribut correspondant de tous les objets concernés.

[123]D'après la spécification, seul *id* est obligatoire ; *attr.name* et *attr.type* donnent des informations destinées à l'utilisation du graphe dans un programme. S'il n'est pas interdit explicitement par la spécification de donner la même valeur pour *attr.name* (donc le même nom) à plusieurs attributs, il faut bien sûr l'éviter.

[124]Si un objet ne donne pas de valeur pour un attribut (pas d'élément XML <*data*> fils de l'élément correspondant à l'objet), il prend pour valeur celle éventuellement définie par défaut lors de la déclaration de l'attribut .

2.6 Choix effectué : GraphML

Nous avons décrit les langages de description de graphes les plus répandus. Lequel choisir ?

Puisque les graphes de notre modèle évitent les structures plus compliquées (hypernœuds, hyperarcs, ports), ils pourraient facilement être décrits dans chacun de ces langages. De plus, l'espace de stockage n'étant plus un problème, nous pouvons tout à fait utiliser des langages basés sur XML, comportant des données redondantes.

Les critères qui ont guidé notre choix sont :

- la possibilité de définir nos propres attributs, ce qui exclut DOT, langage fait pour le dessin des graphes et dont les seuls attributs sont définis pour cette utilisation ;
- l'existence de bibliothèques pour lire et écrire facilement des données dans ce format : *Boost Graph Library* permet de lire des graphes aux formats DOT et GraphML ;
- le support par les éditeurs de graphes : *yEd Graph Editor*[125] supporte GML, XGMML, et GraphML, et a pris dans ses dernières versions GraphML comme langage de stockage par défaut ; *JUNG*[126] a également introduit la lecture de fichiers GraphML.

GraphML, qui semble être arrivé à ce pourquoi il avait été créé (être une norme *de facto* de la description de graphes), nous paraît donc le meilleur langage à utiliser pour le stockage des graphes, dans l'implémentation de notre modèle.

3 Points d'implémentation

Nous avons donc réalisé une bibliothèque C++, basée sur Boost, permettant de manipuler des graphes linguistiques multiniveau avec les opérateurs que nous avons définis, et de lire et écrire ces graphes stockés au format GraphML.

Dans la forme actuelle de notre système, on peut effectuer des opérations d'extraction en écrivant un programme C++ faisant appel aux fonctions que nous avons réalisées. Il faut, dans le futur, permettre de faire ces opérations à travers une surcouche de plus haut niveau, afin de rendre l'outil utilisable par des linguistes aux connaissances limitées en programmation.

3.1 Graphe

3.1.1 Format de stockage du graphe

Boost Graph Library (BGL) permet de stocker en mémoire les graphes sous différentes formes :

- *listes d'adjacence* : liste des nœuds, on associe à chacun d'entre eux la liste des arcs dont il est source ;

[125] http://www.yworks.com/en/products_yed_about.html

[126] http://jung.sourceforge.net/

- *matrice d'adjacence* : tableau à deux dimensions (chacune contenant la liste des nœuds du graphe) ; chaque case du tableau contient les arcs entre le nœud correspondant à la ligne et le nœud correspondant à la colonne ;
- *Compressed Sparse Row* (CSR, lignes creuses comprimées) : un format pour les matrices creuses, destiné aux graphes peu denses dont on ne doit pas modifier le contenu.

La densité des graphes MuLLinG étant faible, la matrice d'adjacence ne peut convenir. On ne peut pas non plus utiliser le format CSR, puisque nous modifions le contenu du graphe. L'implémentation doit donc utiliser la structure en listes d'adjacence.

3.1.2 Attributs

Dans BGL, les attributs sont les *propriétés* du graphe, des nœuds, et des arcs. Lorsqu'on déclare un graphe, on doit indiquer la liste des propriétés sur les graphes, la liste de celles sur les nœuds, et la liste de celles sur les arcs.

Nous avons utilisé les propriétés dans deux cas :

- les attributs correspondant à la nature de l'objet, et permettant de le distinguer des autres (par exemple, arcs interniveau vs. intraniveau) ;
- les attributs « normaux » (ceux qui peuvent être spécifiés par l'utilisateur).

3.1.2.1 Représentation simple

Afin de rester dans le cadre d'un graphe simple, avec un seul ensemble de nœuds et un seul ensemble d'arcs, les séparations entre les différents ensembles de nœuds ou d'arcs ont été représentées à l'aide d'attributs particuliers :

- l'attribut *level* permet d'indiquer le niveau des nœuds et des arcs (pour les arcs interniveau, c'est le niveau de la source de l'arc qui est indiqué) ;
- l'attribut *typeglobal* permet de distinguer les arcs intraniveau (valeur : *same*) et interniveau (valeur : *uplevel*) ;
- l'attribut *type* permet à l'utilisateur d'identifier les « types » des nœuds, des arcs intraniveau ou interniveau[127].

3.1.2.2 Représentation complexe

Comme nous l'avons spécifié, les nœuds du graphe matérialisant les relations sont des nœuds comme les autres. Pour les différencier, on peut utiliser simplement un attribut *type* différent.

Le vrai changement est dû à l'introduction d'arcs argument, considérés comme ayant un nouveau type. C'est l'attribut *typeglobal*, celui qui permet déjà la distinction entre arcs intraniveau et interniveau, qui prend une valeur indiquant qu'il s'agit un arc argument et comportant son numéro d'argument (valeurs : *arg1*, *arg2*, etc.).

[127]Un utilisateur peut par exemple définir des types *occurrence*, *classedequivalence*, *materialisation*, etc.

3.2 Opérations

La présentation que nous avons faite des opérateurs du modèle comporte un certain nombre de points où le problème de l'implémentation de l'algorithme a été écartée. Il nous faut maintenant faire des choix pour répondre à ces questions.

Les opérations utilisent l'attribut *typeglobal*, pour savoir quels objets il faut étudier (par exemple, seulement les arcs intraniveau).

3.2.1 Accès et modification des objets

Nous reprenons les fonctions de base définies par BGL permettant d'accéder aux nœuds et aux arcs, d'en ajouter ou d'en supprimer.

3.2.2 Parcours et complexité

BGL permet de définir, sur la liste des nœuds et sur la liste des arcs, des « itérateurs », des pointeurs sur le début et la fin de la liste, et une fonction d'itération. Nous utilisons donc cela pour tout algorithme où l'on doit traiter tout élément d'un certain type ($\forall \ldots$).

Lorsque nous écrivons une fonction, implémentant une opération de MuLLinG, qui nécessite plusieurs parcours du graphe imbriqués, on effectue généralement un *précalcul* des informations nécessaires dans la seconde boucle. Il suffit alors, à chaque passage dans la deuxième boucle, de lire les informations précalculées, ce qui évite de parcourir à nouveau le graphe.

De telles opérations, comme celles d'émergence, nécessitent donc un temps de calcul *linéaire*.

Au contraire, dans les opérations nécessitant l'identification des nœuds et des arcs entre deux graphes (union, intersection, différence), on doit, pour chaque objet du premier graphe, reparcourir le second graphe pour trouver son éventuel identique : le temps de calcul de ces opérations est quadratique (si les deux graphes sont de tailles équivalentes).

3.2.3 Classes d'équivalence

Dans notre modèle, la classe d'équivalence d'un nœud ou d'un arc est utilisée comme paramètre de l'émergence, on suppose donc qu'elle a déjà été calculée.

Notre implémentation doit donc contenir des fonctions de calcul de la classe d'équivalence de chaque objet.

Nous prenons, comme paramètres de l'équivalence entre deux nœuds ou deux arcs A et B, des listes d'attributs :

- *ListEquiv*, une liste de couples <attribut, fonction d'équivalence> (la fonction d'équivalence pouvant être éventuellement décrite à l'aide d'un partitionnement) ;
- *ListÉgal*, une liste d'attributs.

$$A \equiv B \Leftrightarrow$$
$$[\forall \langle atteq_i , f_i \rangle \in ListEquiv : f_i(A.atteq_i , B.atteq_i)] \wedge [\forall atteg_i \in ListEgal : A.atteg_i = B.atteg_i]$$

Prenons l'exemple de l'équivalence de nœuds :

- tout d'abord, pour chaque attribut sur lequel l'équivalence porte, la fonction parcourt les nœuds pour réaliser une partition des valeurs possibles, et donne un identifiant à chacune de ces partitions ;
- ensuite, pour chaque nœud :
  - on génère un identifiant de sa classe d'équivalence, à partir des identifiants des classes correspondant à chacun des attributs évalués ;
  - cet identifiant est placé sur le nœud, comme valeur de l'attribut `classe`.

L'équivalence d'arcs repose sur le même principe.

Il y a besoin, lorsqu'on calcule l'émergence, de savoir quel nœud représente la classe considérée (fonctions *Classe_V*, *Classe_E* et *Classe_VMater* du modèle). Cela se fait en maintenant une table de hachage (associant à l'identifiant de la classe, et aux extrémités dans le cas des arcs, l'objet matérialisant cette classe), qu'on lit pour savoir s'il y a déjà un objet matérialisant cette classe (s'il n'y en a pas, on le crée).

3.2.4 Filtre

Dans notre implémentation, le filtre des applications de fonctions utilise des listes[128] :

- liste d'associations entre un attribut et la liste des valeurs acceptables pour cet attribut ;
- liste d'associations entre un attribut et la liste des valeurs non acceptables pour cet attribut (« stoplist »).

Dans les opérations comme l'émergence, le filtre est évalué à travers les valeurs d'attributs précisés. Ces valeurs peuvent être fixées par une autre fonction, ou en utilisant l'opération d'application.

3.2.5 Identité

La représentation de la fonction d'identité (sur les arcs intraniveau, interniveau, ou sur les nœuds) se fait à l'aide d'une liste d'attributs. Pour que deux objets soient considérés comme identiques, il faut que, pour chacun des attributs de la liste, les valeurs soient égales.

Comme dans le cas des classes d'équivalence, il faut précalculer les valeurs de ces arguments, ainsi que les associations entre objets aux valeurs identiques, pour éviter de parcourir un graphe à chaque nouvel objet traité.

Il faut également maintenir une table de hachage associant un nœud d'un des graphes de départ au nœud qui le représente, après copie ou fusion, dans le graphe résultat ; cela correspond à la fonction *Nouveaunœud* du modèle.

3.2.6 Copie et fusion sur les attributs

Les opérations d'union, d'intersection et de fusion sont basées sur la copie et la fusion de nœuds et d'arcs.

Pour la copie, nous procédons simplement à la création d'un nouveau nœud, reprenant tous les attributs du nœud copié.

[128]Dans la version actuelle de l'implémentation, on ne peut, pour des raisons pratiques, spécifier des ensembles de valeurs infinis.

Pour la fusion, nous avons choisi de simplifier pour l'instant le problème, en ne conservant que les attributs pour lesquels il n'y a pas de conflit de valeur (ceux ayant une valeur identique pour les deux objets, et ceux n'étant définis que pour un seul des deux).

3.2.7 Commentaire

Quelques fonctions sont définies dans notre bibliothèque de façon plus contraignante que l'opérateur correspondant ne l'est dans notre modèle. En effet, notre but étant avant tout d'obtenir un prototype fonctionnel pour le tester sur différentes applications, nous avons fait certaines hypothèses sur les paramètres des opérateurs pour simplifier l'écriture du code. Une version plus avancée de la bibliothèque est prévue, elle s'affranchira de ces contraintes.

Toutefois, si notre bibliothèque est encore à l'état de prototype, elle permet non seulement de gérer les processus habituels d'extraction de connaissances, mais aussi des cas plus complexes.

Conclusion

Nous avons réalisé un prototype de bibliothèque en C++, basée sur la bibliothèque *Boost Graph Library*, implémentant le modèle MuLLinG que nous avons défini précédemment, en faisant des choix pour les principaux problèmes de mise en œuvre qui n'étaient pas résolus dans le modèle.

Cette bibliothèque lit et écrit les graphes au format GraphML, que nous avons choisi parmi les formats existants parce qu'il semble s'imposer comme un standard, tant au niveau des logiciels d'édition de graphes, qu'à celui des bibliothèques pour la manipulation (y compris la *Boost Graph Library*).

Cette implémentation doit nous permettre de tester l'utilité de notre modèle sur différents problèmes, comme l'extraction de connaissances linguistiques, ou des problèmes de traitement des langues plus généraux.

Chapitre VIII - Extraction de collocations par manipulation de graphes

Après avoir présenté une spécification de représentation des données linguistiques sous la forme de graphes (multiniveau), et les opérateurs de manipulation de tels graphes, nous allons chercher à en démontrer l'utilité. Nous le ferons tout d'abord sur l'extraction de collocations et de bicollocations à partir de corpus, car, comme nous allons le voir, cette tâche se prête tout à fait à une telle représentation.

Comment voir l'extraction de collocations comme une manipulation de graphes ? Nous allons d'abord présenter un cas simple, l'extraction monolingue (un arc entre deux termes suffit à représenter une collocation), et un cas relativement plus complexe (la bicollocation faisant intervenir 4 termes au total, cela implique une structure plus compliquée à représenter et à extraire). Nous verrons ensuite comment nous avons mis en œuvre cette solution, et les résultats que nous avons obtenus.

1 Représentation des données

Dans les expérimentations que nous réalisons, on suppose qu'on a les outils nécessaires pour produire les liens monolingues (par exemple avec un analyseur syntaxique) ou bilingues (à l'aide d'un dictionnaire).

1.1 Extraction monolingue

Dans le cas de l'extraction monolingue, il faut représenter les relations entre termes présentes à l'intérieur d'un corpus. Cela correspond, de manière assez simple, à un graphe dont les nœuds sont les termes et les arcs les relations entre ces termes. Le type de connaissances à extraire (collocations) prenant la forme d'une relation entre deux termes, un candidat à produire aura la forme d'un arc.

1.2 Extraction bilingue

1.2.1 Corpus

La situation est plus complexe dans le cas bilingue.

On a alors, à la base, deux types de nœuds (on peut, par commodité, utiliser un attribut *langue* sur les nœuds pour les distinguer facilement):

- *les nœuds « français »* (issus de la partie française des corpus utilisés) ;
- *les nœuds « anglais »* (issus de la partie anglaise des corpus utilisés).

Il y a alors 3 types d'arcs :

- les liens monolingues français, représentant des relations entre termes du corpus français ;
- les liens monolingues anglais, représentant des relations entre termes du corpus anglais ;
- les *liens de traduction* entre termes français et anglais.

Ces différentes relations correspondent aux branches de l'extraction bilingue décrite lors de nos précédentes extractions. Il faut en effet séparer les deux études monolingues (dans les deux langues) de la collocabilité, et l'étude bilingue de l'associabilité ; chacune correspond à la manipulation d'un des types d'arcs précédents.

1.2.2 Bicollocations

Une bicollocation étant un couple de collocations monolingues, on peut la voir comme un arc entre deux objets représentant des collocations, en l'occurrence, deux nœuds matérialisant un lien entre deux termes de la même langue.

Le processus d'extraction des bicollocations doit comprendre des opérations faisant passer de liens simples (monolingues ou bilingues) à cette structure plus complexe.

2 Propagation

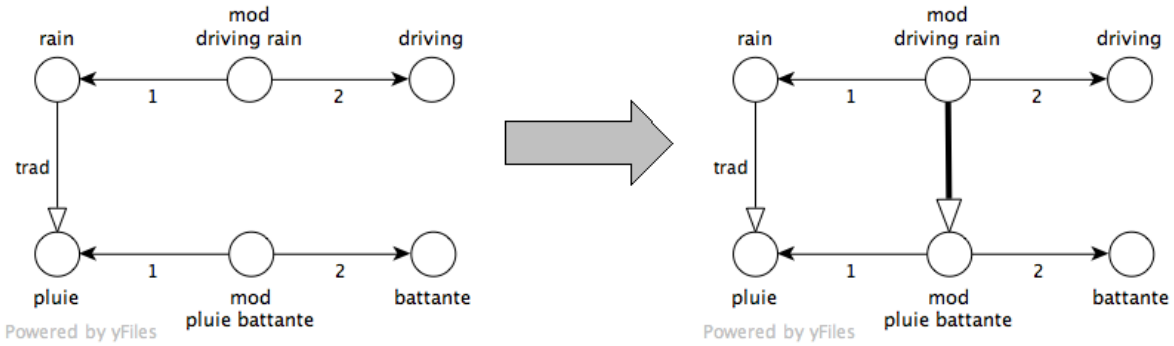


Figure 40: Propagation de traduction (représentation complexe)

Nous définissons une opération de propagation[129], utilisable dans une représentation complexe, qui produit un lien entre deux matérialisations s'il y a un lien entre leurs premiers arguments respectifs. Elle est utile pour le problème de l'extraction bilingue de collocations, quand on souhaite passer du lien de traduction entre deux bases monolingues à un lien entre les collocations respectives.

[129]Cet opérateur n'a pas été introduit dans la présentation générale du modèle, parce qu'il est trop lié au problème des bicollocations.

L'opérateur de propagation parcourt les arcs intraniveau et, pour chacun de ceux sur lesquels on veut appliquer la propagation, cherche si sa source et sa cible sont les premiers arguments de nœuds-matérialisation (dans le cas des bicollocations : la base de collocation), et ajoute le cas échéant un arc similaire entre ces nœuds-matérialisation.

Propagation

Paramètres d'entrée :

entier *Niv*,
fonction($E \rightarrow bool$) *filtre*,
liste(<attribut,fonction($attribut \times V \rightarrow valeur$ >) *CalculAttributs.*

Définition :

Effectue la propagation des arcs du niveau Niv *vérifiant* filtre *(la propagation d'un arc (s,t) est la création d'arcs similaires entre tous les couples possibles de nœuds matérialisation (rs, rt) tels que les premiers arguments respectifs de rs et rt soient s et t) ; la valeur des attributs des arcs ainsi créés est calculée grâce à* CalculAttributs*.*

Variables : arc *p*, nœud v_{classe} , Tableau[nœuds] *ClassArgs*=[ ];

Corps :

$\forall e \in E_{Niv}$:
si filtre(e) :
$\forall s \in V_{Niv} \mid Arg1(t) = source(e)$:
$\forall t \in V_{Niv} \mid Arg1(t) = cible(e)$:
ajouterArcIntra(*p*,Niv, s, t) ;
$\forall \langle att, f \rangle \in CalculAttributs$:
$p.att = f(att, e)$

3 Séquence d'opérations

3.1 Extraction monolingue

Les opérations principales de manipulation du graphe linguistique pour l'extraction (monolingue) de collocations sont illustrées dans le tableau 21.

3.1.1 Identification

Pour commencer l'extraction de collocations, il faut repérer les relations pouvant exprimer des collocations. Cela correspond, au niveau des graphes, à un filtrage des objets pour ne garder que ce qui nous intéresse :

- *filtrage des arcs :* ne conserver que ceux exprimant la relation voulue (selon le type, les parties du discours des extrémités, etc.) ;

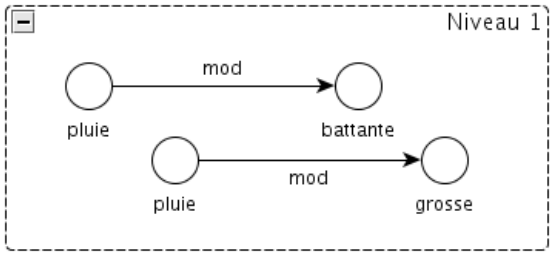
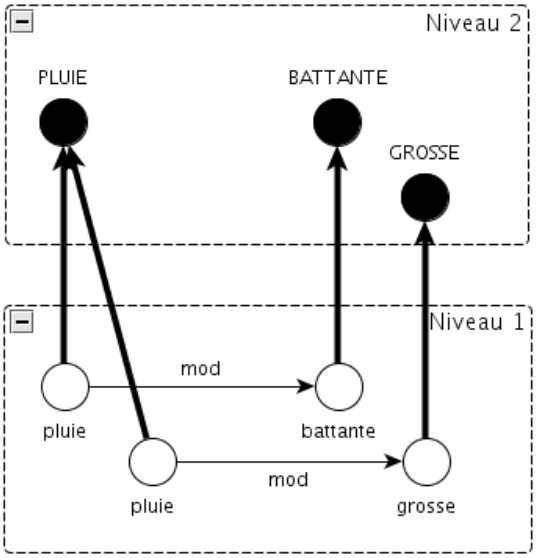
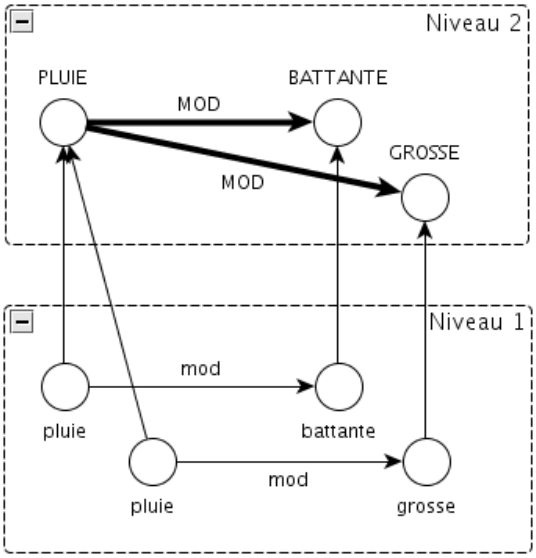
	
Émergence de nœuds	
Émergence d'arcs	

Tableau 21: Manipulation du graphe linguistique pour l'extraction de collocations

- *filtrage des nœuds* : conserver seulement les nœuds non « orphelins », c'est à dire qui sont à l'extrémité d'au moins un des arcs conservés.

3.1.2 Production

Il s'agit, dans cette étape, de passer du stade d'occurrences de relations (supposées être des collocations) à celui de modèles des collocations du corpus (en regroupant les occurrences d'une même collocation supposée) ; cela correspond à un passage au *niveau supérieur* dans notre représentation.

Cela se fait :

- tout d'abord en regroupant les objets *par classes d'équivalence* :
  - *regroupement éventuel des arcs* (pour considérer, par la suite, différentes relations comme équivalentes) ;
  - *regroupement des nœuds par classes*, et création des nœuds au niveau supérieur ; ces derniers représentent (de manière unique) des termes présents dans le corpus ;
- ensuite en réalisant *l'émergence des arcs au niveau supérieur*, à partir des arcs du niveau inférieur.

Les arcs ainsi produits au niveau supérieur renferment les candidats collocations ; reste alors à les classer.

3.1.3 Classement

L'étape de classement est celle qui provoque le moins de modifications sur la structure du graphe :

- *Calcul de mesure simple* à partir des nombres d'occurrences (des nœuds) et de co-occurrences (occurrences des arcs).

Après cette étape, il suffit de récupérer les arcs au niveau supérieur et d'y lire (sur l'attribut correspondant) les valeurs respectives de la mesure pour classer les résultats.

Identification	Filtrage des arcs (garde la relation fixée)
	Filtrage des nœuds (garde les non orphelins)
Production	Émergence de nœuds
	Émergence d'arcs
Classement	Calcul de mesure

Tableau 22: Récapitulatif des opérations nécessaires pour une extraction monolingue de collocations

3.1.4 Mise en œuvre

Le tableau 22 présente l'enchaînement d'opérations à réaliser. Nous détaillons ici comment cela doit se traduire en terme d'opérateurs du modèle MuLLinG.

3.1.4.1 Opérations d'identification

On effectue le filtrage des arcs grâce à *SupprimeArcsSi*, avec comme paramètre :

- *f_arcs(e)* : vrai si *e* n'exprime pas la relation qu'on étudie.

On effectue le filtrage des nœuds grâce à *SupprimeNœudsSi*, avec comme paramètre :

- *f_nœud(v)* : vrai si $\neg\exists e \mid v = source(e) \vee v = cible(e)$ (élimine les orphelins).

3.1.4.2 Opérations de production

L'émergence de nœuds (production des classes de termes) se réalise en utilisant *ÉmergenceDeNœuds*, avec comme paramètres :

- niveau : *1* ;
- filtre : *vrai* ;
- classe d'équivalence : *CEN*, renvoie la concaténation des attributs *lemme* et *pos* (partie du discours)
- calcul des attributs : *CAN*, liste de couples :
  - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "classedenœuds" ;
  - pour les autres attributs *att* : <*att, Copie*>.

L'émergence des arcs (production des classes de collocations) se réalise en utilisant *ÉmergenceDArcs*, avec comme paramètres :

- niveau : *1* ;
- filtre : *vrai* ;
- classe d'équivalence : *CEE*, renvoie le type de la relation ;
- calcul des attributs :
  - sur les arcs : *CAa*, liste de couples :
    - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "classedarcs" ;
    - pour les autres attributs *att* : <*att, Copie*> ;
  - sur les sources : *CAg*, liste de couples :
    - pour l'attribut *d+* (degré sortant) : <*d+, Incrémente*> ;
  - sur les cibles : *CAd*, liste de couples :
    - pour l'attribut *d-* (degré entrant) : <*d-, Incrémente*>.

3.1.4.3 Opérations de classement

Le calcul de la mesure de collocabilité se fait à l'aide de *CalculeF12*, avec comme paramètres :

- attribut : *mesure* ;
- niveau : *2* ;
- filtre : *f_classearcs(e)* : vrai si *e.type* ="classedarcs" ;
- relation : celle étudiée ;
- fonction : celle souhaitée.

3.2 Extraction bilingue

La complexité de l'extraction bilingue vient de la multiplicité des tâches à réaliser ; elle contient en effet trois extractions plus simples : celles des candidats collocations monolingues dans une langue et dans l'autre, ainsi que celle des liens bilingues de traduction entre les collocations monolingues (d'après les traductions entre bases). Cette complexité se ressent bien sûr sur la modélisation du processus comme une manipulation de graphes ; toutefois, si cette manipulation est plus longue, elle continue d'utiliser les seules opérations élémentaires que nous avons définies.

L'expérimentation réalisée utilise un graphe modélisant les relations de dépendances et de traduction d'un corpus parallèle. La production des relations de traductions est faite, *avant* l'expérimentation, à partir des alignements de phrases du corpus et d'un dictionnaire : on relie les occurrences de termes dans deux phrases alignées lorsque le dictionnaire indique que les deux termes sont en relation de traduction.

3.2.1 Identification

Pour commencer une extraction bilingue, il faut tout d'abord filtrer les objets :

- *filtrage des arcs :* ne conserver que ceux exprimant les relations voulues (selon le type, les parties du discours des extrémités, etc.) : relations monolingues dans les deux langues, relations bilingues de traduction ;
- *filtrage des nœuds* : conserver seulement les nœuds non « orphelins ».

Il faut ensuite représenter les relations de manière complexe (un nœud avec des arguments), afin d'obtenir une forme correspondant à une partie de la structure voulue pour les candidats bicollocations. Pour cela, il faut d'abord :

- *matérialiser* les liens de relations monolingues au niveau inférieur ;
- *propager* les informations de traduction entre des bases (premier argument des nœuds collocations) en informations de « traduction éventuelle » entre les collocations correspondantes.

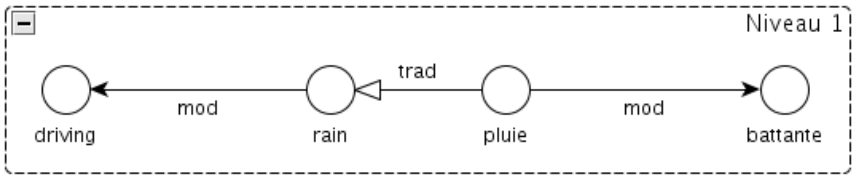
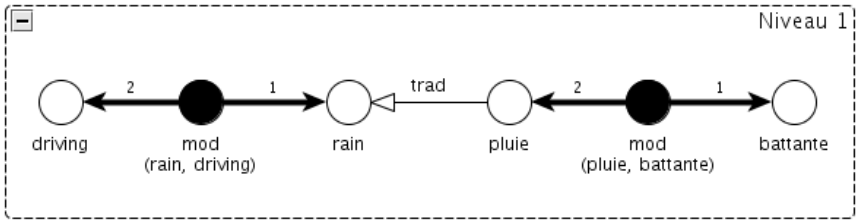
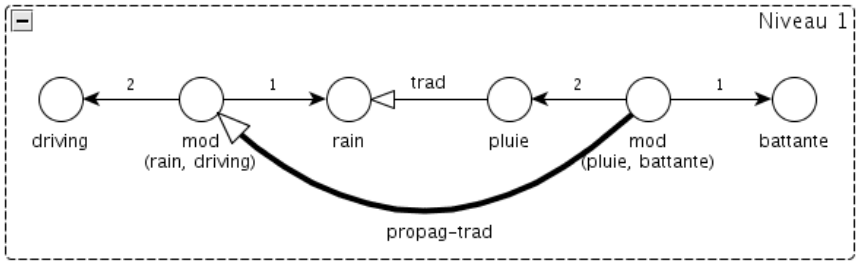
	
Matérialisation des arcs monolingues	
Propagation de l'information de traduction des bases vers les collocations	

Tableau 23: Manipulation du graphe linguistique pour l'extraction de bicollocations : identification des liens entre occurrences de collocations

3.2.2 Production

Le rôle de cette étape est de produire des objets, au niveau supérieur, correspondant à des candidats bicollocations. Cela doit se faire en plusieurs étapes, en générant successivement par *émergence* les diverses classes d'équivalence dont nous avons besoin :

- *classes de termes* : émergence de nœuds ;
- *classes de collocations* (à partir de la représentation complexe des relations au niveau inférieur) : émergence de relations matérialisées .

Le processus se comporte jusqu'ici comme dans le cas monolingue, à ceci près que nous sommes dans le cas d'une représentation complexe.

Il reste maintenant à produire les classes de bicollocations. Pour cela, nous avons les classes de collocations monolingues juste produites (matérialisées sous forme de nœuds), et les liens de traduction (obtenus grâce à la propagation) entre les occurrences de collocations.

Nous produisons donc les *classes de bicollocations* grâce à une émergence d'arcs classique.[130]

[130]On voit ici la puissance du modèle, qui permet de représenter la tâche relativement complexe d'extraction de bicollocations (dont, en particulier, la partie concernant la production des bicollocations à partir des collocations monolingues), comme une succession de modifications de graphe simples.

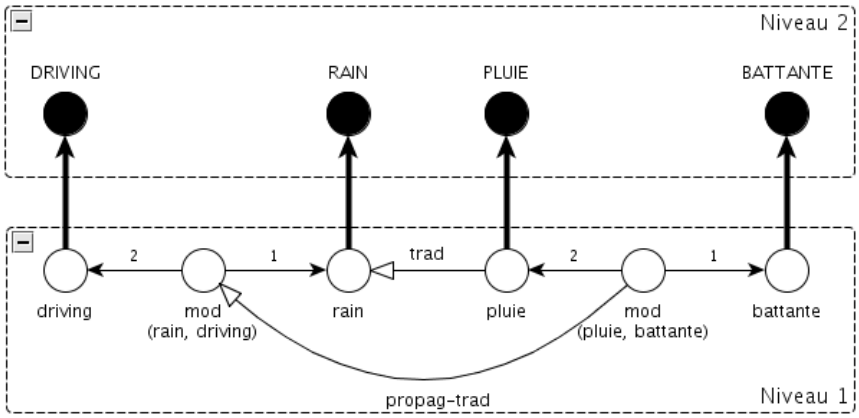
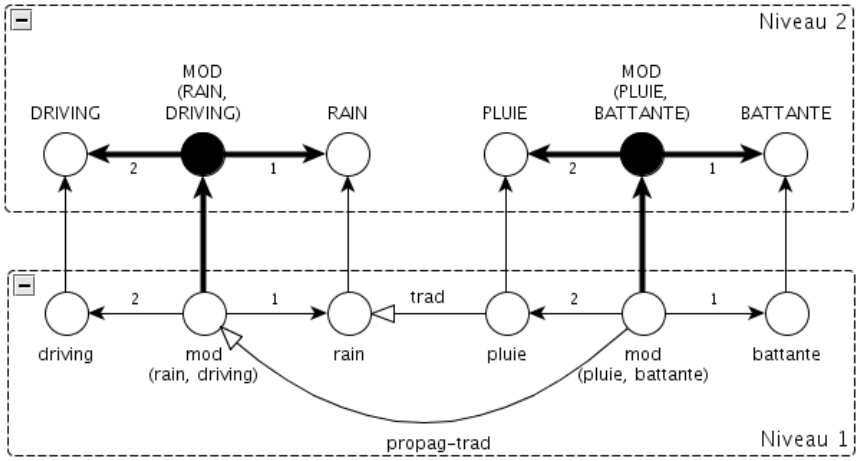
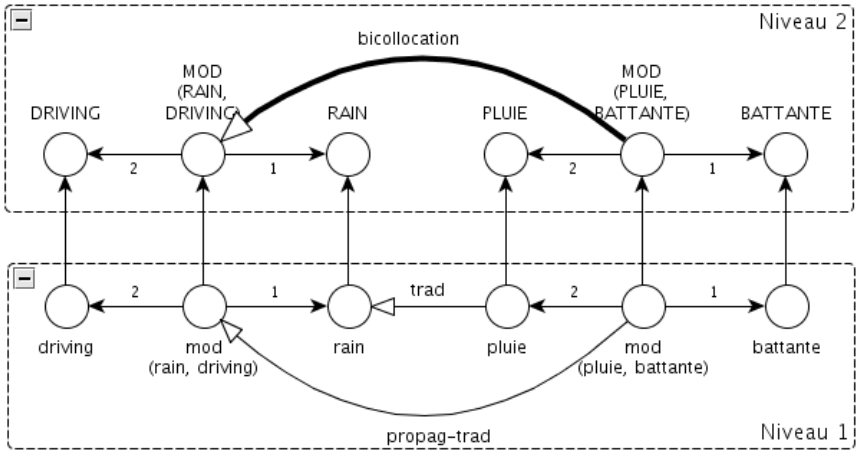
(suite du tableau 23) Émergence simple de nœuds : création des termes	
Émergence complexe d'arcs : création des collocations monolingues	
Émergence simple d'arcs : création des bicollocations	

Tableau 24: Manipulation du graphe linguistique pour l'extraction de bicollocations : production des candidats

3.2.3 Classement

Le classement des bicollocations doit s'effectuer à partir de mesures de bicollocabilité, qui dépendent des collocabilités monolingues (dans les deux langues) et de l'associabilité des collocations. Cela se traduit sur le graphe, pour chaque arc (au niveau supérieur) représentant une bicollocation (c'est-à-dire reliant deux collocations monolingues), par :

- *calcul de la collocabilité* de la composante en langue 1 du candidat (source de l'arc) ;
- *calcul de la collocabilité* de la composante en langue 2 du candidat (cible de l'arc) ;
- *calcul d'associabilité* des deux composantes monolingues (arc lui-même).

Les deux premiers calculs sont à faire dans une représentation complexe, le dernier dans une représentation simple.

Identification des liens entre occurrences de collocations	*Filtrage*	Filtrage des arcs (garde la relation fixée) Filtrage des nœuds (garde les non orphelins)
	(passage à la représentation complexe)	Matérialisation Propagation
Production	*Émergence*	Monolingue : émergence de nœuds (classes de termes) Monolingue : émergence de relations matérialisées (classes de collocations) Bilingue : émergence d'arcs (classes de bicollocations)
Classement	*Composantes*	Monolingue Langue 1 : calcul complexe de mesure Monolingue Langue 2 : calcul complexe de mesure Bilingue : calcul simple de mesure
	Calcul final	Calcul de mesure combinée

Tableau 25: Récapitulatif des opérations nécessaires pour une extraction monolingue de collocations

Une fois que toutes ces mesures intermédiaires ont été calculées, on peut *calculer la mesure combinée* de bicollocabilité finale, pour chaque arc représentant les bicollocations, à partir de la mesure d'associabilité calculée pour l'arc, et des mesures de collocabilité calculées pour ses extrémités.

3.2.4 Mise en œuvre

Le tableau 25 présente l'enchaînement d'opérations à réaliser. Nous détaillons ici comment cela doit se traduire en terme d'opérateurs du modèle MuLLinG.

3.2.4.1 Opérations d'identification

On effectue le filtrage des arcs grâce à *SupprimeArcsSi*, avec comme paramètre :

- *f_arcs(e)* : vrai si *e* n'exprime pas la relation qu'on étudie.

On effectue le filtrage des nœuds grâce à *SupprimeNœudsSi*, avec comme paramètre :

- *f_nœud(v)* : vrai si $\neg\exists e \mid v = source(e) \vee v = cible(e)$ (élimine les orphelins).

3.2.4.2 Opérations de passage à la représentation complexe

On réalise la matérialisation des relations (de co-occurrence monolingue) à l'aide de *MatérialisationArcs*, avec comme paramètres :

- niveau : *1* ;
- filtre : *f_mat(e)* vrai si *e* exprime la relation qu'on étudie (si *e.type* a le type voulu);
- calcul des attributs : *CAM*, liste de couples :
  - pour l'attribut type : <*type, NouvTypeMat*>, avec *NouvTypeMat*(att,v) → "mod-mater" ;
  - pour les autres attributs *att* : <*att, Copie*>.

On réalise la propagation de la relation de traduction (entre nœuds représentant des bases de collocations, aux nœuds représentant les collocations elle-même) à l'aide de *MatérialisationArcs*, avec comme paramètres :

- Propagation : *Propagation*, avec comme paramètres :
- niveau : *1* ;
- filtre : *f_mat(e)* vrai si *e* exprime la traduction (si *e.type* a le type correspondant);
- calcul des attributs : *CAP*, liste de couples :
  - pour l'attribut type : <*type, NouvTypePropag*>, avec *NouvTypePropag*(att,v) → "propag-trad" ;
  - pour les autres attributs *att* : <*att, Copie*>.

3.2.4.3 Opérations de production

Comme dans le cas monolingue, la production des classes de termes se fait en utilisant *ÉmergenceDeNœuds*, avec comme paramètres :

- niveau : *1* ;
- filtre : *f_emergnoeud(v)* vrai si *v* représente une occurrence de terme (si *v.type* a le type correspondant) ;
- classe d'équivalence : *CEN*, renvoie la concaténation des attributs *lemme* et *pos* (partie du discours)
- calcul des attributs : *CAN*, liste de couples :
  - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "classedenœuds"
  - pour les autres attributs *att* : <*att, Copie*>.

Ensuite, la production des classes de collocations se réalise en utilisant *ÉmergenceDeRelationsMaterialisées*, avec comme paramètres :

- niveau : *1* ;

- filtre : *f_emergmater(e)* vrai si *e* représente une matérialisation de relations (si *e.type* = "mod-mater") ;
- classe d'équivalence : *CEM*, renvoie le type de la relation ;
- calcul des attributs :
  - sur les matérialisations (nœuds) : *CAm*, liste de couples :
    - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "classedemater"
    - pour les autres attributs *att* : <*att, Copie*> ;
  - sur les arguments : *CAarg* tableau de listes de couples :
    - *CAarg[1]* liste des calculs correspondant au premier argument :
      - pour l'attribut *d+* (degré sortant) : <*d+, Incrémente*> ;
    - *CAarg[2]* liste des calculs correspondant au second argument :
      - pour l'attribut *d-* (degré entrant) : <*d-, Incrémente*>.

Enfin, la production des classes de bicollocations se réalise en utilisant *ÉmergenceDArcs*, avec comme paramètres :

- niveau : *1* ;
- filtre : *f_emergbicoll(e)* vrai si *e* représente une classe de collocations (si *v.type*=classedemater) ;
- classe d'équivalence : *CEB*, renvoie le type de la relation porté par l'arc ;
- calcul des attributs : *CAB*, liste de couples :
  - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "classebicoll" ;
  - pour les autres attributs *att* : <*att, Copie*>.

3.2.4.4 Opérations de classement

On calcule tout d'abord les mesures monolingues de collocabilité grâce à *CalculeF12Complexe*[131], avec comme paramètres :

- attribut : *mesure* ;
- niveau : *2* ;
- filtre : *f_coll_langue(v)* : vrai si *v.type* ="mod-mater" et si la langue est bien celle voulue ;
- relation : celle étudiée ;
- fonction : la mesure de collocabilité souhaitée ;

On calcule ensuite la mesure d'associabilité (bilingue) grâce à *CalculeF12*, avec comme paramètres :

- attribut : *mesure* ;
- niveau : *2* ;
- filtre : *f_classebicoll(e)* : vrai si *e.type* ="classebicoll" ;

[131]Deux fois (avec un filtre différent) : l'une pour la « langue 1 », l'autre pour la « langue 2 »

- relation : celle étudiée ;
- fonction : la mesure d'associabilité souhaitée ;

Enfin, on calcule la mesure globale (à partir des précédentes) en utilisant *CalculeF3*, avec comme paramètres :

- attribut : *mesureglob* ;
- niveau : *2* ;
- filtre : *f_classebicoll(e)* : vrai si *e.type* ="classebicoll" ;
- attribut de l'arc, de la source, de la cible : *mesure* ;
- fonction : la mesure monolingue souhaitée.

4 Expérimentations

Nous avons réalisé des expérimentations d'extraction de collocations et de bicollocations similaires à celles déjà présentées au chapitre IV.

Ces nouvelles expérimentations, monolingues et bilingues, sont basées sur le modèle de graphe linguistique multiniveau MuLLinG et les opérations associées (telles que décrites dans les pages précédentes). Elles utilisent pour cela notre bibliothèque C++ implémentant ce modèle.

Si nous présentons les résultats de ces expérimentations (et les commentons), *nous ne présentons pas d'évaluation en détail de la qualité des résultats produits, puisque celle-ci dépend des filtrages employés ou des mesures de classement, qui ne sont que des paramètres de notre outil*.

4.1 Monolingue

Les expérimentations sont réalisées à partir du corpus LeMonde95[132]. Nous produisons un graphe modélisant le corpus, où les nœuds sont les occurrences des termes et les arcs correspondent aux relations de dépendance entre ces occurrences, d'après l'analyseur de dépendances XIP.

Expérimentations	*Niveau 1*		*Niveau 2*	
	Nœuds	*Arcs*	*Nœuds*	*Arcs*
verbe-adverbe	1 155 824	1 780 759	6 813	144 586
nom-adjectif	1 319 474	2 009 051	33 132	273 655

Tableau 26: Arcs et nœuds des graphes linguistiques utilisés pour les expérimentations d'extraction monolingue de collocations

[132]Les caractéristiques du corpus ont été données dans le tableau 8 (page 85).

Implémentation et applications

Nous avons effectué deux expérimentations, où l'on cherchait à extraire des collocations :

- avec base verbale et collocatif adverbial ;
- avec base nominale et collocatif adjectival.

Dans les tableaux présentant les résultats de l'expérimentation, les candidats corrects, c'est-à-dire ceux qui sont bien des collocations, sont en italique.

Pour chaque expérimentation, deux mesures de collocabilité ont été calculées : *Information mutuelle* et *WMI*.

Information mutuelle	*WMI*	*WMI+filtrage*
nickeler par tête de pipe	mettre fin	pouvoir difficilement
médire doucereusement	valoir mieux	travailler ensemble
conceptualiser adéquatement	arriver en tête	renvoyer dos à dos
diriger opérationnellement	coûter cher	chercher en vain
cavaler agilement	mettre en avant	*répéter à l'envi*
ligoter politiquement	aller loin	devoir impérativement
vernisser et tout le bataclan	entrer en vigueur	*accueillir favorablement*
trémousser lascivement	lire ci-dessous	*dépasser largement*
répudier *offciellement	avoir également	*participer activement*
dilater grandiosement	lire ci-contre	*régner sans partage*

Tableau 27: Dix premiers résultats de l'extraction de collocations verbe-adverbe, d'après les mesures de collocabilité

Les expérimentations produisent respectivement 144 586 candidats verbe-adverbe et 273 655 candidats nom-adjectif (voir tableau 26).

Les premiers candidats d'après les mesures de collocabilité sont présentés dans les tableaux 27 (verbe-adverbe) et 28 (nom-adjectif). Pour l'expérimentation portant sur les collocations verbes-adverbes, les résultats sont présentés sous deux formes : sans filtrage (sauf pour éliminer les auxiliaires et l'adverbe « pas »), et avec un filtrage destiné à retirer les adverbes ne modifiant pas le sens du verbe (temps, lieu, doute, etc.).

Comme attendu, l'information mutuelle favorise les paires impliquant des mots rares. Cela explique le bruit dû aux fautes de frappe ou aux expressions latines ou anglaises. On pourrait certainement éliminer les cas les plus inutiles en fixant (de manière arbitraire) un seuil minimal sur le nombre d'occurrences des termes et leur co-occurrence, mais les paires les plus rares parmi celles gardées resteraient surévaluées. C'est d'ailleurs l'intérêt de la pondération présente dans WMI qui évite ce phénomène.

L'extraction des collocations nom-adjectif donne des résultats moins bons. Toutefois, on peut constater que, avec WMI, la plupart des candidats mis en avant, s'ils ne sont pas des collocations, sont des « expressions multi-mots » (MWE) et fourniraient un résultat très satisfaisant pour une telle tâche.

Information mutuelle	*WMI*
jupette ultra-courte	premier ministre
*balckcurrent crumble	première foi
wet leasing	*droite extrême*
*discu ssion	second tour
orbis pictus	année dernière
gladiateur nubien	premier tour
centesimus annus	ancien ministre
thermostables photosensibles	secrétaire général
papet vaudois[133]	casque bleu
passings virevoltants	certain nombre

Tableau 28: Dix premiers résultats de l'extraction de collocations nom-adjectif, d'après les mesures de collocabilité

Les résultats de l'extraction de collocations, vus dans leur ensemble, peuvent sembler difficiles à exploiter directement pour la construction de ressources. C'est un problème que nous avons observé lors de nos précédentes expérimentations : on rencontre un bruit important quand on ne filtre pas, mais on provoque du silence en éliminant les collocations les plus opaques dès qu'on filtre.

Il faut toutefois relativiser ce constat : les collocations sont décrites en fonction de leur base. En particulier, un lexicographe peut être intéressé par la liste de tous les collocatifs éventuels d'une base, si possible classés par la collocabilité du couple base-collocatif. Cela correspond, dans notre représentation de graphes linguistiques, à *l'ensemble des voisins d'une base avec lesquels il est relié par un arc pouvant exprimer la collocation*.

L'un des avantages de disposer de l'information sous la forme de graphes MuLLinG est que, pour obtenir de telles informations, il n'est pas nécessaire de modifier le processus d'extraction ni d'écrire un script les extrayant du résultat renvoyé.

En effet, à partir d'un nœud représentant un terme, on peut obtenir l'ensemble de ses collocatifs possibles simplement en considérant les nœuds voisins vers lesquels part un arc correspondant à la classe des relations de modification considérées.

Le tableau 29 présente des exemples de ce que permet d'obtenir facilement cette vue concentrée sur une seule base. On peut constater que les mesures de collocabilité permettent bien de faire ressortir les candidats les plus intéressants.

Pour les deux exemples présentés, choisis au hasard[134], on peut constater que les classements des collocatifs selon l'*information mutuelle* et selon *WMI* sont presque identiques (pas de différence pour « polluer », une seule permutation pour « foisonnement »). Cela peut paraître surprenant, mais il faut rappeler que WMI consiste en la pondération de l'information mutuelle par la probabilité que les deux termes soient en relation (pour tenter d'éliminer le biais de l'information mutuelle sur les couples rares). Quand on considère une base donnée

[133]L'analyseur de dépendances affirme, de manière erronée, que « papet » est un adjectif qui modifierait le nom « vaudois ».

[134]Mais le même phénomène a été constaté sur de nombreux autres mots.

qui n'est pas rare dans le corpus, l'information mutuelle est moins sensible aux erreurs : plus cohérents, les résultats sont peu modifiés par la pondération de WMI.

Collocatifs de « polluer »		*Collocatifs de « foisonnement »*	
Information mutuelle	*WMI*	*Information mutuelle*	*WMI*
moralement	moralement	concomitant	concomitant
dangereusement	*dangereusement*	instrumental	instrumental
physiquement	physiquement	génial	génial
en plus	en plus	sympathique	sympathique
gravement	*gravement*	documentaire	documentaire
considérablement	*considérablement*	parallèle	parallèle
un peu plus	un peu plus	associatif	associatif
voire	voire	*incroyable*	*incroyable*
tant	tant	technologique	technologique
davantage	davantage	*riche*	*riche*
mieux	mieux	présenté	présenté
beaucoup	*beaucoup*	immense	court
peut-être	peut-être	court	immense
moins	moins	vrai	vrai
même	même	tel	tel
donc	donc	public	public
aussi	aussi		
aujourd'hui	aujourd'hui		
encore	encore		
plus	plus		

Tableau 29: Collocatifs de « polluer » (verbe) et de « foisonnement » (nom) classés par mesure de collocabilité

4.2 Bilingue

Les expérimentations bilingues sont réalisées à partir du corpus parallèle *EuroParl*[135]. Pour cela, nous produisons un graphe correspondant aux occurrences de termes dans les sections française et anglaise du corpus, et les arcs correspondant aux relations entre ces termes. Nous avons créé des liens bilingues (de traduction) entre occurrences de termes français et anglais, d'après l'alignement des phrases et un dictionnaire bilingue.

Nous avons effectué deux expérimentations, où nous cherchions à extraire des *bicollocations* formées de collocations à :

[135]Les caractéristiques du corpus ont été données dans le tableau 13 (page 98).

- base verbale et collocatif adverbial ;
- base adjectivale et collocatif adverbial.

Nous n'avons pas utilisé de filtre sémantique (comme les vecteurs conceptuels dans ces expérimentations.

Expérimentations	*Niveau 1*		*Niveau 2*	
	Nœuds	*Arcs*	*Nœuds*	*Arcs*
verbe-adverbe	670 998	1 120 868	66 202	120 319
sans repr. complexe :	*447 332*	*673 536*	*6 385*	*685*
adjectif-adverbe	1 301 054	2 203 121	93 258	170 262
sans repr. complexe :	*862 628*	*1 326 269*	*10 089*	*3 924*

Tableau 30: Arcs et nœuds des graphes linguistiques utilisés pour les expérimentations d'extraction monolingue de collocations

Dans les tableaux présentant les résultats de l'expérimentation, les composantes monolingues correctes (celles qui sont bien des collocations) sont en italique ; les candidats corrects (ceux qui sont bien des bicollocations d'intensification) sont en gras.

Pour chaque expérimentation, nous calculons :

- la collocabilité des composantes monolingues des collocations, avec l'Information mutuelle (MI) et WMI ;
- l'associabilité des collocations, avec le cosinus bilingue[136].

Bic-MI		*Bic-WMI*	
mesurer concomitamment	*measure concomitantly*	*travailler conjointement*	*work together*
prier publiquement	pray ostentatiously	*travailler collectivement*	*work together*
attendre impatiemment	*wait impatiently*	fonctionner efficacement	work together
placer alternativement	*place alternatively*	passer outre	take seriously
renvoyer ad[137]	postpone indefinitely	prendre conjointement	take seriously
prolonger indéfiniment	***prolong indefinitely***	faire justement	do indeed
attendre patiemment	*wait patiently*	faire inévitablement	do indeed
payer cher	pay dearly	mettre effectivement	put together
traiter bestialement	*treat abominably*	*adopter unanimement*	*adopt unanimously*
frapper durement	***hit hard***	*travailler étroitement*	*work closely*

Tableau 31: Dix premiers résultats de l'extraction de bicollocations verbe-adverbe, d'après les mesures de bicollocabilité

[136]Mesure introduite page 96.

[137]Partie de l'expression « (r)envoyer ad patres »

Ces mesures sont combinées pour obtenir les mesures finales suivantes :

- $Bic_{WMI}(\langle b_{1,} c_1\rangle, \langle b_{2,} c_2\rangle) = [WMI(\langle b_{1,} c_1\rangle) + WMI(\langle b_{2,} c_2\rangle)] \times \cos_{bil}(\langle b_{1,} c_1\rangle, \langle b_{2,} c_2\rangle)$;
- $Bic_{MI}(\langle b_{1,} c_1\rangle, \langle b_{2,} c_2\rangle) = [MI(\langle b_{1,} c_1\rangle) + MI(\langle b_{2,} c_2\rangle)] \times \cos_{bil}(\langle b_{1,} c_1\rangle, \langle b_{2,} c_2\rangle)$.

Les expérimentations produisent respectivement 685 candidats verbe-adverbe et 3924 candidats adjectif-adverbe.

Les premiers candidats d'après les mesures de collocabilité sont présentés dans les tableaux 31 (verbe-adverbe) et 32 (adjectif-adverbe). Les candidats dont les composantes sont traductions l'une de l'autre sont en italique ; les candidats qui sont des bicollocations sont en gras.

On peut constater qu'on obtient beaucoup de candidats où les collocatifs sont traduction l'un de l'autre. Cela n'est pas surprenant dans la mesure où il s'agit de traductions de débats, et on peut supposer que, même lorsqu'on pourrait utiliser plusieurs termes pour exprimer un sens, les traducteurs vont plutôt choisir la traduction littérale du collocatif si elle conserve le sens.

Bic-MI		*Bic-WMI*	
étrangement poilu	*strangely hairy*	mieux possible	as possible
inversement proportionnel	*inversely proportional*	efficacement possible	as possible
vaguement racial	*vaguely racial*	étroitement possible	as possible
vaguement mythique	*slighty mythical*	précisément possible	as possible
gravement malade	chronically sick	bref possible	as possible
tellement poilu	strangely hairy	largement possible	as possible
extraordinairement talentueux	*amazingly talented*	clair possible	as possible
apparemment interminable	*seemingly endless*	judicieusement possible	as possible
pratiquement ininterrompu	*nearly uninterrupted*	clairement possible	as possible
comme invincible	militarily invincible	partiellement possible	as possible

Tableau 32: Dix premiers résultats de l'extraction de bicollocations adjectif-adverbe, d'après les mesures de bicollocabilité

Tous les candidats les mieux placés pour l'expérimentation sur les bicollocations adjectif-adverbe avec la mesure *Bic-WMI* ont la même composante. Il s'agit de collocations quaternaires « le plus X possible »/« as X as possible ». Le traitement retient malheureusement seulement « X possible » en français et « as possible » en anglais, ce qui n'est pas du tout exploitable.

Cela montre en tout cas les limites de cette mesure, qui ne pondère pas suffisamment les résultats pour éviter un tel biais. De manière générale, on constate une moins bonne qualité avec Bic-WMI qu'avec Bic-MI.

Le tableau 33 présente des exemples des résultats obtenus en concentrant la vue sur une seule base.

Bicollocations candidates avec base française « participer »			
Bic-MI		*Bic-WMI*	
participer activement	***participate actively***	***participer pleinement***	***participate fully***
participer consciencieusement	*participate conscientously*	***participer activement***	***participate actively***
		participer directement	*participate directly*
participer pleinement	***participate fully***	***participer activement***	***participate effectively***
participer directement	*participate directly*	*participer consciencieusement*	*participate conscientously*
participer activement	***participate effectively***		
participer positivement	*participate positively*	*participer positivement*	*participate positively*
participer activement	participate closely	participer activement	participate closely
Bicollocations candidates avec base française « correct »			
Bic-MI		*Bic-WMI*	
formellement correct	***formally correct***	*politiquement correct*	*politically correct*
politiquement correct	*politically correct*	certes correct	politically correct
juridiquement correct	morally correct	juridiquement correct	not correct
certes correct	politically correct	***absolument correct***	***entirely correct***
juridiquement correct	*legally correct*	*juridiquement correct*	*legally correct*
absolument correct	***entirely correct***	juridiquement correct	morally correct
juridiquement correct	not correct	***vraiment correct***	***quite correct***
vraiment correct	***quite correct***	***très correct***	***quite correct***
très correct	***quite correct***	***formellement correct***	***formally correct***
très correct	***very proper***	***très correct***	***very proper***
plus correct	more correct	plus correct	most correct
plus correct	most correct	plus correct	more correct

Tableau 33: Bicollocations ayant pour base « participer » (verbe) et « correct » (adjectif) classées par mesure de collocabilité

Là encore, on peut constater des erreurs dues à un mauvais découpage : « plus correct » est en réalité employé dans la forme comparative (« plus correct que »/« more correct than ») ou superlative (« le plus correct »/« the most correct »). On peut toutefois observer que les mesures utilisées rejettent bien ces cas dans les dernières positions.

Par contre, même en utilisant un corpus parallèle aligné par phrases, on produit encore un certain nombre de couples qui ne sont pas traductions l'un de l'autre (« juridiquement correct »/« morally correct »). On peut critiquer les mesures employées sur le fait qu'elles peuvent mettre en bonne place de telles erreurs au lieu de les repousser.

Enfin, si la qualité est satisfaisante, il faut néanmoins remarquer que, pour une base donnée, on a beaucoup moins de bicollocations que de collocations et qu'on couvre donc moins tous les collocatifs possibles dans les deux langues. Cela ne rend pas pour autant ce genre

d'informations inutile : en réalité, les descriptions des comportements monolingue et bilingue d'une base sont complémentaires.

Conclusion

Nous avons présenté dans ce chapitre des expérimentations d'extraction de collocations et de bicollocations basée sur les opérations de manipulation de graphes linguistiques que nous avons décrites. Pour cela, nous avons introduit une opération, la propagation, destinée à transmettre l'information sur la traduction de bases de collocations aux collocations elles-mêmes. Nous avons ensuite décrit les processus d'extraction de collocations, puis de bicollocations, comme une succession de modifications simples de graphe, définies par notre modèle MuLLinG.

Nos expérimentations montrent qu'on peut, avec ces opérations simples, reproduire un processus complexe d'extraction de connaissances lexicales.

Cependant, cet exemple, s'il permet de valider nos suppositions sur la faisabilité d'une telle démarche, ne montre qu'un aspect limité de l'intérêt de notre représentation. Il reste en effet à démontrer qu'elle peut être utilisée pour d'autres problèmes linguistiques relativement différents, et que les opérations de fusion sur les graphes linguistiques présentent un réel intérêt.

Chapitre IX - Une autre application : mesure d'association bilingue à partir de WordNet

Après avoir montré que notre modèle de graphes linguistiques permettait bien d'effectuer l'extraction de collocations ou de bicollocations à partir de corpus, nous présentons ici une autre expérimentation, qui combine plusieurs graphes linguistiques représentant des données de différents types. Cela permet de montrer la diversité d'emploi des graphes linguistiques et leur intérêt quand on a à résoudre des problèmes linguistiques divers.

Le problème choisi est la pondération de liens de traduction (tels qu'on peut en trouver dans des ressources linguistiques sur des données bilingues basées sur WordNet). Après l'avoir présenté, nous montrerons comment on peut le résoudre dans la représentation des graphes linguistiques avec les opérateurs existants et d'autres que nous introduirons, puis nous traiterons de la mise en œuvre de cette solution et des résultats obtenus.

1 Caractériser les liens de traduction

On a parfois besoin, quand on utilise des données linguistiques afin d'obtenir des relations de traduction pour un processus comme la recherche d'information multilingue, de pondérer les liens en fonction de la confiance plus ou moins forte qu'on peut avoir dans leur correction.

Pour obtenir une telle information, nous proposons de combiner des liens de traduction venant d'une ressource de référence et d'autres liens non validés[138], pondérés selon leur apparition dans des entités textuelles traductions l'une de l'autre.

1.1 Liens de référence : WordNet

Notre idée est d'utiliser les données bilingues basées sur la ressource largement répandue qu'est *WordNet*.

Développée à l'université de Princeton, WordNet est une base de données lexicales (Fellbaum, 1998) qui a la forme d'un réseau dont :

- les nœuds sont des *synsets*, ensembles de « sens de mots » (lexies) quasi-synonymes ;
- les arcs représentent des relations paradigmatiques (hyponymie, méronymie, etc.).

Nous avons fait une présentation assez détaillée de cette ressource dans le troisième chapitre (page 30).

[138]Il s'agit de liens produits avec des processus automatiques, et qui n'ont pas été évalués par un spécialiste.

Suite au succès de WordNet, de nombreux autres projets de bases lexicales ont repris la même structure, si bien que le terme désigne à la fois le type de base lexicale et la base originale. C'est pourquoi, dans la suite, pour éviter les confusions, « wordnet » désignera le type de base lexicale, et nous emploierons « PWN» (pour « Princeton WordNet ») pour parler de la base originale.

Pour le français, il existe deux tels « réseaux de mots » d'envergure reliés à WordNet (ce sont ces liens qui nous intéressent) : le volume français d'*EuroWordNet* et *Wolf*.

1.1.1 EuroWordNet

La première ressource à considérer est *EuroWordNet* (EWN), produite par un projet européen éponyme mené par l'université d'Amsterdam à la fin des années 90. Ce projet a permis de développer des réseaux de mots (similaires à PWN) pour plusieurs langues européennes (italien, espagnol, néerlandais, français, suédois, tchèque, estonien) et de compléter le wordnet original de l'anglais (PWN, dans sa version 1.5) (Vossen, 2004); ces ressources sont payantes.

Chaque réseau de mots a sa propre structure interne, indépendante de PWN. En effet, se baser sur PWN biaiserait la description de l'organisation du lexique, chaque langue représentant le monde de manière différente. La correspondance entre les entrées des « wordnets » monolingues d'EWN se fait grâce à une structure pivot nommée « *Inter-Lingual-Index* » (basée sur les synsets de PWN), contenant des concepts interlingues reliés aux concepts monolingues correspondants dans chaque langue.

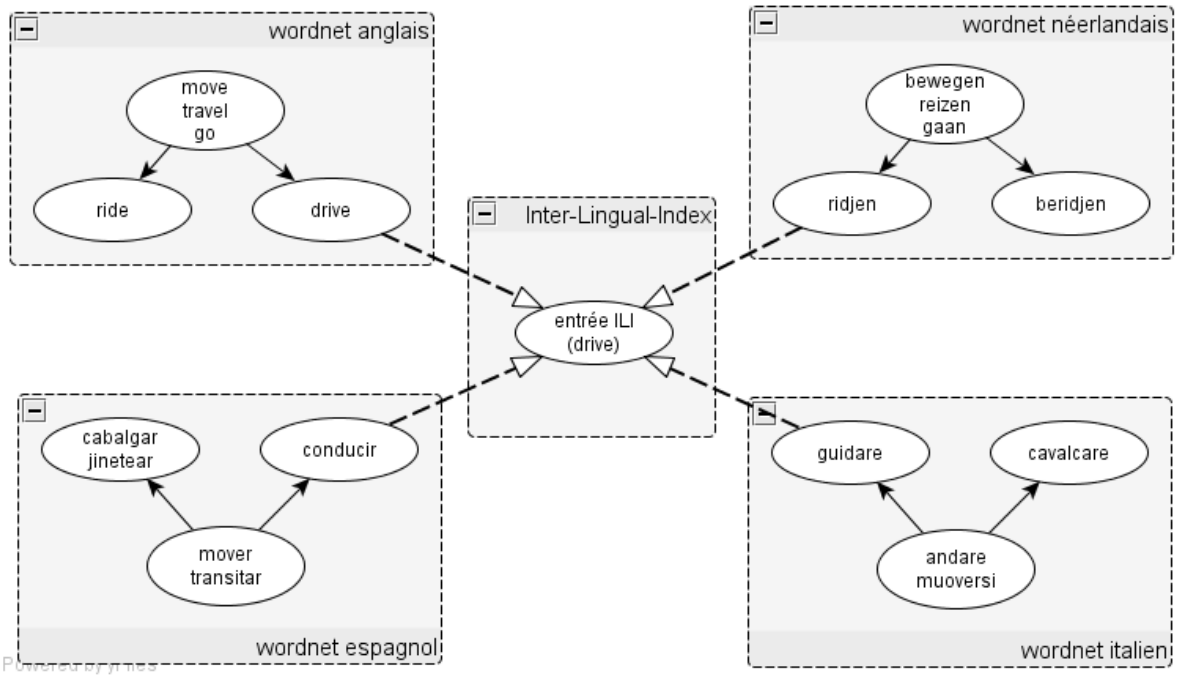


Figure 41: Architecture multilingue (simplifiée) d'EuroWordNet, d'après (Vossen, 2004)

1.1.2 Wolf

Les principaux freins à l'utilisation du volume français d'EuroWordNet sont sa licence payante, et l'absence des lexèmes adjectivaux et adverbiaux. Cela a mené au développement de *WoLF* (Wordnet libre[139] du français), une ressource gratuite visant à devenir l'équivalent de référence au PWN pour le français, développée à partir de thésaurus et des entrées de Wikipedia (Sagot et Fišer, 2008).

Wolf est réalisé à partir de la version 2.0 de PWN, dont il reprend la structure interne[140] (les mêmes synsets), considérant que le biais provoqué par ce choix est faible, étant donné la proximité entre les langues française et anglaise. Si cette décision est discutable, elle permet en tout cas de faciliter nettement le développement.

Wolf 0.1.4 contient 32351 synsets non vides[141]. Cela peut sembler peu par rapport aux 119221 synsets de PWN 2.0, mais cela dépasse déjà largement les 22745 synsets du volume français d'EuroWordNet. Toutefois, si la qualité des données est relativement honorable pour un résultat de processus automatiques (environ 80% de précision selon Sagot & Fišer, 2008), il reste encore un bruit important, gênant pour l'utilisation de telles données dans un processus automatique de traitement de la langue.

L'annexe C présente les entrées de PWN 2.0, EWN, et Wolf, correspondant à un même synset.

1.2 Utilisation : sources de liens de traduction

1.2.1 Principe

Dans les expérimentations que nous avons déjà réalisées concernant l'extraction de bicollocations, intervient le calcul d'une mesure d'associabilité bilingue entre collocations, basée sur les fréquences d'apparition des collocations dans des entités textuelles associés. Ici, nous reprenons l'idée d'un calcul d'associabilité bilingue, mais plus simple, entre les termes du corpus.

Nous souhaitons produire *une pondération des informations de traduction venant des wordnets d'après leur apparition conjointe dans des corpus alignés*.

1.2.2 Gestion des informations

Ce qui nous intéresse réellement dans ces « réseaux de mots » du français, ce sont les liens qu'entretiennent les lexèmes qui les composent avec ceux qui composent le WordNet de Princeton. Un problème se pose alors : il n'y a pas de liens directs entre ces lexèmes, les liens s'effectuant entre sens. Un terme du volume français sera synonyme d'un terme anglais :

- dans *EuroWordNet* : si les deux termes sont reliés au même concept interlingue de l'Inter-Lingual-Index (ILI) ;

[139]Licence CeCILL-C. La base est téléchargeable à l'adresse http://gforge.inria.fr/projects/wolf

[140]Le même choix a été fait par les concepteurs du wordnet du hindi (Narayan et al., 2002) et de Balkanet, wordnet des balkans (Tufis, 2004).

[141]Wolf contient tout les synsets de PWN 2.0, même ceux pour lesquels aucun équivalent en français n'a encore été trouvé.

- dans *Wolf* : si ils appartiennent au même synset.

1.2.2.1 Liens entre le volume français d'EuroWordNet et Princeton WordNet 1.5

L'utilisation d'ILI, la structure pivot d'EuroWordNet faisant correspondre entre elles les acceptions (« sens de mot ») monolingues présentes dans les différents volumes, a pour conséquence qu'un terme d'une langue ne correspond pas nécessairement à un et un seul terme d'une autre langue.

De plus, les liens entre un terme d'un volume monolingue d'EWN et un concept de la structure pivot peuvent non seulement exprimer la synonymie, mais aussi la quasi-synonymie, la généralisation, la métonymie, etc. Ces derniers types de liens peuvent être intéressant à étudier quand il n'y a pas de synonyme au terme dans l'ILI, ce qui peut signifier que le terme n'a pas de traduction exacte, mais seulement une traduction approchée. Toutefois, dans nos expérimentations, nous utiliserons seulement les relations de synonymie.

Les choses sont toutefois facilitées par le fait que les concepts interlingues composant l'ILI sont basés sur les synsets de l'anglais. Les traductions d'un terme français donné sont donc les lexies qui composent le synset associé au concept de l'ILI auquel il est relié.

1.2.2.2 Liens entre Wolf et Princeton WordNet 2.0

Les choses sont un peu plus simples pour obtenir des relations de traduction à l'aide de Wolf, surtout en ce qui concerne l'obtention du synset anglais correspondant, puisque le synset auquel il appartient est défini (de manière unique) en référence à un synset de PWN.

On peut d'ailleurs considérer que Wolf, qui englobe PWN, contient des synsets bilingues. Les liens de traduction qui nous intéressent sont alors les liens bilingues entre termes français et anglais d'un même synset.

2 Représentation des données

Dans cette approche, nous considérons deux graphes différents, que nous ferons fusionner.

Celui issu du wordnet (EWN ou Wolf) contient :

- des nœuds « français » et des nœuds « anglais » ;
- des arcs (entre un nœud français et un nœud anglais) correspondant aux liens de synonymie bilingue existant dans la ressource originale.

Celui issu du corpus contient :

- des nœuds « français » et des nœuds « anglais » ;
- des arcs monolingues correspondant aux liens entre termes, obtenus par exemple à partir d'une analyse de dépendances (nous n'utiliserons pas ces arcs)

3 Nouvelles opérations (pré-traitement des associations bilingues)

Comme pour la propagation au chapitre précédent, il nous faut introduire des opérateurs de manipulation de graphes MuLLinG utiles pour la tâche que nous souhaitons réaliser.

Nous présentons les opérations liées à la représentation des entités textuelles et des alignements d'entités textuelles phrases dans un graphe MuLLinG, puis celles concernant les liens d'apparition conjointe (dans deux entités alignées) qu'on peut en déduire.

3.1 Création des nœuds-entités

Nous introduisons tout d'abord une opération permettant de créer, à partir d'un attribut, de nouveaux nœuds, chacun correspondant à une valeur d'un attribut, et se comportant comme le père des nœuds portant la bonne valeur pour cet attribut.

Cette opération fonctionne à peu près comme *ÉmergenceDeNœuds*, mais n'utilise qu'un attribut pour identifier les « classes », et crée les nœuds correspondant à ces classes au même niveau.

On peut employer cet opérateur pour générer des « nœuds-entités », à partir d'un attribut identifiant l'entité textuelle à laquelle appartient l'occurrence représentée par le nœud, afin de matérialiser les entités textuelles du corpus utilisé.

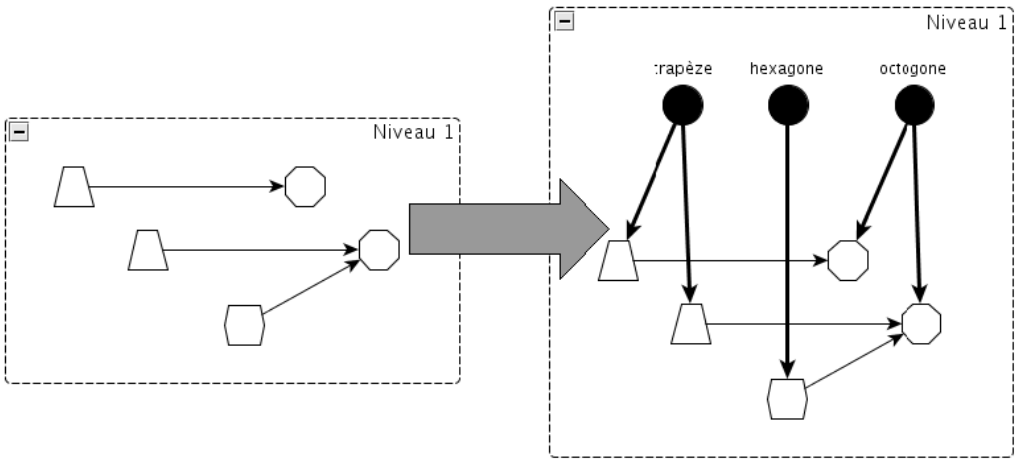


Figure 42: Création de nœuds selon l'attribut de forme

CreeNœudsSelonAttribut

Paramètres d'entrée :

entier *Niv*,
fonction($V \rightarrow bool$) *filtre*,
attribut *attr*,
liste(<attribut,fonction($attribut \times V \times V \rightarrow valeur$ >) *CalculAttNœud*
liste(<attribut,fonction($attribut \times V \times E \rightarrow valeur$ *>) CalculAttArc.*

Définition :

Crée, à partir des nœuds du niveau Niv *vérifiant* filtre*, et, pour chaque valeur présente de leur attribut* attr*, de nouveaux nœuds au même niveau correspondant à ces différentes valeurs ; les valeurs des attributs de ces nouveaux nœuds et celles des attributs des arcs les reliant aux nœuds de base sont calculées respectivement grâce à* CalculAttNœud *et* CalculAttArc*.*

Variables : nœud v_{classe} , arc e.

Corps :

$\forall v \in V_{Niv}$:
si filtre(v) :
si (v.attr est non nul) :
$v_{classe} \leftarrow$ NœudClasse(v.att, Niv);
AjouteArcIntra(*e*, Niv, v_{classe} , v);
$\forall \langle att, f \rangle \in CalculAttNœud$:
$v_{classe}.att \leftarrow f(att, v, v_{classe})$;
$\forall \langle att, f \rangle \in CalculAttArc$:
$e.att \leftarrow f(att, v, e)$.

3.2 Création des nœuds-alignements

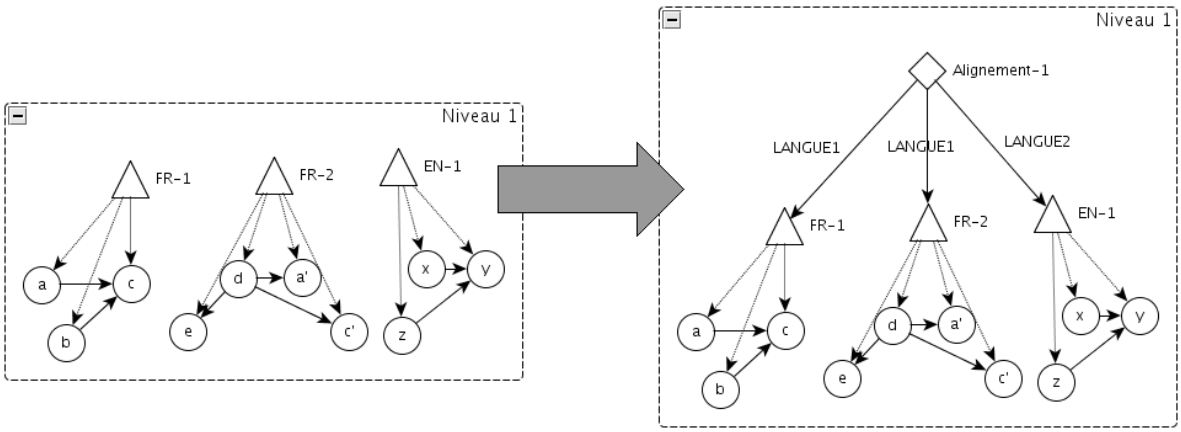


Figure 43: Création de nœuds-alignement

La mise en correspondance d'entités textuelles (il s'agit le plus souvent d'alignements de phrases) n'associe pas toujours une seule entité d'une langue à une seule entité de l'autre langue, et ne peut pas être représentée de manière standard par un arc entre deux nœuds-entités.

Nous représentons donc les correspondances entre entités textuelles issues de l'alignement comme des nœuds à part entière. L'opération qui suit est illustrée par la figure 43 : pour un alignement *<{FR-1, FR-2}, {EN-1}>* entre deux phrases françaises et une phrase anglaise, on crée un nœud-alignement, relié aux nœuds-entités français (*FR-1, FR-2*) et anglais (*EN-1*) par des arcs aux attributs distincts.

Dans cette opération, on suppose que l'alignement du corpus par entités textuelles a déjà été réalisé, c'est d'ailleurs pourquoi cet alignement est un paramètre de l'opération.

CréeNœudsAlignement

Paramètres d'entrée :

entier *Niv*,
fonction($V \rightarrow bool$) *filtre-entité*,
attribut *attr-entité*,
liste(<attribut,fonction($attribut \times V \times V \rightarrow valeur$ >) *CalculAttNœud*
liste(<attribut,fonction($attribut \times V \times E \rightarrow valeur$ *>) CalculAttArcL1, CalculAttArcL2,*
liste(<liste(chainecars), liste(chainecars)>) *ALIGN*

Définition :

Crée, à partir de ALIGN*, liste d'alignements (couples de listes d'identifiants d'entités), les nœuds matérialisant les alignements, reliés aux nœuds (du niveau* Niv *vérifiant* filtre-entité*) matérialisant les entités (identifiées par leur attribut* att-entité*) ; les valeurs des attributs de ces nouveaux nœuds et celles des attributs des arcs les reliant aux nœuds-entités (des 2 langues) sont calculées respectivement grâce à* CalculAttNœud*,* CalculAttArcL1 *et* CalculAttArcL2*.*

Variables : nœud v_{align} , v_{l1} , v_{l2} , arc e.

Corps :

$\forall \langle list1, list2 \rangle \in ALIGN$:
AjouteNoeud(v_{align} , Niv);
$\forall \langle att, f \rangle \in CalculAttNœud$:
$v_{align}.att \leftarrow f(att, v, v_{align})$;
$\forall\ id1 \in list1$:
soit $v_{l1} \in V_{Niv} \mid filtre-entité(v_{l1}) \wedge v_{l1}.att-entité = id1$:
AjouteArcIntra(e, Niv, v_{align} , v_{l1});
$\forall \langle att, f \rangle \in CalculAttArc1$: $e.att \leftarrow f(att, v_{align}, e)$;
$\forall\ id2 \in list2$:
soit $v_{l2} \in V_{Niv} \mid filtre-entité(v_{l2}) \wedge v_{l2}.att-entité = id2$:
AjouteArcIntra(e, Niv, v_{align} , v_{l2});
$\forall \langle att, f \rangle \in CalculAttArc1$: $e.att \leftarrow f(att, v_{align}, e)$.

3.3 Création de liens d'apparition conjointe

Lorsqu'on a des nœuds représentant des alignements et des entités textuelles, on peut vouloir générer les liens entre les occurrences de termes apparaissant dans des entités textuelles et les occurrences des termes apparaissant dans les entités textuelles alignées avec les premières entités. Cela permet d'obtenir des relations d'apparition conjointe entre les termes de langue différente.

CréeLiensApparitionConjointe

Paramètres d'entrée :

entier *Niv*,
fonction($V \rightarrow bool$) *estnoeudalign, estnoeudentité*.
fonction($E \rightarrow bool$) *estarcalign1*, *estarcalign2*, *estarcdansentité*,
fonction($V \times V \rightarrow bool$) *filtrecouple*,
liste(<attribut,fonction(attribut $\times V \times V \rightarrow$ *valeur >) CalculAttributs*

Définition :

Crée, pour chaque nœud-alignement, à partir des ensembles des nœuds correspondant aux termes présents respectivement dans les entités de la 1ère langue et de la seconde langue pour cet alignement, tous les liens possibles vérifiant filtrecouple *entre ces deux ensembles (dont les valeurs des attributs sont calculées grâce à* CalculAttibuts*).*

Supprime à la fin tous les nœuds et arcs correspondant à la matérialisation d'un alignement.

Variables : nœud v, arc e, ensemble(noeuds) *L1*, *L2*, *T1*, *T2*.

Corps :

$\forall v \in V_{Niv} \mid estnoeudalign(v)$:
$L1=\{v_e \mid \exists e \mid source(e)=v \wedge cible(e)=v_e \wedge estarcalign1(e)\}$;
$L2=\{v_e \mid \exists e \mid source(e)=v \wedge cible(e)=v_e \wedge estarcalign2(e)\}$;
$T1=\{v_t \mid \exists e \mid source(e)=v_e \wedge cible(e)=v_t \wedge v_t \in L1 \wedge estarcdansentité(e)\}$;
$T2=\{v_t \mid \exists e \mid source(e)=v_e \wedge cible(e)=v_t \wedge v_t \in L2 \wedge estarcdansentité(e)\}$;
$\forall v1 \in T1$:
$\forall v2 \in T2$:
si filtrecouple(v1,v2) :
AjouteArcIntra(e, Niv, v_{align} , v_{l1});
$\forall \langle att, f \rangle \in CalculAttArc1$:
$e.att \leftarrow f(att, v1, v2)$;

{Suppression des arcs et nœuds matérialisant l'alignement}

$\forall e \in E_{Niv} \mid estarcalign1(v) \vee estarcalign2(v) \vee estarcdansentité(v)$:
SupprimeArcIntra(e);
$\forall v \in V_{Niv} \mid estnoeudalign(v) \vee estnoeudentité(v)$:
SupprimeNœud(v);

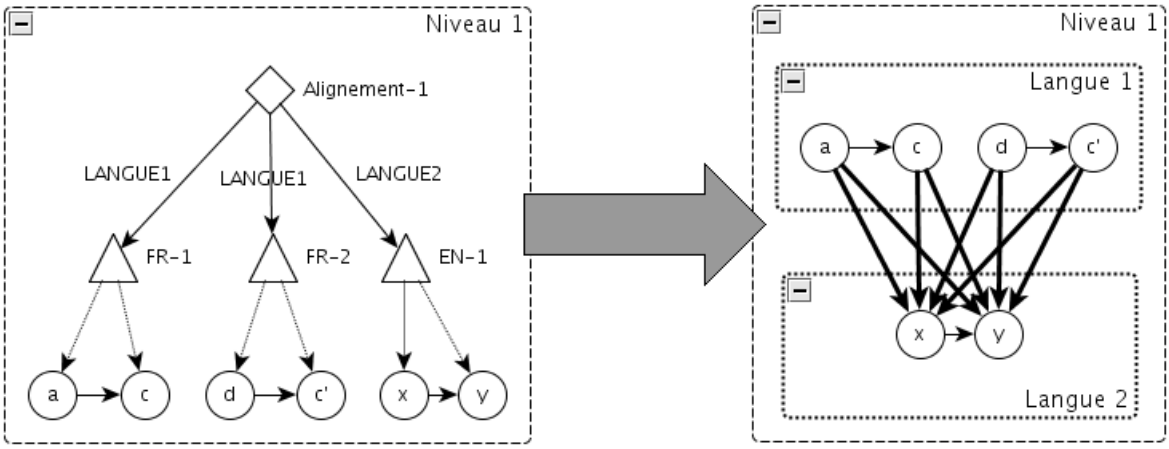


Figure 44: Création de liens d'apparition conjointe

4 Séquence d'opérations

Nous présentons maintenant comment nous utilisons notre modèle MuLLinG pour obtenir, à partir d'un corpus parallèle, les informations de pondération des traductions venant des wordnets.

Dans des expérimentations d'extraction faisant intervenir plusieurs langues, les nœuds et des arcs doivent porter un attribut de langue. Pour plus de lisibilité, nous regroupons, dans les tableaux que nous présentons dans la suite pour illustrer la séquence d'opérations, les nœuds et les arcs dans des sous-graphes de langue ; cela ne fait pas partie du modèle MuLLinG.

4.1 Identification des informations utiles

Le processus que nous souhaitons réaliser doit d'abord commencer par le traitement du graphe du corpus, de manière à ne garder que les informations intéressantes avant de le faire interagir avec celui issu du wordnet.

On doit en particulier s'assurer que les deux graphes contiendront le même type de données. Or, à l'état initial, le graphe du wordnet relie des mots alors que celui du corpus relie des occurrences de mots. Il faut donc générer, grâce à notre opération d'émergence, les mots du corpus.

En outre, le passage de la matérialisation de l'alignement de phrases à celui de liens entre les termes qui les composent étant relativement coûteux[142], il faut donc éviter au maximum de produire des liens inutiles.

[142]S'il y a n termes d'un côté de l'alignement, et m de l'autre, cela fait $n \times m$ couples possibles, dont un certain nombre seront inutiles si l'un des termes n'est pas dans le graphe issu de Wordnet.

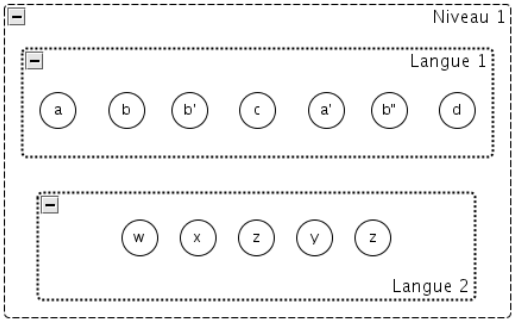
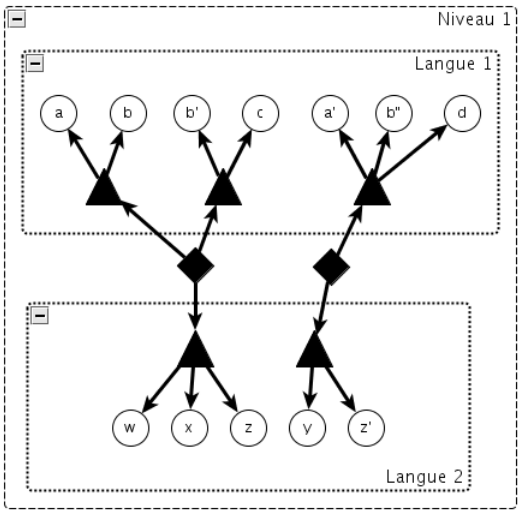
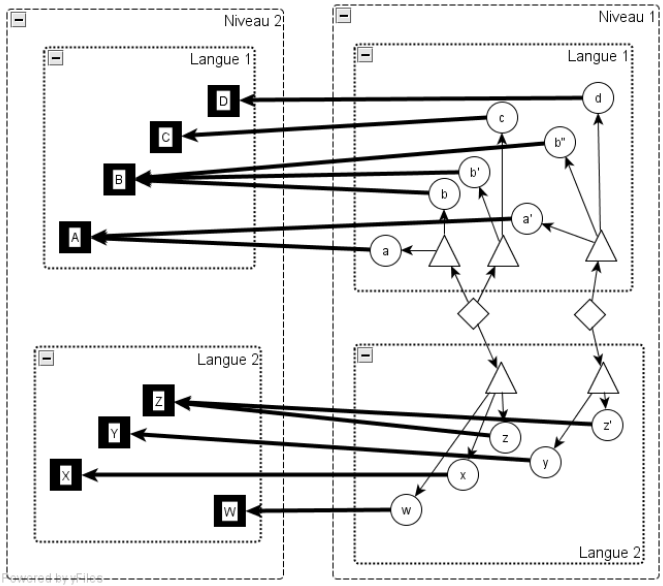
Graphe de départ (filtré)	
Matérialisation des entités et des alignements	
Émergence de nœuds	

Tableau 34: Préparation du graphe du corpus

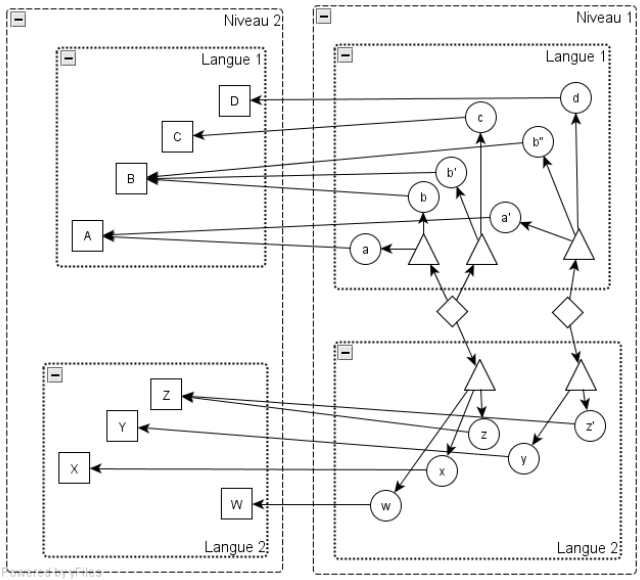
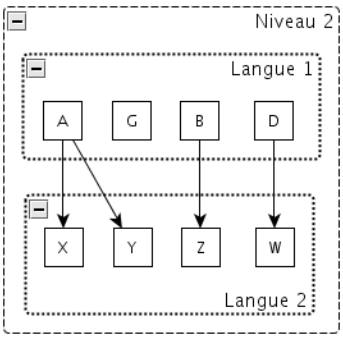
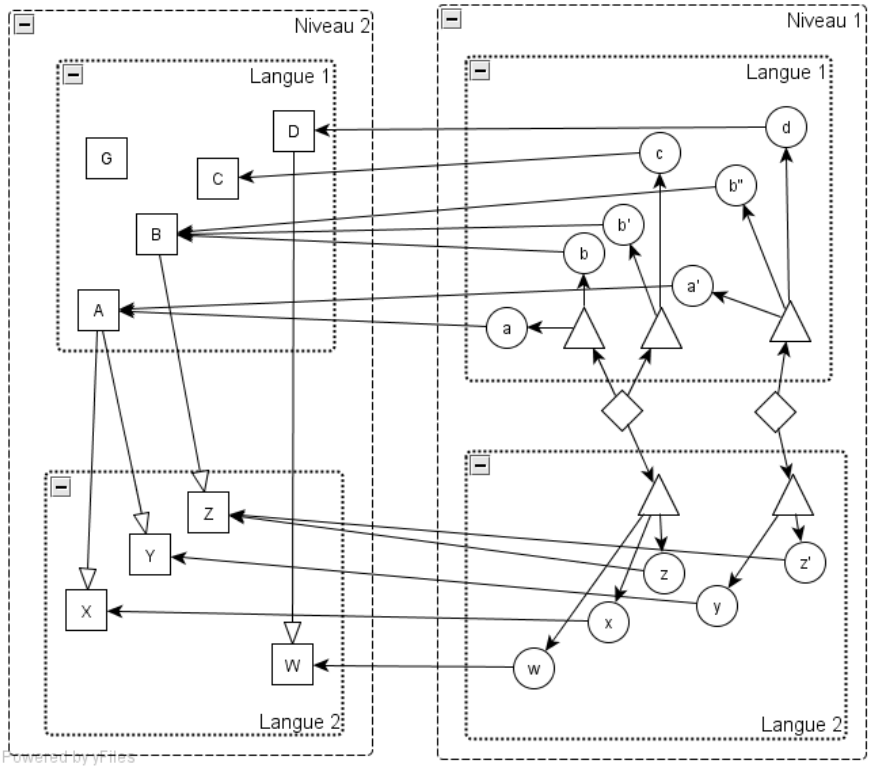
	Corpus	**Wordnet**
Graphes avant union		
Union des graphes		

Tableau 35: Union des graphes du corpus et du wordnet

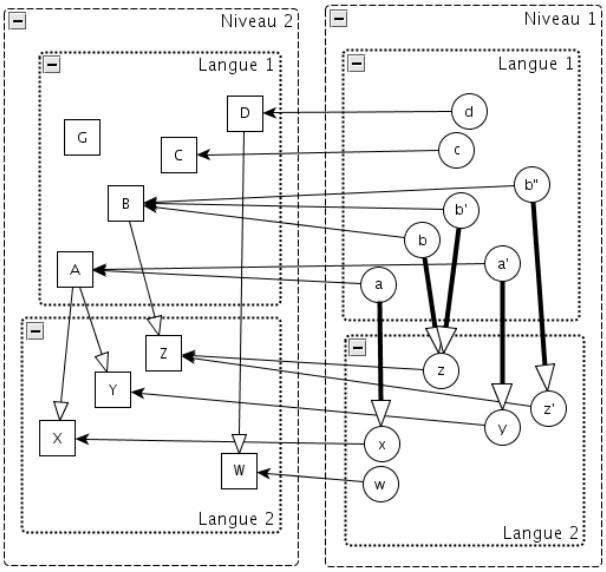
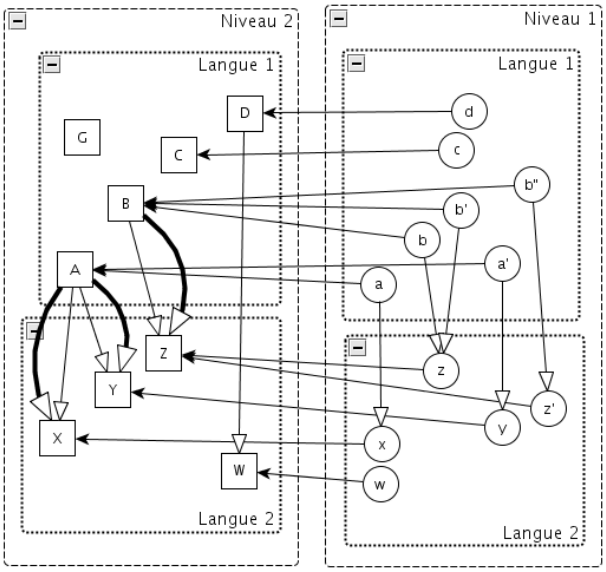
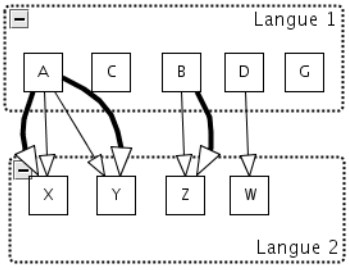
(suite du tableau 35) Création conditionnelle des liens d'apparition conjointe	
Émergence (simple) des arcs	  (vue restreinte au niveau 2)

Tableau 36: Production des liens d'apparition conjointe entre mots

Pour cela, on peut utiliser le graphe du wordnet dans un rôle de filtre (nous détaillerons comment par la suite). Nous ne générerons donc pas les liens d'apparition conjointe (dans des entités alignées) avant l'interaction entre les deux graphes.

L'identification des objets utiles se fait par les opérations suivantes, *sur le graphe du corpus* :

- *suppression des arcs* (les informations de dépendance sont inutiles dans ce processus) ;
- *génération des nœuds-entité* ;
- *génération des nœuds-alignement* ;
- *émergence des nœuds* représentant des *occurrences de mots* de manière à obtenir, au niveau supérieur, des classes d'occurrences qui correspondront aux *lemmes* du corpus.

Cette démarche est illustrée par le tableau 34.

4.2 Union des graphes

Une fois le graphe du corpus traité, on peut faire l'*union* entre ce graphe et le graphe du Wordnet. Cela permet d'obtenir un nouveau graphe qui contient à la fois :

- les informations d'alignement entre occurrences de mots, venant du corpus ;
- les informations de traduction entre mots, venant du wordnet.

Le tableau 35 présente une telle union.

4.3 Production des liens d'apparition conjointe entre mots

Pour pouvoir obtenir ce que nous souhaitons (une pondération des informations de traduction venant des wordnets d'après leur apparition conjointe dans des corpus alignés), il nous faut tout d'abord obtenir les liens d'apparition conjointe dans des entités alignées.

Il nous faut éviter de générer les liens qui nous seront inutiles, c'est-à-dire ceux ne correspondant pas à une information de traduction d'après les wordnets. Nous utilisons donc l'opération de *création de liens d'apparition conjointe*, avec la condition, pour qu'un lien entre occurrences de mots soit créé, que leurs classes d'équivalence respectives soient reliées (c'est à dire que les mots soient traductions l'un de l'autre d'après le wordnet).

On peut alors effectuer l'*émergence d'arcs* pour obtenir les relations d'apparition conjointe entre mots, puis effectuer le *calcul d'associabilité*.

Cette partie du processus est illustrée par le tableau 36.

4.4 Mise en œuvre

Nous détaillons ici la manière de traduire le processus de pondération des liens de traduction sous forme d'opérations du modèle MuLLinG.

	Graphe du corpus	**Graphe du wordnet**
Identification des informations utiles	Filtrage des arcs (on les supprime tous) Filtrage des nœuds (on garde ceux qui ont la bonne partie du discours) Matérialisation des entités textuelles Matérialisation des alignements Émergence des nœuds	(rien)
Interaction entre les informations	Union entre le graphe du corpus et le graphe du wordnet	
Identification des liens d'association bilingue	Création conditionnelle des liens d'apparition conjointe (si les classes des nœuds sont elle-mêmes reliées)	
Production	Émergence des arcs	
Classement	Calcul de mesure simple	

Tableau 37: Récapitulatif des opérations nécessaires pour la production d'informations de traduction pondérées, à partir d'un wordnet et de corpus bilingues.

4.4.1 Opérations d'identification (sur le graphe du corpus)

On effectue le filtrage des arcs grâce à *SupprimeArcsSi*, avec comme paramètre :

- *f_vrai(e)* : vrai quel que soit *e*.

On effectue le filtrage des nœuds grâce à *SupprimeNœudsSi*, avec comme paramètre :

- *f_nœud(v)* : vrai si *v* a la partie du discours recherchée.

On matérialise ensuite les entités textuelles (en l'occurrence les phrases) à l'aide de l'opération *CréeNœudsSelonAttribut*, avec comme paramètres :

- niveau : *1* ;
- filtre : *f_vrai(v)* : vrai quel que soit *v* ;
- attribut sur lequel l'opération se base : celui qui indique l'identifiant de la phrase ;
- calcul des attributs :
  - sur les matérialisations des phrases (nœuds) :
    - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "sentence" ;
    - pour l'attribut indiquant l'identifiant de la phrase : <*idphrase, CopieId*>, avec *CopieId*(att,v1,v2) → *v2.idphrase* (copie de l'identifiant indiqué par les nœuds formant la phrase) ;
  - sur les arcs reliant les nœuds représentant les phrases aux nœuds qui composent la phrase :
    - pour l'attribut type : <*type, NouvTypeArc*>, avec *NouvTypeArc*(att,v,e) → "sentence".

La matérialisation des alignements se fait, elle, en utilisant l'opération *CréeNœudsAlignement*, avec comme paramètres :

- niveau : *1* ;
- filtre : *f_ent(v)* : vrai si le type de *v* est "sentence";
- attribut sur lequel l'opération se base : celui qui indique l'identifiant de la phrase ;
- calcul des attributs :
  - sur les matérialisations des alignements (nœuds) : *CAp*, liste de couples :
    - pour l'attribut type : <*type, NouvTypeAlign*>, avec *NouvTypeAlign*(att,v1,v2) → "align" ;
  - sur les arcs reliant les nœuds matérialisant les alignements à ceux matérialisant les phrases :
    - pour les phrases de langue 1 :
      - pour l'attribut type : <*type, NouvTypeL1*>, avec *NouvTypeL1*(att,v,e) → "align1" ;
    - pour les phrases de langue 2 :
      - pour l'attribut type : <*type, NouvTypeL2*>, avec *NouvTypeL2*(att,v,e) → "align2" ;
- liste d'alignements (liste de couples de listes d'identifiants d'entités) : généré à partir d'un fichier d'alignement.

On produit les classes de termes en utilisant *ÉmergenceDeNœuds*, avec comme paramètres :

- niveau : *1* ;
- filtre : *f_emergnoeud(v)* vrai si *v* représente une occurrence de termes (si *v.type* a le type correspondant) ;
- classe d'équivalence : *CEN*, renvoie la valeur de l'attribut *lemme* ;
- calcul des attributs : *CAN*, liste de couples :
  - pour l'attribut type : <*type, NouvType*>, avec *NouvType*(att,v1,v2) → "classedenœuds"
  - pour les autres attributs *att* : <*att, Copie*>.

4.4.2 Interaction entre les informations (union)

On effectue l'union des deux graphes grâce à l'opérateur *Union*, prenant comme paramètres :

- graphes : le graphe issu du corpus (après les traitements précédents et celui issu de Wordnet) ;
- fonctions d'identité :
  - sur les nœuds : *f_idv(v1,v2)* vrai si les deux nœuds ont le même identifiant ;
  - sur les arcs[143] : *f_ide(e1,e2)* vrai si les arcs ont des sources identiques et des cibles identiques (d'après la fonction d'identité sur les nœuds) et que les types de relation sont les mêmes ;
- fonctions de copie :

[143]Cette fonction est inutile dans notre processus, les deux graphes n'ayant pas d'arc en commun.

  - sur les nœuds : $cop-v(v)\rightarrow v_{cop}$ copie tous les attributs de v dans v_{cop} ;
  - sur les arcs : $cop-e(e)\rightarrow e_{cop}$ copie tous les attributs de e dans e_{cop} ;
- fonctions de fusion :
  - sur les nœuds : $fus-v(v_{1,}v_2)\rightarrow v_{cop}$ copie tous les attributs de v_1 dans v_{cop} (seules les informations, notamment statistiques, venant du graphe du corpus nous intéressent) ;
  - sur les arcs : $fus-e(e_{1,}e_2)\rightarrow e_{cop}$ quelconque (par exemple, copie tous les attributs de e_2 dans e_{cop}) : elle n'est jamais appelée.

4.4.3 Identification des liens d'association bilingue

Après avoir réalisé l'union des deux graphes, on crée les liens d'apparition conjointe (entre occurrences de termes) à partir de l'alignement, en se basant sur les liens entre termes issus de l'union. On utilise pour cela *CréeLiensApparitionConjointe*, avec comme paramètres :

- niveau : *2* ;
- fonction d'identification des objets nécessaires :
  - pour les nœuds-alignement : *f_alignv(v)* vrai si *v.type="align"* ;
  - pour les nœuds-entité : *f_entv(v)* vrai si *v.type="sentence"* ;
  - pour les arcs entre nœuds-entité et nœuds de base : *f_ente(e)* vrai si *e.type="sentence"* ;
  - pour les arcs entre nœuds-alignement et nœuds-entité : *f_align1(e)* et *f_align2(e)* , vrais respectivement si *e.type="align1"* et si *e.type="align2"* ;
- filtre : *f_couple(v1,v2)* vrai s'il y a un lien au niveau 1 entre les nœuds matérialisant les classes respectives de *v1* et de *v2* ;
- calcul des attributs : *CAlac*, liste de couples :
  - pour l'attribut type : <*type, NouvTypeApConj*>, avec *NouvTypeApConj*(att,v) → "trad".

4.4.4 Production

La production des liens entre termes d'après l'apparition conjointe s'effectue à travers une émergence d'arcs, utilisant *ÉmergenceDArcs*, avec comme paramètres :

- niveau : *2* ;
- filtre : *f_vrai(e)* vrai quel que soit si *e* ;
- classe d'équivalence : *CEB*, renvoie le type de la relation portée par l'arc ;
- calcul des attributs : *CAB*, liste de couples :
  - pour l'attribut type : <*type, NouvTypeCT*>, avec *NouvTypeCT*(att,v1,v2) → "classetrad" ;
  - pour les autres attributs *att* : <*att, Copie*>.

4.4.5 Classement

Le calcul de la mesure d'associabilité (bilingue) grâce à *CalculeF12*, avec comme paramètres :

- attribut : *mesure* ;
- niveau : *2* ;
- filtre : *f_classebicoll(e)* : vrai si *e.type* ="classetrad" ;
- relation : celle étudiée ;
- fonction : la mesure d'associabilité souhaitée.

5 Expérimentations

Pour réaliser nos expérimentations, nous avons utilisé :

- le corpus français-anglais *EuroParl* pour la production de liens d'apparition conjointe ;
- des réseaux de mots, pour les liens de traduction de référence :
  - *EuroWordNet* (volume français), avec PWN 1.5 ;
  - *Wolf*, avec PWN 2.0.

Nous faisons ici la supposition que les liens d'association bilingue issus du corpus doivent concerner des termes ayant la même partie du discours. En effet, les ressources étant tirées de PWN, où la partie du discours est un élément discriminant des synsets, elles ne peuvent relier entre eux des termes qui n'ont pas la même partie du discours.

Nous avons mené des expérimentations sur :

- les *verbes* (avec EWN, avec Wolf) ;
- les *noms* (avec EWN, avec Wolf) ;
- les *adjectifs* (avec Wolf) ;
- les *adverbes* (avec Wolf).

Pour chaque expérimentation, deux mesures d'associabilité ont été calculées pour obtenir les « poids » de chaque traduction : *Information mutuelle* et *WMI*.

5.1 Préparation des données

5.1.1 Corpus

Le graphe modélisant le corpus et les relations qu'il contient sera filtré pour ne conserver que les objets utiles, en l'occurrence les nœuds représentant les mots (simples ou composés) ayant la partie du discours choisie, à partir desquels on créera les nœuds représentant les phrases et les alignements, puis les liens d'association bilingue.

Il est à noter que, de par la nature même du corpus (débats entre députés européens s'exprimant dans leur propre langue), la langue d'origine des textes est très variable.

En raison de contraintes techniques (taille de la mémoire, temps de calcul), nous avons utilisé seulement une partie du corpus EuroParl dans nos expérimentations. Le détail des informations sur l'emploi d'EuroParl est donné dans le tableau 38.

Expérimentations		**Noms**	**Verbes**	**Adjectifs**	**Adverbes**
Corpus (années d'EuroParl)		2003	2003	2003 à 2004	2003 à 2005
Documents alignés		61	61	111	168
Alignements de phrases		136 477	136 477	232 083	353 736
Français	*Phrases*	141 466	141 466	239 654	365 373
	Lemmes	8 344	2 968	4 904	931
	Occurrences	849 203	582 085	777 136	590 303
Anglais	*Phrases*	141 818	141 818	240 640	366 345
	Lemmes	13 143	3 855	8 421	1700
	Occurrences	897 153	639 979	736 651	488 692

Tableau 38: Statistiques sur le corpus utilisé

Dans chaque expérimentation, le nombre de lemmes distincts en français est nettement inférieur au nombre de lemmes distincts en anglais, alors que les nombres totaux d'occurrences sont proches. Cela semble s'expliquer par une moins bonne qualité du lemmatiseur employé pour l'anglais.

5.1.2 Wordnet

L'import des données issues du wordnet est plus compliqué puisque, comme nous l'avons dit précédemment, il n'y a pas de liens de traduction directement exploitables entre mots : EuroWordNet et Wolf ne contiennent que des liens de traduction entre synsets[144].

On pourrait, dans notre modèle, générer les liens entre « sens de mots » composant les synsets à partir des nœuds-« synsets », des liens entre tous les couples possibles de nœuds-« sens de mots » de langues différentes.

Cependant, une telle opération dépendrait beaucoup de la tâche et de la structure du graphe de départ, et serait donc peu générique ; son développement n'est donc pas une priorité. Nous préférons travailler ici directement avec les graphes contenant les liens bilingues de traduction entre mots du réseau, en considérant que cette étape d'import au bon format a été faite.

[144]Chaque synset français contient une information indiquant à quel synset de PWN il correspond.

Wordnet utilisé	**Noms**		**Verbes**		**Adjectifs**	**Adverbes**
	Wolf	**EWN**	**Wolf**	**EWN**	**Wolf**	**Wolf**
Termes français	39 335	24 178	2 430	8 282	1 959	941
Termes anglais	59 518	34 420	4 025	13 146	3 260	1 365
Couples de traduction	87 103	51 244	6 857	24 155	4 887	2 146

Tableau 39: Statistiques sur les traductions issues des wordnets

Nous présentons dans le tableau 39, outre les nombres de termes anglais et français, les nombre de couples de traduction générés à partir des synsets de PWN et de leurs équivalents en français dans Wolf ou EuroWordNet.

5.2 Résultats

Le tableau 40 présente le nombre d'arcs et de nœuds produits lors des différentes expérimentations. On peut constater que, aussi bien pour les noms que pour les verbes, EWN permet d'obtenir plus de nœuds et d'arcs de niveau 2, c'est-à-dire de termes et de relations de traductions (pondérées) entre eux.

On peut également remarquer que les adjectifs et les adverbes produisent bien moins de relations entre termes (arcs de niveau 2) que les verbes et surtout les noms, bien que les nombres d'occurrences utilisées pour les produire soient du même ordre de grandeur.

Expérimentations		*Niveau 1*		*Niveau 2*	
		Nœuds	*Arcs*	*Nœuds*	*Arcs*
Noms	**Wolf**	1 746 356	45 694	2 478	2 427
	EWN	1 746 356	45 356	2 866	2 735
Verbes	**Wolf**	1 222 064	317 480	1 068	1 528
	EWN	1 222 064	166 249	1 376	1 846
Adjectifs	**Wolf**	1 513 787	12 215	685	525
Adverbes	**Wolf**	1 078 995	40 872	378	385

Tableau 40: Arcs et nœuds des graphes linguistiques utilisés pour les expérimentations de pondération de liens de traduction

Nous présentons dans la suite plusieurs tableaux illustrant les résultats de nos expérimentations, dans lesquels, pour un terme français donné, on montre les termes anglais avec lesquels il est en relation de traduction, classés d'après leur associabilité selon les mesures utilisées.

Pour plus de lisibilité, les valeurs obtenues de MI (information mutuelle) et de WMI sont multipliées respectivement par 10 et 10 000.

Qu'on utilise Wolf ou EWN pour générer les traductions, l'information mutuelle d'un couple de termes (par exemple « résultat »-« outcome » dans le tableau 42) reste identique. En effet, si on utilise le même corpus, on obtiendra les mêmes nombres d'occurrences de termes, et les relations entre ces occurrences (si elles sont générées grâce au wordnet français employé) apparaîtront aussi souvent dans un cas que dans l'autre.

Au contraire, la valeur de WMI dépend du wordnet utilisé. En effet, cette mesure se base entre autres[145], pour une relation *r* entre des termes *a* et *b*, sur le nombre de fois qu'une occurrence de *a* est le premier argument (et sur le nombre de fois qu'une occurrence de *b* est le second argument) d'une relation de type *r*, ce qui dépend directement des relations de traduction produites par le processus, et donc du wordnet employé pour cette production.

On ne pourrait classer objectivement les traductions d'un terme, décider que l'une est meilleure que l'autre, puisque le choix de la traduction à utiliser dans un cas précis dépend du contexte. C'est pourquoi il n'est pas possible de réaliser une comparaison des mesures utilisées à travers les classements qu'elles ont produits.

Une telle expérience permet d'obtenir des données de traduction classées selon le domaine du corpus. De telles informations peuvent guider les processus de traductions naïfs utilisés en recherche d'information multilingue.

Rapport							
Avec Wolf				**Avec EuroWordNet**			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Report	15,10	Report	129,64	Theme	15,66	Report	126,43
Account	9,81	Account	13,02	Report	15,10	Account	11,62
Study	8,71	Relation	10,55	Account	9,81	Relation	9,98
Story	7,03	Study	4,14	Study	8,71	Statement	4,46
Relationship	5,05	Relationship	2,11	Connection	7,71	Study	3,63
Relevance	4,18	Story	0,55	Story	7,03	Connection	2,84
Relation	0,39	Relevance	0,25	Paper	6,29	Argument	2,06
				Composition	5,62	Relationship	1,99
				Statement	5,60	Link	1,41
				Relationship	5,05	Theme	1,34
				Relevance	4,18	Paper	1,25
				Link	3,21	Composition	1,13
				Relation	0,39	Story	0,55
				Argument	-1,17	Relevance	0,24

Tableau 41: Traductions du nom « rapport » (d'après les wordnets) classées d'après l'information mutuelle et WMI.

[145]La mesure a été présentée en détail page 80.

La qualité des résultats produits étant fortement dépendantes du corpus utilisés, on ne peut faire de généralisation sur ce qui permet d'obtenir les meilleurs résultats.

Effet							
Avec Wolf				**Avec EuroWordNet**			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Effect	14,13	Issue	8,56	Effect	14,13	Result	8,50
Result	10,33	Result	6,81	Result	10,33	Effect	7,96
Event	9,53	Effect	6,51	Consequence	9,16	Consequence	2,94
Consequence	9,16	End	3,08	Outcome	7,62	Outcome	1,53
Purpose	7,97	Event	2,42	Side	1,23	Side	1,04
Outcome	7,62	Consequence	2,39				
Aim	7,28	Aim	1,71				
Goal	6,71	Purpose	1,41				
Issue	6,51	Outcome	1,22				
End	4,74	Goal	1,06				
Intention	-3,04	Intention	0,87				
Impression	-13,38	Impression	0,13				
Résultat							
Avec Wolf				**Avec EuroWordNet**			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Outcome	17,37	Result	9,34	Outcome	17,37	Result	11,42
Result	15,84	Issue	5,60	Result	15,84	Outcome	3,39
Gain	13,49	Solution	3,03	Effect	8,82	Effect	2,90
Event	10,69	Outcome	2,82	Finding	8,29	Consequence	0,68
Effect	8,82	Resolution	2,79	Consequence	0,31	Finding	0,22
Finding	8,29	Effect	2,30				
Resolution	7,81	Event	1,92				
Issue	6,31	Answer	0,62				
Answer	3,64	Gain	0,56				
Solution	1,44	Consequence	0,52				
Consequence	0,31	Profit	0,45				
Profit	-3,37	Finding	0,16				

Tableau 42: Traductions des noms « effet » et « résultat » (d'après les wordnets) classées d'après l'information mutuelle et WMI.

Aller							
Avec Wolf				**Avec EuroWordNet**			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Go	11,20	Go	30,69	Go	11,20	Do	47,74
Function	9,89	Work	14,24	Range	11,03	Go	16,89
Work	7,48	Move	3,73	Lead	8,00	Come	10,68
Operate	6,61	Operate	2,35	Do	7,12	Lead	6,33
Travel	6,05	Run	1,96	Come	6,87	Move	2,43
Run	5,97	Travel	1,24	Accomodate	6,37	Run	1,14
Move	4,79	Function	0,64	Travel	6,05	Proceed	1,02
Fit	3,21	Fit	0,23	Proceed	6,03	Travel	0,96
				Run	5,97	Range	0,20
				Move	4,79	Fit	0,17
				Suit	4,28	Suit	0,17
				Fit	3,21	Accomodate	0,13
Mettre							
Avec Wolf				**Avec EuroWordNet**			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Nominate	18,02	Make	32,17	Place	11,07	Have	76,54
Release	12,12	Put	19,23	Put	8,70	Put	12,80
Threaten	12,05	Set	12,35	Lay	7,69	Set	8,67
Name	11,44	Establish	9,23	Set	7,49	Apply	4,21
Place	11,07	Lay	4,85	Have	6,57	Let	3,63
Update	9,50	Place	4,29	Get	6,18	Lay	3,08
Establish	8,85	Threaten	2,54	Let	5,41	Place	2,65
Put	8,70	Release	1,50	Apply	2,03	Get	2,44
Lay	7,69	Update	1,09	Pose	-3,85	Pose	0,20
Jeopardise	7,58	Name	0,89				
Set	7,49	Jeopardise	0,86				
Make	6,86	Nominate	0,42				
Kill	-3,53	Pose	0,36				
Pose	-3,85	Kill	0,33				

Tableau 43: Traductions des verbes « aller » et « mettre » (d'après les wordnets) classées d'après l'information mutuelle et WMI.

Autre				*Important*			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Other	3,67	Other	32,23	Significant	-4,01	Important	22,62
Elaborate	0,41	Different	5,34	Considerable	-4,28	Significant	2,25
Different	0,40	Detailed	0,72	Major	-5,23	Major	2,05
Detailed	-1,24	Elaborate	0,04	Important	-7,61	Crucial	1,87
Spare	-3,38	Spare	0,04	Crucial	-11,74	Considerable	0,99

Tableau 44: Traductions des adjectifs « autre » et « important » (d'après les wordnets) classées d'après l'information mutuelle et WMI.

Aussi				*Toutefois*			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Likewise	7,84	Also	58,86	Notwithstanding	11,67	However	20,64
Besides	7,64	As	25,94	However	7,63	Still	4,88
Also	4,79	Too	11,79	Nonetheless	5,22	Yet	2,27
Too	4,45	Equally	1,43	Nevertheless	2,77	Nevertheless	1,55
As	3,78	Likewise	0,49	Yet	2,69	Nonetheless	0,48
Equally	3,37	Besides	0,17	Still	2,36	Notwithstanding	0,07
Entièrement				*Uniquement*			
MI (x10)		**WMI** (x10000)		**MI** (x10)		**WMI** (x10000)	
Wholly	20,80	Fully	1,37	Solely	17,40	Only	4,63
Altogether	16,08	Entirely	0,56	Exclusively	13,33	Just	1,77
Entirely	14,78	Completely	0,42	Alone	10,75	Solely	0,70
Completely	8,98	All	0,40	Only	6,84	Alone	0,68
Fully	8,81	Wholly	0,21	Just	3,83	Simply	0,41
All	1,53	Altogether	0,09	Merely	0,19	Exclusively	0,26
Totally	0,37	Totally	0,07	Simply	0,17	Merely	0,16
				Entirely	-0,14	Entirely	0,12

Tableau 45: Traductions des adverbes « aussi », « toutefois », « entièrement » et « uniquement » (d'après les wordnets) classées d'après l'information mutuelle et WMI.

5.3 Observations

On peut tout d'abord observer le bruit causé par les auxiliaires « do » et « have ») dans les expérimentations sur les verbes (tableau 43). Cela est dû à l'analyseur syntaxique utilisé, qui ne distingue pas les auxiliaires des verbes « normaux » : puisqu'on emploie souvent les auxiliaires, on les retrouve fréquemment dans des phrases alignées avec celles comportant le verbe considéré, ce qui explique les fortes valeurs d'associabilité obtenues d'après les mesures utilisées. Cela se remarque d'ailleurs surtout quand on emploie WMI, mesure destinée à favoriser les couples les plus fréquents. Ce problème n'apparaît bien sûr que dans le cas où les verbes auxiliaires sont considérés comme traductions d'après les informations issues des wordnets.

On retrouve le défaut habituel de la mesure d'information mutuelle, qui favorise les couples mettant en jeu des termes plus rares. Ainsi, dans l'expérimentation sur les verbes utilisant Wolf (tableau 43), la traduction de « mettre » qui domine nettement les autres est « nominate » ; de même, la première traduction renvoyée pour « toutefois » est « notwithstanding » (tableau 45). Ces traductions rares sont au contraire reléguées en fin de classement avec la mesure WMI.

WMI permet en effet de distinguer plus nettement les traductions les plus fréquentes : par exemple, le couple « rapport »-« report » a un poids 10 fois plus fort que « rapport »-« account » avec WMI, alors qu'ils étaient proches avec MI (tableau 41).

WMI permet même de faire remonter ces traductions les plus fréquentes à la première place alors qu'elles étaient mal classées avec l'information mutuelle, comme « mettre »-« put » (tableau 43) ou « aussi »-« also » (tableau 45).

Il faut enfin remarquer que, faute d'analyse sémantique du corpus suffisante, les expérimentations n'ont pu utiliser directement les « sens de mots » présents dans les wordnets, n'en déduisant que des relations de traduction entre mots. Par exemple, le mot « rapport » apparaît dans 7 synsets de Wolf et 13 de EWN, avec des significations telles que « relation », « compte rendu », « étude », « quotient », « argumentaire ».

On ne peut, dans les classements produits pour « rapport » (tableau 41), distinguer un sens de traduction qui serait plus présent qu'un autre dans le corpus ; quand bien même on pourrait le faire, cela devrait être utilisé très prudemment : si cela peut convenir pour certaines tâches, cela pourrait causer des erreurs dans d'autres (comme la traduction). En effet, ce n'est pas parce qu'un sens est souvent employé qu'il faut le privilégier : il faut se baser avant tout sur le contexte pour désambiguïser.

Conclusion

Dans ce chapitre, nous avons présenté des expérimentations de pondération de liens de traduction, à partir de différentes sources d'information (corpus parallèles, wordnets). Nous avons introduit pour cela plusieurs opérations spécifiques au traitement de l'alignement (dans des corpus parallèles). Nous avons ensuite décrit le processus menant au résultat souhaité, en

utilisant, outre ces opérations spécifiques, un certain nombre d'opérations de base de notre modèle, telles que l'union de graphes.

Ces expérimentations montrent que notre modèle n'est pas seulement utile pour de simples tâches d'extraction, mais qu'il permet également de combiner les informations linguistiques de différentes sources (ici un graphe contenant des relations de co-occurrence entre occurrences de termes dans un corpus, et un autre graphe contenant des relations de traduction entre termes venant d'un wordnet). Cela confirme la généricité des opérations de notre modèle, puisqu'elles permettent la gestion d'informations qui ne sont pas du même type et ne proviennent pas de la même source.

Conclusion

Nous avons présenté dans cette thèse une modélisation par graphes linguistiques (multiniveau), et un outil informatique générique pour résoudre les problèmes concernant l'extraction de connaissances linguistiques, basé sur la manipulation des graphes représentant ces données.

Pour cela, nous avons d'abord étudié un problème particulier (celui des collocations) et les besoins que présente leur extraction, à travers plusieurs expérimentations, monolingues et bilingues. Nous en avons déduit un cahier des charges que doit respecter un outil pour l'extraction : il doit gérer divers types de ressources (et ne doit donc pas être spécifique à l'une d'elles) et avoir une représentation de l'information simple et facilement compréhensible, aussi bien par l'humain que par la machine ; il doit également permettre de réaliser un processus à partir d'opérations simples et génériques (indépendantes du type de ressource).

À partir des contraintes fixées, nous avons défini *MuLLinG*, un modèle de graphes linguistiques multiniveau, dont les nœuds et les arcs sont respectivement les objets linguistiques et les relations entre eux ; les différents niveaux sont utilisés afin de représenter plusieurs vues de l'information.

Nous avons également introduit un ensemble d'opérations associées à ce modèle, écrites de manière à être les plus génériques possibles. Il s'agit de permettre de voir un processus d'extraction d'information comme une succession de tâches simples de manipulation de graphes.

Nous avons également décrit une version « complexe » du modèle qui permet de gérer les cas où les relations ne sont pas représentées par des arcs mais par des nœuds, comme lorsqu'on doit représenter des relations n-aires.

Nous avons ensuite mis en œuvre le modèle et les opérations sous la forme d'une bibliothèque C++, puis réalisé avec celle-ci plusieurs expérimentations, sur l'extraction de collocations et sur la pondération de liens de traduction, pour démontrer l'utilité de notre modèle, et en particulier son aspect générique.

Les travaux que nous avons présentés présentent de nombreuses perspectives d'amélioration.

La première concerne la bibliothèque C++ mettant en œuvre le modèle et ses opérations : nous avons avant tout cherché à obtenir un prototype fonctionnel, mais celui-ci reste trop compliqué à employer pour un linguiste avec des connaissances limitées en informatique. Il faut donc rendre les fonctions plus simples à comprendre, les messages d'erreur plus clairs. Il est également nécessaire de s'affranchir des hypothèses que nous avons faites pour simplifier la mise en œuvre.

Un premier public à viser est celui des informaticiens effectuant des tâches linguistiques et à la recherche d'outils informatiques facilitant leur programmation. Ensuite, il faudra rendre cette bibliothèque réellement accessible à des personnes peu initiées à la programmation. Pour cela, il faudra réaliser une couche supplémentaire, qui permette à l'utilisateur de spécifier simplement les opérations et les paramètres du processus, et fasse appel aux fonctions correspondantes de la bibliothèque.

Cela peut se faire à travers un langage spécialisé pour la programmation linguistique (LSPL), de plus haut niveau, avec des primitives simples qui correspondraient aux opérations plus complexes du modèle ; on peut envisager l'emploi d'un langage de requête. On pourrait éventuellement développer un LPSL encore plus appropriable par des non-informaticiens, qui soit un langage « narratif » (Bellynck, 1999). Il faut aussi développer une interface graphique à manipulation directe associée au modèle, fût-ce dans une version simplifiée.

Plusieurs problèmes concernent plus directement la programmation de la bibliothèque : l'outil fourni devra s'attacher à faciliter l'import des données au format souhaité, permettre une spécification des tâches plus simple (en définissant efficacement des opérateurs plus complexes) et rendre plus rapide le déroulement des processus (en optimisant la gestion de la mémoire).

Il nous faut également étendre le modèle MuLLinG et l'appliquer à d'autres tâches, tant pour prouver son intérêt que pour l'améliorer. En effet, si le modèle a été écrit de manière à être générique (et à s'affranchir de la tâche de départ qui concernait les collocations), il reste assez lié au problème de l'extraction.

Afin de prendre du recul, nous devons donc nous pencher sur la manière de perfectionner le modèle, en premier lieu pour permettre de considérer encore bien davantage de processus basés sur l'information linguistique avec notre modèle de graphes multiniveau..

La représentation de l'information (sous forme de graphes) et l'extraction de connaissances pertinentes à partir de celle-ci n'étant pas spécifiques à la linguistique, nous pouvons également envisager d'utiliser notre modèle dans d'autres domaines, notamment la recherche d'information de différents types (texte, image, vidéo, etc.).

Bibliographie

(Agosti & Crestani, 1993) **Maristella Agosti & Fabio Crestani** (1993). A Methodology for the Automatic Construction of a Hypertext for Information Retrieval. In *Proceedings of the 1993 ACM Symposium on Applied Computing*. pp. 745-753.

(Al-Adhaileh & Tang, 2002) **Mosleh Al-Adhaileh & Enya Kong Tang** (2002). Synchronous Structured String-Tree Correspondence (S-SSTC). In *Proceedings of the 20th IASTED02 International Conference*, Innsbruck. 6 p.

(Andreewsky et al., 1975) **Alexandre Andreewsky, Françoise Combrisson & Christian Fluhr** (1975). *Le problème de l'identification automatique de concepts*. Rapport d'études : Note CEA-N-1816.

(Andreewsky et al., 1977) **Alexandre Andreewsky, Fathi Debili & Christian Fluhr** (1977). Computational Learning of Semantic Lexical Relations for the Generation and Automatical Analysis of Content. In *Proceedings of IFIP Congress 1977*, Toronto. pp. 667-672

(Andreewsky et al., 1982) **Alexandre Andreewsky, M. Desi & Christian Fluhr** (1982). Méthodes d'apprentissage pour l'analyse automatique morphosyntaxique et lexicale-sémantique de la langue espagnole. In *Proceedings of COLING 1982, Abstracts*. pp. 11-16

(Austin, 1962) **John Langshaw Austin** (1962). *How to do things with words*. Oxford University Press. 192 p.

(Baldwin et al., 2003) **Timothy Baldwin, Colin Bannard, Takaaki Tanaka & Dominic Widdows** (2003). An empirical model of multiword expression decomposability. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*. pp. 89–96.

(Bellynck, 1999) **Valérie Bellynck** (1999). *Introduction d'une vue textuelle synchronisée avec la vue géométrique primaire dans le logiciel Cabri-II*, Thèse de doctorat, Université Joseph Fourier, Grenoble. 250 p.

(Berge, 1987) **Claude Berge** (1987). *Hypergraphes. Combinatoires des ensembles finis*. Gauthier-Villars. 240 p.

(Blanchon et al., 2006) **Hervé Blanchon, Christian Boitet & Ali Choumane** (2006). Traduction automatisée fondée sur le dialogue et documents auto-explicatifs : bilan du projet LIDIA. *Traitement Automatique des Langues (TAL), 47:3*. pp. 175-204

(Brandes et al., 2001) **Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt & M. Scott Marshall** (2001). GraphML Progress Report - Structural Layer Proposal. In *Proceedings of 9th International Symposium Graph Drawing (GD'01)*, Vienne (Autriche), Springer-Verlag. pp. 501-512.

(Bray et al., 1998) **Tim Bray, Jean Paoli & C.M. Sperberg-McQueen** (1998). *Extended Markup language (XML) 1.0*. W3C Recommendation.

(Bobrow & Winograd, 1977) **Daniel G. Bobrow & Terry A. Winograd** (1977). An overview of KRL, a Knowledge Representation Language. *Cognitive Science, 1:1*.

(Boguslavsky et al., 2000) **Igor Boguslavsky, Nadezhda Frid, Leonid Iomdin, Leonid Kriedkin, Irina Sagalova & Victor Sizov** (2000). Creating a Universal Networking Language module within an advanced NLP system. In *Proceedings of the 18th International conference on Computational Linguistics (COLING-2000),* Saarbrücken. Morgan Kaufmann. pp. 83-89.

(Boguslavsky et al., 2005) **Igor Boguslavsky, Jesús Cardeñosa, Carolina Gallardo & Luis Iraola** (2005). The UNL Initiative: An Overview. *Lecture Notes in Computer Science, 3406,* Springer. pp. 377-387

(Boitet & Zaharin, 1988) **Christian Boitet & Yusoff Zaharin** (1988). Representation trees and string-tree correspondences. In *Proceedings of COLING-88*, Budapest. pp. 59-64.

(Boitet et al., 2002) **Christian Boitet, Mathieu Mangeot-Lerebours & Gilles Sérasset** (2002). The PAPILLON project: cooperatively building a multilingual lexical data-base to derive open source dictionaries & lexicons. In *Proceedings of COLING Worshop on NLP and XML (NLPXML-2002)*, Taipei. pp. 93-96.

(Boitet, 2008) **Christian Boitet** (2008). Les architectures linguistiques et computationnelles en traduction automatique sont indépendantes. In *Actes de la conférence TALN 2008*, Avignon. 10 p.

(Bourigault, 1994) **Didier Bourigault** (1994). *LEXTER, un Logiciel d'EXtraction de Terminologie. Application à l'acquisition des connaissances à partir de textes*. Thèse de doctorat, École des Hautes Études en Sciences Sociales, Paris.

(Bourigault & Fabre, 2000) **Didier Bourigault & Cécile Fabre** (2000). Approche linguistique pour l'analyse syntaxique de corpus. *Cahiers de grammaire, 25*. pp. 131-151.

(Bourigault et al., 2005) **Didier Bourigault, Cécile Fabre, Cécile Frérot, Marie-Paule Jacques & Sylwia Ozdowska** (2005), Syntex, analyseur syntaxique de corpus. In *Actes des 12èmes journées sur le Traitement Automatique des Langues Naturelles*, vol. II, Dourdan. pp. 17-20.

(Brunet-Manquat & Sérasset, 2006) **Francis Brunet-Manquat & Gilles Sérasset** (2006). Création d'une base terminologique juridique multilingue à l'aide de la plateforme générique Jibiki : le projet LexALP. In *Actes de TALN 2006*, Louvain. pp. 435-444.

(Cabré et al., 2001) **Maria Teresa Cabré, Rosa Estopà & Jordi Vivaldi** (2001). Automatic term detection : A review of current systems. In *Recent Advances in Computational Terminology*, John Benjamins. pp. 53–87.

(Charest et al., 2007) **Simon Charest, Éric Brunelle, Jean Fontaine & Bertrand Pelletier** (2007). Élaboration automatique d'un dictionnaire de cooccurrences grand public. In *Actes de TALN 2007*. pp. 283-292

(Chauché, 1990) **Jacques C hauché** (1990). Détermination sémantique en analyse structurelle : une expérience basée sur la définition de distance. *TA Information, 31:1*. pp. 17-24.

(Church & Hanks, 1989) **Kenneth Ward Church & Patrick Hanks** (1989). Word Association Norms, Mutual Information, and Lexicography. In *Proceedings of the 27th. Annual Meeting of the Association for Computational Linguistics*. pp. 76-83.

(Church & Hanks, 1990) **Kenneth Ward Church & Patrick Hanks** (1990). Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics, 16:1*. pp. 22-29.

(Collins & Quillian, 1969) **Allan M. Ross & M. Ross Quillian** (1969). Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior, 8:2*. pp. 240-248

(Collins & Quillian, 1910) **Allan M. Ross & M. Ross Quillian** (1970). Does category size affect categorization time ? *Journal of verbal learning and verbal behavior, 9:4*. pp. 432-438

(Colmerauer, 1970) **Alain Colmerauer** (1970). *Les Systèmes-Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur*, publication interne 43, groupe TAUM, département d'informatique de l'Université de Montréal.

(Colmerauer & Roussel, 1992) **Alain Colmerauer & Philippe Roussel** (1992). *La naissance de Prolog*, consultée le 16/07/2009. <http://alain.colmerauer.free.fr/ArchivesPublications/HistoireProlog/24juillet92.pdf>

(Conchon et al., 2005) S**ylvain Conchon, Jean-Christophe Filliâtre & Julien Signoles** (2005). Le foncteur sonne toujours deux fois. In *Actes des Seizièmes Journées Francophones des Langages Applicatifs*, Obernai. pp. 79-94.

(Croft, 2001) **William Croft** (2001). *Radical Construction Grammar: syntactic theory in typological perspective*. Oxford University Press. 448 p.

(Daille, 1994) **Béatrice Daille** (1994). *Approche mixte pour l'extraction automatique de terminologie : statistiques lexicales et filtres linguistiques*. Thèse de Doctorat. Université Paris 7.

(Deerwester et al., 1990) **Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, & Richard Harshman** (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41:6*. pp. 391-407.

(Dolan et al., 1993) **William Dolan, Lucy Vanderwende & Stephen D. Richardson** (1993). Automatically deriving structured knowledge bases from on-line dictionaries. In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, Vancouver. pp. 5-14.

(Dong & Dong, 2003) **Zhendong Dong & Qiang Dong** (2003). HowNet - a hybrid language and knowledge resource. In *Proceedings of 2003 International Conference on Natural Language Processing and Knowledge Engineering*, Pékin. pp. 820-824.

(Dorr & Katsova, 1998) **Bonnie Dorr & Maria Katsova** (1998). Lexical Selection for Cross-Language Applications: Combining LCS with WordNet. In *Proceedings of AMTA'1998*, Langhorne (États-Unis), Springer. pp. 438-447.

(Duchet & Kraif, 2006) **Jean-Louis Duchet & Olivier Kraif** (2006). Alinea : a language independant tool for bi-text processing, In *Proceedings of JRC EU-Enlargement Workshop: Exploiting parallel corpora in up to 20 languages*, Arona, Italie.

(Dunning, 1993) **Ted Dunning** (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics, 19:1*. pp. 61-74.

(Evert, 2004) **Stefan Evert** (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Thèse de doctorat, Université de Stuttgart. 353 p.

(Fellbaum, 1998) **Christiane Fellbaum** (1998). *Wordnet : An Electronic Lexical Database*. MIT Press. 446 p.

(Fillmore, 1982) **Charles J. Fillmore** (1982). Frame semantics, *Linguistics in the Morning Calm*, Hanshin Publishing Co., pp. 111-137.

(Fillmore et al., 2003) **Charles J. Fillmore, Christopher R. Johnson & Miriam R.L. Petruck** (2003). Background to Framenet. *International Journal of Lexicography, 16-3*, pp. 235-250.

(Fischer & Fuhr, 2002) **Gudrun Fischer & Norbert Fuhr** (2002). An RDF Model for Multi-Level Hypertext in Digital Libraries. In *Proceedings of the 32. GI-Jahrestagung*, Dortmund. pp. 156-160.

(Fluhr et al., 1997) **Christian Fluhr** (1997). SPIRIT-W3 : A distributed Cross-Lingual Indexing and Search Engine. In *Proceedings of the INET 97 « The Seventh Annual Conference of the Internet Society »*, Kuala Lumpur, Malaysie. <http://www.isoc.org/inet97/proceedings/A8/A8_1.HTM>

(Frantzi & Ananiadou, 1995) **Katerina Frantzi & Sophia Ananiadou** (1995). Statistical measures for terminological extraction. In *Proceedings of JADT'95*, Rome. pp. 297-308.

(Fung & McKeown, 1997) **Pascale Fung & Kathleen McKeown** (1997). A technical word- and term-translation aid using noisy parallel corpora across language groups. *Machine Translation, 12:1-2*.5 pp. 3-87.

(Fujitsu, 2007) *ATLAS Lineup* (2007). Fujitsu, consultée le 16/07/2009. <http://www.fujitsu.com/global/services/software/translation/atlas/lineup/>

(Gale & Church, 1993) **William A. Gale & Kenneth W. Church** (1993). A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics, 19:1*. pp. 177-184.

(Gansner et al., 2006) **Emden Gansner, Eleftherios Koutsofios & Stephen North** (2006). *Drawing graphs with dot*, consultée le 16/07/2009. <http://www.graphviz.org/pdf/dotguide.pdf>

(Gansner & North, 2000) **Emden R. Gansner & Stephen C. North** (2000) - An Open Graph Visualization System and its Applications to Software engineering. *Software Practice and Experience, 30:11*. pp. 1203-1234

(Graphviz) *The DOT Language*. Graphviz, consultée le 16/07/2009. <http://www.graphviz.org/doc/info/lang.html>

(Grishman et al., 1994) **Ralph Grishman, Catherine Macleod & Adam Meyers** (1994). COMLEX Syntax: Building a Computational Lexicon. In *Proceedings of Coling 1994*, Kyoto. pp. 268-272.

(Gross, 1981) **Maurice Gross** (1981). Les bases empiriques de la notion de prédicat sémantique. *Langages, 63*. pp. 7-52.

(Gross, 1990) **Maurice Gross** (1990) La caractérisation des adverbes dans un lexique-grammaire. *Langue Française, 86*. pp. 90-102.

(Grossmann & Tutin, 2003) **François Grossmann & Agnès Tutin** (2003). Quelques pistes pour le traitement des collocations. In A. Tutin & F. Grossmann (éd.), *Les collocations. Analyse et traitement,* De Werelt. pp. 5-21.

(Hassan et al., 2006) **Hany Hassan, Ahmed Hassan & Sara Noeman** (2006). Graph based semi-supervised approach for information extraction. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, New York. pp. 9-16.

(Hausmann, 1989) **Franz Josef Hausmann** (1989). Le dictionnaire de collocations. In F.J. Hausmann, O. Reichmann, H.E. Wiegand & L. Zgusta (éd.), *Wörterbucher : ein internationales Handbuch zur Lexicographie. Dictionaries. Dictionnaires*, De Gruyter. pp. 1010-1019.

(Himsolt, 1997) **Michael Himsolt** (1997). *GML: A portable graph file format*. Rapport technique, Université de Passau.

(Hjelmslev, 1971) **Louis Hjelmslev** (1971). *Essais linguistiques*. Les éditions de Minuit. 296 p.

(Holt et al., 2000) **Richard C. Holt, Andreas Winter & Andy Schürr** (2000). GXL: Towards a Standard Exchange Format. In *Proceedings 7th Working Conference on Reverse Engineering (WCRE 2000)*, Brisbane. pp. 23-25.

(Hunston, 2002) **Susan Hunston** (2002). *Corpora in applied linguistics*. Cambridge : Cambridge University Press. 254 p.

(Inkpen & Hirst, 2002) **Diana Zaiu Inkpen & Graeme Hirst** (2002). Acquiring collocations for lexical choice between near synonyms. In *Proceedings of the SIGLEX Workshop on Unsupervised Lexical Acquisition, 40th meeting of the Association for Computational* Linguistics. pp. 67-76.

(Järvinen, 1994) **Timo Järvinen** (1994). Annotating 200 Million Words: The Bank Of English Project. In *Proceedings of COLING 1994*, Kyoto. pp.565-568.

(Jijkoun & De Rijke, 2007) **Valentin Jijkoun & Maarten de Rijke** (2007). Learning to Transform Linguistic Graph. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, Rochester (États-Unis). pp.53-60.

(Justeson & Katz, 1995) **John S. Justeson & Slava M. Katz** (1995), Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering, 1*. pp. 9-27.

(Kahane & Polguère, 2001) **Sylvain Kahane & Alain Polguère** (2001). Formal foundation of lexical functions. In *Proceedings of COLLOCATION: Computational Extraction, Analysis and Exploitation*, Toulouse. pp. 8-15.

(Kahane, 2003) **Sylvain Kahane** (2003) Une « blessure profonde » dans le Dictionnaire Explicatif et Combinatoire : sur le lien entre la définition lexicographique et les fonctions lexicales. In A. Tutin & F. Grossmann (éd.), *Les collocations. Analyse et traitement,* De Werelt. pp. 61-73.

(Katz & Giesbrecht, 2006) **Graham Katz & Eugenie Giesbrecht** (2006). Automatic identification of non-compositional multi-word expressions using Latent Semantic Analysis. In *Proceedings of the COLING/ACL'06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, Sydney, pp. 12-19.

(Kay & Röscheisen, 1993) **Martin Kay & Martin Röscheisen** (1993). Text-translation alignment. *Computational Linguistics, 19:3*. pp. 121-142.

(Killgarriff et al., 2004) **Adam Kilgarriff, Pavel Rychly, Pavel Smrz & David Tugwell** (2004). The Sketch Engine. In *Proceedings of EURALEX 2004*, Lorient. pp. 105-116.

(Kingsbury & Palmer, 2002) **Paul Kingsbury & Martha Palmer** (2002). From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas (Espagne), ELDA/ELRA. pp. 1989-1993.

(Kipper et al., 2006). **Karin Kipper, Anna Korhonen, Neville Ryant & Martha Palmer** (2006). Extending VerbNet with Novel Verb Classes. In *Proceedings of 5th international conference on Language Resources and Evaluation*, Genova, Italy. pp. 1027-1032.

(Koehn & Knight, 2002) **Philip Koehn & Kevin Knight** (2002). Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, Philadelphia. pp. 9-16.

(Koeva et al., 2007) **Svetla Koeva, Denis Maurel & Max Silberztein** (2007). *Formaliser les langues avec l'ordinateur : de INTEX à NooJ*. Cahiers de la MSH Ledoux, Presses Universitaires de Franche-Comté. 438 p.

(Korhonen & Briscoe, 2004) **Anna Korhonen & Ted Briscoe** (2004). Extended Lexical-Semantic Classification of English Verbs. In *Proceedings of the HLT/NAACL Workshop on Computational Lexical Semantics*, Boston. pp. 38-45.

(Kupiec, 1993) **Julian Kupiec** (1993). An algorithm for finding noun phrase correspondences in bilingual corpora. In *Proceedings of 31st Annual Meeting of the Association for Computational Linguistics*, Colombus (États-Unis). pp. 17-22.

(Kraif, 2008) **Olivier Kraif** (2008). Comment allier la puissance du TAL et la simplicité d'utilisation ? L'exemple du concordancier bilingue ConcQuest. In *Actes de JADT 2008*, Lyon, Presses Universitaires de Lyon. pp. 625-33.

(Kipper-Schuler, 2005) **Karin Kipper-Schuler** (2005). *VerbNet: A broad-coverage, comprehensive verb lexicon.* Thèse de doctorat, Université de Pennsylvanie. 146 p.

(Lafourcade & Prince, 2001) **Mathieu Lafourcade & Violaine Prince** (2001). Synonymies et vecteurs conceptuels. In *Actes de TALN'2001*, Tours, ATALA. pp. 233-242.

(Lafourcade & Joubert, 2008) **Mathieu Lafourcade & Alain Joubert** (2008). JeuxDeMots : un prototype ludique pour l'émergence de relations entre termes. In *Proceedings of JADT'2008*. Lyon, Presses Universitaires de Lyon. pp. 657-666.

(Larousse 1992) **Larousse** (1992) . *Thésaurus Larousse – des idées aux mots, des mots aux idées*, Larousse. 1146 p.

(Lenat & Greiner, 1980) **Douglas B. Lenat & Russel D. Greiner** (1980). RLL : a Representation Language Language. In *Proceedings of the 1st NCAI,* Stanford. pp. 165-169.

(Léon & Millot, 2005) **Stéphanie Léon & Chrystel Millon** (2005). Acquisition semi-automatique de relations lexicales bilingues (français-anglais) à partir du Web. In *Rencontre des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL 2005)*, Dourdan. pp. 595-604.

(Levin, 1993) **Beth Levin** (1993). *English Verbs Classes and Alternations : A Preliminary Investigation*. The University of Chicago Press. 366 p.

(L'Homme, 2003) **Marie-Claude L'Homme** (2003) Les combinaisons lexicales spécialisées (CLS). Description lexicographique et intégration aux banques de terminologie. In A. Tutin & F. Grossmann (éd.), *Les collocations. Analyse et traitement,* De Werelt. pp. 89-103.

(Li et al., 2003) **Mingqin Li, Juanzi Li, Zhendong Dong, Zuoying Wang & Dajin Lu** (2003). Building a large Chinese corpus annotated with semantic dependency. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, Sapporo, Japon. pp. 84-91.

(Lin, 1998) **Dekang Lin** (1998). Extracting Collocations from Text Corpora. In *Proceedings of First Workshop on Computational Terminology*. pp. 57-63.

(Lu & Zhou, 2004) **Yajuan Lü & Ming Zhou** (2004). Collocation translation acquisition using monolingual corpora. In *Proceedings of ACL 2004*, Barcelone. pp. 57-63.

(Ma, 2006) **Xiaoyi Ma** (2006). Champollion: A Robust Parallel Text Sentence Aligner. In *Proceedings of LREC 2006: Fifth International Conference on Language Resources and Evaluation*, Gênes, ELRA/ELDA. pp. 489-492.

(Macklovitch et al., 2000) **Elliott Macklovitch, Michel Simard & Philippe Langlais** (2000). TransSearch: A Free Translation Memory on the World Wide Web. In *Proceedings of Second International Conference on Language Resources and Evaluation (LREC)*. Athènes, ELRA/ELDA. pp. 1201-1208.

(Mangeot et al., 2003) **Mathieu Mangeot, Gilles Sérasset & Mathieu Lafourcade** (2003). Construction collaborative de données lexicales multilingues, le projet Papillon. *TAL, édition spéciale. Les dictionnaires électroniques : pour les personnes, les machines ou pour les deux ? 44:2*. pp. 151-176.

(Mangeot, 2005) **Mathieu Mangeot** (2005). Back to the roots: building a French-Japanese dictionary. In *Proceedings of Papillon 2005 Workshop*, Chiang Rai, Thailand. pp. 52-58.

(Mangeot & Chalvin, 2006) **Mathieu Mangeot & Antoine Chalvin** (2006). Dictionary Building with the Jibiki Platform: the GDEF case. In *Proceedings of LREC 2006*. Gênes, ELRA/ELDA. pp. 1666-1669.

(McEnery et al., 2005) **Tony McEnery, Richard Xiao & Yukio Tono** (2005). *Corpus-based Language Studies: An Advanced Resource Book*. Routledge, Taylor & Francis. 386 p.

(Melhorn & Näher, 1999) **Kurt Mehlhorn & Stefan Näher** (1999). *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press.

(Mel'čuk et al., 1984) **Igor Mel'čuk, Nadia Arbatchewsky-Jumarie, Léo Elnitsky, Lidija Iordanskaja, Adèle Lessard, Marie-Noëlle Lefebvre & Suzanne Mantha** (1984). *Dictionnaire explicatif et combinatoire du français contemporain, Recherches lexico-sémantiques, I*. Presses de l'Université de Montréal. 344 p.

(Mel'čuk et al., 1988) **Igor Mel'čuk, Nadia Arbatchewsky-Jumarie, Louise Dagenais, Léo Elnitsky, Lidjia Iordanskaja, Marie-Noëlle Lefebvre & Suzanne Mantha** (1988). *Dictionnaire explicatif et combinatoire du français contemporain, Recherches lexico-sémantiques, II*. Presses de l'Université de Montréal. 174 p.

(Mel'čuk et al., 1992) **Igor Mel'čuk, Nadia Arbatchewsky-Jumarie, Lidija Iordanskaja & Suzanne Mantha** (1992). *Dictionnaire explicatif et combinatoire du français contemporain, Recherches lexico-sémantiques, III*. Presses de l'Université de Montréal. 325 p.

(Mel'čuk et al., 1995) **Igor Mel'čuk, André Clas & Alain Polguère** (1995). *Introduction à la lexicologie explicative et combinatoire*. Duculot. 264 p.

(Mel'čuk, 1997) **Igor Mel'čuk** (1997). *Vers une linguistique sens-texte*. Leçon inaugurale au Collège de France, Paris. 43 p.

(Mel'čuk et al., 1999) **Igor Mel'čuk, Nadia Arbatchewsky-Jumarie, Lidija Iordanskaja, Suzanne Mantha & Alain Polguère** (1999). *Dictionnaire explicatif et combinatoire du français contemporain, Recherches lexico-sémantiques, IV*. Presses de l'Université de Montréal. 347 p.

(Mel'čuk, 2003) **Igor Mel'čuk** (2003). Collocations : définition, rôle et utilité. In A. Tutin & F. Grossmann (éd.), *Les collocations. Analyse et traitement,* De Werelt. pp. 23-31.

(Mel'čuk & Polguère, 2006) **Igor Mel'čuk & Alain Polguère** (2006). Dérivations sémantiques et collocations dans le DiCo/LAF. In P. Blumenthal & F. J. Hausmann (éd.), *Langue française, 50 (Collocations, corpus, dictionnaires)*. pp.66-83.

(Miller, 2000) **Christopher Miller** (2000). Regards sur la phonologie des langues signées, *Recherches linguistiques de Vincennes*, *19*, pp.101-120.

(Minsky, 1975) **Marvin Minsky** (1975) A framework for representing knowledge. In P.H. Winston (éd.), *The Psychology of Computer Vision*, McGraw-Hill, New York. pp. 211-277.

(Miyoshi et al., 1996) **Hideo Miyoshi, Kenji Sugiyama, Masahiro Kobayashi & Takano Ogino** (1996) An Overview of the EDR Electronic Dictionary and the Current Status of its Utilization. In *Proceedings of the 16 th COLING*, Copenhague. pp. 1090-1093.

(Morris, 1938) **Charles Morris** (1938). *Foundations of the theory of signs*. Chicago University Press. 59 p.

(Narayan et al., 2002) **Dipak Narayan, Debasri Chakrabarti, Prabhakar Pande & Pushpak Bhattacharyya** (2002), An Experience in Building the Indo WordNet - a WordNet for Hindi. In *Proceedings of First International Conference on Global WordNet*, Mysore, Inde. 8 p.

(Nerima et al. 2006) **Luka Nerima, Violeta Seretan, & Eric Wehrli** (2006). Le problème des collocations en TAL. *Nouveaux cahiers de linguistique française, 27*. pp. 95-115.

(Nguyen et al., 2007) **Hong-Thai Nguyen, Christian Boitet & Gilles Sérasset** (2007). PIVAX, an online contributive lexical database for heterogeneous MT systems using a lexical pivot. In *Proceedings of SNLP 2007*, Bangkok, Thaïlande. 6 p.

(Och & Ney, 2003) **Franz Josef Och & Hermann Ney** (2003). A Systematic Comparison of Various Statistical Alignment Models, *Computational Linguistics, 29:1*. pp. 19-51.

(Paumier, 2003) **Sébastien Paumier** (2003). *De la reconnaissance de formes linguistiques à l'analyse syntaxique*. Thèse de doctorat, Université de Marne-la-Vallée, 197 p.

(Pearce, 2001) **Darren Pearce** (2001). Synonymy in collocation extraction. In *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources : Applications, Extensions and Customizations*, Pittsburgh. pp.41-46.

(Pecina & Schlesinger, 2006) **Pavel Pecina & Pavel Schlesinger** (2006). Combining Association Measures for Collocation Extraction. In *Proceedings of the 21th International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, Sydney. pp .651-658.

(Peirce, 1931- 1935) **Charles S. Peirce** (1931-1935). *Collected Papers of C. S. Pierce*, vols. I-VI, éd. Charles Hartshorne & Paul Weiss, Harvard University Press, Cambridge.

(Planas, 1998) **Emmanuel Planas** (1998). *TELA : Structure et algorithmes pour la traduction fondée sur la mémoire*. Thèse de doctorat, Université Joseph Fourier, Grenoble. 371 p.

(Plante, 1983) **Pierre Plante** (1983). Le système de programmation DEREDEC. *Mots, 6*. pp. 101-134.

(Polguère, 2000) **Alain Polguère** (2000). Towards a theoretically-motivated general public dictionary of semantic derivations and collocations for French. In *Proceedings of EURALEX'2000*, Stuttgart. pp. 517-527

(Polguère, 2003) **Alain Polguère** (2003). *Lexicologie et sémantique lexicale. Notions fondamentales*, Presses de l'Université de Montréal. 266 p.

(Polguère, 2006) **Alain Polguère** (2006). Structural Properties of Lexical Systems: Monolingual and Multilingual Perspectives. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability (COLING/ACL 2006)*, Sydney. pp.50-59.

(Preece, 1981) **Scott Preece** (1981). *A Spreading Activation Model for Information Retrieval*. Thèse de doctorat, Université de l'Illinois.

(Punin et al., 1999) **John Punin, Yongxing Wang & Mukkai Krishnamoorthy** (1999). WWWPal Client-Server System for Webgraphs. In *Poster Proceedings of the 8th International World Wide Web Conference*, Toronto.

(Punin et al., 2001) **John Punin, Mukkai Krishnamoorthy & Mohammed J. Zaki** (2001). LOGML - Log Markup Language for Web Usage Mining, *in Proceedings of WEBKDD Workshop 2001: Mining Log Data Across All Customer TouchPoints (with SIGKDD01)*, San Francisco, Springer. pp. 88-112.

(Punin & Krishnamoorthy, 2001) **John Punin & Mukkai Krishnamoorthy** (2001). *XGMML 1.0 Draft Specification*, consultée le 16/07/2009. <http://www.cs.rpi.edu/~puninj/XGMML/draft-xgmml.html>

(Pustejovsky, 1993) **James Pustejovsky** (1993). *The Generative Lexicon*. MIT Press. 318 p.

(Quillian, 1968) **M. Ross Quillian** (1968) Semantic memory. In M. Minsky (éd.), *Semantic information processing*, MIT Press, Cambridge. pp. 227-270.

(Rapp, 1999) **Reinhard Rapp** (1999). Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of ACL-99*, College Park, Maryland (États-Unis). pp. 519-526.

(Richardson et al., 1993) **Stephen D. Richardson, Lucy Vanderwende & William Dolan** (1993), Combining Dictionary-Based and Example-Based Methods for Natural Language Analysis. In *Proceedings of TMI '93*, Kyoto. pp. 69-79.

(Richardson et al., 1998) **Stephen D. Richardson , William B. Dolan & Lucy Vanderwende** (1998). MindNet: acquiring and structuring semantic information from text. In *Proceedings of the 17th international conference on Computational linguistics*, Montréal. pp. 1098-1102.

(Roget, 1852) **Peter Mark Roget** (1852). *Roget's Thesaurus of English Words and Phrases*, Longman. 1280 p.

(Sag et al., 2002) **Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake & Dan Flickinge**. (2001). Multiword Expressions: A Pain in the Neck for NLP. In *Proceedings of CICLING-2002*, Mexico. pp.1-15.

(Sagot et Fišer, 2008) **Benoît Sagot & Darja Fišer** (2008). Construction d'un Wordnet libre du français à partir de ressources multilingues. In *Actes de TALN 2008*, Avignon. 10 p.

(Salton et al., 1975) **Gerard Salton, A. Wong & C. S. Yang** (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM, 18:11*. pp. 613-620.

(Salton & McGill,1986) **Gerard Salton & Michael J. McGill** (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill. 448 p.

(Saussure, 1916) **Ferdinand de Saussure** (1916). *Cours de linguistique générale*, Payot.

(Schmid, 1994) **Helmut Schmid** (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester. pp. 44-49.

(Schone & Jurafsky, 2001). **Patrick Schone & Daniel Jurafsky** (2001). Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh. pp. 100-108.

(Schwab , 2001) **Didier Schwab** (2001). *Vecteurs Conceptuels et Fonctions Lexicales : application à l'antonymie*. Mémoire de DEA, Université Montpellier II. 62 p.

(Schwab et al., 2002) **Didier Schwab, Mathieu Lafourcade & Violaine Prince** (2002). Vers l'apprentissage automatique, pour et par les vecteurs conceptuels, de fonctions lexicales. In *Actes de TALN'2002*, Nancy. pp. 125-134.

(Schwab et al., 2004) **Didier Schwab, Mathieu Lafourcade & Violaine Prince** (2004). Hypothèse pour la construction et l'exploitation conjointe d'une base lexicale sémantique basée sur les vecteurs conceptuels. In *Proceedings of JADT 2004 : 7èmes Journées internationales d'Analyse statistique des Données Textuelles*, Louvain-La-Neuve, Presses Universitaires de Louvain. pp. 1008-1018.

(Schwab, 2005) **Didier Schwab** (2005). *Approche hybride - lexicale et thématique - pour la modélisation, la détection et l'exploitation des fonctions lexicales en vue de l'analyse sémantique de texte*. Thèse de doctorat, Université de Montpellier. 390 p.

(Sébillot, 2002) **Pascale Sébillot** (2002). *Apprentissage sur corpus de relations lexicales sémantiques - La linguistique et l'apprentissage au service d'applications du traitement automatique des langues*. Habilitation à diriger des recherches, Université de Rennes. 109 p.

(Seo et al., 2003) **Hee-Chol Seo, Sang-Bum Kim, Baeg-Il Kim, Hae-Chang Kim & Sang-Zoo Lee** (2003). KUNLP system for NTCIR-3 English-L
Korean Cross-Language Information Retrieval. In K. Oyama, E. Ishida, & N. Kando (éd.), *Proceedings of the Third NTCIR Workshop on research in Information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo. pp. 63-78.

(Sérasset, 1994) **Gilles Sérasset** (1994). *SUBLIM : un système universel de bases lexicales multilingues et NADIA : sa spécialisation aux bases lexicales interlingues par acceptions*. Thèse de doctorat, Université Joseph Fourier, Grenoble. 194 p.

(Sérasset, 1997) **Gilles Sérasset** (1997). Informatisation du Dictionnaire Explicatif et Combinatoire. In *Actes de TALN 1997*, Grenoble. pp.194-198.

(Sérasset & Boitet, 1999) **Gilles Sérasset & Christian Boitet** (1999). UNL-French deconversion as transfer & generation from an interlingua with possible quality enhancement through offline human interaction. In *Proceedings of the MT Summit VII*, Singapore. pp. 220-228.

(Seretan & Wehrli, 2006) **Violeta Seretan & Eric Wehrli** (2006). Accurate collocation extraction using a multilingual parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*. 953-960.

(Seretan & Wehrli, 2007) **Violeta Seretan & Eric Wehrli** (2007). Collocation translation based on sentence alignment and parsing. In *Actes de la 14e conférence sur le Traitement Automatique des Langues Naturelles (TALN 2007)*, Toulouse. pp. 401-410.

(Seretan, 2008) **Violeta Seretan** (2008). *Collocation extraction based on syntactic parsing*. Thèse de doctorat, Université de Genève. 265 p.

(Shinyama & Sekine, 2004) **Yusuke Shinyama & Satoshi Sekine** (2004). Named Entity Discovery Using Comparable News Articles. In *Proceedings of COLING 2004*, Genève. pp. 848-853.

(Silberztein, 1993) **Max Silberztein** (1993). *Dictionnaires électroniques et analyse automatique de textes : le système INTEX*. Masson, Paris. 233 p.

(Silberztein, 1999) **Max Silberztein** (1999). Text Indexing with INTEX, *Computers and the Humanities, 33:3*, Kluwer Academic Publishers. pp. 265-280.

(Silberztein, 2005) **Max Silberztein** (2005). NooJ's Dictionaries. In *Proceedings of LTC 2005*, Poznan. pp. 291-295.

(Simpson, 1943) **George Gaylord Simpson** (1943). Mammals and the nature of continents. *American Journal of Science, 241*, pp. 1-31.

(Sinclair et al., 1970) **John Sinclair, Susan Jones & Robert Daley** (1970). English lexical studies: report to OSTI on project C/LP/08. Rapport technique, Department of English, University of Birmingham. 208 p.

(Smadja, 1991) **Franck A. Smadja** (1991). From N-Grams to Collocations: An Evaluation of Xtract. In *Proceedings of 29th Annual Meeting of the Association for Computational* Linguistics, Berkeley. pp. 279-284.

(Sowa, 1976) **John F. Sowa** (1976). Conceptual graphs for a database interface, *IBM Journal of Research and Development*, *20:4*, pp. 336-357.

(Sowa, 1984) **John F. Sowa** (1984). *Conceptual structures : Information Processing in Mind and Machine*, Addison-Wesley, Reading. 481 p.

(Sowa, 2000) **John F. Sowa** (2000). *Conceptual Graph Standard*, consultée le 16/07/2009. <http://www.jfsowa.com/cg/cgstandw.htm>

(Traum & Habash, 2000) **David Traum & Nizar Habash** (2000). Generation from Lexical Conceptual Structures. In *Proceedings of Workshop on Applied Interlinguas, NAACL/ANLP-2000*, Seattle. pp. 34-41

(Tsai, 2004) **Wang-Ju Tsai** (2004). *La coédition langue↔UNL pour partager la révision entre langues d'un document multilingue.* Thèse de doctorat, Université de Grenoble. 333 p.

(Tufis, 2004) **Dan Tufis** (2004). *Romanian Journal of Information Science and Technology. Special Issue on Balkanet,* 7:1-2. 248,).

(Turtle, 1990) **Howard R. Turtle** (1990). *Inference networks for document retrieval*. Thèse de doctorat, Université du Massachusetts.

(Tutin & Grossmann, 2002) **Agnès Tutin & François Grossmann** (2002). Collocations régulières et irrégulières : esquisse du phénomène collocatif, *Revue française de linguistique appliquée*, *VII:1*. pp. 7-25.

(Uchida, 1989) **Hiroshi Uchida** (1989). ATLAS-II : A Machine Translation System Using Conceptual Structure as an Interlingua. In *Proceedings of Second Machine Translation Summit*, Munich. pp. 152-157.

(Uchida, 2001) **Hiroshi Uchida** (2001), The Universal Networking Language beyond Machine Translation, In *Proceedings of International symposium on language in cyberspace*, Seoul. 14 p.

(UNL, 2005) *Universal Networking Language (UNL) Specification, Version 2005* (2005). UNDL Foundation, consultée le 16/07/2009. <http://www.undl.org/unlsys/unl/unl2005/>

(Véronis, 2000) **Jean Véronis** (2000). Alignement de corpus multilingues. In Jean-Marie Pierrel. (éd.), *Ingénierie des langues*, Hermès. pp. 151-171.

(Von Ahn & Dabbish, 2004) **Luis von Ahn & Laura Dabbish** (2004). Labeling Images with a Computer Game. In Proceedings of *ACM CHI 2004*, Vienne (Autriche). pp. 319-326.

(Vossen, 2004) **Piek Vossen** (2004). EuroWordNet: a multilingual database of autonomous and language-specific wordnets connected via an Inter-Lingual-Index. *International Journal of Linguistics (semi-special issue on multilingual databases), 17:2*. pp. 161-173

(Wanner et al., 2006) **Leo Wanner, Bernd Bohnet & Mark Giereth** (2006). What is beyond Collocations? Insights from Machine Learning Experiments. In *Proceedings of EURALEX 2006*, Turin. 16 p.

(Wehrli, 2004) **Eric Wehrli** (2004). Un modèle multilingue d'analyse syntaxique. In A. Auchlin et al. (éd.), *Structures et discours - Mélanges offerts à Eddy Roulet*, Nota bene, Québec. pp. 311–329.

(Widdows & Dorow, 2002) **Dominic Widdows & Beate Dorow** (2002). A graph model for unsupervised lexical acquisition. In *Proceedings of 19th International Conference on Computational Linguistics*, Taipei. pp. 1093-1099.

(Widlöcher & Bilhaut, 2007) **Antoine Widlöcher et Frédérik Bilhaut** (2007). La plate-forme LinguaStream. In *Autour des langues et du langage : perspective pluridisciplinaire*, Presses Universitaires de Grenoble. pp. 447-454.

(Williams, 2003) **Geoffrey Williams** (2003). Les collocations et l'école conceptualiste britannique. In A. Tutin & F. Grossmann (éd.), *Les collocations. Analyse et traitement,* De Werelt. pp. 33-44.

(Wikimedia, 2008) *List of Wiktionaries* (2008). Fondation Wikimedia, consultée le 22/05/2008. <http://meta.wikimedia.org/wiki/List_of_Wiktionaries>

(Wikimedia, 2009a) *Statistiques Wiktionnaire - Français* (2009). Fondation Wikimedia, consultée le 16/07/2009. <http://stats.wikimedia.org/wiktionary/FR/TablesWikipediaFR.htm>

(Wikimedia, 2009b) *Wikipedia Statistics - French* (2009). Fondation Wikimedia, consultée le 16/07/2009. <http://stats.wikimedia.org/EN/TablesWikipediaFR.htm>

(Witschel, 2007) **Hans Friedrich Witschel** (2007). Multi-level Association Graphs - A New Graph-Based Model for Information Retrieval. In *Proceedings of the HLTNAACL-07 Workshop on Textgraphs-07*. pp. 9-16.

(Witschel, 2008) **Hans Friedrich Witschel** (2008). *Global and Local Resources for Peer-to-Peer Text Retrieval*. Thèse de doctorat, Université de Leipzig. 204 p.

(Wu & Zhou, 2003) **Hua Wu & Ming Zhou** (2003). Synonymous collocation extraction using translation information. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, Sapporo. pp. 120–127.

(Yang, 2003) **Shouxun Yang** (2003). Machine learning for collocation identification. In *Proceedings of 2003 International Conference on Natural Language Processing and Knowledge Engineering (ICNLP-2003)*, Pékin. pp.315-320.

(Zeman, 1964) **Jay J. Zeman** (1964). *The Graphical Logic of C. S. Peirce*. Thèse de Ph.D., Université de Chicago. <http://www.clas.ufl.edu/users/jzeman/graphicallogic/>

(Zinsmeister & Heid, 2003) **Heike Zinsmeister & Ulrich Heid** (2003). Identifying predicatively used adverbs by means of a Statistical Grammar Model. In *Proceedings of the Corpus Linguistics 2003 conference*, Lancaster. pp. 932-939.

(Žolkovskij & Mel'čuk, 1967) **Aleksandr Žolkovskij & Igor Mel'čuk** (1967). O semantičes kom sinteze. *Problemy kibernetiki, 19*, pp. 177-238. [Traduction française (1970) : Sur la synthèse sémantique. *T.A.Informations, 2*, pp. 1-85.]

Annexes

Annexe A – Les fonctions lexico-sémantiques

D'après (Mel'čuk et al., 1999). Les relations *paradigmatiques* sont exprimées par les fonctions lexicales 1 à 20, les relations *syntagmatiques* par les fonctions lexicales 21 à 51.

	Nom	**Signification**	**Exemple**
1	**Syn**	Synonyme	Syn(voiture)=automobile
2	**$Conv_{ij}$**	Conversif	$Conv_{21}$(inclure)=faire partie
3	**Anti**	Antonyme	Anti(égal)=inégal
4	**Contr**	Contrastif	Cont(feu)=glace
5	**Epit**	Épithète pléonastique	Epit(océan)=immense
6	**Gener**	Générique	Gener(armoire)=meuble
7	**Figur**	Figuratif	Figur(fumée)=rideau de [~]
8	**S_0**, **V_0**, **A_0**, **Adv_0**,	Nominalisation, verbalisation, adjectivisation, adverbialisation	S_0(dormir)=sommeil V_0(serment)=jurer A_0(correction)=correct Adv_0(rapide)=rapidement
FLS nominales			
9	**S_i**	Dérivé sémantique nominal actanciel	S_1(parler)=locuteur S_2(parler)=paroles, propos, discours S_3(parler)=allocuteur, destinataire
10	**S_{instr}**, **S_{loc}**, **S_{med}**, **S_{mod}**, **S_{res}**,	Circonstant (de l'instrument, du lieu, du moyen, du mode, du résultat)	S_{instr}(peindre)=pinceau**1.2** S_{res}(copier)=copie
11	**Sing**	Portion régulière	Sing(flotte)=bateau
12	**Mult**	Collectif	Mult(poisson)=banc de [~s]
13	**Cap**	Nom de chef	Cap(université)=président
14	**Equip**	Nom d'équipe	Equip(avion)=équipage
15	**Germ**	Nom de démarrage	Germ(colère**I**)=ferment, levain [de la ~]
16	**Centr**	Nom du centre	Centr(problème)=cœur [du ~]
17	**Culm**	Point culminant	Culm(joie**I**)=comble**II** [de la ~]

FLS adjectivales			
18	**A_i**	Dérivé sémantique adjectival actanciel	A_1(mépris)=plein, rempli [de ~] A_2(mépris)=couvert [de ~]
19	**$Able_i$**	Dérivé sémantique adjectival potentiel (= possible)	$Able_1$(peur)=peureux $Able_2$(peur)=effrayant
20	**$Qual_i$**	Dérivé sémantique adjectival virtuel (= probable)	$Qual_1$(tromper)=malhonnête $Qual_2$(tromper)=naïf
21	**Magn**	Intensificateur	Magn(peur)=bleue
22	**Plus, Minus**	Comparatif	(seulement en combinaison avec d'autres FLS) IncepPredPlus(fièvre)=augmente IncepPredMoins(fièvre)=baisse, diminue (*Pred* : 31, *Incep* : 38)
23	**Ver**	Confirmateur	Ver(peur)=justifiée
24	**Bon**	Laudatif	Bon(conseil)=précieux
25	**Pejor**	Péjoratif	IncepPredPejor(santé)=détériore
26	**Pos_2**	Positif	Pos_2(avis)=favorable
FLS adverbiales			
27	**Adv_i**	Dérivé sémantique adverbial actanciel	Adv_1(joie**I**)=avec [~] Adv_2(joie**I**)=à la [~]
28	**Instr**	Instrumental	Instr(main)=à, avec [la ~], de [la ~]
29	**Loc**	Locatif[146]	$Loc_{in/ad}$(campagne)=à [la ~]
30	**Propt**	Consécutif	Propt(alcool)=sous l'empire [de ~]
FLS verbales			
31	**Pred**	Verbe ayant le sens de « être »	(en combinaison avec d'autres FLS ; voir notamment *Plus* et *Minus* : 22)
32	**$Oper_i$**	Verbe support dont le mot-clé est le complément d'objet principal	$Oper_1$(conseil)=donner $Oper_3$(conseil)=recevoir
33	**$Func_i$**	Verbe support dont le mot-clé est le sujet	$Func_0$(réunion)=est en cours $Func_1$(aide)=vient [de N] $Func_2$(danger)=menace [N]

[146]Correspond à plusieurs sens différents : **Loc_{in}** (« se trouvant dans »), **Loc_{ab}** (« se déplaçant à partir de »), **Loc_{ad}** (« se déplaçant pour se trouver dans »).

34	**Labor$_{ij}$**	Verbe support dont le mot-clé est le COI	Labor$_{12}$(estime)=tenir [N en ~]
35	**Involv**	Verbe d'implication	Involv(odeur)=remplit [la pièce]
36	**Incep, Cont, Fin**	Verbes phasiques : Commencer, Continuer, Cesser	IncepOper$_1$(suprématie)= arriver [à ART ~], obtenir [ART ~] ContOper$_1$(suprématie)=retenir [ART ~] FinOper$_1$(suprématie)=perdre [ART ~]
37	**Caus, Liqu, Perm**	Verbes causatifs: Causer, Liquider, Permettre	CausOper$_1$(forme)=mettre [N sous ART ~] CausFunc$_1$(forme)=donner [à N ART ~] LiquOper$_1$(envie)=priver LiquFunc$_1$(envie)=ôter Caus$_2$Func$_2$(attention)=accaparer [ART$_{déf}$ ~]
38	**Real$_i$**	Verbe de réalisation dont le mot-clé est le COD	Real$_1$(peine)=imposer, infliger [ART ~] Real$_2$(peine)=purger [ART ~]
39	**Fact$_i$**	Verbe de réalisation dont le mot-clé est le sujet	Fact$_0$(rêve)=se réalise
40	**Labreal$_{ij}$**	Verbe de réalisation dont le mot-clé est le COI	Labreal$_{12}$(balle)=atteindre [N avec ~]
41	**Manif**	Verbe de manifestation	Manif(joie)=inonde [ART$_{déf}$ visage]
42	**Prepar**	Préparer	PreparFact$_2$(piège)=tendre [un ~ à N]
43	**Prox**	Être sur le point de	ProxFunc$_0$(orage**1**)=s'approcher
44	**Degrad**	Se dégrader	Degrad(lait)=tourner
45	**Obstr**	Fonctionner avec difficulté (temporaire)	Obstr(souffle)=manquer
46	**Excess**	Fonctionner d'une façon excessive	Excess(moteur)=s'emballer
47	**Son**	Émettre le son typique	Son(chien)=aboyer
48	**Imper**	Ordre[147]	Imper(tirer)=Feu !

[147]Autrement qu'en conjugant le verbe à l'impératif.

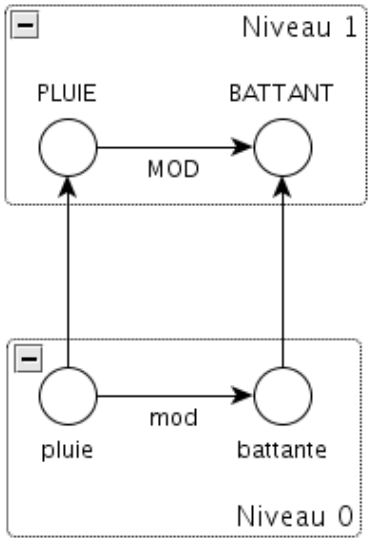
49	**Perf**	Action complétée	S_1Perf(marier)=époux
50	**Result**	Résultat escompté de l'événement	Result$_1$Perf(promettre)= //[148] être obligé [de V_{inf}]
51	**Sympt$_{ijk}$**	Symptôme	Excess(cheveux**I.2**)—Sympt$_{13}$(horreur)= ses cheveux se dressent (d'horreur)

[148]« // » note que la valeur est une expression *fusionnée* de la FL, contenant à la fois du sens du mot-clé et de la FL (par exemple : AntiMagn(vent)= léger, //brise) ; la valeur ne doit donc pas être employée conjointement avec le mot-clé.

Annexe B – Fichier GraphML représentant un graphe MuLLinG

```
<?xml version="1.0" encoding="utf-8"?>
<graphml    xmlns="http://graphml.graphdrawing.org/xmlns"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
            http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
      <key id="nat" for="node" attr.name="term" attr.type="string"/>
      <key id="nal" for="node" attr.name="lemma" attr.type="string"/>
      <key id="nap" for="node" attr.name="pos" attr.type="string"/>
      <key id="naf" for="node" attr.name="feat" attr.type="string"/>
      <key id="nas" for="node" attr.name="idxsent" attr.type="string"/>
      <key id="nagt" for="node" attr.name="globaltype" attr.type="string">
            <default>class</default>
      </key>
      <key id="nalg" for="node" attr.name="lang" attr.type="string"/>
      <key id="nalvl" for="node" attr.name="level" attr.type="string"/>
      <key id="eat" for="edge" attr.name="typerel" attr.type="string"/>
      <key id="eaf" for="edge" attr.name="feat" attr.type="string"/>
      <key id="eagt" for="edge" attr.name="globaltype" attr.type="string"/>
      <key id="ealg" for="edge" attr.name="lang" attr.type="string"/>
      <key id="ealvl" for="edge" attr.name="level" attr.type="string"/>
      <key id="eail" for="edge" attr.name="interlevel" attr.type="string"/>
      <graph id="EWN" edgedefault="directed">
            <node id="n1">
                  <data key="nat">pluie</data>
                  <data key="nal">pluie</data>
                  <data key="nap">NOUN</data>
                  <data key="nalg">fr</data>
                  <data key="nlvl">0</data>
                  <data key="nagt">occ</data>
            </node>
            <node id="n2">
                  <data key="nat">battante</data>
                  <data key="nal">battant</data>
                  <data key="nap">ADJ</data>
                  <data key="nalg">fr</data>
                  <data key="nlvl">0</data>
                  <data key="nagt">occ</data>
            </node>
```

```
            <node id="n3">
                  <data key="nat">pluie%NOUN</data>
                  <data key="nalg">fr</data>
                  <data key="nlvl">1</data>
                  <data key="nagt">class</data>
            </node>
            <node id="n4">
                  <data key="nat">battant%ADJ</data>
                  <data key="nalg">fr</data>
                  <data key="nlvl">1</data>
                  <data key="nagt">class</data>
            </node>
            <edge source="n1" target="n2">
                  <data key="eagt">mono</data>
                  <data key="eat">mod</data>
                  <data key="ealg">fr</data>
                  <data key="ealvl">0</data>
                  <data key="eail">same</data>
            </edge>
            <edge source="n1" target="n3">
                  <data key="ealvl">0</data>
                  <data key="eail">uplevel</data>
            </edge>
            <edge source="n2" target="n4">
                  <data key="ealvl">0</data>
                  <data key="eail">uplevel</data>
            </edge>
            <edge source="n3" target="n4">
                  <data key="eagt">classe</data>
                  <data key="eat">classe-mod</data>
                  <data key="ealg">fr</data>
                  <data key="ealvl">1</data>
                  <data key="eail">same</data>
            </edge>
      </graph>
</graphml>
```



Annexe C – Entrées de WordNet

Représentation du synset correspondant au sens de « mesure » qu'on retrouve dans « prendre des mesures ».

WordNet de Princeton (2.0)

```
00164617 04 n 02 measure 1 step 2 004 @ 00158985 n 0000 ~ 00164868 n 0000 ~
00165810 n 0000 ~ 00772673 n 0000 | any maneuver made as part of progress
toward a goal; "the situation called for strong measures"; "the police took
steps to reduce crime"
```

Identifiant du synset	00115661				
Identifiant du type de fichier du lexicographe correspondant	04				Fichier du lexicographe : noun.act (noms d'actions)
Type de synset	n				Nom
Nombre de « sens de mots » dans le synset	02				
Mot	measure		step		Sens 1 de « measure », sens 2 de « step »
Identifiant des sens d'un lemme	1		2		
Nombre de liens de ce synset vers d'autres	004				
Type du lien	@	~	~	~	@ : hyperonymie, ~ : hyponymie
Identifiant du synset lié	00158985	00164868	00165810	00772673	
Type du synset lié	n	n	n	n	Noms
Direction	0000	0000	0000	0000	Du synset courant au synset indiqué
Gloss : définition, examples	any maneuver made as part of progress toward a goal; "the situation called for strong measures"; "the police took steps to reduce crime"				

Tableau 46: Format de représentation d'un synset (PWN)

EuroWordNet

L'identifiant de synset (98113) est celui que le synset avait dans la version 1.5 de PWN, il est différent de celui de la version 2.0 (115661).

EuroWordNet français :

```
0 @571@ WORD_MEANING
  1 PART_OF_SPEECH "n"
  1 VARIANTS
    2 LITERAL "démarche"
      3 SENSE 3
      3 EXTERNAL_INFO
        4 SOURCE_ID 1
          5 TEXT_KEY "98113-n"
    2 LITERAL "mesure"
      3 SENSE 10
  1 INTERNAL_LINKS
    2 RELATION "has_hyperonym"
      3 TARGET_CONCEPT
        4 PART_OF_SPEECH "n"
        4 LITERAL "manoeuvre"
          5 SENSE 1
    2 RELATION "has_hyponym"
      3 TARGET_CONCEPT
        4 PART_OF_SPEECH "n"
        4 LITERAL "contre-mesure"
          5 SENSE 1
  1 EQ_LINKS
    2 EQ_RELATION "eq_synonym"
      3 TARGET_ILI
        4 PART_OF_SPEECH "n"
        4 WORDNET_OFFSET 98113
```

Inter-Lingual-Index :

```
0 @571@ ILI_RECORD
  1 PART_OF_SPEECH "n"
  1 WORDNET_OFFSET 98113
  1 GLOSS "any maneuver made as part of progress toward a goal; "the police
took steps to reduce crime"& 03 04 2ndOrderEntity Agentive BoundedEvent
Cause Dynamic Mental Purpose SituationType"
  1 VARIANTS
    2 LITERAL "measure"
      3 SENSE 2
    2 LITERAL "step"
      3 SENSE 3
```

Wolf (0.1.4)

```
<SYNSET>
      <ID>ENG20-00164617-n</ID>
      <POS>n</POS>
      <SYNONYM>
            <LITERAL>
                  étape
                  <SENSE>21/5:fr.csbgen,fr.csen,fr.rocsbgen,fr.rocsen,
                  fr.roen</SENSE>
            </LITERAL>
            <LITERAL>
                  mesure
                  <SENSE>4294/2:fr.csbgen,fr.csen</SENSE>
            </LITERAL>
      </SYNONYM>
      <ILR>
            <TYPE>hypernym</TYPE>
            ENG20-00158985-n
      </ILR>
      <DEF>any maneuver made as part of progress toward a goal</DEF>
      <USAGE>the situation called for strong measures</USAGE>
      <USAGE>the police took steps to reduce crime</USAGE>
      <BCS>3</BCS>
      <DOMAIN>factotum</DOMAIN>
      <SUMO>
            IntentionalProcess
            <TYPE>=</TYPE>
      </SUMO>
</SYNSET>
```

Résumé

Pour modéliser au mieux les phénomènes linguistiques dans les systèmes de traitement automatique des langues, il faut notamment pouvoir disposer de ressources de qualité. Or, les ressources existantes sont très souvent incomplètes et ne permettent pas de traiter les données de façon adéquate dans des processus comme la traduction, l'analyse, etc.

Cette thèse s'intéresse à l'acquisition de connaissances linguistiques, et plus précisément à leur extraction à partir de corpus. Nous étudions en particulier l'extraction de collocations, ces couples de termes dont l'un est choisi en fonction de l'autre pour exprimer un sens particulier (comme « pluie battante » où « battante » sert à exprimer l'intensification).

Pour permettre l'acquisition de données à grande échelle, il faut la rendre facile à réaliser de manière automatique, et simple à paramétrer par des linguistes aux connaissances limitées en programmation. On doit pour cela se baser sur une modélisation adaptée et précise des données et des processus réalisés.

Nous avons, dans ce but, conçu et implémenté MuLLinG, un modèle de graphes linguistiques multiniveau, où chaque niveau représente l'information d'une manière différente (de plus en plus abstraite en général), ainsi que des opérations de manipulation de ces graphes.

Ce modèle, fondé sur une structure riche mais assez simple, permet de représenter, traiter et gérer divers types de ressources. De plus, les opérations associées ont été écrites de façon à être les plus génériques possibles, c'est à dire qu'elles sont indépendantes de ce que peuvent représenter les nœuds et les arcs du graphe, ainsi que de la tâche à réaliser.

Cela permet donc à notre modèle, qui a été mis en œuvre et utilisé pour plusieurs expérimentations, dont certaines concernent l'extraction de collocations, de voir un processus (parfois complexe) d'extraction de connaissances linguistiques comme une succession d'opérations simples de manipulation de graphes.

Mots-clés

extraction, acquisition de connaissances, modèle de graphe, manipulation de graphes, généricité, collocations

Abstract

In order to model the linguistic phenomena at best, in natural language processing systems, one needs high quality ressources. Butexisting ressources are most often incomplete and do not allow to manipulate data adequately in processes like translation, analysis, etc.

This thesis is about acquisition of linguistic knowledge, and more precisely about the extraction of that knowledge from corpora where it appears. We especially study the extraction of collocations, pairs of terms where one term is chosen in function of the other one to express a particular meaning (as in « driving rain », where « driving » is used to express intensification).

To allow large-scale data acquisition, it is necessary to make it easy to realize automatically, and simple to configure by linguists with limited knowledge in computer programming. For that reason, we have to rely on a precise and suitable model for data and process.

To that effect, we have designed and implemented MuLLinG, a model of multilevel linguistic graphs, where each level represents information in a different manner (more and more abstract in general), and operations for manipulating these graphs.

That model, based on a rich but quite simple structure, allows to represent, manipulate, and manage diverse kinds of ressources. Moreover, associated operations have been written in order to be as generic as possible, which means that they are independent of what nodes and edges represent, and of the task to fulfil.

That enables our model, which has been implemented and used for several experiments, some concerning collocation extraction, to view a (sometimes complex) process of linguistic knowledge extraction as a succession of small graph manipulation operations.

Keywords

extraction, knowledge extraction, graph model, graph manipulation, generic programming, collocations