Méthode et outils pour la création et l'évaluation automatiques de structures de bases lexicales multilingues (symétriques) à lexies et axies

Aree Teeraparbseree

UNIVERSITÉ JOSEPH FOURIER – GRENOBLE 1
UFR D'INFORMATIQUE ET MATHÉMATIQUES APPLIQUÉES

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER
Discipline : INFORMATIQUE

préparée au laboratoire GETA-CLIPS (IMAG, UJF & CNRS)
présentée et soutenue publiquement

par

Aree Teeraparbseree

le 27 septembre 2005

Méthode et outils pour la création automatiques et l'évaluation de structures de bases lexicales multilingues (symétriques) à lexies et axies

Directeur de thèse :
Christian BOITET

JURY

Mme Marie-Christine Fauvet	Président
M. Jacques Chauché	Rapporteur
Mme Béatrice Daille	Rapporteur
M. Christian Boitet	Directeur de thèse
M. Gilles Sérasset	Co-directeur de thèse
M. Mathieu Lafourcade	Examinateur
M. Mathieu Mangeot-Lerebours	Examinateur

À mes parents et Romain

pour leur soutien et leur affection

Remerciements

En premier lieu, je tiens à remercier Christian Boitet, directeur de cette thèse, pour ses critiques pertinentes et détaillées sur mon travail, et notamment sur ce mémoire. Elles m'ont apporté un éclairage complémentaire et enrichissant.

Je remercie également Gilles Sérasset, co-directeur de cette thèse, pour l'encadrement qu'il a effectué tout au long de ce travail, ainsi que nos discussions constructives.

Je remercie vivement les membres du jury, Jacques Chauché, Béatrice Daille, Mathieu Lafourcade, Mathieu Mangeot-Lerebours, et Marie-Christine Fauvet qui ont accepté d'être rapporteurs, examinateurs et présidente de cette thèse, et qui ont lu le manuscrit en l'accompagnant de nombreuses suggestions pertinentes.

Je souhaite remercier chaleureusement Romain Lenglet pour m'avoir aidée à surmonter les moments difficiles, pour m'avoir fait partager ses nombreuses idées intéressantes, et pour m'avoir toujours soutenue, motivée, et encouragée tout au long de ces travaux.

Je tiens aussi à remercier Jean-Philippe Guilbaud et Seksun Suwanmanee pour m'avoir aidée à créer une partie des données lexicales pour mes expérimentations.

Je remercie Didier Schwab et Jean-Michel Delorme pour nos nombreuses discussions sur le vecteur conceptuel.

Merci également à Sutee Sudprasert pour son aide et sa contribution aux expérimentations de cette thèse, et surtout à la récupération et la segmentation des données thaïes.

Je tiens enfin plus généralement à remercier tous les membres de l'équipe GETA qui m'ont accueillie et aidée durant ces quatre années de thèse.

Table des matières

Table des figures

Liste des tableaux

Introduction

Les ressources lexicales sont un élément important dans les systèmes de traitement automatique des langues naturelles (TALN). Depuis plusieurs années, le besoin en ressources lexicales de grande taille, riches en information, et couvrant de nombreuses langues, augmente de plus en plus. Par contre, le coût de construction d'un nouvel ensemble lexical de grande taille est assez élevé, voire prohibitif. Par exemple, le projet Electronic Dictionary Research (EDR, [EDR93, Yok95]) de construction de deux gros dictionnaires japonais-anglais et anglais-japonais associés à un dictionnaire conceptuel a nécessité plus de 1200 hommes-années pour un coût total d'environ 14 milliards de Yens (environ 104 millions euros). Son prix de vente, environ 14 000 euros, est assez inférieur aux coûts réels de construction mais trop élevé pour un particulier. On a donc besoin de réduire les coûts de création de bases lexicales multilingues (BDLM) libres et riches en information.

Il existe plusieurs projets de construction de bases lexicales de grande taille et libres, par exemple, JMDict [Breb], WordNet [Pri], HowNet [Zhe], et le projet Papillon [TMLP00, MLSL03]. Parmi ces projets, le projet Papillon semble utiliser les concepts les plus avancés. En particulier, sa macrostructure en étoile facilite l'ajout de nouvelles langues et sa microstructure basée sur une structure issue de la théorie sens-texte d'Igor Mel'čuk et ses collègues [Mel81, MP87, Pol98] permet des usages très détaillés d'informations.

Le moyen le plus rapide et le moins coûteux pour construire une nouvelle base lexicale est de l'initialiser à partir de ressources existantes. Dans ce contexte, les problèmes déjà résolus sont la récupération d'un ensemble de ressources « tel quel », et la mise à disposition, et l'extraction de ressources plus ou moins spécialisées. Les problèmes difficiles, encore non résolus, sont ceux liés à la construction d'une BDLM de lexies (sens de mot) et d'axies (liens interlingues), c'est-à-dire la base NADIA de Papillon. Ces problèmes sont divisés en trois parties :

- la construction initiale des lexies et des axies à partir d'un graphe global de vocables (ou de lexies) représentant les relations traductionnelles.
- l'évolution incrémentale par contribution.
- l'évaluation de la « qualité ».

Parmi ces trois parties, la deuxième partie reste un travail principalement manuel. La première et la dernière parties peuvent s'informatiser. Ces deux problèmes sont fortement liés car une construction initiale ne peut fonctionner sans disposer d'un moyen d'évaluer les résultats produits. Par conséquent, il est nécessaire de progresser sur la

notion de « qualité » d'une BDLM en même temps qu'automatiser sa construction initiale.

En ce qui concerne la construction, le traitement totalement automatique d'une BDLM peut être très gros et difficile. Il existe, en effet, une variété d'algorithmes proposés et combinables. D'où l'idée de construire une plate-forme logicielle permettant de les combiner facilement.

Dans ce travail, nous nous intéressons au cas concret de la structuration de bases lexicales multilingues, de manière automatique, par combinaison de différentes méthodes. Nous proposons Jeminie, un système logiciel d'aide à la construction d'une base lexicale multilingue, et plus précisément à la production d'un « premier jet » d'une organisation des vocables en lexies et à la transformation des liens traductionnels entre vocables en liens interlingues entre lexies. Cette première version de la base nécessite ensuite la correction et l'amélioration par des linguistes.

Jeminie a deux aspects principaux : combinaison et adaptabilité. Comme dans plusieurs domaines [BSAG98, Ped00, BM04], la combinaison des attributs ou des algorithmes donne, la plupart du temps, un résultat meilleur que chaque algorithme utilisé isolément. Nous utilisons donc la combinaison de plusieurs algorithmes pour créer le meilleur résultat possible. D'un autre côté, l'adaptabilité donne la possibilité de combiner aléatoirement des algorithmes en fonction des ressources disponibles.

De plus, Jeminie possède une partie dédiée à l'évaluation de la qualité d'une base lexicale produite par des méthodes semi-automatiques ou automatiques. Une étude du domaine nous a montré qu'il n'existe pas d'approche générale ou de critère reconnu pour l'évaluation des bases lexicales. Ainsi, nous proposons dans cette thèse une catégorisation de critères et des définitions de méthodes possibles pour une telle évaluation.

Organisation de la thèse

Cette thèse comporte trois parties.

La première partie porte sur les problèmes cruciaux en bases lexicales multilingues. Nous présentons d'abord une vue globale sur les projets de bases lexicales multilingues, puis les problèmes de la structuration en lexies et en axies, et les problèmes de construction des données dictionnairiques.

La deuxième partie donne dans un premier temps une brève analyse des principales approches de la structuration et une proposition de critères de qualité. Dans un deuxième temps, nous exposons les idées-clés à la base de notre conception d'un système d'aide à la structuration et à l'évaluation des bases lexicales multilingues.

La dernière partie présente en détail notre système d'aide à la structuration et à l'évaluation des bases lexicales multilingues. Nous terminons en présentant diverses expérimentations sur la structuration des bases lexicales bilingues et multilingues.

Première partie

Problèmes cruciaux en bases lexicales multilingues

Chapitre 1

Évolution des idées

Dans ce chapitre, nous explorons l'histoire du dictionnaire et ses évolutions. Les idées pour faire évoluer le dictionnaire seront aussi présentées. La section 1.1 présente l'histoire du dictionnaire depuis la pré-informatique jusqu'aux débuts de l'informatisation ainsi que ses problèmes. La section 1.2 donne une vue globale sur des projets visant à la construction des bases lexicales multilingues (BDLM). La section 1.3 présente une catégorisation du dictionnaire pour ensuite donner un but idéal, les BDLM.

1.1 Évolution du dictionnaire

1.1.1 Terminologie

Dictionnaire Un ensemble d'articles où chaque article est composé d'un mot-vedette et d'une information associée. Les articles sont classés dans un ordre spécifique. Généralement, c'est l'ordre alphabétique des mots-vedettes.

Lexique Un dictionnaire succinct établi pour un domaine spécialisé, par exemple, le lexique informatique, le lexique de la justice, le lexique économique, etc.

Base lexicale Un terme plus large que celui de dictionnaire. Une base lexicale peut être de plusieurs natures : des dictionnaires, des lexiques, des corpus, des thésaurus, etc.

Acception Une acception monolingue est une unité sémantique d'une langue. Une base d'acceptions fournit un lien entre les acceptions monolingues des différents dictionnaires. L'ensemble des acceptions interlingues est l'union des ensembles des acceptions monolingues des différents dictionnaires de la base.

1.1.2 Histoire « pré-informatique »

L'histoire des dictionnaires sur papier a commencé depuis longtemps. Les premières « listes lexicales » monolingues datent des Sumériens. On remarque aussi le *Vocabulario Degli Accademici della Crusca* en italien (1612), le *dictionnaire de l'Académie Française* (1694) et le *Diccionario de la Lengua Española* en espagnol (1726≃1736).

Un des dictionnaires bilingues les plus anciens est le *Promptorium Parvulorum Sive Clericorum* (1440), un dictionnaire anglais-latin d'environ 10 000 entrées. Puis, un dictionnaire trilingue breton-français-latin rédigé par Jehan Lagadeuc est apparu en 1464. En 1755, Samuel Johnson produisit le *Dictionary of the English Language* qui contient environ 40 000 définitions. C'était une première tentative pour normaliser la prononciation et l'orthographe des mots anglais, qui resta un modèle de la lexicographie anglaise pendant plus d'un siècle, avant la création de l'Oxford Dictionary.

Depuis, les dictionnaires ont beaucoup évolué. Il existe plusieurs types de dictionnaires tels que les dictionnaires d'acronymes [dA, Kin, qqc], les dictionnaires de synonymes [Man, dC02], les dictionnaires spécialisés dans des domaines spécifiques comme le dictionnaire de bactériologie vétérinaire [Euz], le dictionnaire d'hydrologie [CNF], etc. Les dictionnaires ont aussi évolué pour prendre en compte plusieurs langues. En effet, on trouve actuellement des lexiques multilingues (ou terminologiques multilingues) qui permettent à partir d'une langue, et de trouver la traduction dans d'autres langues. On ne trouve pas sous forme papier traditionnelle de dictionnaire d'usage réellement multilingue, c'est-à-dire, permettant l'accès à partir de chaque langue. Au mieux, on a des dictionnaires multicible (en fourche).

1.1.3 Débuts de l'informatisation

Dès l'arrivée de l'informatique, les lexicographes[1] ont essayé de stocker leurs dictionnaires et leurs bases lexicales sous forme électronique. Les deux motivations principales de la construction de dictionnaires sous forme électronique sont :

- la simplification de l'accès aux données et leur intégration à des environnements informatiques de rédaction,
- la possibilité de construire des dictionnaires pour des processus de traitement automatique.

Dans le premier cas, un dictionnaire sur support informatique renferme essentiellement le même contenu qu'un dictionnaire papier. Les dictionnaires électroniques diffèrent des dictionnaires papier par leur utilisation, leur présentation, leurs capacités de recherche et leurs fonctions bureautiques. Par ailleurs, l'informatique permet aussi d'ajouter des informations multimédia telles que le son (pour la prononciation), le court métrage vidéo, et l'image (comme leurs versions papier). Dans le second cas, les dictionnaires sont structurés et encodés selon leur applications.

Au début, les bases lexicales et les dictionnaires ont été stockés dans des bases de données en réseau ou relationnelles. Par exemple, la base de données multilingue européenne *EuroDicAutom*[2] (1973) et la banque de données *Termium*[3] (1975). Puis, les premiers dictionnaires sur cédérom ont fait leur apparition vers la fin des années 80. Le premier dictionnaire de langue française en version électronique, *Le Grand Robert électronique*, a été publié en 1989. Par la suite, d'autres dictionnaires électroniques ont vu le jour : *Larousse multimédia encyclopédique*, *Dictionnaire Hachette multimédia*,

[1] Les lexicographes sont les personnes qui rédigent les dictionnaires.

[2] `http://europa.eu.int/eurodicautom/`

[3] `http://www.termium.gc.ca/`

Oxford English Dictionary, *American Heritage Dictionary*, ou encore *Le Trésor de la Langue Française informatisé* (*TLFi*). L'informatisation de ce dernier a commencé depuis 1965, avec stockage sur rubans magnétiques. Avec l'évolution de l'informatique, en 1992, le TLFi a été adapté sous forme de base relationnelle et transformé en un multidictionnaire. Après plus de dix ans de travail, le TLFi « nouveau » a été produit, avec environ 100 000 mots, 270 000 définitions, et plus de 430 000 exemples. C'est probablement le plus complet des dictionnaires sur cédérom.

Les dictionnaires sur support informatique peuvent être représentés sous forme de fichiers de caractères lisibles par l'humain, ou sous forme de fichiers binaires nécessitant des applications spécifiques pour les décoder. Dans la suite du document, le terme dictionnaire électronique réfère seulement à un dictionnaire sous forme d'un fichier de caractères.

Malgré tous les efforts consentis pour informatiser des dictionnaires, il reste de nombreux problèmes :

- la *spécialisation* et la *fixité* des dictionnaires. Un dictionnaire est souvent spécifique à une application et à un système précis. Le développement d'un système de traduction impose souvent de construire un nouveau dictionnaire. Il serait préférable d'avoir un dictionnaire ou une base lexicale adaptable à des systèmes différents ou bien une méthode pour réutiliser des données existantes.
- le *choix des entrées* des dictionnaires, qui diffèrent entre les dictionnaires. Par exemple, dans le dictionnaire FeM [YRY+96] les mots « *idéal* » et « *idéaliser* » se trouvent dans deux entrées différentes, alors que dans le dictionnaire bilingue français-anglais Oxford-Hachette il n'y a qu'une entrée pour « *idéal* », dont « *idéaliser* » est une sous-entrée.
- La *granularité des entrées* est un problème supplémentaire. Par exemple, dans des dictionnaires généraux, les entrées sont au niveau des mots, mais dans le dictionnaire DEC [Mel88, Mel92] on trouve des entrées pour des expressions telles que « *à la tête* », « *au cœur* », « *prendre de vitesse* », « *sur pied* », etc. Il faut connaître ces détails pour développer un système qui profite bien des informations présentes dans les dictionnaires utilisés.

1.2 Projets de construction de BDLM

Parmi les bases lexicales informatisées, nous nous intéressons aux BDLM (bases de données lexicales multilingues). Un début de BDLM a été entrevu dans les années 80. Depuis, de nombreux projets ont travaillé sur la construction de BDLM. Nous détaillerons ici certains projets, ceux qui sont les plus intéressants pour offrir une vision globale sur les caractéristiques générales des BDLM, sur les approches utilisées et sur les problèmes rencontrés :

1.2.1 EDR (1986-1995)

Le projet EDR [EDR93, Yok95, MSKO96] est un grand projet de construction d'une base lexicale bilingue à « pivot conceptuel » réalisé à Tokyo. L'objectif principal est de créer une grande base lexicale (japonaise et anglaise) avec une infrastructure bien établie pour un usage automatisé dans le traitement avancé de la langue naturelle ou le traitement des connaissances. Les caractéristiques du dictionnaire EDR sont :

- une grande taille, avec une couverture la plus grande possible des termes généraux,
- un dictionnaire pour des applications générales, non spécialisé pour une application ou un algorithme particulier,
- un dictionnaire fourni avec une base de connaissances pour l'analyse sémantique,
- un dictionnaire d'une grande objectivité, car basé sur un grand volume de textes,
- un dictionnaire facilement extensible à des langues différentes.

EDR comprend deux corpus et cinq types de dictionnaires : dictionnaires de mots, dictionnaires bilingues, dictionnaires de concepts, dictionnaires de cooccurrences et dictionnaires de termes techniques. La structure du dictionnaire EDR est illustrée dans la figure 1.1.

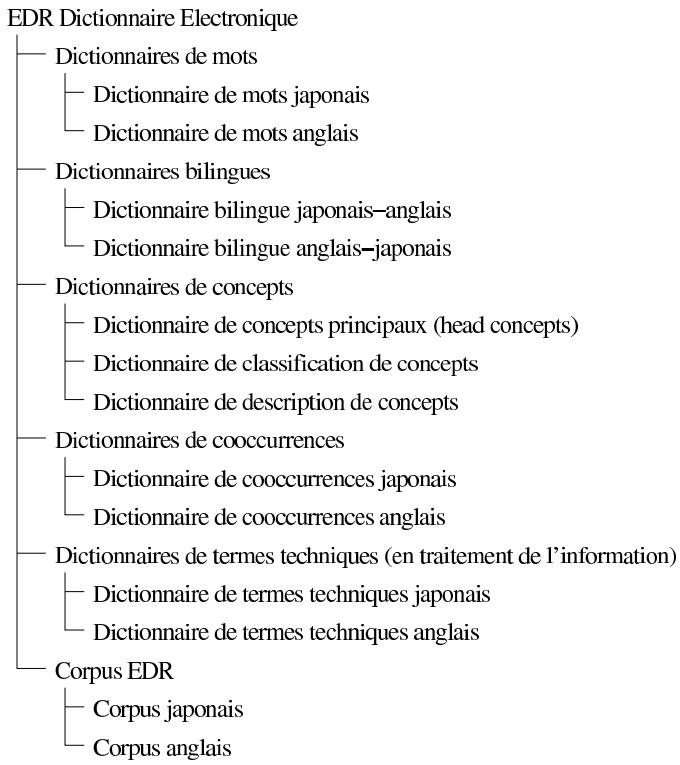

FIG. 1.1 : Structure de la base EDR

1. Un dictionnaire de mots présente le concept (signification) d'un mot, et décrit ses caractéristiques grammaticales. Chaque entrée du dictionnaire de mots comprend les quatre champs suivants :

- `<headword Information>` indique la forme du mot-vedette et la prononciation du mot,
- `<Grammatical Information>` indique la partie du discours[4] de l'entrée,
- `<Semantic Information>` donne les informations liées au dictionnaire de concepts telles que le numéro de concept correspondant, le Headconcept[5] et l'explication du concept,
- `<Pragmatic and Supplementary Information>` donne éventuellement des informations supplémentaires comme l'usage du mot ou la fréquence du mot dans le corpus EDR, ou la fréquence du mot pour ce concept précis dans le corpus EDR.

Un mot peut correspondre à N concepts. Dans ce cas, on a N entrées pour ce mot. Par contre, le nombre de mots dans un dictionnaire est défini par le nombre de mot-vedettes qu'il contient. Le dictionnaire de mots japonais contient 270-000 mots, et le dictionnaire de mots anglais contient 190 000 mots. La figure 1.2 illustre un exemple d'entrée de dictionnaire de mots anglais.

2. Un dictionnaire bilingue comprend, pour chaque entrée, l'information de l'entrée, l'information grammaticale, l'information sémantique, et l'information bilingue dans les quatre champs suivants :
   - `<headword Information>` indique la forme du mot,
   - `<Grammatical Information>` indique la partie du discours du mot,
   - `<Semantic Information>` donne les informations liées au dictionnaire de concepts telles que le numéro de concept correspondant, le Headconcept et l'explication du concept,
   - `<Correspondence Word Information>` indique le mot dans la langue cible correspondant au mot dans le champ `<headword Information>` ainsi que l'information sur la partie du discours de ce mot dans la langue cible.

   La figure 1.3 illustre un exemple d'entrée du dictionnaire anglais-japonais. Le dictionnaire bilingue japonais-anglais contient 230 000 mots, et le dictionnaire bilingue anglais-japonais contient 160 000 mots.
3. Le but du dictionnaire de concepts est de décrire, classifier et relier les concepts provenant des dictionnaires de mots. Le dictionnaire de concepts, contenant environ 410 000 concepts, est divisé selon le type de l'information en dictionnaire de Headconcept, dictionnaire de classification de concepts, et dictionnaire de description de concepts.
   - Le dictionnaire de Headconcept (figure 1.4) décrit l'information sur les concepts eux-mêmes, pour les faire comprendre aux lecteurs humains.
   - Le dictionnaire de classification de concepts (figure 1.5) décrit les relations d'hyponymie (*sub-concept*) ou d'hyperonymie (*super-concept*) entre les concepts. La difficulté de créer le dictionnaire de classification de concepts est de choisir la relation d'hyperonymie pour des concepts. Pour cela, EDR a regroupé les concepts qui partagent un certain attribut, afin de repérer les groupes qui sont

[4] Anglais : part-of-speech, ou encore « catégorie morpho-syntaxique »

[5] Un mot représentatif qui est le plus approprié pour exprimer le concept correspondant.

```
<Record Number>                          EWD1364642
<Headword Information>
        <Headword>                       supply
        <Invariable Portion of Headword and Adjacency Attributes Pair>
                                         suppl(Verb with Initial Consonant Sound,
                                         Invariable Portion of Verb Headword -
                                         Inflection Pattern y)
        <Syllable Division>              sup/ply
        <Pronunciation>                  sXepl'ai
<Grammar Information>
        <Part of Speech>                 Verb
        <Syntactic Tree>
        <Word Form and Inflection Information>
        <Word Form Information>          Invariable Portion of Verb
        <Inflection Information>         Inflection Pattern y
        <Grammatical Attributes>
        <Sentence Pattern Information>   Must take a direct object (direct object
                                         is a noun phrase);Takes a prepositional
                                         phrase beginning with the preposition 'to'
        <Function and Position Information>
        <Function Word Information>
<Semantic Information>
        <Concept Identifier>             0ec944
        <Headconcept>
                <Japanese Headconcept>   支給する[シキュウ・スル]
                <English Headconcept>    supply
        <Concept Explication>
                <Japanese Concept Explication>
                                         物をあてがう
                <English Concept Explication>
                                         to supply goods
<Pragmatic and Supplementary Information>
        <Usage>
        <Frequency>                      122/234
<Management Information>
        <Management History Record>      3/4/93
```

FIG. 1.2 : Exemple d'entrée du dictionnaire de mots anglais EDR

```
<Record Number>                          EJB1083615
<Headword Information>
        <Headword>                       claim
<Grammar Information>
        <Part of Speech>                 Common Noun
<Semantic Information>
        <Concept Identifier>             3d01c7
        <Headconcept>
                <English Headconcept>    claim
                <Japanese Headconcept>   権利[ケンリ]
        <Concept Explication>
                <English Concept Explication>
                                         a lawful power which enables a person to
                                         claim his profit
                <Japanese Concept Explication>
                                         法律上で特定の利益を主張しうる力
<Correspondence Information>
        <Correspondence Word Information>
                <Correspondence Word Category>
                                         0
                <Correspondence Word Notation>
                                         (当然の)権利

                <Correspondence Word Category>
                                         0
                <Correspondence Word Notation>
                                         (当然の)資格

                <Correspondence Word Category>
                                         0
                <Correspondence Word Notation>
                                         要求権
<Management Information>
        <Management History Record>      DATE="95/3/10"
```

FIG. 1.3 : Exemple d'entrée du dictionnaire bilingue anglais-japonais EDR

les plus représentatifs. Un même concept peut appartenir à plusieurs groupes. Il s'agit donc d'une hiérarchie.

– Le dictionnaire de description de concepts décrit les relations sémantiques entre les concepts verbaux et les concepts nominaux telles que `AGENT`, `OBJECT`, `IMPLEMENT`, `PLACE`, etc. La figure 1.6 donne un exemple de relation entre concepts dans le dictionnaire de description de concepts.

```
<Record Number>                          CPH0314159
<Concept Identifier>                     3d0ecb
<Headconcept>
        <English Headconcept>            borrow
        <Japanese Headconcept>           借りる[カリ・ル]
<Concept Explication>
        <English Concept Explication>    to use a person's property after
                                         promising to ...
        <Japanese Concept Explication>   返す約束で,他人のものを使う
<Management Information>
        <Management History Record>      Date of record update "93/04/26"
```

FIG. 1.4 : Exemple d'entrée du dictionnaire EDR de concepts principaux (headconcept)

```
<Record Number>                          CPC0271828
<Super-concept Identifier>               4445bc [Concept identifier that
                                         indicates 'something written']
<Sub-concept Identifier>                 4445a0 [Concept identifier that
                                         indicates 'piece of correspondence']
<Management Information>
        <Management History Record>      Date of record update "92/03/05"
```

FIG. 1.5 : Exemple d'entrée du dictionnaire EDR de classification de concepts

4. Le dictionnaire de cooccurrences fournit une information collocationnelle sur les mots. Le dictionnaire de cooccurrences japonais contient 900 000 phrases, et le dictionnaire de cooccurrences anglais contient 460 000 phrases.
5. Le corpus EDR contient des phrases analysées aux niveaux morpho-syntaxique et sémantique. Le corpus japonais contient 220 000 phrases, et le corpus anglais contient 120 000 phrases.

EDR adopte une approche mixte : une approche bilingue et une approche interlingue. L'approche bilingue concerne les dictionnaires bilingues, tandis que l'approche interlingue concerne le dictionnaire de concepts. Les dictionnaires bilingues de EDR possèdent à la fois des liens vers le dictionnaire de concepts et des liens bilingues.

Le point faible de cette architecture est la difficulté d'ajouter une nouvelle langue dans la base, parce qu'il faut à la fois la relier avec toutes les autres langues dans la base et la relier avec un dictionnaire de concepts. En revanche, le point fort est la taille des dictionnaires qui est assez grande pour couvrir tous les vocables japonais de la vie quotidienne. De plus, la base EDR ne dépend pas d'une théorie linguistique

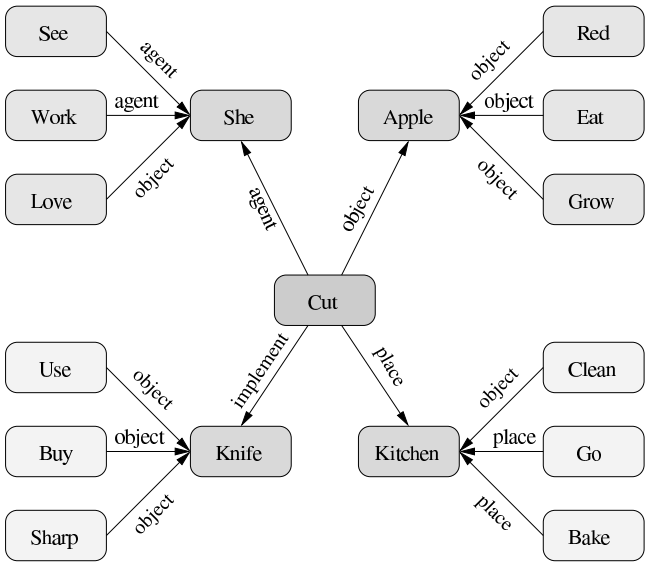

FIG. 1.6 : Exemple de relations entre concepts dans le dictionnaire de concepts EDR

particulière, et la traduction entre deux langues est assez précise car chaque entrée est basée sur le niveau sémantique (concept).

1.2.2 ULTRA (1985-1993)

ULTRA [FGW92, FGW93] est un système de traduction automatique multilingue réalisé à l'université du Nouveau Mexique (New Mexico State University ou NMSU). Le système traduit entre cinq langues : anglais, allemand, chinois, espagnol et japonais avec environ 10 000 acceptions pour chaque langue. ULTRA introduit l'approche interlingue pour relier des dictionnaires monolingues en un dictionnaire interlingue. Les acceptions interlingues, appelées aussi IR[6], sont construites à partir des sens de mots (acceptions) du dictionnaire monolingue anglais LDOCE[7].

Chaque langue dans le système ULTRA est indépendante des autres, mais reliée aux autres à travers les acceptions interlingues. De plus, chaque langue possède sa propre méthode pour associer des entrées avec les IRs. Les entrées correspondantes dans les langues différentes devraient être reliées à une même IR et chaque IR devrait générer les entrées correspondantes dans chaque langue.

ULTRA comprend deux types de données : acception interlingue et acception monolingue. Chaque acception (monolingue ou interlingue) se présente sous la forme d'une clause Prolog :

```
category (Form, F1, F2, ...).
```

`category` est une catégorie d'acception. ULTRA distingue huit catégories d'acception : `entities` (nom), `relations` (verbe, ou adjective), `entity specifiers` (déterminant), `relation specifiers` (auxiliaire), `case relations` (préposition), `proposition`

[6]Interlingual Representation
[7]Longman Dictionary of Contemporary English (LDOCE)

`specifiers` (complémentaire), `proposition modifiers` (locution adverbiale), et `conjunctions` (conjonction).

`Form` est une forme orthographique.

`F1, F2, ...` sont des contraintes syntaxiques pour les entrées monolingues. Pour les entrées interlingue, `F1, F2, ...`, correspondent aux contraintes sémantiques et pragmatiques.

Voici, deux exemples d'entrées monolingues des mots espagnols *banco* et *ingresó* :

```
noun (banco, third_singular, masculine, bank4_1).
verb (ingresó, third_singular, finite, past, simple, indicative,
active, deposit1_3).
```

Le dernier paramètre de chaque entrée monolingue représente une forme d'une IR correspondant à cette entrée. Les deux entrées IR de *bank4_ 1* et *deposit1_ 3* sont les suivantes :

```
entity (bank4_1, class, countable, institution, abstract_object,
economics_banking).
relation (deposit1_3, dynamic, placing, agent, patient, human,
amount, human, abstract_object, economics_banking)
```

ULTRA utilise une approche interlingue dans le sens où ses acceptions interlingues sont les acceptions du dictionnaire anglais LDOCE. C'est un point faible de cette architecture, car il est probable que les sens de mot dans LDOCE ne couvrent pas tous les sens de mot dans les dictionnaires monolingues de langues différentes. Par exemple, dans LDOCE, on ne trouve pas de sens de mot pouvant distinguer les mots français « fleuve » et « rivière ». Il est donc préférable qu'un dictionnaire interlingue soit indépendant des langues pour qu'on puisse exprimer les différents sens de mot de différentes langues. Par contre, une architecture interlingue permet d'ajouter facilement une nouvelle langue sans effets secondaires négatifs sur les langues existant dans la base, ni sur la performance globale du système, puisque chaque langue est indépendante des autres. Du fait que chaque entrée de chaque langue est reliée à une acception interlingue qui représente une sémantique, la traduction entre langues est assez précise.

1.2.3 WordNet et EuroWordNet

1.2.3.1 WordNet (1990 -)

Princeton WordNet [Fel98, Pri] est un projet de construction d'un dictionnaire monolingue anglais de grande taille. Il ne contient que des mots des catégories nom, verbe, adverbe et adjectif, qui ont été organisés séparément en quatre ensembles de synonymes appelés « *synset* », et en relations sémantiques entre ces synsets.

Les synsets sont des ensembles de mots qui ont la même partie du discours (nom ou verbe, etc.) et qui peuvent se substituer dans un certain contexte. Par exemple

dans WordNet 1.5 les mots `{car ; auto ; automobile ; machine ; motorcar}` sont dans le même synset car ils peuvent être employés pour un même concept. Un synset est souvent décrit par une *glose*[8], par exemple la description de l'exemple précédent de synset est « `4-wheeled ; usually propelled by an internal combustion engine` ». Des synsets peuvent être associés entre eux par des relations sémantiques, telles que l'hyponymie, l'hyperonymie (entre concepts spécifiques et concepts généraux), l'holonymie, la méronymie (entre une partie et une totalité), et l'antonymie. La figure 1.7 illustre un exemple de relations autour du synset `{car ; auto ; automobile ; machine ; motorcar}`. Ce synset est lié :

- au concept plus général, (ou hyperonyme) : `{motor vehicle ; automotive vehicle}`,
- aux concepts plus spécifiques, (ou hyponymes) : `{cruiser ; squad car ; patrol car ; police car ; prowl car}` et `{cab ; taxi ; hack ; taxicab}`,
- aux concepts correspondant aux pièces dont se compose une voiture, par exemple : `{bumper}`, `{car door}`, `{car mirror}` et `{car window}`.

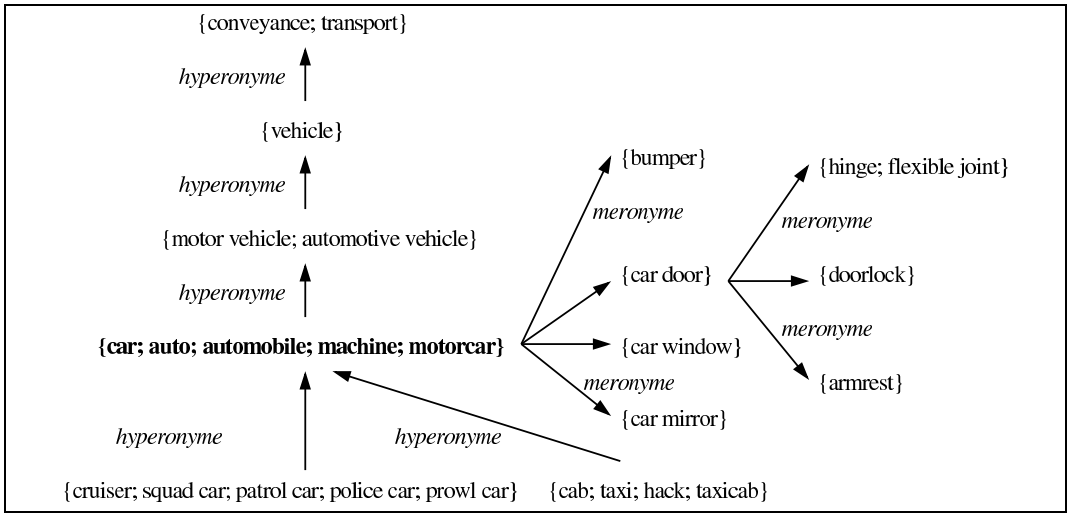

FIG. 1.7 : Exemple de synset d'un sens de mot « `car` » dans WordNet

À l'aide des relations sémantiques, toutes les acceptions dans une langue peuvent être interconnectées et constituent un grand réseau ou « wordnet ». Depuis plusieurs années, de nombreux contributeurs ont développé WordNet. Le projet WordNet est une base lexicale monolingue mais il a donné l'idée, ensuite, de lancer le projet EuroWordNet pour l'aspect multilingue.

1.2.3.2 EuroWordNet (1996-1999)

Le projet EuroWordNet [Vos98, Eur], est la suite du projet WordNet [Fel98], et a pour but de construire une base lexicale multilingue reliant des *wordnets* pour plusieurs langues européennes telles que l'anglais, le hollandais, l'italien, l'espagnol, l'allemand, le français, le tchèque et l'estonien. Les wordnets sont structurés de la même

[8]une définition et/ou des phrases d'exemple pour un synset (*gloss* en anglais)

manière que le Princeton WordNet [Pri] avec des synsets et des relations sémantiques entre synsets.

Chaque synset d'un wordnet est également lié au synset qui a la relation sémantique la plus proche dans le WordNet américain. Chaque langue est ainsi liée au WordNet américain, dont les synsets servent d'index interlingue[9]. Grâce à ces liens interlingues, il est possible de passer d'un mot d'une langue à un autre mot similaire dans une autre langue. Cependant, les distinctions sémantiques de WordNet sont souvent trop subtiles. Il est difficile parfois même pour un humain de déterminer quel sens il voudrait exactement. Voici un exemple de quelques-uns des 37 sens du verbe « `hold` ».

```
hold (v) keep from exhaling or expelling.
hold (v) hold the attention of.
hold (v) remain in a certain state.
hold (v) stop dealing with.
hold (v) keep in mind or convery as a conviction or view .
hold (v) organize or be responsible for.
```

Le projet est déjà terminé, mais il y a encore plusieurs groupes de recherche qui continuent de développer des wordnets dans d'autres langues, européennes ou non-européennes, en utilisant les spécifications de l'EuroWordNet. S'ils sont compatibles, ces wordnets pourront être ajoutés à une base lexicale de l'EuroWordNet à travers des index, et reliés à d'autres wordnets. La taille de l'EuroWordNet est d'environ 45-000 concepts et 75 000 acceptions. La plupart des vocables sont des mots courants de chaque langue.

1.2.4 HowNet (1988-2003)

HowNet [Zhe] est un travail initié par Dong ZhenDong qui défend le point de vue qu'une base de connaissances est très importante pour l'analyse finale dans le traitement automatique des langues naturelles. Le projet HowNet vise à la construction d'une base de connaissances en chinois et en anglais, gratuite pour un usage non commercial, et uniquement destinée à un usage automatisé. La base de connaissances est représentée sous forme d'un ensemble de graphes. Un graphe comprend plusieurs concepts, et des relations entre ces concepts. Chaque concept est construit par des *sémèmes*, un sémème étant une unité de lexique représentant un seul sens. HowNet contient environ 65 000 concepts du chinois et environ 75 000 concepts de l'anglais.

Chaque entrée lexicale de HowNet comprend quatre types d'information :

`W_X` = forme de mot ou de phrase en langue X,

`E_X` = exemple d'usage,

`G_X` = partie du discours,

`DEF` = définition avec une liste de concepts et/ou de pointeur,

où les pointeurs sont des relations entre concepts ou attributs de concepts. Par exemple, « `#` » indique la relation « *associé* », « `*` » indique la relation « *agent* ».

[9] Inter-Lingual-Index (ILI)

Le champ `DEF` est interprété selon la catégorie du concept. HowNet divise les concepts en six catégories : entité, événement, attribut, quantité, valeur d'attribut, et valeur de quantité. Par exemple, pour les entrées de type entité, le premier paramètre dans le champ `DEF` indique la classe générale du concept.

La figure 1.8 donne un exemple d'entrée de HowNet. Dans l'exemple, l'entrée « `journalist` » a la classe générale « `human` ». « `journalist` » a une relation *associée* avec « `occupation` » et « `news` », et a une fonction « `compile` » et « `gather` ».

FIG. 1.8 : Exemple d'entrée de HowNet

La stratégie de construction de la base HowNet consiste à construire d'abord une base de connaissances générale qui représente des concepts généraux et à établir des relations entre eux. À partir de cette base, les entrées sont enrichies par des utilisateurs pour des domaines spécifiques. La base permet aussi aux utilisateurs d'ajouter des relations de synonymie, d'antonymie ou de traduction, en utilisant une liste de règles prédéfinies.

À ce jour, HowNet est devenu une base de connaissances de grande taille couvrant les vocables courants en chinois. Plus la taille de la base est grande, plus les difficultés augmentent sur le choix des relations (hyponymie, hyperonymie, etc) entre entrées lorsqu'on ajoute une nouvelle entrée, et sur la définition des concepts pour chaque entrée.

1.2.5 PARAX et PARAX-UNL

1.2.5.1 PARAX (1989-1997)

PARAX [Bla95] est une maquette de base lexicale interlingue par acceptions construite par Étienne Blanc. La base PARAX comprenait cinq langues : l'allemand, l'anglais, le français, le chinois, et le russe. Cette maquette a été implémentée en $HyperCard^{TM}$. Chaque acception monolingue est associée à une information linguistique et une acception interlingue correspondante. L'accès au dictionnaire interlingue se fait soit à travers la liste des acceptions interlingues définies, soit via un dictionnaire monolingue. La base lexicale PARAX contient peu de données (environ 600 entrées), car c'était une base d'expérimentation pour l'approche interlingue.

1.2.5.2 PARAX-UNL (1998-2003)

PARAX-UNL [Bla99] est la suite de la maquette PARAX, construite en utilisant des données du projet UNL [Uch01] comme entrées. Cette fois, la taille est beaucoup plus grande. PARAX-UNL comprend 7 langues : l'anglais, le chinois, l'espagnol, le français, l'italien, le japonais, et le russe. Comme les données monolingues proviennent des données UNL de chaque langue, la taille de la base dépend de la taille des données UNL existant pour chaque langue. Pour l'anglais et le japonais, on a un grand nombre d'entrées, et un peu moins pour d'autres langues (voir les détails dans le tableau 1.1).

PARAX-UNL utilise les UW (Universal Word) d'UNL comme un langage pivot. Les UW sont les « lexèmes » du langage UNL. Chaque UW représente un concept ou un ensemble de concepts (sens de mots). Une UW est formée d'un mot anglais suivi d'une liste de contraintes, qui précise le sens de l'UW. Chaque dictionnaire monolingue dans PARAX-UNL est lié aux UW.

L'accès à un dictionnaire monolingue se fait soit par un mot-vedette (`entrée : lemme`), soit par une UW (`entrée : uw`) (figure 1.9). Si on accède à partir d'un mot-vedette, on obtient une liste des différents sens pour ce mot-vedette. Par exemple, la figure 1.10 montre une liste d'UW pour le mot-vedette français « `centre` ». Si on accède à un dictionnaire monolingue à partir d'une UW, on obtient le détail de cette UW.

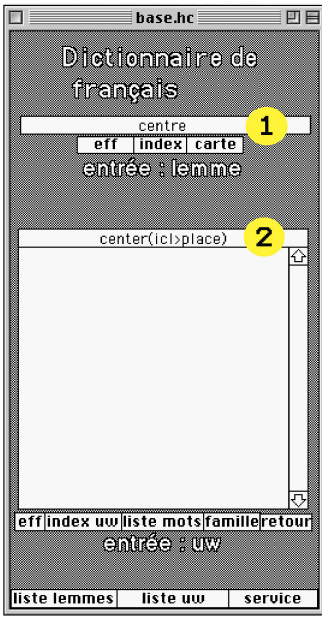

FIG. 1.9 : Page d'entrée du dictionnaire monolingue français de PARAX-UNL

La consultation multilingue se fait à partir d'une page d'entrée de dictionnaire monolingue (de n'importe quelle langue). Après avoir cliqué sur le bouton « `carte` » (accès par un mot-vedette) ou le bouton « `famille` » (accès par une UW), on obtient une fenêtre comme dans la figure 1.10. Pour trouver la traduction correspondante

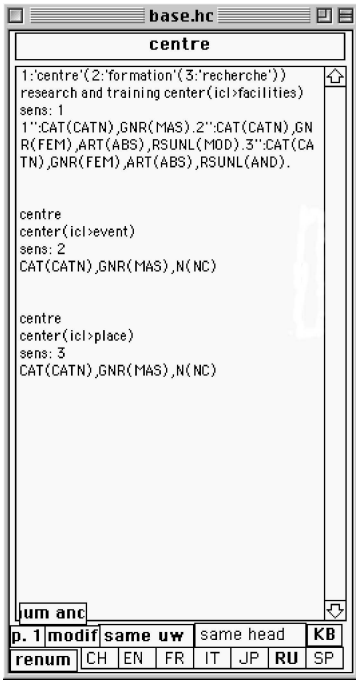

FIG. 1.10 : Liste d'UW pour le mot français « `centre` »

dans une autre langue, il faut d'abord sélectionner la langue, puis choisir « `same uw` », puis cliquer sur l'article choisi. Dans cet exemple, on a choisi le 3^{e} sens du mot-vedette « `centre` » avec la langue « `russe` ». On obtient alors l'équivalent russe de l'UW comme illustré dans la figure 1.11.

1.2.6 Papillon (2000 -)

Le projet Papillon [TMLP00, MLSL03] vise à la construction collaborative d'une base lexicale multilingue de grande taille pour un usage à la fois humain et machine. Les données lexicales de Papillon sont gratuites pour un usage non commercial. La base Papillon comprend l'allemand, l'anglais, le français, le japonais, le malais, le lao, le thaï, le vietnamien et le chinois. Le projet Papillon comprend la création d'un environnement coopératif permanent pour le développement et la consultation libre et personnalisable d'une base lexicale multilingue sur Internet. Papillon utilise l'approche interlingue pour représenter une base lexicale multilingue. Chaque acception monolingue dans la base est prévue pour être très détaillée, et suit le format DiCo [Pol00a, Lar02], version simplifiée du DEC [MCP95].

La structure commune des acceptions monolingues de la base Papillon est définie par un schéma XML (figure 1.12). Dans la suite de ce document, nous appelons une base Nadia les données de la base Papillon, qui suivent ce schéma.

Des balises <element> ayant la valeur minOccurs= « 1 » sont les balises d'informations obligatoires pour chaque lexie. Il s'agit des trois éléments « headword », « pos », et « semantic-formula ». Ensuite, le détail de chaque élément dans cette structure est

FIG. 1.11 : Équivalent en russe de UW « enter(icl->place) »

défini. Par exemple, l'élément « pos » est défini comme une chaîne de caractères, l'élément « lexical-functions » est défini comme une liste de « functions », etc. Chaque langue de la base peut définir sa propre liste de valeurs possibles pour chaque élément. Par exemple, dans l'élément « language-level », on a le type « usageType » défini différemment entre le français et le thaï.

– pour le français

```
<simpleType name="usageType">
  <restriction base="d :usageType">
    <enumeration value="vulgaire"/>
    <enumeration value="familier"/>
    <enumeration value="neutre"/>
    <enumeration value="formel"/>
  </restriction>
</simpleType>
```

– pour le thaï

```
<simpleTye name="usageType">
  <restriction base="d :usageType">
    <enumeration value="acronym"/>
    <enumeration value="foreign word"/>
    <enumeration value="colloquial express"/>
    <enumeration value="monk"/>
    <enumeration value="official palace language"/>
  </restriction>
</simpleType>
```

Le projet Papillon propose une plate-forme pour la construction manuelle des données lexicales, où chaque contributeur peut ajouter, modifier ou supprimer des données lexicales via Internet. Ces manipulations des données sont contrôlées.

Le tableau 1.1 synthétise les projets de base lexicale multilingue présentés dans cette section.

```
<element name="lexie">
 <complexType>
  <sequence>
   <element ref="d :headword" minOccurs="1" maxOccurs="1"/>
   <element ref="d :writing" minOccurs="0" maxOccurs="1"/>
   <element ref="d :reading" minOccurs="0" maxOccurs="1"/>
   <element ref="d :pronunciation" minOccurs="0" maxOccurs="1"/>
   <element ref="d :pos" minOccurs="1" maxOccurs="1"/>
   <element ref="d :language-level" minOccurs="0" maxOccurs="1"/>
   <element ref="d :semantic-formula" minOccurs="1" maxOccurs="1"/>
   <element ref="d :government-pattern" minOccurs="0" maxOccurs="1"/>
   <element ref="d :lexical-functions" minOccurs="0" maxOccurs="1"/>
   <element ref="d :examples" minOccurs="0" maxOccurs="1"/>
   <element ref="d :full-idioms" minOccurs="0" maxOccurs="1"/>
   <element ref="d :more-info" minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute ref="d :id" use="required"/>
 </complexType>
</element>
```

Fig. 1.12 : Structure générale des lexies Papillon

1.3 BDLM idéale

1.3.1 Catégorisation des bases dictionnairiques

Certaines caractéristiques des différentes bases lexicales ont été présentées dans la section précédente. Afin de fixer une base dictionnairique de caractéristique idéale, on peut examiner les dictionnaires selon les axes suivants : langue, structure, granularité, disponibilité, richesse de l'information, taille et usage.

Langue La langue est une des caractéristiques principales d'un dictionnaire. Globalement, les dictionnaires sont divisés en trois catégories : monolingues, bilingues et multilingues. Les dictionnaires monolingues contiennent des informations dans une langue. Les dictionnaires bilingues décrivent les données d'une langue, en donnant leur équivalent dans une autre langue. Les dictionnaires multilingues regroupent les données de plus de deux langues.

Projet	**Nombre de langues**	**Nombre d'entrées**	**Nombre d'acceptions**	**Licence**
EDR	2	250 000 ja 190 000 en	410 000	payant
ULTRA	5	7 000	10 000	non diffusé
EuroWordNet	8	45 000	75 000	payant
HowNet	2	81 000 zh 76 000 en	95 000	gratuit
PARAX	5	135 fr 300 en 380 de 390 ru n/a zh	590	non diffusé
PARAX-UNL	7	25 500 fr 20 900 it 90 000 ja 3 900 ru 700 es 6 100 zh 95 200 en		non diffusé
Papillon	9	821 fr 186 en 108 ja 67 ms 105 zh (en cours)	45 (en cours)	gratuit

* de = allemand, en = anglais, es = espagnol, fr = français, it = italien, ja = japonais, ru = russe, zh = chinois, ms = malais

TAB. 1.1 : Synthèse des caractéristiques des bases lexicales multilingues des projets présentés

Structure Cette dimension concerne la macrostructure d'une base dictionnairique, c'est-à-dire son organisation en volumes, où un volume est l'ensemble des entrées d'une même langue. La macrostructure la plus simple consiste en un seul volume. C'est le cas des dictionnaires monolingues et de certains dictionnaires bilingues. La macrostructure est plus complexe pour les dictionnaires multilingues, où nous distinguons deux types :

- dictionnaire d'une langue source vers N langues cible,
- dictionnaire de N langues source vers N langues cible.

Dans le premier cas, nous parlons de dictionnaire multicible ou *furcoïde* [BN86a, BN86b]. Cette structure permet aux utilisateurs de passer d'une langue source à plusieurs langues cibles. Le dictionnaire regroupe un volume pour la langue source et plusieurs volumes pour les langues cible. Notons que les dictionnaires bilingues peuvent être considérés comme un cas particulier de la structure en fourche, avec une seule langue cible. La figure 1.13 illustre la structure en fourche.

Pour le second cas, celui du dictionnaire de N langues vers N langues, il existe deux approches principales : *l'approche interlingue* et *l'approche (par) transfert.*

L'approche interlingue est une approche où plusieurs langues sont liées à un seul langage artificiel appelé *interlangue* et utilisé comme *langage pivot.* Sa structure globale est illustrée dans la figure 1.14.

L'approche (par) transfert (ou l'approche bilingue) a une structure globale formée d'un volume bilingue ou d'un ensemble de volumes bilingues, comme illustré dans la figure 1.15.

L'approche interlingue facilite l'extension et la maintenance : par exemple, si nous souhaitons ajouter une nouvelle langue dans un dictionnaire interlingue, la seule chose que nous devons faire est de créer des liens entre la nouvelle langue et le langage pivot du dictionnaire. Au contraire, dans une approche bilingue, pour ajouter une nouvelle langue, il faut la relier avec toutes les autres langues de la base. Il est donc plus facile d'ajouter une nouvelle langue dans une base interlingue que dans une base multilingue d'approche bilingue.

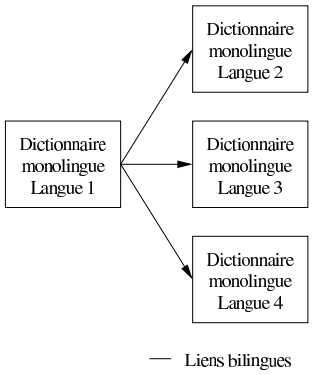

FIG. 1.13 : Structure en fourche

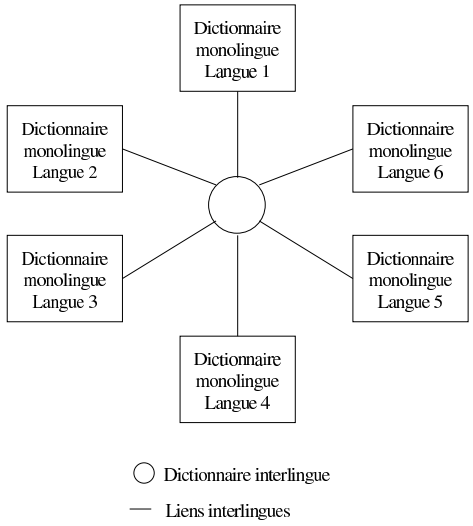

FIG. 1.14 : Approche interlingue d'une base lexicale multilingue

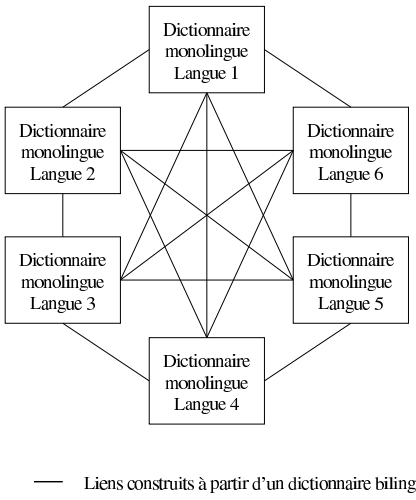

FIG. 1.15 : Approche de transfert d'une base lexicale multilingue

Granularité Cette dimension concerne la microstructure d'un dictionnaire, au niveau des entrées de chaque dictionnaire monolingue. Nous séparons les dictionnaires en deux catégories selon leur granularité : 1) les dictionnaires d'acceptions et 2) les dictionnaires de mots. Une acception est une représentation d'un seul sens. Dans les dictionnaires d'acceptions, chaque article représente un sens. Dans les dictionnaires de mots, chaque article a un mot-vedette différent, et regroupe un ou plusieurs sens de ce mot. Par exemple, l'entrée pour le mot français « avocat » contient deux acceptions : 1) avocat (homme de loi) et 2) avocat (fruit).

Richesse Les dictionnaires peuvent être aussi distingués suivant la richesse de l'information qu'ils contiennent. Le contenu d'un dictionnaire est soit général, soit spécialisé. Les dictionnaires généraux sont des dictionnaires de langue, dont l'information donne des renseignements sur les mots de la langue commune et leurs emplois. Les dictionnaires spécialisés sont, par exemple, des dictionnaires de synonymes, d'antonymes, d'homonymes, des dictionnaires étymologiques, des dictionnaires thématiques ou des dictionnaires orthographiques. Les dictionnaires spécialisés offrent une information différente selon leur type. Le dictionnaire DEC [Mel99] est un exemple de dictionnaire riche en information.

Taille Nous classons les dictionnaires en trois « tailles », suivant leur nombre d'articles. Nous rangeons les dictionnaires contenant moins de 10 000 vocables dans ceux de « petite taille ». En général, les entrées de ces dictionnaires sont des mots courants considérés comme essentiels. Les dictionnaires entre 10 000 et 50 000 vocables sont dits de « taille moyenne ». Ils comprennent les mots de la vie quotidienne, plus des mots pour les utilisateurs ayant atteint un certain niveau de scolarité, ou encore les termes techniques les plus fréquents ou les plus difficiles (pour les traducteurs) d'un domaine. Enfin, les dictionnaires de grande taille ont plus de 50 000 vocables.

Usage Globalement, les dictionnaires peuvent être séparés en deux grandes catégories, dictionnaires à usage automatisé, et dictionnaires à usage humain. Les dictionnaires à usage automatisé sont plus structurés et moins lisibles par les humains. Chaque information dans le dictionnaire à usage automatisé doit être annotée pour qu'elle puisse être interprétée par le programme. L'information dans le dictionnaire diffère selon les applications. Par exemple, un dictionnaire dans un système de traitement automatique des langues naturelles ne donne pas les informations telles que l'étymologie ou des exemples d'usage. En ce qui concerne les utilisateurs humains, les informations présentes/absentes dans un dictionnaire dépendent de la compétence supposée des utilisateurs, par exemple 1) les utilisateurs dont c'est la langue maternelle, 2) les apprenants d'une langue étrangère, et 3) les traducteurs. Les détails des informations destinées à des utilisateurs différents sont donc différents. Par exemple, un dictionnaire bilingue anglais-français général pour apprenants ou étrangers contient la prononciation, mais si l'utilisateur est supposé bien connaître l'anglais (c'est le cas des traducteurs), un dictionnaire bilingue anglais-français ne comprendra pas la

prononciation, comme dans le dictionnaire *Guide anglais-français de la traduction* [Mee04].

Disponibilité Nous faisons trois distinctions : 1) les dictionnaires disponibles et payants, 2) les dictionnaires disponibles et gratuits, et 3) les dictionnaires non diffusés. Dans le premier cas, ce sont des dictionnaires conçus dans un but commercial : les utilisateurs doivent acheter les données ou un droit d'utilisation. Le dictionnaire EDR[10], le dictionnaire Hachette-Oxford, le New Oxford Dictionary of English sont des exemples de ce type de dictionnaire. Le deuxième type de dictionnaire est formé des dictionnaires que l'on peut souvent récupérer via Internet. Enfin, les dictionnaires non diffusés sont les dictionnaires construits pour un usage restreint dans un groupe de recherche ou dans une application particulière, par exemple le dictionnaire de la maquette PARAX [Bla95, Bla99] ou du projet ULTRA [FW91].

1.3.2 Fixation progressive d'un but idéal des BDLM

Nous sommes maintenant en mesure de définir les caractéristiques idéales d'une base lexicale multilingue par rapport aux catégorisations définies dans la section 1.3.1. Nous considérons les caractéristiques suivantes comme les caractéristiques idéales qu'il est préférable de prendre en compte dans la construction de nouvelles bases lexicales multilingues.

- multi-usage : une base lexicale doit être à la fois utilisable par une machine et compréhensible par un humain.
- multi-théorie (multiniveau ; multicodage) : une base lexicale doit être ouverte à toutes les théories possibles. Par exemple, il existe deux théories contradictoires pour interpréter la langue thaï : l'une postule qu'il n'existe pas d'adjectifs dans la langue thaï, l'autre postule le contraire. Il faut que la base lexicale soit utilisable par (adaptée à) différents systèmes pour éviter de reconstruire une nouvelle base lexicale pour chaque système. Un moyen est, par exemple, de dupliquer et de coder différemment les informations pour chaque théorie, dans une même base lexicale.
- incrémentale : une base lexicale doit permettre aux développeurs de manipuler seulement une partie de la base lexicale sans avoir à considérer le reste de la base. Ainsi, pour modifier une partie de la base, il ne doit pas être nécessaire de recompiler toute la base.
- fondée sur la « sémantique linguistique », une branche de la sémantique qui étudie en particulier le sens des mots d'une langue, et sur les théories lexicologiques modernes pour assister la tâche de désambiguïsation dans la traduction.
- de grande taille : une base lexicale doit supporter un grand nombre d'entrées.
- très détaillée : chaque article dans une base lexicale doit être riche en informations.

[10]Electronic Dictionary Research : dictionnaire bilingue bidirectionnel japonais-anglais.

- en ressource libre : une base lexicale doit permettre d'accéder directement à ses données, et pas seulement offrir une visualisation des données qui nécessite une application spécifique.
- à construction collaborative : une base lexicale doit pouvoir être construite par des développeurs ou lexicographes de manière collaborative dans un style « bazar[11] », de la même manière que le développement du système Linux par exemple. Le travail en collaboration est très intéressant surtout dans le cas d'une base multilingue, dont la construction nécessite plusieurs lexicographes compétents dans des langues différentes.

1.4 Conclusion

Ce chapitre montre l'évolution des dictionnaires depuis le dictionnaire simple jusqu'au dictionnaire idéal qu'on souhaite. Certains projets de construction d'une base lexicale ont résolu les problèmes fondamentaux de type accès ou visualisation, mais le problème dur comme le choix des entrées de dictionnaires reste toujours difficiles à résoudre.

En comparant chaque projet avec une base lexicale idéale, le tableau 1.2 nous montre que le projet Papillon semble être le projet le plus proche du but idéal de BDLM et de la lexicographie multilingue informatique. Nous nous concentrerons donc sur ce projet. Une autre raison est que le projet Papillon a des fondations théoriques solides, qu'il s'agit de sa macrostructure ou de sa microstructure.

- La macrostructure de Papillon repose sur une structure *« pivot »* définie dans [Sér94]. Une « base Papillon » est composée de *lexies* (acceptions monolingues) et *axies* (acceptions interlingues qui sont des liens entre lexies).
- La microstructure de Papillon est fondée sur la structure *« DiCo »* qui est une simplification non abusive du dictionnaire DEC [Mel84, Mel88, Mel92, Mel99] visant à la production de masse.

Il existe déjà, dans le projet Papillon, des données et une plate-forme web de manipulation des données. La plate-forme Papillon offre une présentation unifiée de N dictionnaires existants. Elle offre également la possibilité de filtrage et de présentation dynamique des données ainsi que la possibilité de contribution à certains de ces dictionnaires. Nous étudierons plus en détail ce projet dans le chapitre 2.

[11] `http://www.catb.org/~esr/writings/cathedral-bazaar/`

Caractéristique	**EDR**	**ULTRA**	**EuroWordNet**	**HowNet**	**PARAX**	**PARAX-UNL**	**Papillon** (Nadia-DiCo)
multi-usage	-	+	+	-	+	+	+
multi-théorie	-	-	-	-	-	-	-
sémantique	+	+	+	+	+	+	+
de grande taille	+	-	+	+	-	+	+
très détaillée	-	-	-	+	-	-	+
en source ouverte	+	-	-	-	-	-	+
collaborative	-	-	-	-	-	-	+

(+) possède cette caractéristique, (-) ne possède pas cette caractéristique

TAB. 1.2 : Comparaison entre les projets et une base lexicale idéale

Chapitre 2

Les problèmes de structuration par acceptions

Les problèmes de structuration par acceptions concernent soit la partie des acceptions monolingues (lexies), soit la partie des acceptions interlingues (axies). Pour mieux comprendre la structuration par acceptions, la première section présente le projet Papillon de manière approfondie, et notamment sa stratégie de construction des données. La section 2.2 illustre ensuite les difficultés de la définition de lexies et d'établissement des axies, et motive notre travail. La dernière section introduit plus généralement les trois volets d'une solution à ce problème.

2.1 Approfondissement du concept de Papillon

Parmi les caractéristiques du projet Papillon, la suite présente l'architecture linguistique et lexicale, et la stratégie de construction des données. Dans le cadre de Papillon, notre travail concerne plus particulièrement une sous-partie des problèmes que Papillon tente de résoudre, et plus précisément la phase d'amorçage de Papillon.

2.1.1 Architectures linguistique et lexicale

Cette section présente les concepts de lexie et d'axie. En ce qui concerne la macrostructure, une structure pivot similaire à celle de Papillon a déjà été expérimentée dans PARAX [Bla99]. Une « base Papillon » est composée d'un ensemble d'acceptions monolingues (lexies) pour chaque langue et d'un ensemble de liens interlingues qui relient les lexies. Ces liens interlingues peuvent aussi être reliés à une liste d'éléments d'autres systèmes de codage ou de description sémantique (synsets de WordNet, universal words d'UNL, etc.). La figure 2.1 illustre un ensemble d'axies reliant des lexies de dictionnaires monolingues français, anglais et japonais.

Les problèmes contrastifs de l'équivalence lexicale sont traités grâce à des liens entre axies. La figure 2.2 illustre cela sur l'exemple de la traduction du mot « *chair* » en anglais par les mots « *chaise* » et « *fauteuil* » en français. Les mots « *chaise* » et

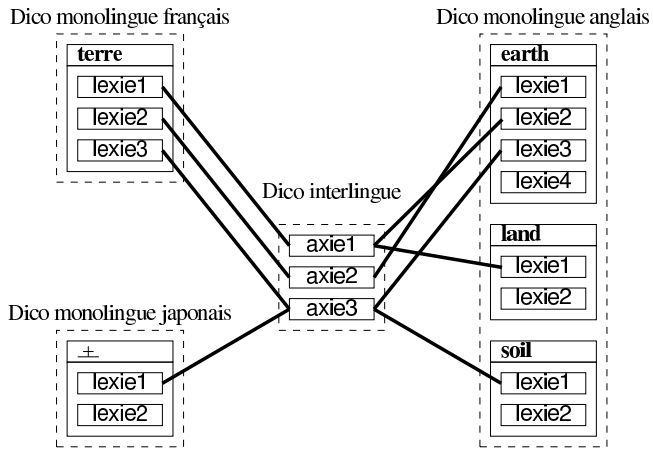

FIG. 2.1 : Fragment d'une base Papillon

« *fauteuil* » sont traduits par « *chair* » mais nous ne pouvons pas lier « *chaise* » et « *fauteuil* » à une même axie à moins que nous ne souhaitions les considérer comme des synonymes (ce qui serait une erreur ici). Il faut donc créer trois axies différentes pour relier ces acceptions monolingues, comme illustré dans la figure 2.2 : la première correspond à l'acception de « *chair* » ; la deuxième correspond à l'acception de « *chaise* » et la dernière correspond à l'acception de « *fauteuil* ».

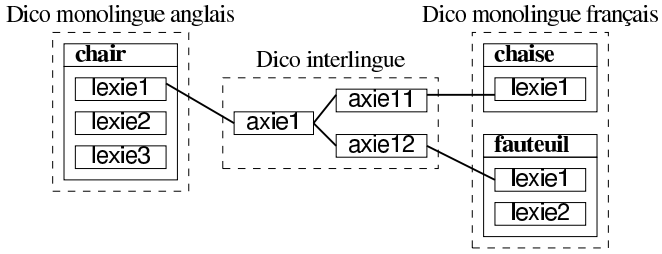

FIG. 2.2 : Utilisation de liens entre axies pour représenter les phénomènes contrastifs de l'équivalence lexicale

Quant à la microstructure de Papillon, la structure de chaque unité lexicale est issue de la théorie sens-texte d'Igor Mel'čuk et ses collègues [Mel81, MP87, Pol98]. Cette théorie fournit les informations nécessaires pour passer d'une idée (le sens) à sa réalisation dans une langue donnée (le texte). Le dictionnaire explicatif et combinatoire (DEC) [Mel84, Mel88, Mel92, Mel99] est basé sur cette théorie. Il comporte peu de vocables mais chacun est très détaillé. Les vocables sont divisés en lexies qui constituent les unités de base du dictionnaire. La microstructure du DEC [MCP95] est trop complexe pour être utilisée à grande échelle. Alain Polguère a simplifié les structures utilisées dans le DEC pour construire la base DiCo [Pol00b, Lar02]. La structure d'article dans la base Papillon suit la structure d'article de DiCo, et représente les entrées dans le format XML. La figure 2.3 donne un exemple d'un article de

Papillon. Ces données sont importées, à partir de fichiers XML, dans une base de données relationnelle. La base de données de Papillon est accessible principalement sur le site `http://www.papillon-dictionary.org/`. Ce site contient aussi une collection de dictionnaires « *classiques* », accessibles par une interface unifiée.

```
<lexie>
<headword>ABANDON</headword>
<pos>n.m.</pos>
<semantic-formula>acte :~ PAR personne X DE fait Y(X)</semantic-formula>
<government-pattern>
    X = I = par N, A-poss
    Y = II = de N, A-poss
</government-pattern>
<lexical-functions>
    <function name="QSyn">
        <valgroup><value>arrêt</value>
            <value>cessation</value>
            <value>suspension</value>
        </valgroup>
        <valgroup><value>abdication</value>
        </valgroup>
    </function>
    <function name="QAnti">
        <valgroup><value>continuation</value>
            <value>maintien</value>
            <value>poursuite</value>
        </valgroup>
    </function>
    <function name="Magn">
        <valgroup><value>complet</value>
            <value>total</value>
        </valgroup>
    </function>
    <function name="AntiMagn">
        <valgroup><value>partiel</value>
        </valgroup>
    </function>
</lexical-functions>
<example>
    Le syndicat réclame l'abandon complet de la réforme des lycées.
</example>
</lexie>
```

Fig. 2.3 : Exemple de lexie de la base Papillon.

2.1.2 Stratégie pour la construction de la base

La structuration en lexies et axies se fait en deux phases successives, la phase d'amorçage et la phase de contribution.

Phase d'amorçage Cette phase a pour but d'obtenir, à partir de dictionnaires existants, une première base lexicale contenant de nombreuses entrées associées à des informations minimales. Cette phase comprend à la fois la construction d'entrées monolingues (lexies) et celle d'entrées interlingues (axies). Plusieurs ressources lexicales sont disponibles, telles que des dictionnaires monolingues (4 000 acceptions françaises DiCo de l'université de Montréal, 10 000 vocables thaï de l'université Kasetsart), des dictionnaires bilingues (70 000 vocables japonais-anglais de JMDICT [Bre04] en format XML de Jim Breen et 10 000 vocables japonais-français de Jean-Marc Desperrier [Des02, Des]) et de dictionnaires multicible (20 000 vocables et 50 000 acceptions anglais-français-malais [FeM]).

Chaque ressource est transformée au format DML [ML01] (un schéma et un espace de noms XML) de Papillon, puis vers la structure de DiCo, avec plusieurs champs vides (notamment le régime et les fonctions lexicales).

L'étape suivante, la création des axies, est semi-automatique. Ce sont les problèmes de cette phase que notre travail de thèse essaie de résoudre.

Phase de contribution Une fois que la phase d'amorçage est terminée, et qu'il existe un ensemble minimal d'informations lexicales dans la base du serveur Papillon, la phase de contribution peut commencer. Dans cette phase, des utilisateurs coopèrent via Internet pour apporter des modifications à la base lexicale du serveur Papillon.

Trois tâches sont effectuées lors de l'évolution de la base :

- la contribution : tout utilisateur peut proposer des modifications telles que l'ajout d'informations dans des lexies existantes, l'ajout ou la suppression de lexies.
- la validation : les modifications effectuées par les utilisateurs seront acceptées directement dans le cas d'utilisateurs spécialistes, et après validation dans le cas d'utilisateurs non-spécialistes.
- l'intégration : des utilisateurs de confiance acceptent ou rejettent les contributions (validées ou non) pour qu'elles soient effectivement appliquées à la base.

La stratégie de la construction de la base Papillon [MLSL03] est illustrée dans la figure 2.4.

2.2 Problème de structuration par acceptions

2.2.1 Difficultés pour définir les lexies

La première difficulté est le choix de la façon de diviser un mot en un ou plusieurs sens. Les dictionnaires monolingues ont souvent différentes divisions en sens. Par exemple, dans le petit Larousse, le mot « *poésie* » a quatre sens, illustrés dans la figure 2.5. Par contre, dans le petit Robert, le mot « *poésie* » est divisé en six sens, comme illustré dans la figure 2.6.

Nous ne pouvons pas juger de la supériorité d'un dictionnaire par rapport à l'autre. Le problème est de déterminer au « mieux » le nombre de lexies pour un mot-vedette

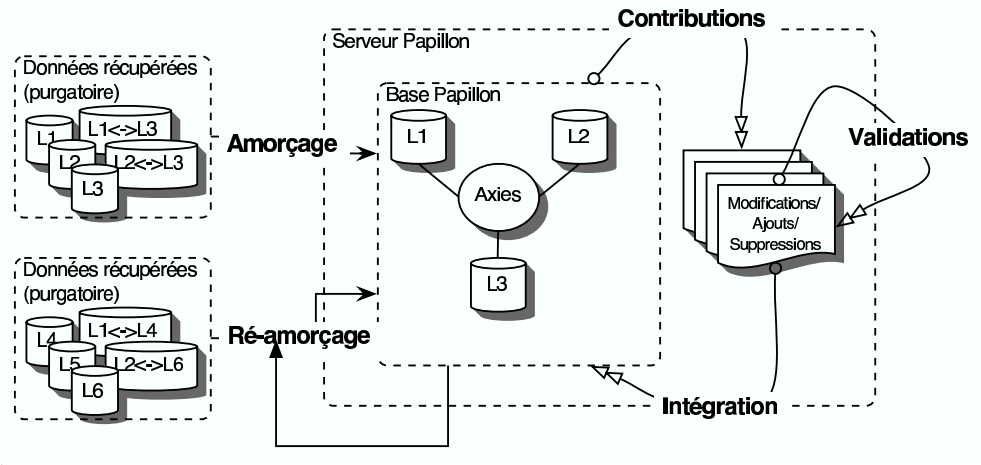

FIG. 2.4 : Stratégie de construction de la base Papillon

donné en associant si possible à chaque lexie une information (gloses si les données en contenaient, traits sémantiques, ou encore exemple(s) d'usage).

En outre, certains dictionnaires n'ont pas de division en sens explicite, mais les sens sont définis « par l'usage » ou « par l'exemple ». Il est donc plus difficile de diviser ces données en lexies. Ce problème est commun à tous les dictionnaires qui utilisent la notion de sens.

POÉSIE n.f

1. Art de combiner les sonorités, les rythmes, les mots d'une langue pour évoquer des images, suggérer des sensations, des émotions.
2. (Qualifié) Genre poétique.
3. Œuvre, poème en vers de peu d'étendue.
4. Caractère de ce qui touche la sensibilité, émeut.

FIG. 2.5 : Définition du mot « *poésie* » dans le dictionnaire Le petit Larousse

2.2.2 Difficultés pour établir des axies

Le problème d'établir des axies est le problème que tous les projets à pivot doivent résoudre. Pour établir la correspondance entre les lexies dans deux ou plusieurs langues, le problème est de trouver le moyen de comparer deux ou plusieurs lexies et de décider si elles représentent un même sens ou au moins deux sens différents. En effet, le problème de la désambiguïsation de sens d'un mot est un des problèmes les plus difficiles dans le domaine du traitement automatique des langues naturelles.

Par ailleurs, comme les langues sont différentes, il faut aussi prendre en compte le fait qu'il est possible que certains sens dans une langue ne trouvent pas de correspondance directe dans une autre langue. Par exemple, les mots français « fleuve »

POÉSIE n.f

1. Art du langage, visant à exprimer ou à suggérer par le rythme (surtout le vers), l'harmonie et l'image.
2. Manière propre à un poète, une école, de pratiquer cet art ; l'ensemble des œuvres où se reconnaît cette manière.
3. Poème (généralement assez court).
4. Propriétés poétiques qui peuvent se manifester dans toute œuvre d'art.
5. Qualité d'émotion esthétique (que peut éveiller un spectacle, un lieu, une situation).
6. Aptitude (d'une personne) à éprouver l'état, l'émotion poétique.

FIG. 2.6 : Définition du mot « *poésie* » dans le dictionnaire Le petit Robert

et « rivière » dans leur sens concret le plus commun, sont traduits en anglais par le mot « river » (dans son sens le plus commun). Les deux mots français ont deux sens différents que l'anglais ne distingue pas.

2.2.3 Trois volets à ce problème

Pour résoudre le problème de la structuration par acceptions, nous proposons de le diviser en trois volets.

Le premier volet est la création d'un graphe de lexies et d'axies initiales à partir de données existantes. Les problèmes sont ici 1) la récupération de données existantes, 2) l'intégration des données, et 3) la création des lexies et axies à partir des données récupérées.

Par exemple, supposons que nous avons plusieurs dictionnaires monolingues dans plusieurs langues. Chaque dictionnaire a sa propre représentation et sa propre structure. Il faut récupérer le maximum des informations contenues dans chacun des dictionnaires et les transformer en notre structure pour que nous puissions les traiter. Cependant, le problème de l'hétérogénéité peut se produire au moment de combiner deux dictionnaires pour créer des lexies. Une fois que des données monolingues sont prêtes, le problème suivant est d'établir des liens axies entre ces données monolingues.

Le deuxième volet est l'amélioration incrémentale des données initiales. C'est le problème à résoudre après avoir produit les données initiales. Dans cette partie, plusieurs linguistes ou contributeurs travaillent en collaboration sur une même base lexicale. Nous souhaitons que chacun puisse corriger des parties de données sans causer de problème ni modifier le reste de la base. Par exemple, nous voulons pouvoir vérifier et modifier une lexie ou une axie sans devoir recalculer toute la base à chaque fois que nous voulons modifier un élément dans la base.

Le dernier volet est l'évaluation des données produites. Lorsqu'on a construit la base de données, il faut vérifier la qualité de la base pour déterminer si c'est acceptable ou s'il faut encore des corrections. La qualité idéale d'une base interlingue peut être

définie par le fait que : 1) chaque base lexicale monolingue ne contient pas de lexies en doubles, 2) chaque lexie est liée à une seule axie, 3) chaque axie ne regroupe que des lexies qui ont une même signification.

2.3 Conclusion

Ce chapitre présente le projet Papillon et ses difficultés de la construction des données. Parmi les trois volets présentés dans la section précédente, nous nous intéressons au premier et au dernier volet : la création initiale et l'évaluation des données produites, car si nous voulons un environnement d'amélioration incrémentale collaborative, il faut :

- savoir signaler aux contributeurs les « endroits douteux », donc savoir évaluer la qualité de la base, globalement et localement,
- étudier les mécanismes de création, ce qui mènera à la définition d'actions élémentaires susceptibles d'être les « outils de base » de l'amélioration incrémentale, par exemple, la division d'une lexie ou d'un vocable en 2 lexies, le regroupement de deux axies qui ont une même signification, ou la création d'une nouvelle axie, etc.

Nous allons donc travailler sur :

- la création d'une structure de lexies/axies à partir de ressources multiples,
- l'évaluation d'une telle structure globalement et localement.

L'amélioration incrémentale reste finalement un travail manuel par des contributeurs. Le chapitre suivant présente plus en détail les problèmes de construction des données dictionnairiques.

Chapitre 3

Problèmes de construction des données dictionnairiques

Dans ce chapitre, nous abordons les problèmes de la construction de données dictionnairiques, et plus particulièrement chaque article dans un dictionnaire. Du fait qu'il existe de nombreuses données dictionnairiques, il est intéressant de construire un dictionnaire à partir de ces données existantes, pour réduire le temps de construction de la base. La section 3.1 présente des travaux pour la récupération des données existantes. La section 3.2 présente les travaux existants qui visent à compléter les informations manquantes dans les données récupérées. La section 3.3 présente les travaux sur la construction d'informations lexicales complémentaires pour déterminer le sens des données lexicales.

3.1 Récupération

Nous parlons ici de la récupération des dictionnaires sous forme électronique. Les dictionnaires sur papier sont d'ailleurs de plus en plus informatisés. Mais ils sont souvent sous la forme de fichiers non structurés au niveau des données linguistiques, comme par exemple des fichiers Word ou Excel. Il se pose donc le problème de l'analyse et de la récupération de ces données non structurées. Nous mentionnons ici trois travaux existants.

3.1.1 RÉCUPDIC

RÉCUPDIC [Hai98a, Hai98b, Tee02] est un logiciel développé au GETA pour la récupération de données dictionnairiques non structurées afin de produire des données utiles dans un format structuré. Le système s'est montré efficace en pratique, pour la récupération d'une grande quantité de ressources de diverses complexités. Environ 33 ressources ont été récupérées, soit au total 1.7 millions d'articles dans 12 langues, en 18 mois.

La méthode de « récupération » est constituée de 2 étapes :

1. On « normalise » la ressource de façon ad hoc, sous un éditeur (éventuellement en utilisant des macros), ou par des scripts perl, awk, tcsh, etc. Par exemple, si la ressource est en Word, on remplace chaque suite de blancs par un seul blanc, on met tout sous format texte en ajoutant au début de chaque paragraphe une balise contenant le nom de son style, ou passe en Unicode/UTF-8, etc.
2. On décrit la grammaire des articles en H-grammaire, et on utilise RÉCUPDIC. Les articles corrects sont transformés dans une représentation structurée, et les articles incorrects sont mis de côté pour une 2^{e} phase de normalisation et correction manuelle.

Voici quelques détails sur RÉCUPDIC et un exemple de normalisation RÉCUPDIC.

RÉCUPDIC se compose d'un formalisme de spécification de traduction (H-grammar) et d'un moteur d'exécution. Cet outil est un traducteur gouverné par la syntaxe[1] [ASU86].

Les entrées sont dans un fichier contenant la ressource lexicale qu'on veut récupérer et leur grammaire est décrite en H-grammar. Le résultat obtenu est un fichier dans le format LISPO (un format structuré spécifique de RÉCUPDIC).

Le formalisme H-grammar permet de définir des grammaires hors-contexte. Une grammaire de récupération se compose de six sections, introduites par des mots-clefs :

- `#grammar` : indique le nom de la grammaire,
- `#syntax-rules` : permet de définir des règles d'analyse syntaxique pour la récupération,
- `#start-symbol` : indique le symbole de départ (axiome) de la grammaire,
- `#lexical-rules` : permet de définir des règles d'analyse lexicale pour construire les items lexicaux,
- `#lexical-order` : permet de définir un ordre de préférence entre les items lexicaux,
- `#working-code` : permet d'écrire des fonctions Common Lisp et de les intégrer dans les règles syntaxiques.

Le cœur de H-grammar est la définition des règles d'analyse syntaxique. Ces règles sont de la forme :

```
:nom : A(ai1 ai2 ...; ao1 ao2 ...) ->
           B(bi1 bi2 ...; bo1 bo2 ...)
           C(ci1 ci2 ...; co1 co2 ...) ...
```

Le nom d'une règle d'analyse syntaxique est optionnel ; s'il existe, il est mis entre une paire de « : ».

`A` est un non-terminal ; `B, C, ...` peut être un non terminal, un terminal, le symbole nul §, ou une action. `ai1, ai2, ...` sont les variables d'entrée, qui sont initialisées lorsque la règle est appelée. `ao1, ao2, ...` sont les variables de sortie. `bi1, bi2, ..., ci1, ci2, ...` sont les expressions d'entrée (en syntaxe de LISP), qui peuvent contenir des variables.

Lorsqu'une expansion d'une unité de la partie droite se déroule, ses expressions d'entrée sont calculées.

[1] anglais : syntax-directed translator

- Si l'unité est un non-terminal, une règle dans la partie gauche sera choisie et développée.
- Si l'unité est un terminal, un token correspondant est cherché et retourné comme valeur de sa variable de sortie.
- Si l'unité est une action qui est en fait une fonction LISP, la fonction est appliquée aux valeurs de ses expressions d'entrée, et les résultats sont stockés dans ses variables de sortie.
- Si l'unité est un symbole nul §, les valeurs de ses expressions d'entrée sont affectées à ses variables de sortie, dans l'ordre.

Les valeurs des variables de sortie de l'unité non terminale de la partie gauche (`ao1, ao2, ...`) sont gardées et retournées comme résultat de son expansion.

Exemple de normalisation RÉCUPDIC nécessite une normalisation ad hoc des données avant son utilisation. Par exemple, dans la source de BABEL[2], les articles sont séparés par une nouvelle ligne, mais certains articles sont trop longs et occupent plusieurs lignes consécutives. Pour pouvoir utiliser le caractère de nouvelle ligne comme séparateur d'articles dans H-grammar, il faut supprimer les nouvelles lignes internes aux articles. En effet, les articles qui occupent plusieurs lignes ont une espace de 10 caractères blancs avant la ligne suivante. Pour supprimer les nouvelles lignes internes aux articles, on peut simplement remplacer une nouvelle ligne et les 9 caractères blancs qui suivent par une chaîne vide. Les données ainsi normalisées peuvent être ensuite récupérées par RÉCUPDIC.

```
COM       Commercial (organization Domain name) [Internet] +
          Common Object Model [Microsoft]
GMS       Global Management System +
          Global Messaging Service [Novell]
GMT       Greenwich Mean Time
GND       Ground (signal/system)
```

Tab. 3.1 : Articles extraits de BABEL

Un article de BABEL se compose d'une entrée qui est une abréviation/acronyme, et d'un ou plusieurs sens (Figure 3.1). Un sens se compose d'une expansion de l'abréviation/acronyme, d'une petite explication optionnelle entre parenthèses, et d'une spécification optionnelle de sujet ou domaine entre crochets.

Une grammaire pour les articles de BABEL est présentée dans la figure 3.1. Les règles d'analyse syntaxique sont les suivantes :

- La règle n° 1 produit un article BABEL `babel-entry` à partir du mot-vedette `hwd` et d'un corps `body`.
- La règle n° 2 produit un corps `body` à partir d'une liste de sens `sense*`.

[2]Un glossaire d'abréviations/acronymes informatiques anglais de Irving & Richard Kind (anglais : A Glossary of Computer Oriented Abbreviations and Acronyms)

```
#grammar babel-glossary

#syntax-rules

  :1 : babel-entry(;entry) -> >hwd(;hwd) body(;body)
                -- (#!babel((trim-whites hwd) body); entry).
  :2 : body(;body) -> sense(;S1) sense*(;S*)
                 -- cons(S1 S*; body).
  :3 : sense(;S) -> >exps(;exps) expl?(;expl) subj?(;subj)
               -- (#!sense((trim-whites exps)
                           (if expl (trim-whites expl))
                           (if subj (trim-whites subj))); S).
  :4 : expl?(;expl) -> "(" >to-cparen(;expl) ")".
  :5 : expl?(;expl) -> §(nil; expl).
  :6 : subj?(;subj) -> "[" >to-cbrak(;subj) "]".
  :7 : subj?(;subj) -> §(nil; subj).
  :8 : sense*(;S*) -> "+" sense(;S1) sense*(;S*1)
                  -- cons(S1 S*1; S*).
  :9 : sense*(;S*) -> §(nil; S*).

#start-symbol babel-entry

#lexical-rules

  hwd -> _^10.      /* mot-vedette prend 10 charactères */
  exps ->![+([]*.
  to-cparen -> >[)].
  to-cbrak -> >[\]].

#lexical-order ("+" "(" ")" "[" "]" hwd exps expl subj)

#working-code

  (sia-defclass babel () (hwd body))

  (sia-defclass sense () (exps expl subj))

  (defun trim-whites (string)
    (string-trim '(#\Space #\Tab #\Newline) string))
```

FIG. 3.1 : Exemple de grammaire en H-grammar

- La règle n° 3 produit un sens à partir d'une définition `exps`, d'une explication `expl` et d'un domaine `subj`.
- Les règles n° 4 et 5 produisent une explication `expl` à partir d'un texte entre parenthèses « `()` ».
- Les règles n° 6 et 7 produisent un domaine `subj` à partir d'un texte entre crochets « `[]` ».
- Les règles n° 8 et 9 produisent une liste de sens `sense*` à partir de 2 sens `sense` séparés par un « `+` ».

Cette grammaire est ensuite interprétée par un compilateur Lisp qui produit des objets LISP correspondant aux articles récupérées.

La figure 3.2 montre le résultat de la récupération de l'article BABEL original après compilation avec H-grammar :

Cet article BABEL après transformation est un objet LISP. Toutes les informations sont marquées explicitement. Il est alors très facile de les réutiliser automatiquement pour produire de nouveaux ensembles lexicaux.

```
(BABEL
   (HWD . ".com")
   (BODY LIST
   (SENSE
      (EXPS . "Command")
      (EXPL . "file name extension")
      (SUBJ . NIL))
   (SENSE
      (EXPS . "Commercial Business")
      (EXPL . "Domain Name")
      (SUBJ . "Internet"))))
```

Fig. 3.2 : Article de BABEL après récupération (objet LISP)

3.1.2 CDM de Papillon

Dans le projet Papillon [MLSL03], l'étape de la récupération des dictionnaires ou des ressources existantes se fait en trois parties, chacune constituée de tâches pouvant être réalisées en parallèle :

1. récupération « primaire » de toutes les ressources disponibles avec transformation du format source vers le format XML / DML et de l'encodage d'origine vers UTF-8,
2. fusion et intégration des données dans les dictionnaires monolingues de Papillon,
3. évolution par travail coopératif sur le Web.

Dans la première étape, la structure est récupérée et un maximum d'informations sont balisées de façon à pouvoir ensuite les réutiliser. Les ressources récupérées en XML / DML sont traduites vers le format CDM (Common Dictionary Markup) défini

dans [ML01]. CDM est un mécanisme de pointeurs dans une structure XML avec des balises pour identifier les différentes informations contenues dans les dictionnaires. Ce format est suffisamment général pour structurer des données d'une grande variété de dictionnaires. Le tableau 3.2 donne un exemple de la correspondance entre les éléments de CDM et les éléments des autres dictionnaires balisés tels que les dictionnaires TEI [IV96], FeM (français-anglais-malais), Oxford-Hachette (OHD) et Oxford (NODE). À partir du format CDM, on a l'accès unifié en lecture.

Le dictionnaire FeM est un exemple de ressource récupérée par le projet Papillon[3]. Le dictionnaire FeM a été récupéré du format original au format LISPO selon la méthode RÉCUPDIC (cf. section 3.1.1). Le résultat du format LISPO a été converti vers XML avec un programme LISP. La figure 3.3 illustre un extrait de l'article `abandonner` après cette conversion.

```
<HFEM>
    <FRE>abandonner</FRE>
    <PRNC>aban-done-</PRNC>
    <BODY lispo="BODY1">
      <SENSE* lispo="LIST">
        <SENSE>
          <CAT* lispo="LIST">v.tr.</CAT*>
          <SENSE1* lispo="LIST">...</SENSE1*>
          <SENSE1* lispo="LIST">
            <SENSE1>
              <GLOSS>renoncer à</GLOSS>
              <TRANS* lispo="LIST">
              <TRANS>
                <ENG* lispo="LIST">to give up</ENG*>
                <ENG* lispo="LIST">to abandon</ENG*>
              </TRANS>
              </TRANS*>
              <EXPL* lispo="LIST">
              <EXPL>
                <FRE>il a abandonné son projet</FRE>
                <ENG>he had gave up his project</ENG>
              </EXPL>
              </EXPL*>
            </SENSE1>
          </SENSE1*>
          <SENSE1* lispo="LIST">...</SENSE1*>
          <SENSE1* lispo="LIST">...</SENSE1*>
        </SENSE>
      </SENSE*>
    </BODY>
</HFEM>
```

FIG. 3.3 : Extrait de l'article `abandonner` converti de LISPO vers XML

[3]L'exemple est tiré de la thèse [ML01]

Elément CDM	équivalent TEI	FeM	OHD	NODE
<entry>	(entry)	<fem-entry>	<se>	<se>
<headword>	(hom)(orth)	<entry>	<hw>	<hw>
<pronunciation>	(pron)	<french_pron>	<pr><ph>	<pr><ph>
<etymology>	(etym)			<etym>
<syntactic-sense>	(sense level="1")		<sense n=1>	<s1>
<pos>	(pos)(subc)	<french_cat>	<pos>	<ps>
<lexie>	(sense level="2")		<sense n=2>	<s2>
<indicator>	(usg)	<gloss>	<id>	
<label>	(lbl)	<label>		<la>
<example>	(def)	<french_sentence>	<ex>	<ex>
<definition>	(eg)			<df>
<translation>	(trans)(tr)	<english_equ> <malay_equ>		<tr>
<collocate>	(colloc)		<co>	
<link>	(xr)	<cross_ref_entry>	<xr>	<xg> <vg>
<note>	(note)		<ann>	

TAB. 3.2 : Correspondance entre les éléments CDM et des éléments des dictionnaires TEI, FeM, Oxford-Hachette, et Oxford

Cet article est réparti automatiquement en lexies et axies qui sont ensuite intégrées dans la base lexicale (cf figure 3.4).

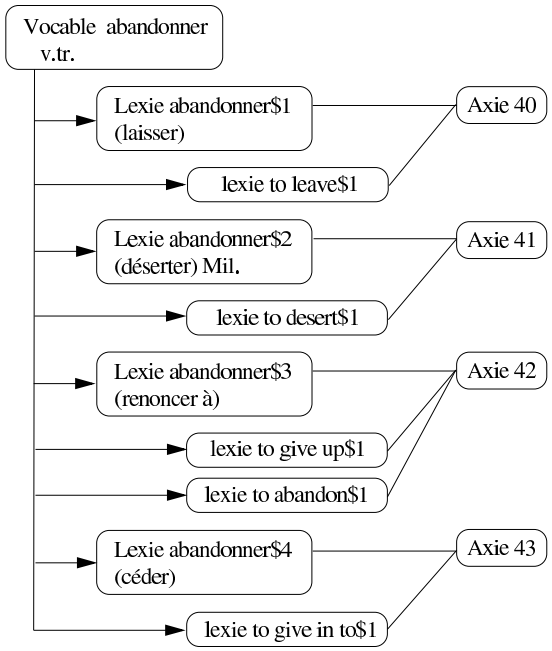

FIG. 3.4 : Répartition d'un article du FeM en lexies et axies

L'article `abandonner` a généré automatiquement quatre lexies françaises correspondant à tous les sens de l'article. Les identificateurs de ces sens (lexies) sont numérotés de `abandonner$1` à `abandonner$4`. Les informations spécifiques au FeM sont stockées dans l'élément `<fem>`. Elles serviront par exemple à régénérer ensuite l'article original. Voici par exemple la lexie `abandonner$3` :

```
<lexie id="abandonner$3" basic="no">
    <headword>abandonner</headword>
    <pronunciation encoding="GETA">aban-done-</pronunciation>
    <pos>v.tr.</pos>
    <fem><gloss>renoncer à</gloss></fem>
    <axies><refaxie href="a42"/></axies>
</lexie>
```

Cette lexie est reliée à l'axie dont l'identificateur est `a42`.

Le traitement de l'article du FeM a généré automatiquement cinq lexies anglaises correspondant aux cinq traductions anglaises se trouvant dans l'article. Voici par exemple la lexie anglaise `to abandon$1` reliée à la lexie française précédente par l'intermédiaire de l'axie `a42` :

```
<lexie id="to abandon$1" basic="yes">
    <headword>to abandon</headword>
    <fem-data><gloss>renoncer à</gloss></fem-data>
    <axies><refaxie href="a42"/></axies>
</lexie>
```

Le traitement de l'article du FeM a généré automatiquement cinq axies reliant chacune une lexie française et une lexie anglaise. Voici par exemple l'axie `a42` reliant les deux lexies précédentes.

```
<axie id="a42">
    <fra><reflexie href="abandonner$3"/></fra>
    <eng><reflexie href="to abandon$1"/></eng>
</axie>
```

3.1.3 Réutilisation du lexique-grammaire du LADL

Contrairement aux deux travaux généraux présentés ci-dessus qui visent à la récupération de tous types de ressources, le travail présenté dans cette section vise à la récupération d'une seule ressource particulière : le lexique-grammaire du français [Gro75] du LADL[4]. Le lexique-grammaire (LG) des verbes et des autres éléments prédicatifs (adjectifs, noms, et adverbes) a été construit sous la forme de tables de propriétés. Les tables de LG comportent un nombre significatif d'entrées (plus de 10 000 entrées). Chaque table représente un groupe de verbes qui ont la même structure de base.

La figure 3.5 donne un extrait d'une table de LG. Chaque ligne représente une entrée V, et chaque colonne correspond à une propriété définie dans la table. Par exemple, dans la figure 3.5, la première colonne spécifie le constituant de sujet de chaque entrée V. Les colonnes 8 à 11 spécifient les suffixes adjectifs de V. Pour chaque colonne, la marque « + » signifie que cette entrée a la propriété correspondant à cette colonne, la marque « - » signifie le contraire.

sujet																				
N0 =: Nhum	N0 =: Nnr	N0 =: le fait Qu P	NO =: V1 W		V "concret"	N0 V	Adjectif				Comp. direct			N1 se V de ce Qu P	N1 se V auprès de N3 de ce Qu P	N1 est Vpp de ce Qu P	[passif par]	[passif de]	N0 V N1 contre Nhum
							-a =: -ant	-a =: -able	-a =: -eux	-a =: -(at)eur	N1 =: Nhum	N1 =: N-hum	N1 =: le fait Qu P						
+	+	+	+	abasourdir	−	+	+	−	−	−	+	−	−	−	−	+	+	−	−
+	+	+	+	abattre	+	+	−	−	−	−	+	+	−	+	−	+	+	−	−

Fig. 3.5 : Un extrait de la table 4 de LG

[4]Laboratoire d'Automatique Documentaire et Linguistique (LADL, laboratoire du CNRS)

Cependant, ces données sont difficiles à récupérer parce que leur représentation n'est pas triviale à interpréter. Le but des travaux de N. Hathout et F. Namer [HN98] est donc de normaliser et transformer LG vers le format de lexique intermédiaire (IL) du formalisme PATR-II, pour mieux représenter les informations codées dans les tables de LG.

La manière de procéder consiste à considérer les tables les unes après les autres et à convertir le contenu de chaque table en un ensemble d'entrées lexicales, chaque entrée étant associée un usage particulier d'un verbe pour l'ensemble des propriétés linguistiques que lui attribue la table. Plus précisément, une procédure générale de conversion peut être décrite par deux étapes, pour une table T quelconque :

- pour chaque ligne L de la table T, créer une entrée lexicale associant le verbe V concerné par la ligne L au cadre des fonctions syntaxiques de base associé à T ;
- enrichir chaque entrée lexicale créée à l'étape précédente en utilisant le contenu des colonnes de la table T.

La traduction des tables LG en un ensemble d'entrées lexicales IL est elle-même réalisée en deux étapes. La première étape se fait manuellement par l'étude des en-têtes des colonnes : on cherche à déterminer l'effet d'une colonne (enrichissement d'une entrée lexicale existante ou création d'une nouvelle entrée) et à formaliser cet effet par le biais d'une procédure. Comme on retrouve certaines en-têtes d'une table à l'autre, on peut réutiliser les procédures créées pour des tables précédentes. La deuxième étape consiste à répéter l'application des procédures ainsi créées à l'ensemble des verbes et à l'ensemble des colonnes d'une table donnée.

Les IL sont utilisés comme entrée d'un système qui génère deux ressources lexicales, basées respectivement sur les théories linguistique HPSG [PS88, PS94] et TAG [Jos87], et destinées à être utilisées dans des applications de traitement automatique du français.

3.1.4 Discussion

Les lexicologues ont besoin d'outils « guidés par les attentes » plutôt que d'outils « guidés par les données » tels que RÉCUPDIC, car les données sont trop variées et incohérentes. De plus, RÉCUPDIC est un outil pour informaticiens spécialisés qui est difficile à utiliser. La difficulté avec RÉCUPDIC est de décrire la grammaire d'un dictionnaire avec H-grammar. La conclusion pour la récupération est que, soit on récupère au cas par cas avec un programme ad hoc, soit on peut faciliter la construction d'un analyseur / convertisseur automatique, mais on ne peut jamais complètement automatiser.

Par ailleurs, bien que nous sachions structurer un dictionnaire existant (par exemple avec RÉCUPDIC) et unifier des éléments de structure de dictionnaires différents (par exemple avec CDM), cela ne suffit pas pour la récupération des données. Il reste encore le problème, par exemple, de l'identification des sens de deux dictionnaires différents.

3.2 Construction d'informations « usuelles » manquantes

Dans le cas où des données lexicales sont récupérées à partir de dictionnaires existants, il est nécessaire de compléter des informations manquantes telles que, par exemple, des informations sur les classes morpho-syntaxiques, des exemples d'usage, des synonymes dans la même langue ou des traductions dans une autre langue. De telles informations peuvent être nécessaires dans le dictionnaire en cours de production, mais non disponibles dans les ressources récupérées.

Les travaux visant à la construction des dictionnaires ont essayé d'inclure le maximum d'informations usuelles provenant des données lexicales.

Le projet Papillon n'est à présent pas encore entré dans la phase de contribution utilisant l'interface de contribution individuelle. En revanche, cet aspect collaboratif a été un succès dans certains projets tels que le projet JEDict [Bre04, Breb] sur le modèle du dictionnaire JEDict anglais-japonais, et le projet de dictionnaire français-anglais-japonais [Des02, Des] dirigé par Jean-Marc Desperrier qui contient environ 14 700 entrées courantes du projet EDict [Brea] traduites par des volontaires.

On peut aussi mentionner le succès récent de l'outil ITOLDU [BK04, BBK05] pour l'enseignement des vocabulaires techniques d'une langue étrangère. ITOLDU est un site web conçu au départ pour aider l'enseignement de l'anglais technique à des élèves ingénieurs français.

Les étudiants doivent ajouter la traduction de termes anglais proposés dans leur « dictionnaire personnel » ainsi qu'un exemple d'usage en anglais. Ils peuvent voir ce qu'ont proposé les autres et « adopter » la proposition d'un autre étudiant. Un étudiant gagne des points quand il ajoute quelque chose à son dictionnaire, ou quand une de ses contributions est adoptée. Pour la première expérimentation, portant sur la traduction de vocabulaires techniques anglais-français, ITOLDU a permis de collecter plus de 10 000 entrées avec le travail de 250 étudiants en 4 mois, et plus de 18 000 entrées en 8 mois.

3.3 Construction d'informations lexicales complémentaires

En complément des informations usuelles considérées dans la section 3.2, des informations plus détaillées, telles que la formule sémantique, le régime, la fonction lexicale, et les collocations, sont presque toujours manquantes dans les ressources récupérées. La construction de ces informations est un problème de recherche difficile. Jusqu'à présent, la seule façon de les obtenir est manuelle et très artisanale. Il n'existe donc qu'un petit nombre de travaux sur ce sujet : le projet DEC, le travail de Jurij D. Apresjan et al., et le système ETAP [ABI+03].

DEC Le projet DEC (Dictionnaire Explicatif et Combinatoire du français contemporain [Mel88, Mel92, Mel99, Mel84]) est un résultat de recherche sur la description formelle du lexique français, selon une approche sémantique. Le DEC est élaboré à

partir des principes de la lexicographie explicative et combinatoire [MCP95] issue de la théorie linguistique Sens-Texte [MP87, Pol98]. Comme il s'agit d'un travail de recherche en lexicographie, le DEC comprend peu de vocables, mais chacun d'entre eux est très détaillé. La figure 3.6 illustre un exemple d'article du DEC. Chaque article est une lexie. Chaque lexie comprend trois types d'informations, ou sections principales :

- section sémantique : définition,
- section syntactico-combinatoire : schéma de régime,
- section lexico-combinatoire : fonctions lexicales.

Chaque lexie comprend également des connotations, des marques d'usage, ainsi que des spécifications orthographiques, prosodiques, pragmatico-culturelles et même encyclopédiques. Le schéma de régime est un tableau qui présente explicitement tous les actants syntaxiques du lexème, en spécifiant pour chacun sa forme de surface (infinitif, syntagme prépositionnel) et son interprétation sémantique (l'actant sémantique qui lui correspond). De plus, le schéma de régime mentionne toutes les restrictions sur la cooccurrence des différents actants syntaxiques du même lexème.

En général, les articles du DEC sont écrits par des étudiants du séminaire de lexicologie de Igor Mel'čuk. Chacun prend un ou plusieurs vocables et écrit tout l'article. Chaque personne travaille sur un champ lexical donné. Par exemple, un étudiant décrira les vocables liés au domaine des télécommunications, un autre les vocables liés aux émotions, etc. Ensuite, les articles sont revus par des lexicographes expérimentés comme Igor Mel'čuk, Lidjia Iordanskaja, Alain Polguère, etc. Finalement, le tout est revu par des collègues du département[5] et on obtient le dictionnaire final.

Tolkovanija Le travail de Jurij D. Apresjan et de ses 10-12 collaborateurs à Moscou, sur le dictionnaire russe, concerne la description du lexique. Le principal résultat est *Tolkovanija* [Apr94], le dictionnaire de définitions, ou d'explications des mots ou des entrées lexicales. Dans les années 60-70, Jurij Apresjan et Igor Mel'čuk ont travaillé dans la même équipe à Moscou sur les recherches fondamentales en sémantique lexicale et en lexicologie, ce qui fait que le travail du DEC et du Tolkovanija s'inspirant de deux théories lexicographiques très proches.

Tolkovanija comprend quatre fonctions :

1. expliquer la signification d'une unité linguistique donnée,
2. fournir l'information de base pour trouver la sémantique, correspondant à une unité linguistique donnée,
3. fournir l'information de base pour mettre en relation la présentation syntaxique et la présentation sémantique,
4. donner l'information de base pour la combinaison sémantique entre une unité linguistique donnée et d'autres unités linguistiques.

Chaque entrée de dictionnaire est donc très détaillée, et le dictionnaire comporte seulement quelques milliers d'entrées.

[5] L'Observatoire de linguistique Sens-Texte (OLST), département de linguistique et de traduction, Université de Montréal

ACHAT, nom, masc.

1a. S_0 (*acheter 1*) [*l'achat par Marie d'une robe*]

1b. Activité commerciale - ensemble de tous les achats 1a ... [*les achats, par l'URSS, de céréales au Canada*]

2. Ce que X a acheté 1 ... [*Pierre m'a montré ses derniers achats*]

1a *Achat par X de Y à Z pour W* = S_0(*acheter 1*)

Régime

1 = X	2 = Y	3 = Z	4 = W
1. *de* N 2. *par* N 3. A_{poss}	1. *de* N	1. *à* N 2. $Loc._{in}$ N	1. *de* Num N 2. *pour* N

1) $C_{3.2}$	:	*N désigne un commerçant ou une entreprise commerciale*
2) $C_{4.2}$ sans C_2	:	*impossible*
3) $C_{1.2}$ sans C_2	:	*non souhaitable*
C_1	:	*les achats de Pierre, ses achats*
C_2	:	*un achat de marchandises*
C_3 + C_4	:	*un achat de 15 dollars à cette entreprise <chez un épicier >*
C_1+ C_2 + C_3 + C_4	:	*l'achat par Marie d'une robe chez la couturière pour cinquante dollars*
Impossible	:	**l'achat pour 3 000 dollars (2) [= l'achat de 3 000 dollars]*
Non souhaitable	:	*les derniers achats par l'entreprise (3) [= les derniers achats de l'entreprise]*

Fonctions lexicales

Syn_o	:	acquisition 1
Syn_p	:	emplette 1
$Conv_{3214i}$	:	vente 1a
Gener	:	transaction
Mult	:	achats 1b
$Magn^{quant_2}$	:	massif
$Magn_4$	:	important, grand | prépos
$Oper_1$	:	faire, effectuer [ART ~]
$Func_4$	:	s'élever [à N]

Exemples

C'était justement l'achat d'un trousseau qui retardait un peu son arrivée [F. Mauriac]. La reine le prie de lui avancer l'argent pour l'achat de cette parure dont elle rêve. Le gouvernement autorise l'achat de nouvelles machines agricoles pour soixante millions de francs.

FIG. 3.6 : Exemple d'article du DEC

ETAP Le processeur linguistique ETAP-3 [ABI+03, ABIT03, BIS04] constitue un environnement multifonctionnel pour le TALN. Cet environnement est essentiellement basé sur la théorie Sens-Texte d'Igor Mel'čuk [Mel81, Pol98], et la théorie lexicographique systématique de Jurij D. Apresjan [Apr00]. Les dictionnaires principaux utilisés dans ETAP sont des dictionnaires combinatoires de l'anglais et du russe, avec chacun plus de 65 000 entrées. Chaque entrée de ces dictionnaires comprend les informations suivantes : parties du discours, traduction dans une autre langue, caractéristique syntaxique, caractéristique sémantique, régime (government pattern), fonctions lexicales, ainsi que plusieurs règles pour aider à la traduction de cette entrée vers une autre langue. La figure 3.7 donne un extrait d'une entrée du dictionnaire anglais de l'ETAP, qui illustre la quantité d'information disponible dans chaque entrée.

```
CONTROL 1, a noun.
V0 : CONTROL 2
MAGN : STRICT / RIGID
ANTIMAGN : LAX
OPER1 : HAVE
INCEPOPER1 : ESTABLISH / SET UP
FINOPER1 : LOSE
LIQUOPER1 : DEPRIVE (somebody of control)
OPER2 : BE (under control)
INCEPOPER2 : FALL (under control)
FINOPER2 : GO OUT (of control)
LIQUOPER2 : FREE (somebody of control)
LABOR1-2 : KEEP (somebody under control)
INCEPLABOR1-2 : TAKE (somebody under control)
LIQUFUNC2 : CANCEL (control over something)
```

FIG. 3.7 : Extrait d'une entrée du dictionnaire anglais de l'ETAP

Pour résoudre le problème de la construction d'informations spécialisées, il est également possible de chercher à étendre l'approche ITOLDU (avec l'aide des étudiants, voir la section 3.2), qui va bien pour des équivalents de termes techniques, à des informations telles que les fonctions lexicales ou d'autres collocations plus ou moins libres. Cependant, dans notre cas, c'est un problème à résoudre après celui de la structuration.

3.4 Conclusion

Le succès d'un projet de construction manuelle de dictionnaire en collaboration dépend de la capacité à inciter des contributeurs à contribuer au projet. Par exemple, le projet ITOLDU [BBK05] a été un succès grâce à la contribution d'étudiants dans le cadre de leur formation. Le projet Yakushite.Net d'Oki Electric[6] [SKSM01, MKFS03] a eu du succès en offrant l'accès au système de traduction automatique PENSEE [SMI+99] aux contributeurs pour la traduction bilingue anglais-japonais. A priori, le projet Papillon [TMLP00, MLSL03] risque d'avoir un succès limité, parce qu'il n'y a pas d'incitations à contribuer au projet. Le tableau 3.3 résume cette discussion.

[6] `http://www.yakushite.net/`

Projet	Volontaire	Gratuité	Succès	Raison de contribuer
Papillon	oui	oui	-	-
ITOLDU	non	oui	oui	pour les études
yakushite.net	oui	non	oui	accès à l'outil de traduction
JMDict	±	oui	oui	intérêts personnel

TAB. 3.3 : Synthèse des projets et de leur succès du point de vue de la construction collaborative

Conclusion

Dans cette première partie, nous avons mis en relief deux problèmes cruciaux pour la production d'une base lexicale multilingue à acceptions, et par là de dictionnaires entre toutes les paires de langues traitées :

- le problème de la structuration de la base lexicale,
- le problème de la construction des données manquantes ou spécialisées.

Notre travail est centré sur l'automatisation de la production de dictionnaires. Dans la suite de ce document, nous abordons seulement le premier problème, car le deuxième problème ne peut pas être traité automatiquement par informatisation.

Le chapitre 2 a détaillé les trois étapes de la structuration :

1. la production automatique,
2. l'amélioration incrémentale par des contributeurs individuels, éventuellement organisés en groupes intéressés par tel ou tel aspect,
3. l'évaluation des données produites.

Nous nous intéressons seulement à la création et à l'évaluation, car l'amélioration incrémentale par des contributeurs individuels est un travail principalement manuel, comme indiqué dans la conclusion, page 35.

Pour résumer, le travail présenté dans la suite concerne la création d'une structure de lexies et d'axies à partir de différentes ressources, et l'évaluation d'une telle structure globalement et localement. Notre objectif est d'automatiser au maximum ce travail, car la construction manuelle de bases lexicales est extrêmement coûteuse.

Les problèmes principaux de la construction manuelle de bases lexicales que l'on peut rencontrer sont par exemple : 1) le temps de construction, 2) le manque de lexicographes compétents et 3) l'édition collaborative.

Pour le premier problème, les travaux antérieurs comme [DJ96, CBV+95] montrent que la construction manuelle d'une entrée lexicale (monolingue ou multilingue) peut prendre jusqu'à 30 minutes. Le temps de construction d'une base entière est donc très grand. Néanmoins, le temps de construction peut diminuer en partageant des données entre les lexicographes. Dans ce cas, il faut contrôler la cohérence entre ceux-ci.

Le deuxième problème pour la rédaction manuelle se pose dans le cas de dictionnaires multilingues de plus de deux langues. Pour établir un lien entre plusieurs langues, le lexicographe doit connaître toutes les langues concernées. Pour les langues bien répandues telles que l'anglais, l'espagnol et le français, il est facile de trouver des lexicographes qui ont une connaissance approfondie de ces langues. Mais pour des

langues moins connues telles que le malais et le thaï, il est difficile voire impossible de trouver des lexicographes qui sont compétents dans toutes ces langues.

Les lexicographes compétents sont d'autant plus difficiles à trouver que le nombre de langues considérées augmente. Une façon de construire un dictionnaire multilingue avec un grand nombre de langues est de réunir des lexicographes qui ont chacun une connaissance sur les langues différentes mais qui couvrent ensemble toutes les langues de la base lexicale, pour établir des liens multilingues.

Un problème difficile de l'édition collaborative est d'assurer la cohérence d'un dictionnaire lorsqu'il est rédigé par plusieurs lexicographes. Il s'agit de la cohérence sur la forme spécifiée par le lexicologue, comme les abréviations, les balises, etc., et aussi sur le fond, à savoir de sélection des sens, les critères de décomposition en entrées et sous-entrées, etc.

Le contrôle de la cohérence est plus difficile dans le cas de la construction d'un dictionnaire multilingue que dans celui de la construction d'un dictionnaire monolingue. En effet, dans le cas d'un dictionnaire multilingue, il peut y avoir des lexicographes de plusieurs nationalités qui travaillent sur une partie différente concernant des langues qu'ils connaissent.

L'objectif de la construction automatique ou semi-automatique de dictionnaires est 1) de pallier les limitations de la construction manuelle, par exemple l'absence de lexicographes compétents pour toutes les langues considérées, et/ou 2) d'accélérer le travail des lexicographes.

En effet, la plupart des méthodes de construction automatique ont été proposées pour la construction de dictionnaires bilingues ou multilingues. Pour rendre automatique la construction, l'étape la plus difficile est de relier les lexies qui ont une même signification. Les méthodes proposées pour la création automatique sont donc des techniques de désambiguïsation et de sélection des liens.

Cependant la méthode manuelle est toujours nécessaire pour construire les ressource de base, par exemple les dictionnaires monolingues. La construction manuelle est donc complémentaire de la construction automatique.

Deuxième partie

Approches du problème de la structuration d'une BDLM en lexies et axies

Chapitre 4

Approches de la structuration et de l'évaluation

L'objectif de ce chapitre est de présenter les approches existantes de la structuration et de l'évaluation. À partir de l'étude de ces approches, nous proposons un ensemble d'opérations de base qui permettent de décrire toute approche pour la structuration et l'évaluation. La première section donne une vue globale sur les approches existantes de la structuration, et la section 4.2 propose une catégorisation des critères d'évaluation des bases lexicales ainsi que les méthodes utilisables. Nous expliquons ensuite dans la section 4.3 la similitude entre les critères pour la structuration et l'évaluation.

4.1 Approches de la structuration

Le problème de la structuration est d'abord un problème de sémantique d'un mot. Pour pouvoir relier des articles de plusieurs langues, il faudra distinguer le sens de ces articles, ce qui mène au problème de la désambiguïsation. Depuis plusieurs dizaines d'années, le problème de la désambiguïsation est un des problèmes difficiles dans le domaine du traitement automatique des langues naturelles. De nombreux efforts ont été consentis pour résoudre ce problème. En ce qui concerne la construction des bases lexicales, nous décrivons dans cette section l'approche ontologique, l'approche linguistique et traductionnelle, et l'approche sémantique.

4.1.1 Opérations de base

En général, la structuration d'une BDLM en lexies et axies se décompose en deux parties : la distinction des sens de chaque vocable en lexie(s), et la création des liens entre plusieurs lexies de plusieurs langues : les liens axies.

Pour la création des liens axies, les opérations de base que nous avons considérées sont les opérations que l'on peut effectuer lorsqu'il y a un problème structural, par exemple, si une lexie est reliée à deux axies. Les opérations de base qui peuvent être

effectuées sont le groupage et le raffinement.

Groupage d'axies Cette opération fusionne deux axies qui sont reliées à un même ensemble de lexies. Comme une lexie représente un sens unique, si deux axies sont liées à un même ensemble de lexies, elles devraient représenter un même sens, on peut alors fusionner ces deux axies. Cette opération peut se faire automatiquement si les deux axies ont exactement les mêmes liens vers les mêmes lexies. De plus, si une axie est liée à un ensemble de lexies qui est un sous-ensemble des lexies d'une autre axie, le groupage d'axies peut être effectué automatiquement. Sinon il faut un critère pour vérifier avant de grouper deux axies. Par ailleurs, les axies peuvent aussi être groupées au cas où des lexies de plusieurs langues sont reliées aux mêmes axies et où le graphe obtenu contient une boucle. La figure 4.1 donne un exemple de groupage d'axies.

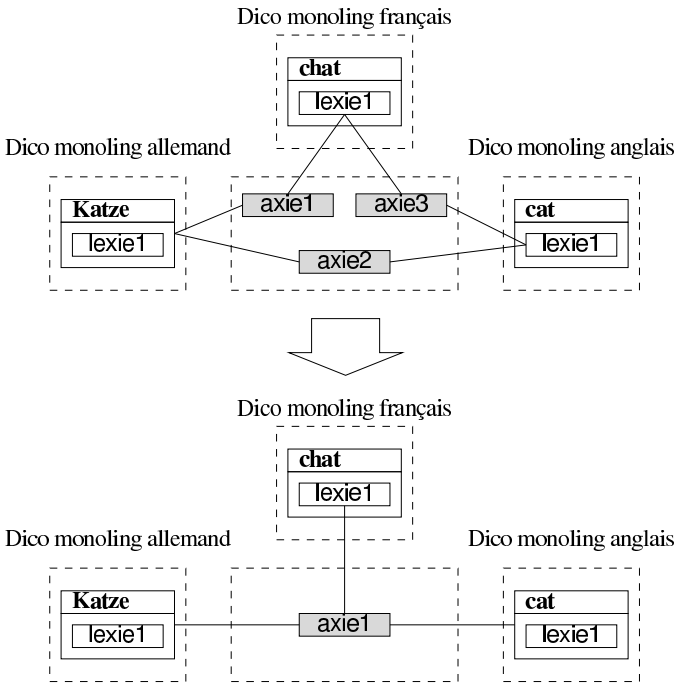

FIG. 4.1 : Groupage d'axies

Raffinement d'axies Cette opération est effectuée lorsqu'une lexie relie deux axies et que les deux axies ne peuvent pas être fusionnées car elles sont chacune liées au moins à une lexie différente dans une même langue (c'est le problème contrastif, cf. section 2.1.1). L'opération de raffinement enlève les liens entre la lexie et les deux axies, puis ajoute une nouvelle axie et relie la lexie et les deux axies à la nouvelle axie. La figure 4.2 donne un exemple de cette opération.

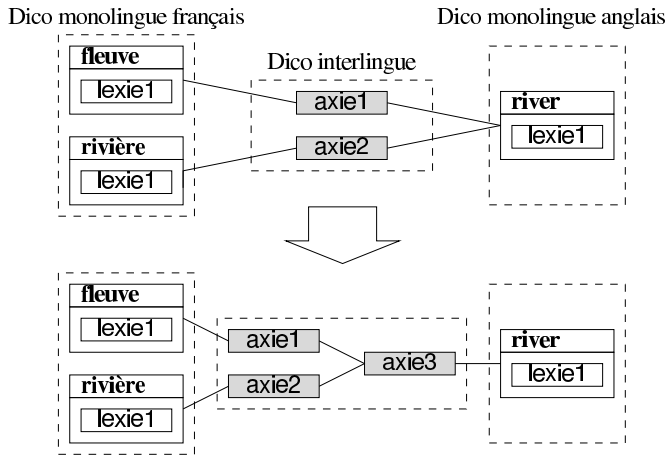

FIG. 4.2 : Raffinement d'axies

4.1.2 Approches « ontologiques »

Cette approche utilise une base de connaissances ou une ontologie pour aider à distinguer le sens des mots. Le schéma algorithmique est le suivant.

Données Des entrées monolingues de chaque langue et une classe sémantique ou une ontologie.

Méthode Relier chaque entrée monolingue à un concept de l'ontologie.

Résultat Des entrées monolingues liées aux concepts de l'ontologie. Une entrée d'une langue peut trouver une autre entrée d'une autre langue à travers le ou les concepts de l'ontologie auquel elle est liée.

Nous donnons ici deux exemples de cette approche.

4.1.2.1 KBMT

Le travail de K. Goodman [Goo89, GN91] à CMU[1] sur le projet KBMT avait pour but de construire une maquette de TAFC[2] utilisant un « pivot interlingue sémantique » relié à une ontologie du domaine, qui était la maintenance du PC et du PC5550 (le PC japonisé d'IBM). Les langues source et cible étaient l'anglais et le japonais.

Dans ce travail, une base lexicale interlingue se présente sous forme d'un réseau lexical lié à un lexique conceptuel [Nir89] (une partie de l'ontologie) représentant un ensemble de concepts du niveau sémantique, indépendant des langues.

La manipulation (développement et maintenance) de l'ontologie du projet KBMT se fait à travers l'outil ONTOS. Cette ontologie est représentée comme un réseau interconnecté et hiérarchisé de cadres (*frames*), chacun représentant un concept. Toutes les représentations de connaissances (les concepts, les vocables, le langage pivot (ILT[3]),

[1] Center for Machine Translation de Carnegie Mellon University
[2] Traduction Automatique Fondée sur la Connaissance
[3] InterLingua Text

et les règles de transfert vers/de l'ILT) sont exprimées avec le système *FRAMEKIT* par des structures de cadres.

Un cadre comprend une ou plusieurs cases (slot) ; une case comprend une ou plusieurs facettes (facet) ; une facette comprend une ou plusieurs vues, et un ou plusieurs remplisseurs (filler). Voici un exemple de structure de cadre :

```
*(make-frame
    cmu
        (is-a (value(common university non-profit-institution)))
        (location   (city (common pittsburgh))
                    (state (common pa))
                    (country (common usa))))
```

Le nom du cadre est « `cmu` ». Ce cadre comprend 2 cases : `is-a` et `location`. La case `is-a` comprend une facette `value` avec `common`[4] comme vue et deux remplisseurs : `university` et `non-profit-institution`. La case `location` comprend trois facettes : `city`, `state`, et `country` avec les vues `common` et les remplisseurs : `pittsburgh`, `pa`, et `usa`, respectivement.

Chaque dictionnaire est un ensemble de cadres où chaque cadre représente une entrée et ses informations. De plus, chaque cadre est lié à un concept dans l'ontologie. Ainsi, les unités lexicales des différents dictionnaires monolingues sont reliées aux unités de la base interlingue.

Voici quelques exemples d'entrées dans les dictionnaires de KBMT. Dans ces exemples, le verbe anglais « `remove` » et le verbe japonais « `torinozoku` » sont liés au concept « `remove` », tandis que le nom anglais « `tape` » et le nom japonais « `teepu` » sont liés au concept « `sticky-tape` ».

– exemple de verbe anglais : « `remove` »

```
("remove" (CAT V)
    (CONJ-FORM INFINITIVE)
    (FEATURES
       (CLASS DEFAUT-VERB-FEAT)
       (all-features (*OR*
         ((FORM INF) (VALENCY TRANS) (COMP-TYPE NO)
           (ROOT REMOVE))
         ((PERSON (*OR* 1 2 3)) (NUMBER PLURAL) (TENSE PRESENT)
           (FORM FINITE) (VALENCY TRANS) (COMP-TYPE NO)
           (ROOT REMOVE)
         ((PERSON (*OR* 1 2)) (NUMBER SIGNGULAR) (TENSE PRESENT)
           (FORM FINITE) (VALENCY TRANS) (COMP-TYPE NO)
           (ROOT REMOVE)))))
    (MAPPING
       (local
         (HEAD (REMOVE)))
       (local
         (slots (SOURCE=(PPADJUNCT (PREP=FROM)))))
       (CLASS CB-TH-VERB-MAP)))
```

[4]Cette vue est visible par tout le monde

– exemple de nom anglais : « tape »

```
("tape" (CAT N)
    (CONJ-FORM SINGULAR)
    (FEATURES
       (CLASS DEFAULT-NOUN-FEAT)
       (all-features ((PERSON 3) (NUMBER SINGULAR) (COUNT YES)
    (MAPPING
       (local
         (HEAD (STICKY-TAPE)))
       (CLASS OBJECT-MAP)))
```

– exemple de nom japonais : « teepu »

```
("teepu" (CAT V)
    (MAPPING
       (local
         (HEAD (STICKY-TAPE)))
       (CLASS OBJECT-MAP)))
```

– exemple de verbe japonais : « torinozoku »

```
("torinozoku" (CAT V)
    (MAPPING
       (local
         (HEAD (REMOVE)))
       (CLASS AGENT-THEME-MAP)))
```

Le problème du paradigme de KBMT est que la construction de cette KB (ontologie, modèle du domaine) est très coûteuse, car elle reste toujours construite manuellement [MNC93].

4.1.2.2 Lexique multilingue de l'UTMK[5]

[LK04] a proposé de construire un lexique multilingue basé sur une ontologie. Ce travail a construit une ontologie basée sur le thésaurus sémantique *GoiTaikei* [IMS+99] en traduisant chaque catégorie sémantique (en japonais) du *GoiTaikei* en anglais pour obtenir une catégorisation sémantique similaire en anglais.

Une entrée dans ce lexique comprend un mot-vedette (en malais), une définition, un mot équivalent en anglais, la partie du discours, l'étymologie, la classe morphologique, et des phrases exemples. Les étapes pour trouver la catégorie sémantique à laquelle cette entrée appartient sont les suivantes :

1. le mot-vedette de l'entrée en malais est traduit en japonais,
2. le mot japonais est utilisé pour rechercher dans *GoiTaikei* la catégorie sémantique à laquelle le mot appartient.
3. on cherche la classe sémantique dans la nouvelle catégorisation sémantique anglaise correspondant à cette catégorie sémantique japonaise,
4. on ajoute cette catégorie sémantique à l'entrée en malais.

[5]Unit Terjemahan Melalui Komputer, à l'Universiti Sains Malaysia, Penang, Malaisie

Autrement dit, la technique de [LK04] est de créer chaque unité lexicale avec un seul sens, et de relier cette unité lexicale à une catégorie sémantique. Les différents sens d'un vocable (nos « lexies ») sont désambiguïsés systématiquement par la catégorie sémantique à laquelle ils appartiennent.

La figure 4.3 montre la partie du thésaurus sémantique où se trouvent les lexies du mot « `hand` », que l'on suppose divisé en quatre lexies suivantes :

- partie d'un corps (*tangan* en malais),
- ouvrier (*pekerja* en malais),
- écriture (*tulisan* en malais),
- aide (*bantuan* en malais)

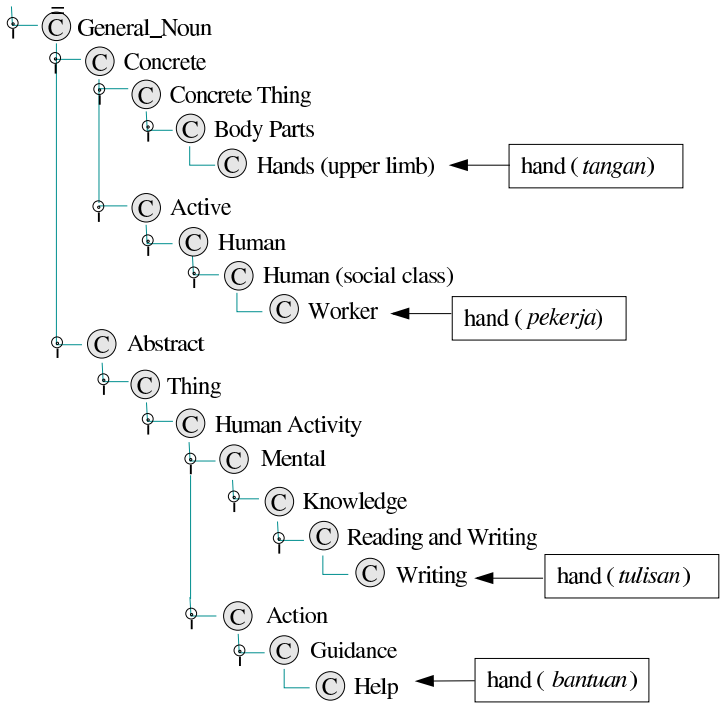

FIG. 4.3 : Partie du GoiTaikei contenant les différents sens du nom « `hand` »

4.1.3 Approche « linguistique et traductionnelle »

Cette approche utilise la traduction obtenue par les dictionnaires bilingues, et/ou l'information sur les synonymes ou les antonymes, pour établir des liens entre des articles de plusieurs langues.

4.1.3.1 Traduction avec dictionnaires bilingues

Données Un dictionnaire bilingue mot à mot de la langue `x` à la langue `y` (noté $Dict_{x \to y}$), les vocables monolingues des langues `x` et `y`.

Résultat Les unités interlingues reliant les vocables des langues `x` et `y` qui sont traduction l'un de l'autre d'après le dictionnaire $Dict_{x \to y}$.

Méthode

unités-interlingues = {liens ou vide}
pour chaque entrée $(mot - vedette_x \to \{mots - vedettes_y\})$ du $Dict_{x \to y}$ faire
  pour chaque E_y dans $\{mots - vedettes_y\}$ faire
    créer une unité interlingue A reliée aux $mot - vedette_x$ et E_y
    ajouter A dans *unités-interlingues*
fin

Cette technique simple utilise un ou plusieurs dictionnaires bilingues pour trouver des mots traductions entre deux langues.

En fait, on désire un résultat de type : une lexie d'un vocable lié à une lexie d'un autre vocable dans une autre langue. Mais avec cette technique, on obtient un graphe avec des liens du type mot à mot (figure 4.4). Le problème est de trouver le moyen de passer d'un lien vers un mot à un ou plusieurs liens vers une ou plusieurs lexies (figure 4.5).

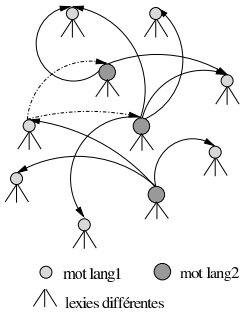

Fig. 4.4 : Exemple du type de graphe du dictionnaire bilingue

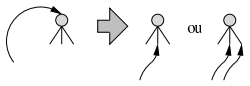

Fig. 4.5 : Problème d'un graphe du type traduction bilingue

4.1.3.2 Traduction par transfert et consultation inverse de dictionnaires bilingues

Les idées de [TU94] sont d'utiliser plusieurs dictionnaires bilingues et de passer par une langue intermédiaire pour relier deux langues lorsqu'on ne dispose pas de dictionnaire bilingue entre elles, ou lorsqu'il manque des vocables dans un dictionnaire bilingue. Par exemple, un dictionnaire japonais-français peut être construit en joignant un dictionnaire japonais-anglais et un dictionnaire anglais-français. Ensuite, le dictionnaire produit pourrait être amélioré par une technique de *consultation inverse*.

Notations

$Dict_{source \to cible}$: un dictionnaire bilingue mot à mot de la langue `source` à la langue `cible`.

$\delta_a(EC, v)$: le nombre de vocables v dans un ensemble EC.

$\delta_b(EC, v)$: le nombre de morphèmes du vocable v dans un ensemble EC.

$\delta_a(EC, SA) = \sum_{sa \in SA} \delta_a(EC, sa)$; où EC et SA sont des ensembles.

Données Des dictionnaires bilingues $Dict_{x \to y}$, $Dict_{y \to z}$, $Dict_{z \to y}$, et $Dict_{y \to x}$, les vocables monolingues des langues `x`, `y`, et `z`.

Résultat Un dictionnaire bilingue $Dict_{x \to z}$

Méthode

$Dict_{x \to z}$ = {liens ou vide}

pour chaque entrée $(mot - vedette_x \to \{EC_y\})$ du $Dict_{x \to y}$ faire
  $EC_{z-final}$ = []
  pour chaque EC_y dans $\{EC_y\}$ faire
    chercher la traduction $(EC_y \to \{EC_z\})$ dans $Dict_{y \to z}$
    // première consultation inverse
    pour chaque EC_z dans $\{EC_z\}$ faire
      chercher la traduction $(EC_z \to \{SA_y\})$ à l'aide du $Dict_{z \to y}$
        si $\{SA_y\}$ contient un élément et se trouve dans $\{EC_y\}$ alors
          ajouter EC_z dans $EC_{z-final}$
          enlever EC_z de $\{EC_z\}$
        sinon si $\delta_a(\{SA_y\}, \{EC_y\}) <= 1$ alors enlever EC_z de $\{EC_z\}$
    // seconde consultation inverse
    SA_{xfinal} = []
    pour chaque EC_{z-tmp} dans $\{EC_z\}$ faire
      chercher la traduction $(EC_z \to \{SA_y\})$ à l'aide du $Dict_{z \to y}$
      pour chaque SA_y dans $\{SA_y\}$ faire
        chercher la traduction $(SA_y \to \{SA_x\})$ à l'aide du $Dict_{y \to x}$
        ajouter $\{SA_x\}$ dans SA_{xfinal}
      si $\delta_a(\{SA_{xfinal}\}, mot - vedette_x) > 1$ alors ajouter EC_{z-tmp} dans $EC_{z-final}$
    pour chaque EC_z dans $EC_{z-final}$ faire
      créer une entrée A reliée aux $mot - vedette_x$ et EC_z
      ajouter A dans $Dict_{x \to z}$
  fin

Par exemple[6] dans la figure 4.6, la liste des mots traductions (EC_y) en anglais pour le mot japonais « `kyousou` » (競争) est « `competition` », « `contest` », et « `race` ». En consultant un dictionnaire $Dict_{e \to f}$, on obtient les candidats français (EC_z) « `compétition` », « `concours` », « `course` », « `race` » et « `hâte` ».

Pour décider quelles sont les traductions possibles en français pour « `kyousou` », [TU94] fait une consultation inverse en utilisant un dictionnaire $Dict_{f \to e}$, et puis un

[6]L'exemple est tiré de l'article [TU94],

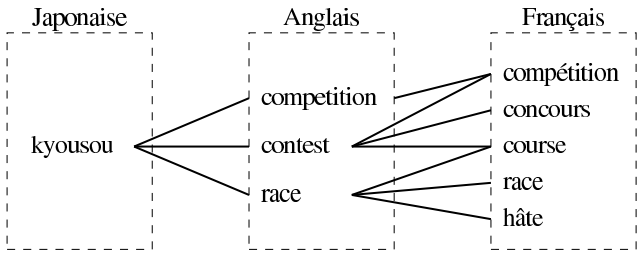

FIG. 4.6 : Équivalences candidates pour « `kyousou` »

dictionnaire $Dict_{e \to j}$. La consultation inverse entre le français et l'anglais donne le résultat (SA_y) comme illustré dans la figure 4.7.

Les mots « `concours` » et « `course` » sont retenus car leur seule traduction en anglais se trouve dans la liste EC_y. Par contre, le mot « `hâte` » est éliminé parce qu'il n'a aucune traduction en commun avec « `kyousou` ».

On continue la consultation inverse deux fois pour les mots « `compétition` » et « $race_{fr}$ ». Le résultat de la consultation inverse deux fois (SA_x) est donné dans la figure 4.8. Le mot « `compétition` » est retenu, alors que le mot « $race_{fr}$ » est éliminé car il ne satisfait pas la condition $\delta_a(\{SA_x\}, kyousou) > 1$.

Avec la correction inverse, le résultat final est que les mots traductions possibles en français pour le mot « `kyousou` » sont « `compétition` », « `concours` » et « `course` ».

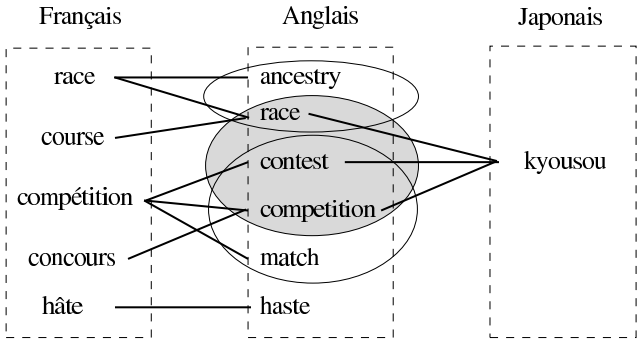

FIG. 4.7 : Consultation inverse une fois

Une autre façon pour compter les éléments en commun est de compter une partie de mot. Par exemple, on compte 競 et 争 comme deux éléments différents. Puisqu'un kanji est un idéogramme, chaque élément a un sens. Pour d'autres langues comme le français ou l'anglais, la méthode correspondante est de compter les morphèmes, par exemple, « inter » et « national » pour le mot « international ».

4.1.4 Approche « sémantique »

Cette approche utilise une représentation formalisée du sens pour comparer et établir les liens entre des articles de plusieurs langues. On peut par exemple utiliser des vecteurs conceptuels [SC96, LS99]. Le travail de [Cha90] a proposé un formalisme pour

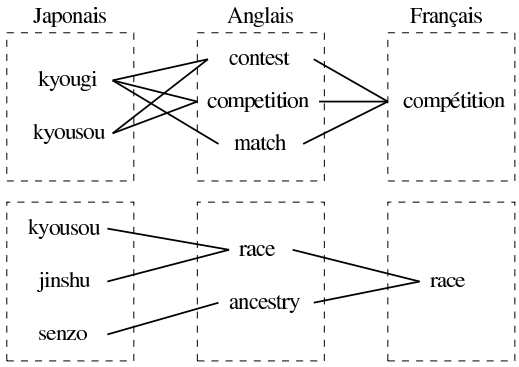

FIG. 4.8 : Consultation inverse deux fois

la projection de la notion sémantique linguistique dans un espace vectoriel, duquel le travail de [Laf02] est inspiré. L'idée de [Laf02] est d'associer à chaque lexie un *vecteur conceptuel*, puis d'utiliser ces vecteurs conceptuels pour comparer les lexies. Cette technique est expérimentée et utilisée actuellement par M. Lafourcade pour le français et l'anglais.

Principe Soit $\mathcal{C}$ un ensemble de n concepts. Un vecteur conceptuel V est une combinaison linéaire des éléments c_i de $\mathcal{C}$[7]. Supposons qu'un mot est associé à un ensemble de concepts c_i dans $\mathcal{C}$, la valeur du vecteur conceptuel associé à ce mot (plus exactement, à l'ensemble des sens de ce mot) est obtenue en projetant les concepts associés à ce mot dans l'espace vectoriel de $\mathcal{C}$. La valeur pour chaque c_i varie entre zéro (0) et un nombre donné (N). La valeur zéro dans une dimension de vecteur signifie qu'il n'y a aucune relation sémantique entre le sens auquel le vecteur est associé et le concept correspondant à cette dimension, tandis que la valeur N représente l'intensité maximale de la relation sémantique entre ce sens et le concept correspondant à cette dimension.

Par exemple[8], le mot « *friction* » est associé aux concepts : *Frottement*, *Soins du corps*, *Désaccord*, *Conflit* et *Coiffure*. Supposons que la valeur N soit égale à 1024, le vecteur conceptuel obtenu pour le mot « *friction* » est le suivant (les `CONCEPT[intensité]` sont ordonnés par valeur décroissante) :

$V_{friction}=$ (`NETTOYAGE`[511], `PROPRETÉ`[510], `COIFFURE`[483], `FROTTEMENT`[400], `DISTINCTION`[205], `ÉVICTION`[108], `BEAUTÉ`[108], `CHIRURGIE`[108], `MUSIQUE`[97], ...)

Dans [Laf02], un vecteur conceptuel a une dimension égale au nombre de concepts feuilles d'un thésaurus ([Laf02] a utilisé les 873 concepts feuilles dans la hiérarchie du thésaurus Larousse [Péc92]).

[7]Définition tirée de [LPS02]
[8]L'exemple est tiré de l'article [LS99]

Distance angulaire[9] Soit $Sim(X, Y)$ la mesure de *similarité* entre deux vecteurs définie selon la formule :

$$Sim(X, Y) = \cos(X, Y) = \frac{X \cdot Y}{|X| \times |Y|}$$

où « $\cdot$ » est le produit scalaire.
La distance angulaire D_A entre deux vecteurs X et Y est définie selon la formule :

$$D_A(X, Y) = \arccos(Sim(X, Y))$$

La distance angulaire entre vecteurs conceptuels est utilisée pour calculer la proximité thématique entre lexies. [MLSL03] montre que deux vecteurs conceptuels sont proches si leur distance est inférieure à un seuil de $\frac{\pi}{4}$. Voici quelques exemples de comparaison entre vecteurs (en degrés) :

$D_A(\text{"}profit\text{"}, \text{"}profit\text{"}) = 0^\circ$	$D_A(\text{"}profit\text{"}, \text{"}market\text{"}) = 28^\circ$
$D_A(\text{"}profit\text{"}, \text{"}benefit\text{"}) = 10^\circ$	$D_A(\text{"}profit\text{"}, \text{"}joy\text{"}) = 39^\circ$
$D_A(\text{"}profit\text{"}, \text{"}finance\text{"}) = 20^\circ$	$D_A(\text{"}profit\text{"}, \text{"}sadness\text{"}) = 65^\circ$

Contextualisation faible Généralement le vecteur conceptuel d'un mot représente tous les sens possibles de ce mot. Pourtant, si un mot se trouve dans un certain contexte, on peut mettre en valeur un des sens de ce mot en appliquant l'opération de *contextualisation faible*. Cette opération augmente la valeur des concepts communs entre ce mot et le contexte. Soit X et Y deux vecteurs, [LPS02] définit $\Gamma(X, Y)$ comme la contextualisation faible de X par Y selon la formule :

$$\Gamma(X, Y) = X \oplus (X \otimes Y)$$

où
$X \oplus Y$ est une somme vectorielle définie par

$$V = X \oplus Y \mid v_i = \frac{(x_i + y_i)}{|V|}$$

$X \otimes Y$ est un produit terme à terme normalisé définie par

$$V = X \otimes Y \mid v_i = \sqrt{(x_i y_i)}$$

Par exemple, le vecteur du mot *avocat* représente deux sens : *un humain* et *un fruit*. Si ce mot se trouve dans un contexte contenant le mot *loi*, en appliquant l'opération de contextualisation entre le mot *avocat* et le mot *loi*, le sens du mot *avocat* pour un humain augmente, tandis que le sens pour un fruit est diminué.
L'algorithme pour cette technique peut être résumé comme suit.

[9] Formules tirées de [LPS02]

Données Les acceptions monolingues de la langue x et y ; un indexage manuel d'un certain nombre de termes dans chaque langue afin de connaître un premier ensemble de vecteurs ; une hiérarchie d'un thésaurus ; un ou deux analyseurs morpho-syntaxique pour la langue x et la langue y.

Résultat Un ensemble d'acceptions interlingues reliées aux acceptions monolingues des langues x et y.

Méthode

étape 1 : associer un vecteur conceptuel à chaque lexie
Pour chaque acception monolingue ($lexie_i$) de la langue x et y faire
    créer un vecteur conceptuel (vc_i) correspondant
    relier le vecteur conceptuel vc_i à la $lexie_i$

étape 2 : créer une axie liée à chaque lexie de la langue source
Pour chaque acception monolingue ($lexie_j$) de la langue x faire
    créer une axie associée à cette $lexie_j$

étape 3 : relier la langue source et cible à l'aide d'un dictionnaire bilingue
pour chaque entrée ($mot_x = \{mot_{xi}\}$) d'un dictionnaire bilingue $Dict_{x \to y}$ faire
pour chaque sous-entrée $mot_{xi} = (pos, glose^*, equivalent_y^+)$ faire
    si glose = 0 alors vc_g = zéro
    sinon vc_g = moyenne des vecteurs conceptuels de ces gloses
    vc_{xi} = résultat de la contextualisation faible entre
            vc_g et vc_i du mot_x
    relier vc_{xi} à la sous-entrée mot_{xi}
    choisir la $lexie_x$ dont le vecteur conceptuel est le plus proche -
            de vc_{xi} avec la distance angulaire D_a
    si la distance D_a est supérieure à $\frac{\pi}{4}$ alors
       afficher un message d'alerte
    sinon
       relier la $lexie_x$ au mot_{xi}
       comparer le vc_{xi} avec tous les vc_i des $lexie_y$ -
                dont le mot-vedette se trouve dans $equivalent_y^+$
       choisir la $lexie_y$ dont le vecteur conceptuel est le plus proche -
                de vc_{xi} avec la distance angulaire D_{ai}
       si la distance D_{ai} est supérieure à $\frac{\pi}{4}$ alors
          afficher un message d'alerte
       sinon relier la $lexie_y$ à l'axie a_x qui est reliée à la $lexie_x$
fin

Remarque : pour pouvoir comparer la distance angulaire entre vecteurs conceptuels, il faut que ces vecteurs conceptuels aient le même nombre de dimensions pour chaque langue, et utilisent des hiérarchies de thésaurus qui aient les mêmes concepts ou des concepts comparables, présentés dans le même ordre.

Cela constitue un frein pour l'inclusion de nouvelles langues. La difficulté est de trouver des thésaurus pour ces langues. Par contre, l'utilisation des vecteurs conceptuels associés aux lexies pour comparer la distance entre deux lexies donne le résultat directement au niveau lexie. Un autre avantage est que cette technique est très efficace.

Synthèse

Nous avons vu que chaque approche a des avantages et des inconvénients différents. De plus, chaque technique a besoin de ressources différentes. Pour utiliser chacune des techniques, nous avons besoin des ressources exigées par cette technique. Cependant, une seule technique n'est pas suffisante pour initialiser une base lexicale interlingue de la qualité qu'on souhaite. Par exemple, avec la seule technique utilisant la traduction bilingue comme montré dans la section 4.1.3.2, nous n'obtenons qu'une base lexicale au niveau des mots et pas au niveau du sens des mots. Cependant, ces techniques sont complémentaires. Nous proposons de composer plusieurs techniques ensembles, pour atteindre la meilleure qualité possible de la base lexicale produite.

4.2 Approche de l'évaluation

4.2.1 Contexte

Notre objectif est maintenant d'évaluer une base lexicale produite par des méthodes semi-automatiques ou automatiques. L'objectif principal de l'évaluation est de vérifier si une base lexicale produite est correcte (et pas si elle est équilibrée, ou couvrante, etc.). Les travaux antérieurs sur l'évaluation des bases lexicales ne sont pas nombreux, et nous n'avons pas trouvé de travail qui fasse une étude globale de l'évaluation des bases lexicales.

Néanmoins, on peut mentionner quelques critères évidents pour mesurer la qualité d'une base lexicale, comme :

- la complétude des données lexicales par rapport à un domaine donné [NMC94],
- la granularité, par exemple, le WordNet [Pri] qui est assez détaillé pour certaines applications comme la recherche d'informations [Mih03],
- le choix des informations grammaticales : par exemple, celle du projet EDR [EDR93, Yok95, MSKO96] ne sont pas adaptées à des langues plus flexionnelles que l'anglais ou le japonais. Le projet EDR a été aussi critiqué sur le risque de la duplication des concepts dans la base de concepts qui est très grosse mais il manque d'outils efficaces pour la gérer.

Étant donné qu'il n'y a pas d'approche générale pour l'évaluation des bases lexicales, nous proposons de catégoriser des critères et de définir des méthodes possibles pour l'évaluation des BDLM. Cette catégorisation est influencée par l'étude de l'évaluation des systèmes de traduction automatique, notamment par les travaux de [NMC94, HKP02, PRWZ02, AMHT00]. Comme nous n'évaluons pas des bases lexicales créées manuellement, nous ne jugeons pas la nature des données. Par exemple, nous n'évaluons pas si une base lexicale a une bonne qualité parce qu'elle contient beaucoup de mots courants ou parce qu'elle contient des définitions des mots qui sont bien expliquées, etc.

À noter que dans la suite, une axie qui n'est reliée qu'à des lexies ayant la même signification est appelée axie « correcte ». Une axie reliée à au moins une lexie qui n'a pas la même signification que les autres est appelée axie « incorrecte ».

4.2.2 Proposition de critères de qualité

Cette section propose des critères pour l'évaluation qualitative des bases lexicales multilingues, et donne une interprétation de ces mesures. Nous proposons une classification des critères d'évaluation d'une base lexicale multilingue en quatre classes, selon leur nature [Tee04a].

4.2.2.1 Critère basé sur une référence

Dans le domaine de la traduction automatique, une manière de plus en plus utilisée (bien que fort critiquée) pour mesurer la qualité d'un système de traduction automatique est de comparer les résultats du système avec un ensemble de traductions de référence considérées comme correctes [PRWZ02, HKP02]. Par analogie, on peut définir une base lexicale multilingue de référence pour comparer à une base lexicale produite par un système automatique, à condition que les deux bases utilisent les mêmes bases lexicales monolingues. Nous considérons que deux axies sont identiques si elles contiennent exactement tous les liens aux mêmes lexies. En comparant avec une base de référence, nous obtenons le nombre d'axies qui sont définies à la fois dans la base lexicale produite et dans la base lexicale de référence (c'est le nombre d'axies correctes). La qualité de la base lexicale multilingue produite par machine peut alors être mesurée avec deux métriques adaptées de l'évaluation de systèmes de recherche d'information (RI) : rappel et précision [AMHT00].

Rappel est le nombre d'axies correctes, divisé par le nombre d'axies dans la base lexicale de référence.

Précision est le nombre d'axies correctes, divisé par le nombre d'axies dans la base lexicale produite.

Cependant, [ALJS99] a mis en évidence les limites de l'approche de comparaison avec une référence, car il est souvent difficile de produire manuellement les ressources de référence précises. Dans le contexte du projet Papillon, une base lexicale multilingue de référence traiterait neuf langues (l'anglais, l'allemand, le français, le japon, le laotien, le malais, le thaï, le vietnamien et le chinois), ce qui la rendrait extrêmement difficile à produire. En outre, puisque la base lexicale multilingue produite dans Papillon définira au moins 40 000 axies, en utilisant des ressources hétérogènes, une comparaison avec une base de référence de seulement 100 axies ne semble pas appropriée.

Au lieu de produire une base de référence entière, nous proposons de considérer une base de référence partielle correspondant à une partie d'une base lexicale multilingue, en utilisant les bases lexicales qui existent déjà. Par exemple, la base EDR pour une référence entre l'anglais et le japonais, la base HowNet pour une référence entre l'anglais et le chinois, ou l'UNL qui comprend déjà plusieurs langues en prenant les UW comme axies.

4.2.2.2 Critère structural

Le critère structural considère l'état des liens entre lexies et axies. Une lexie aura un des trois types de connexion avec axies :

- Incomplétude : une lexie n'a aucune connexion avec aucune axie.
- Correction et complétude : une lexie a une seule connexion avec une seule axie.
- Incorrection : une lexie a plus d'une connexion avec plusieurs axies.

Pour évaluer la base produite, on calcule le rapport des lexies correctes et complètes, celui des lexies incomplètes et celui des lexies incorrectes, avec les formules suivantes :

$$\text{Pourcentage des lexies incomplètes} = \frac{|\{L||conn(L)| = 0\}|}{|L|}$$

$$\text{Pourcentage des lexies correctes et complètes} = \frac{|\{L||conn(L)| = 1\}|}{|L|}$$

$$\text{Pourcentage des lexies incorrectes} = \frac{|\{L||conn(L)| > 1\}|}{|L|}$$

où

L est une lexie.

$|L|$ est le nombre total des lexies dans la base évaluée.

$|conn(L)|$ est le nombre de connexions d'une lexie avec une ou plusieurs axies.

La base lexicale évaluée est bonne si le pourcentage des lexies correctes et complètes est élevé. Une telle métrique est facile à mesurer. De plus, elle concerne une base entière. Par contre, ce type de critère n'aide pas à évaluer la qualité des liens entre les axies et les lexies en termes de sémantique.

4.2.2.3 Critère basé sur un jugement humain

Ce critère d'évaluation est basé sur la mesure du nombre de corrections faites par un linguiste sur une partie d'une base lexicale produite. Par exemple, on peut mesurer la proportion entre le nombre de ces corrections et le nombre total de liens entre les axies et les lexies considérées. Plus la proportion est proche de zéro, plus la qualité de la base produite est élevée.

Il faut remarquer que cette approche diffère fondamentalement de la comparaison avec une référence. En effet, de même qu'en traduction, le réviseur humain de la base lexicale cherchera à minimiser le nombre de corrections permettant d'arriver à un résultat correct. Dans la mesure où de nombreuses solutions sont possibles (il suffit de voir le nombre de façons de diviser les mots en lexies dans des dictionnaires renommés), rien ne garantit que le résultat obtenu sera égal à une « référence » préalable, ni même proche (au sens d'une des mesures précédentes) d'une référence parmi plusieurs.

Cependant, la construction manuelle d'une BDLM de taille suffisante est très coûteuse : l'effort est équivalent à celui de la traduction parfaite. Par contre, corriger des échantillons choisis au hasard ou en fonction de critères variés (auteurs, dates, langues, domaines, etc.) permet certainement d'obtenir une évaluation du même type, à coût « réglable » en fonction de la précision souhaitée.

4.2.2.4 Critère sémantique

Il s'agit de la qualité sémantique des liens entre axies et lexies. On n'a pas besoin de ressources lexicales supplémentaires.

Une des métriques que nous considérons est la distance entre les vecteurs conceptuels des lexies liées à une même axie [Laf02]. Plus la distance entre les vecteurs conceptuels de deux lexies est petite, plus les lexies sont sémantiquement proches. Pour mesurer, nous calculons la distance conceptuelle moyenne entre chaque paire de lexies liées à une même axie. Plus la valeur est petite, plus la base évaluée est bonne en terme de sémantique entre des lexies.

Cependant, le calcul fiable des vecteurs conceptuels se fonde sur les définitions précises et riches des lexies, et sur les ressources lexicales pour calculer les vecteurs initiaux. De plus, la méthode suppose l'analyse complète des définitions, et se heurte au fait qu'il n'existe pas forcément de tels analyseurs librement utilisables pour toutes les langues.

Synthèse

Dans un cadre conceptuel plus général, nous proposons une catégorisation des critères d'évaluation en quatre dimensions, ou caractéristiques :

dimension	description
Automatisation	la mesure associée au critère peut-elle être réalisée automatiquement ?
Portée	le critère permet-il d'évaluer une base entière, ou seulement une partie ?
Sémantique	le critère permet-il d'évaluer le contenu sémantique d'une base, ou seulement sa structure ?
Ressource	le critère nécessite-t-il des ressources lexicales supplémentaires pour pouvoir être mesuré ?

Une base lexicale multilingue telle que Papillon peut être utilisée dans différents contextes, par exemple dans des systèmes de traduction automatique ou dans des systèmes de recherche d'information. Les critères utilisés pour évaluer une base lexicale multilingue devraient aussi être adaptés au contexte dans lequel la base lexicale est utilisée.

Par exemple, si une base lexicale multilingue est très précise et bonne entre le français et le japonais, mais n'est pas tellement bonne pour d'autres langues, elle devrait être jugée comme une bonne base lexicale par les utilisateurs qui évaluent seulement son usage pour du traitement du français et du japonais, mais elle devrait être jugée globalement comme une mauvaise base lexicale.

Puisque la base lexicale produite par notre système ne sera pas destinée à des utilisations spécifiques, le système ne devrait pas imposer de critères d'évaluation prédéfinis. Au lieu de cela, nous proposons de permettre d'utiliser n'importe quel critère, par exemple ceux que nous avons définis dans notre catégorisation dans la

section précédente, et de permettre de les composer arbitrairement pour s'adapter à différents contextes.

De plus, puisque nous visons à exécuter une évaluation automatique, nous ne considérons pas de critères nécessitant du travail humain, même si l'évaluation humaine est certainement valide. Notre approche est semblable à l'approche proposée pour le système pour la structuration des lexies/axies. Nous abordons ce problème de la composition des critères du point de vue du génie logiciel, en utilisant des techniques de programmation par objets pour désigner et implémenter des logiciels modulaires et réutilisables.

4.2.3 Méthodes d'évaluation

Comme indiqué dans la section précédente, nous souhaitons un système d'évaluation qui n'impose pas de critères fixes pour évaluer une base lexicale. Nous proposons donc un système qui permet d'implémenter des algorithmes pour évaluer une base lexicale dans des modules réutilisables (voir le détail dans le chapitre 6). Dans ce système, on considère chaque critère d'évaluation implémenté comme un module. Par convention, chaque module mesurant un critère doit donner comme résultat une valeur numérique Q_i. Il existe deux types de critères : 1) *plus c'est mieux*, et 2) *moins c'est mieux*. En général, les méthodes de décision multicritère nécessitent d'avoir des critères de type « plus c'est mieux ». Par conséquent, les modules associés aux critères de type « moins c'est mieux » doivent produire l'inverse de la valeur mesurée.

Plus la valeur Q_i augmente, meilleure est la base lexicale évaluée.

En utilisant les critères présentés dans la section 4.2.2, les mesures de qualité utilisables sont les suivantes : mesure par rapport à une référence, mesure structurale, mesure vectorielle.

Mesure de qualité par rapport à une référence Les deux métriques possibles sont le rappel et la précision :

$$Q_{i(rappel)} = \frac{Z}{X}$$

$$Q_{i(precision)} = \frac{Z}{Y}$$

où

X est le nombre d'axies dans la base lexicale de référence

Y est le nombre d'axies dans la base lexicale produite

Z est le nombre d'axies qui sont définies à la fois dans la base lexicale produite et dans la base lexicale de référence

Mesure de qualité « structurale »

$$Q_{i(lexies-incompletes)} = \frac{|\{L||conn(L)| = 0\}|}{|L|}$$

$$Q_{i(lexies-correctes-completes)} = \frac{|\{L||conn(L)| = 1\}|}{|L|}$$

$$Q_{i(lexies-incorrectes)} = \frac{|\{L||conn(L)| > 1\}|}{|L|}$$

où
L est une lexie.
$|L|$ est le nombre total des lexies dans la base évaluée.
$|conn(L)|$ est le nombre de connexions d'une lexie avec une ou plusieurs axies.

Mesure de qualité « vectorielle » Comme décrit dans le critère sémantique (page 72), cette formule calcule la distance moyenne entre les vecteurs conceptuels des lexies liées à une même axie. C'est un critère de type « moins c'est mieux », donc la valeur retournée doit être l'inverse de ladite distance.

$$Q_{i(vectoriel)} = 1/(\frac{1}{n} \sum_{i=1}^{n} (\frac{1}{nbpl_{(i)}} \sum_{k=1}^{nbpl_{(i)}} distance_k))$$

où
n est le nombre total d'axies dans la base évaluée.
$nbpl_{(i)}$ est le nombre total de paires de lexies qui sont reliées à une même axie i.
$distance_k$ est la distance angulaire entre les deux vecteurs conceptuels associés aux deux lexies d'une paire k.

Critère global

Nous appliquons une méthode de décision multicritère du domaine de l'analyse de la décision [Bou90, PH97, Kal04] pour définir une valeur de qualité globale. La valeur de qualité globale, notée Q, sera la somme pondérée de chaque valeur de qualité mesurée par chaque module de critère, et pondérée de façon à refléter les objectifs ou le point de vue du lexicologue.

Le choix des critères correspond à un contexte d'usage de la base lexicale évaluée, et le poids (positif) de chaque valeur dans ce contexte est indiqué comme facteur dans la somme :

$$Q = \sum_{i=1}^{nbmodules} poids_i \cdot Q_i$$

L'objectif est de maximiser Q. Le poids de chaque valeur peut être choisi pour souligner l'importance des critères choisis dans le contexte de l'évaluation. Par exemple, en évaluant spécifiquement la qualité des axies entre les lexies françaises et anglaises, le poids de la valeur fournie par un module utilisant comme référence le dictionnaire EN-FR pourrait être plus élevé que les poids pour d'autres valeurs.

En outre, les différentes valeurs ne sont pas normalisées. Il est donc nécessaire d'adapter les poids pour compenser les différences d'échelle entre les valeurs de Q_i. Il est à noter que nous pouvons évaluer la valeur de Q à chaque fois que nous avons modifié une base d'axies, par exemple à chaque étape de création d'axies. Dans ce contexte, ce qui est intéressant est l'évolution de la valeur de Q, et non cette valeur elle-même.

4.3 Similitudes entre la structuration et l'évaluation

Chaque module de création de lexies et d'axies encapsule son propre critère de qualité qu'il essaie d'optimiser explicitement ou implicitement. Un critère pour décider de créer une axie peut aussi être utilisé pour évaluer des axies existantes. Par exemple, si on a un critère de construction qui dit qu'une axie est plausible si elle est liée à des lexies ayant la même partie du discours, alors on peut utiliser ce critère pour vérifier si des axies existantes sont correctes ou non.

En adaptant le travail de [Béd89] avec l'évaluation basée sur une référence, nous définissons quatre mesures : rappel, précision, silence et bruit.

Soit $\mathcal{A}$, un ensemble d'axies possibles a priori.

$\mathcal{A} = \mathcal{P}(\mathcal{L}_1) \times \mathcal{P}(\mathcal{L}_2) \times \ldots \times \mathcal{P}(\mathcal{L}_N)$

où

$\mathcal{L}_i$ est l'ensemble de lexies de la langue i.

N est le nombre total de langues dans la base d'axies $\mathcal{A}$.

$\mathcal{P}(\mathcal{L}_i)$ est un ensemble de parties de $\mathcal{L}_i$.

$\mathcal{A}_{vrai}$ ou $\mathcal{A}_v$ est un ensemble d'axies qu'il faudrait créer.

$\mathcal{A}_{algo}$ ou $\mathcal{A}_a$ est un ensemble d'axies qui sont créées d'après un algorithme particulier.

À partir de ces notions, nous définissons quatre valeurs :

$$\mathcal{A}_1 = \mathcal{A}_{algo} \cap \mathcal{A}_{vrai}$$
$$\mathcal{A}_2 = \overline{\mathcal{A}}_{algo} \cap \overline{\mathcal{A}}_{vrai}$$
$$\mathcal{E}_1 = \overline{\mathcal{A}}_{algo} \cap \mathcal{A}_{vrai}$$
$$\mathcal{E}_2 = \mathcal{A}_{algo} \cap \overline{\mathcal{A}}_{vrai}$$

Posons $\mathcal{A}_j = |\mathcal{A}_j|$ et $\mathcal{E}_j = |\mathcal{E}_j|$. Nous définissons alors :

$$rappel = \frac{\mathcal{A}_1}{\mathcal{A}_v} = \frac{\mathcal{A}_1}{\mathcal{A}_1 + \mathcal{E}_1}$$
$$precision = \frac{\mathcal{A}_1}{\mathcal{A}_a} = \frac{\mathcal{A}_1}{\mathcal{A}_1 + \mathcal{E}_2}$$

$$silence = \frac{\mathcal{E}_1}{\mathcal{A}_v} = \frac{\mathcal{E}_1}{\mathcal{A}_1 + \mathcal{E}_1}$$
$$bruit = \frac{\mathcal{E}_2}{\mathcal{A}_a} = \frac{\mathcal{E}_2}{\mathcal{A}_1 + \mathcal{E}_2}$$

Comme décrit dans la section 4.2.3, la valeur associée à un critère de type « moins c'est mieux » doit être l'inverse de la valeur mesurée, pour pouvoir l'utiliser dans la mesure du critère global. Pour utiliser les mesures de silence et de bruit dans le critère global, il faut donc les inverser. Alors les valeurs associées au silence et au bruit peuvent être définies par :

$$\mathcal{V}_{silence} = \frac{\mathcal{A}_v}{\mathcal{E}_1} = \frac{\mathcal{A}_1 + \mathcal{E}_1}{\mathcal{E}_1}$$
$$\mathcal{V}_{bruit} = \frac{\mathcal{A}_a}{\mathcal{E}_2} = \frac{\mathcal{A}_1 + \mathcal{E}_2}{\mathcal{E}_2}$$

4.4 Conclusion

Ce chapitre a présenté, en premier lieu, les approches pour la structuration des lexies et axies. Étant donné qu'elles sont complémentaires, pour atteindre le meilleur résultat possible, nous avons proposé de les combiner. Ce chapitre a aussi montré plusieurs formules d'évaluation de la qualité des DBLM en utilisant différents critères.

Nous avons également proposé de composer ces différents critères en produisant les valeurs associées, pour obtenir une évaluation globale.

Enfin, nous avons insisté dans la section 4.3, sur la similitude entre la structuration et l'évaluation, ce qui nous a permis de proposer une solution similaire pour les deux tâches.

Le chapitre suivant expliquera plus en détail les idées clés de notre proposition pour la structuration et l'évaluation des BDLM.

Chapitre 5

Propositions pour la structuration et l'évaluation

Ce chapitre présente notre proposition pour la conception d'un système d'aide à la construction automatique des BDLM. La première section présente le contexte général du travail. La section 5.2 présente les idées clés que nous avons utilisées. La section 5.3 présente deux langages : le langage de description de processus de construction de bases lexicales et le langage de description de stratégies d'évaluation de qualité.

5.1 Contexte général

Avant de spécifier une solution « programmation » aux problèmes que nous avons identifiés, il est utile de rappeler la liste des hypothèses que nous faisons ou ne faisons pas dans ce travail.

- Nous ne faisons aucune hypothèse sur la nature des ressources utilisées ou produites. Les ressources peuvent être variées. Le système doit être capable de traiter des ressources différentes telles que des dictionnaires bilingues, multilingues, de synonymes, etc. (cf. la catégorisation du chapitre 1.3.1).
- Nous ne faisons aucune hypothèse quantitative sur la qualité des ressources et sur la taille des ressources utilisées ou produites. Nous imposons cependant que la taille des ressources utilisées ou produites puisse être très grande.
- La structure est celle de la base « centrale » de Papillon (NADIA-DiCo : lexie, axie, etc.). Mais le système doit être ouvert à tout type d'information complémentaire qui peut être utile, telle que parties du discours, domaine, etc. Ainsi, tout type d'information peut être attaché à une lexie, une axie, etc.
- Nous devons faire un compromis entre les performances et la généricité du système ou sa capacité de maintenance et d'évolution. Dans ce cas, nous choisissons d'avoir un système maintenable et évolutif, qui n'est pas forcément performant.

Les différentes techniques étudiées sont complémentaires. Il est possible, et souhaitable, de les composer pour tirer parti au mieux des ressources disponibles. Cependant, il n'existe pas de composition idéale de ces techniques. Notamment, les techniques uti-

lisables dépendent des ressources et des informations disponibles, dans le contexte de la base à créer. Par exemple, il n'est pas possible d'utiliser la technique de traduction bilingue si aucun dictionnaire bilingue n'est disponible.

C'est cette disparité des contextes d'utilisation possibles de notre système qui a motivé notre objectif de capacité de composition arbitraire de techniques de construction d'axies. Nous proposons de réduire ce problème de composition à un problème de composition de modules logiciels du point de vue de l'architecture de notre système. Notre proposition se situe donc à l'intersection des domaines de la lexicographie et de l'architecture logicielle.

5.2 Idées-clés

Cette section présente les trois idées principales que nous utilisons pour la conception d'un système de structuration et d'évaluation des BDLM.

5.2.1 Séparation des préoccupations

Le principe de séparation des préoccupations [Par72, HL95] est l'un des plus importants et des plus classiques parmi les principes de conception de logiciel. Ce principe dit qu'il faut spécifier séparément les parties d'un système qui correspondent à des domaines ou des responsabilités différentes, en préservant la caractéristique d'être composables.

Il y a souvent des points de vue différents sur un même système. Par exemple, dans notre cas, il y a le point de vue du lexicologue, du développeur de module de création d'axies, ou encore celui de l'administrateur de la base de données. Si quelqu'un veut changer quelque chose dans le système, de son point de vue, il ne devrait pas avoir besoin de comprendre ni même de regarder les autres points de vue : il ne devrait avoir à raisonner ou à modifier que des choses qu'il a à connaître. L'avantage de ce principe est la facilité de modifier ou de faire évoluer un logiciel, dans un point de vue, sans avoir besoin de regarder d'autres points de vue.

Trois problèmes se posent pour la séparation des préoccupations : 1) l'identification des différentes préoccupations, 2) la spécification séparée des préoccupations, et 3) leur composition [HL95].

En ce qui concerne l'identification des préoccupations, [HL95] donne une liste non exhaustive de préoccupations couramment considérées : la synchronisation, la localisation dans un système réparti, les contraintes de temps réel et la tolérance aux fautes.

La séparation des préoccupations peut être considérée au niveau de l'implantation et au niveau conceptuel. Dans notre cas, les niveaux conceptuel et d'implantation sont confondus : les préoccupations sont séparées effectivement au niveau de l'architecture du système.

Nous appliquons ce principe à notre système. Au niveau global, on a deux points de vue sur le système : 1) le point de vue du lexicologue et 2) le point de vue du

programmeur. Ces deux rôles différents dans notre système ont des objectifs et des problèmes différents :

lexicologue le travail du lexicologue concerne la partie manuelle du système, donc il n'est pas concerné par notre système qui ne traite que la partie automatisée.

programmeur le travail du programmeur est de développer un système qui facilite le travail des lexicologues.

5.2.2 Adaptabilité et composition des techniques

5.2.2.1 Adaptabilité

Dans la conception, nous souhaitons que le système puisse supporter le changement éventuel de parties du système. Autrement dit, nous voulons éviter la « rigidité » du système, c'est-à-dire que nous souhaitons pouvoir changer une partie de programme du système sans modifier ou casser le reste. Par conséquent, nous devons savoir et définir clairement ce qui doit pouvoir changer dans notre système, au moment de la conception.

Les composants que nous pouvons envisager de changer sont par exemple, le système de gestion de base de données, l'analyseur morpho-syntaxique (lorsque nous avons plusieurs langues, nous devrions avoir plusieurs analyseurs), les formats de fichier des données lexicales (par exemple, ceux des dictionnaires monolingues et bilingues, etc).

C'est la programmation par objets qui permet le mieux le développement d'applications adaptables, en facilitant l'application du principe de polymorphisme [LW94], plus particulièrement via le mécanisme de la délégation abstraite [Mey97]. L'application du principe de polymorphisme dit qu'on doit pouvoir remplacer un objet d'un type A par n'importe quel objet d'un sous-type de A, sans modifier le code qui utilise l'objet de type A. Le mécanisme de la délégation abstraite rend ce principe plus réalisable avec une classe abstraite.

D'abord, nous définissons une classe abstraite I (ou une interface, en java) et toutes les classes « clientes » référencent cette interface de I, et manipulent des objets de type I, sans dépendre des sous-types de I (i.e. des sous-interfaces ou des classes qui implémentent I). Ensuite, nous pouvons définir plusieurs classes qui héritent de I, et les classes « clientes » peuvent manipuler ces objets, sans connaître leur classe réelle. La classe abstraite (ou l'interface) est en fait la définition d'un « point d'adaptabilité » dans notre système où nous pouvons adapter notre système autour de cette interface.

Nous proposons un système appelé « Jeminie ». Jeminie [Tee04b] est un canevas logiciel que nous avons conçu pour automatiser la structuration et l'évaluation de bases lexicales interlingues par acceptions. Nous en parlerons plus en détails dans le chapitre 6.

5.2.2.2 Combinaison des techniques

Nous avons dit plus haut dans la section 4.1 que chaque technique de production d'axies a des avantages et des inconvénients différents. De plus, une seule technique n'est pas suffisante pour initialiser une base lexicale interlingue de la qualité qu'on souhaite. Nous avons donc proposé de combiner plusieurs techniques. Par contre, il n'existe pas de composition idéale de ces techniques, car dans des situations différentes, on peut avoir besoin de compositions différentes selon les ressources disponibles.

Nous présentons deux exemples de composition de techniques pour illustrer et insister sur la nécessité de composer des techniques. Supposons qu'on a la liste des techniques disponibles suivante :

TN_1 : traduction avec dictionnaires bilingues (cf. section 4.1.3.1).

TN_2 : traduction transfert et consultation inverse de dictionnaires bilingues de K. Tanaka [TU94] (cf. section 4.1.3.2).

TN_3 : comparaison de deux vecteurs conceptuels [Laf02, LS99]. L'algorithme présenté dans la section 4.1.4 est utilisé pour créer une base d'axies. Dans cette section, nous modifions l'algorithme pour l'utiliser pour filtrer des axies existantes. Pour chaque axie, l'algorithme enlève toute lexie dont la distance minimale par rapport aux autres lexies est supérieure à un seuil donnée. L'algorithme est le suivant :

Donnée. Une base d'axies DB_a

Résultat. La base d'axies DB_a modifiée

Méthode.

```
pour chaque axie a_i dans la base DB_a faire
  min_distance = 0
  lexie_temp = null
  pour chaque lexie l_i dans axie a_i faire
    C = distance minimale de lexie l_i par rapport aux autres lexies
    si C > min_distance alors
          min_distance = C
          lexie_temp = l_i
    fin-si
  fin
  si min_distance > un seuil donné alors
     enlever l_i (la lexie contenue dans lexie_temp)
  fin-si
fin
```

TN_4 : comparaison de parties du discours.

Bref algorithme :

Données. Une base d'axies BD_a

Résultat. La base d'axies BD_a modifiée

Méthode.

```
pour chaque axie a_i dans BD_a faire
```

```
      pour chaque lexie l_i reliée à l'axie a_i faire
        comparer la partie du discours de l_i avec celles des autres lexies dans a_i
        si la partie du discours de l_i est différente des autres, alors
          enlever l_i de l'axie a_i
        fin-si
      fin
    fin
```

On souhaite créer une base d'axies BD_a en utilisant ces techniques en fonction des ressources disponibles. Certaines techniques manipulent une base lexicale au niveau des mots (TN_1, TN_2), et d'autres manipulent une base lexicale au niveau des acceptions (TN_3, TN_4). Pour arriver à fonctionner au niveau des acceptions, les algorithmes de TN_1 et TN_2 sont légèrement modifiés.

Pour TN_1, au lieu de relier les vocables $mot-vedette_x$ et E_y (cf. page 63), on relie toutes les lexies du vocable $mot-vedette_x$ à toutes les lexies du vocable E_y, et de même pour TN_2. En conséquence, comme illustré dans la figure 5.1, au lieu d'avoir un seul lien entre $mot-vedette_x$ (`mot lang 1`) et E_y (`mot lang 2`), le nombre de liens créés est égal au produit du nombre des lexies du $mot-vedette_x$ et du nombre des lexies du E_y.

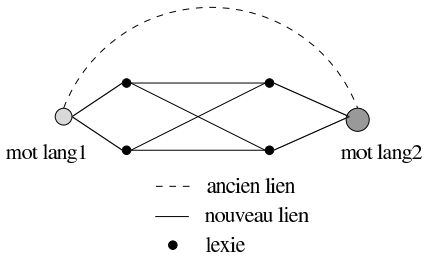

FIG. 5.1 : Liens entre lexies de deux vocables

1re situation Liste des ressources disponibles :
- des dictionnaires bilingues : $Dict_{fr\to en}$, $Dict_{en\to th}$, $Dict_{th\to en}$, $Dict_{en\to fr}$, $Dict_{en\to jp}$.
- des bases de lexies $BDml_{fr}$, $BDml_{en}$, $BDml_{jp}$, $BDml_{th}$.
- l'information sur la partie du discours pour chaque lexie.
- un vecteur conceptuel associé à chaque lexie de chaque base lexicale monolingue.

Pour créer une base d'axies BD_a, on peut définir par exemple la composition de techniques comme illustré dans la figure 5.2.

Étapes :

1. Appliquer TN_2 aux bases $BDml_{fr}$, $BDml_{en}$ et $BDml_{th}$ en utilisant $Dict_{fr\to en}$, $Dict_{en\to th}$, $Dict_{th\to en}$, et $Dict_{en\to fr}$, on obtient la base BD_a.
2. Appliquer TN_4 à la base BD_a, on obtient la base BD_a modifiée.
3. Appliquer TN_3 à la base BD_a, on obtient la base BD_a modifiée.

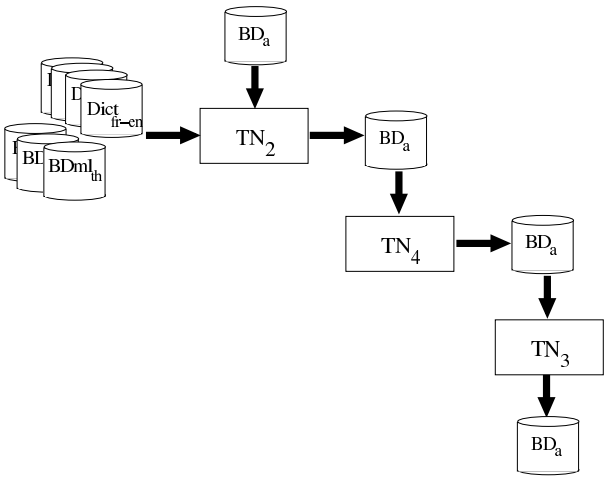

FIG. 5.2 : Processus pour la 1re situation

2e situation Liste des ressources disponibles :
- des dictionnaires bilingues : $DictFA1_{fr\rightarrow en}$, $DictFA2_{fr\rightarrow en}$, $Dict_{en\rightarrow th}$.
- des bases de lexies $BDml_{fr}$, $BDml_{en}$, $BDml_{th}$.
- l'information sur la partie du discours pour chaque lexie.
- un vecteur conceptuel associé à chaque lexie de chaque base lexicale monolingue.

Une des compositions de techniques possible est montrée dans la figure 5.3.

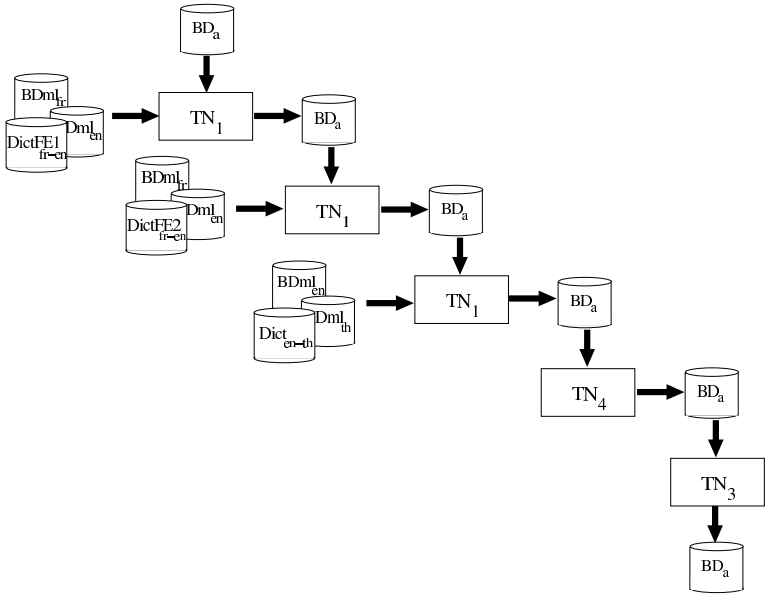

FIG. 5.3 : Processus pour la 2e situation

Étapes :

1. Appliquer TN_1 aux bases $BDml_{fr}$ et $BDml_{en}$ en utilisant $DictFA1_{fr\rightarrow en}$, on

obtient la base BD_a.

2. Appliquer TN_1 aux bases $BDml_{fr}$ et $BDml_{en}$ en utilisant $DictFA2_{fr \to en}$; le résultat est ajouté dans la base BD_a.
3. Appliquer TN_1 aux bases $BDml_{en}$ et $BDml_{th}$ en utilisant $Dict_{en \to th}$; le résultat est ajouté dans la base BD_a.
4. Appliquer TN_4 à la BD_a, on obtient la base BD_a modifiée.
5. Appliquer TN_3 à la BD_a, on obtient la base BD_a modifiée.

Dans ces deux exemples, on a plus de ressources disponibles dans la première situation que dans la deuxième situation. Il est possible d'utiliser la technique TN_2 dans la première, ce qui n'est pas le cas pour la deuxième.

Ces exemples suggèrent qu'il est nécessaire de pouvoir utiliser des compositions de techniques différentes, car le choix d'une composition est un compromis entre la qualité souhaitée de la base produite et le coût de la mise en œuvre des techniques (ressources techniques nécessaires, et ressources lexicales nécessaires).

D'ailleurs, la section 4.2.3 a aussi discuté l'avantage de la composition des critères pour l'évaluation des BDLM. Notre proposition est de développer des modules logiciels pour la structuration et l'évaluation. La composition de techniques est alors résolue dans les deux cas par la composition des modules logiciels.

5.2.3 Similarité entre structuration et évaluation

Comme indiqué dans la section 4.3, les algorithmes / techniques utilisés pour la structuration peuvent aussi être utilisés pour l'évaluation. Ces algorithmes sont fournis sous la forme de modules logiciels, pour pouvoir être composés facilement.

Nous proposons que les modules logiciels pour la structuration et pour l'évaluation aient des formes et des propriétés similaires, et qu'un même algorithme puisse à la fois être implémenté dans un module de structuration et dans un module d'évaluation.

5.3 Outils pour le lexicologue

5.3.1 Langage de description de processus de construction de bases lexicales

Dans l'exemple de la section 5.2.2.2, on exécute une par une des techniques de construction d'une base d'axies. Le linguiste ou la personne qui exécute cette séquence d'exécutions devrait lancer chaque exécution lorsque la précédente est terminée. Il est préférable d'avoir un langage dans lequel on peut définir une séquence d'exécutions et n'en lancer qu'une seule à la fois.

Un des langages existants est celui de PRODUCDIC [Hai98b, Hai98a]. Il propose les opérations de base pour la production d'un dictionnaire à partir des ressources

existantes. Les opérations[1] proposées sont : sélection, extraction, regroupement, inversion, enchaînement, combinaison parallèle, et combinaison en étoile.

Selection Cette opération permet de sélectionner un sous-ensemble d'un ensemble lexical selon un critère donné, par exemple sélectionner des entrées verbales, sélectionner des entrées polysémiques, etc. (figure 5.4)

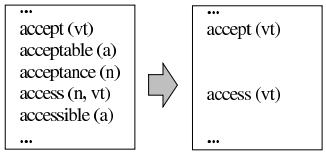

FIG. 5.4 : Opération sélection de Producdic

Extraction L'extraction d'une sous-structure des articles d'un ensemble lexical, par exemple l'extraction de la partie syntaxique des articles d'un dictionnaire ou l'extraction d'un dictionnaire bilingue à partir d'un dictionnaire trilingue. (figure 5.5)

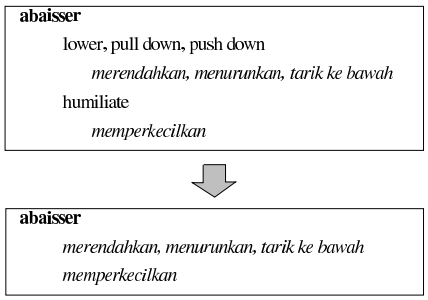

FIG. 5.5 : Opération extraction de Producdic

Regroupement Un élément de l'ensemble cible est formé par l'intégration de plusieurs éléments de l'ensemble source, par exemple un regroupement d'un ensemble d'articles homographes en un ensemble d'articles polysémiques. (figure 5.6)

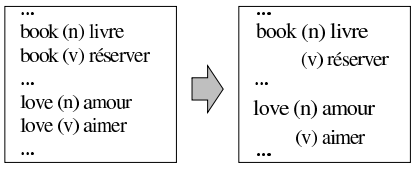

FIG. 5.6 : Opération regroupement de Producdic

[1] Définitions tirées de [Hai98b].

Inversion L'inversion d'un ensemble lexical permet d'obtenir un nouvel ensemble lexical, où l'entrée d'un article source devient un composant d'un ou plusieurs articles cible, et un composant d'un ou plusieurs articles source devient l'entrée d'un article cible. Par exemple, l'inversion d'un dictionnaire bilingue français-anglais pour obtenir un (brouillon de) dictionnaire anglais-français. (figure 5.7)

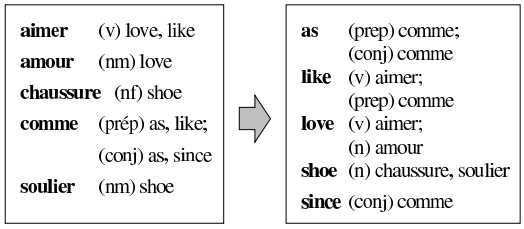

FIG. 5.7 : Opération inversion de Producdic

Enchaînement L'opération enchaîne plusieurs ensembles lexicaux dont l'entrée de l'article de l'un est un composant de l'article de l'autre, pour obtenir un nouvel ensemble ayant plus d'information. Par exemple, l'enchaînement d'un dictionnaire français-anglais et d'un dictionnaire anglais-malais, pour obtenir un dictionnaire français-anglais-malais, ou seulement français-malais. (figure 5.8)

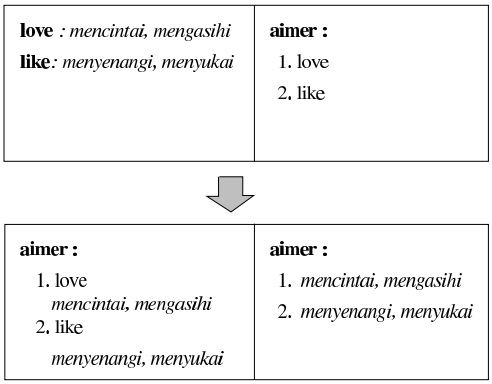

FIG. 5.8 : Opération enchaînement de Producdic

Combinaison parallèle Des ensemble lexicaux dont l'entrée de l'article a la même signification peuvent être combinés en parallèle. Par exemple, la combinaison parallèle de deux dictionnaires français-anglais, pour obtenir un dictionnaire français-(anglais1, anglais2). (figure 5.9)

Combinaison en étoile C'est une généralisation de l'enchaînement et de la combinaison parallèle. L'idée de la combinaison en étoile peut être exprimée par un exemple, où l'on combine des dictionnaires $A \rightarrow B$, $B \rightarrow A$, $A \rightarrow C$, $C \rightarrow A$, $A \rightarrow D$,

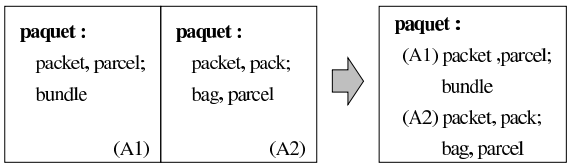

FIG. 5.9 : Opération combinaison parallèle de Producdic

et $D \rightarrow A$, pour créer un dictionnaire multilingue ayant A comme langue pivot : $(B, C, D) - A - (B, C, D)$.

PRODUCDIC utilise une extension de CLOS[2], dans le sens où il utilise la syntaxe et les fonctions de CLOS en ajoutant des opérations décrites ci-dessus. Chaque opération a un formalisme différent. Si on a une nouvelle opération, il faudra définir une nouvelle formule pour cette opération.

PRODUCDIC utilise une opération de base pour produire un nouvel ensemble lexical. Dans notre cas, nous nous intéressons particulièrement à produire une base d'axies. Certaines opérations de PRODUCDIC sont intéressantes à réutiliser comme technique dans notre système, par exemple, l'enchaînement, la combinaison parallèle, et la combinaison en étoile. Par contre, certaines opérations sont plus préférables dans l'étape de la préparation des données, par exemple, la sélection, l'extraction, le regroupement, et l'inversion.

PRODUCDIC convient pour la production des données lexicales générales, mais il ne convient pas à notre objectif. Nous souhaitons un langage pour exécuter une séquence de techniques. Nous appelons *un processus*, une séquence d'exécution . Nous définissons ici un langage simple pour définir de tels processus.

- Un processus est défini par une liste de techniques les unes après les autres, dans l'ordre d'exécution. Chaque technique est séparée par un point virgule et la liste se termine par un point.
- Un nom est donné à chaque technique de création et de filtrage d'axies.
- Chaque nom de technique a une liste de paramètres définie entre parenthèses derrière son nom. Chaque paramètre est séparé du suivant par une virgule. Chaque technique a au moins un paramètre, le nom de la base d'axies à manipuler.

Spécification syntaxique

```
processus        = listetechniques point
listetechniques  = technique
                 | technique ; listetechniques
technique        = nom-module(parametres)
parametres       = parametre
                 | parametre , parametres
```

Spécification lexicale

```
lettre = [['a'..'z']+['A'..'Z']]
```

[2]Common Lisp Object System

```
chiffre = ['0'..'9']
tiret = '-'
tiretbas = '_'
slash = '/'
point = '.'
parametre = (lettre|chiffre|point|tiret|tiretbas|slash)+
nom-module = M-(lettre|chiffre|tiret|tiretbas)+
```

Explication

Dans un processus, chaque technique est exécutée dans l'ordre d'apparition dans la liste. Le résultat de chaque technique est transmis à travers des paramètres.

Reprenant l'exemple de la section 5.2.2.2, supposons que chaque technique soit définie par un nom et une liste de paramètres comme suit :

- `M-link_biling (nom_base_axies, nom_dico_bilingue, langue_source, langue_cible)` (pour la technique TN_1)
- `M-link_biling_inverse (nom_base_axies, nom_dico_bilingue_x-y, nom_dico_bilingue_y-z, nom_dico_bilingue_z-y, nom_dico_bilingue_y-x, langue_x, langue_y, langue_z)` (pour la technique TN_2)
- `M-filtrage_compCV (nom_base_axies, seuil_en_degre)` (pour la technique TN_3)
- `M-filtrage_pos (nom_base_axies)` (pour la technique TN_4)

Le processus pour la première situation est défini comme :

`M-link_biling_inverse(BD1,` *DictFA.dic*, *DictAT.dic*, *DictTA.dic*, *DictAF.dic*, `français, anglais, thai); M-filtrage_compCV(BD1, 45); M-filtrage_pos(BD1).`

Le processus pour la deuxième situation est défini comme :

`M-link_biling(BD2,` *DictFA1.dic*, `français, anglais); M-link_biling(BD2,` *DictFA2.dic*, `français, anglais); M-link_biling(BD2,` *DictAT.dic*, `anglais, thai); M-filtrage_compCV(BD2, 45); M-filtrage_pos(BD2).`

Nous pouvons plus tard envisager un processus plus compliqué qui a besoin de langage de processus plus évolué par exemple un langage avec `if .. then.. else`, et `switch`, etc.

Lorsqu'un processus est défini, le système doit fournir un interpréteur pour interpréter ce langage et exécuter les techniques indiquées.

5.3.2 Langage de description de stratégies d'évaluation de qualité

Dans cette section, on parle bien de *« stratégie »*, parce que c'est un plan d'action, une combinaison d'algorithmes, qui permet d'évaluer la qualité, et donc de déterminer

si l'objectif de qualité défini par le lexicologue est bien réalisé. En réalité, l'objectif de qualité est ici défini comme une valeur numérique, un seuil qui doit être dépassé par la valeur effectivement calculée. Nous proposons ici un langage simple pour exprimer des compositions de critères de qualité. C'est un langage similaire au langage de processus défini dans la section précédente.

- Une stratégie d'évaluation est définie par une liste de critères. Chaque critère est séparé du suivant par un point virgule et la liste est terminée par un point.
- Un nom est donné à chaque module de critère d'évaluation.
- Chaque nom de critère a une liste de paramètres définie entre parenthèses dernière son nom. Chaque paramètre est séparé du suivant par une virgule. Le premier paramètre pour chaque nom de critère est la valeur (positive) du poids associée à ce critère.

Spécification syntaxique

```
strategie      = listecriteres point
listecriteres  = critere
               | critere ; listecriteres
critere        = nom-module(parametres)
               | parametre , parametres
```

Spécification lexicale

```
lettre = [['a'..'z']+['A'..'Z']]
chiffre = ['0'..'9']
tiret = '-'
tiretbas = '_'
slash = '/'
point = '.'
parametre = (lettre|chiffre|point|tiret|tiretbas|slash)+
nom-module = Q-(lettre|chiffre|tiret|tiretbas)+
```

Explication

Dans une stratégie, chaque critère est exécuté dans l'ordre d'apparition dans la liste. Le résultat final est une valeur numérique égale à la somme des valeurs calculée par tous les critères dans la liste. Les modules d'évaluation sont définis avec le nom et paramètre(s) suivants :

- Critère rappel (cf. section 4.3)
  `Q-rappel(poids, nom_base_axies, fichier de référence)`
- Critère présicion (cf. section 4.3)
  `Q-precision(poids, nom_base_axies, fichier de référence)`
- Critère silence (cf. section 4.3)
  `Q-silence(poids, nom_base_axies, fichier de référence)`
- Critère bruit (cf. section 4.3)
  `Q-bruit(poids, nom_base_axies, fichier de référence)`

- Critère structural : pourcentage des lexies correctes et complètes (cf. section 4.2.3)
  `Q-structural (poids, nom_base_axies)`
- Critère vectoriel (cf. section 4.2.3)
  `Q-vecteurmoyen(poids, nom_base_axies)`

Le poids pour chaque module doit être soigneusement choisi, selon l'échelle des valeurs retournées par chaque module, et les objectifs linguistiques.

La section 4.3 a expliqué qu'une même technique (algorithme), ou une même ressource, peut être utilisée pour la création, le filtrage ou l'évaluation de qualité. C'est aussi la raison pour laquelle nous proposons des langages très similaires pour 1) définir des processus de construction, et 2) définir des stratégies d'évaluation de la qualité.

La principale différence entre ces deux parties, se situe dans 1) des préoccupations différentes du lexicologue, et 2) la différence d'implémentation des modules de construction et d'évaluation de la qualité. L'implémentation est le point de vue du programmeur que nous verrons dans le chapitre 6.

5.4 Conclusion

La qualité d'une base lexicale peut être évaluée après qu'elle ait été créée ou modifiée par l'exécution d'un processus de création d'axies. Une telle mesure de qualité peut être utilisée par des linguistes pour décider soit d'exécuter un autre processus de création d'axies pour augmenter la qualité de la base lexicale, soit de s'arrêter si la base de données a atteint la qualité désirée. La création d'une base d'axies consiste donc en des exécutions itératives et alternatives des processus de création d'axies, des évaluations de la qualité, et des décisions.

L'exécution d'un processus de création d'axies n'implique pas toujours une augmentation monotone de la qualité mesurée. Puisque les algorithmes de création d'axies peuvent être mutuellement incohérents, l'ordre des exécutions des modules, dans un processus ou dans plusieurs processus exécutés consécutivement, a un impact sur la mesure de la qualité globale.

Plus précisément, des ressources supplémentaires utilisées par des modules de création d'axies, et/ou par des modules de critère de qualité, peuvent contenir des erreurs et être mutuellement incohérentes. L'exécution d'un module de création d'axies utilisant une ressource R_1, peut faire baisser la valeur de A_1 et augmenter la valeur de E_2 si on mesure la qualité par un module de critère basé sur une ressource R_2 incohérente avec R_1. Cela peut faire diminuer de manière significative la qualité globale évaluée. La base lexicale peut, cependant, être en fait de meilleure qualité si R_2 a une qualité inférieure à R_1, R_1 ayant une bonne qualité. Cela met en valeur le besoin de ressources de bonne qualité pour créer la base de données et pour évaluer sa qualité.

Avec tous ces aspects présentés dans ce chapitre : séparation des préoccupations, adaptabilité, et combinaison des techniques, ce qu'il faut dans le cas concret du système de structuration et l'évaluation des acceptions, c'est définir un vrai algorithme précis pour chaque technique, et définir des algorithmes qui manipulent aussi les lexies,

ou des algorithmes qui manipulent à la fois les lexies et les axies. D'ailleurs, il est nécessaire de développer de nouveaux algorithmes pour augmenter la cohérence interne d'une base d'axies, par exemple un module de groupage d'axies.

Troisième partie

Implémentation et expérimentations

Chapitre 6

Système pour la structuration et l'évaluation de BDLM

Ce chapitre introduit un système pour la structuration et l'évaluation que nous appelons « *Jeminie* ». Nous présentons d'abord, dans la section 6.1.1, une vue globale des fonctions principales du système et des catégories d'utilisateurs. Ensuite la conception du système dans la section 6.2 et l'utilisation du système dans la section 6.3.

6.1 Spécifications externes

6.1.1 Fonctions principales

Nous avons identifié globalement trois fonctions que doit fournir notre système à l'utilisateur : 1) normalisation, 2) production et filtrage d'axies et 3) évaluation la qualité de la base d'axies produite.

6.1.1.1 Normalisation

La première tâche dans le système est de préparer les dictionnaires monolingues en extrayant des informations dans ces dictionnaires et de les importer dans une base de données. Les données dictionnairiques qui seront traitées dans cette étape devront déjà avoir été transformées du format d'origine au format XML de Papillon.

Le travail dans cette étape est motivé par des calculs qui seront exécutés dans l'étape de production et de filtrage d'axies. Comme les informations varient entre les différents dictionnaires monolingues, l'exécution de cette étape varie selon les ressources. La normalisation des lexies signifie à la fois la synthèse des données existantes et l'ajout de nouvelles données, par exemple :

- Analyser les définitions des lexies, en utilisant un analyseur morpho-syntaxique, pour obtenir des listes de mots de définition. Ce processus donne, en conséquence, un ensemble de mots de définition pour chaque lexie. Ces informations pourront servir comme contextes pour distinguer le sens de chaque lexie.

- Analyser les domaines des lexies et les convertir dans un ensemble de domaines commun que nous avons défini. Bien que le domaine ne soit pas toujours présent dans les bases lexicales, il semble que ce soit un bon critère de filtrage pour des axies.
- Analyser les parties du discours des lexies et les convertir en un ensemble de parties du discours commun que nous avons défini tel que nom, adverbe, adjectif, conjonction, interjection, verbe, préposition et autre.
- Calculer un vecteur conceptuel pour chaque lexie, pour pouvoir comparer des sens de façon mathématique. Le concept de vecteur conceptuel a été utilisé dans ce contexte par [LPS02, Laf02], comme nous l'avons détaillé plus haut dans la section 4.1.4.

Le résultat de cette étape est une base de lexies pour chaque langue. Si nous avons plusieurs dictionnaires monolingues pour une langue, il est possible d'avoir des lexies en double que nous devrions fusionner ultérieurement. Cela sera compliqué par le fait que des dictionnaires monolingues différents ont le plus souvent des informations différentes. Les bases de lexies ne sont donc pas en général égales au niveau de la granularité et de la richesse. (cf. section 1.3.1)

6.1.1.2 Production et filtrage d'axies

Dans cette étape, nous regroupons deux fonctions : 1) la production de nouvelles axies et 2) le filtrage d'axies existantes. Une fois l'étape de normalisation achevée, l'étape de production et filtrage d'axies peut commencer. S'il n'existe pas encore de base d'axies, l'utilisateur doit commencer par l'étape de production d'axies, puis d'autres productions ou filtrages. Dans l'étape de la production, seules de nouvelles axies sont créées. Le but de cette étape est de maximiser le nombre d'axies correctes, c'est-à-dire des axies qui relient des lexies de même sens (au sens de l'équivalence traductionnelle, puisqu'il s'agit ici de dictionnaires et pas d'ontologies).

Lorsqu'on produit des axies, il est inévitable que de mauvaises axies soient aussi produites. Nous avons donc une étape de filtrage qui supprime des lexies incorrectement rattachées à une axie ou bien fusionne des axies ayant une même signification selon les critères appliqués. Le but est de minimiser le nombre d'axies non correctes.

La production et le filtrage d'axies sont basés sur un ou plusieurs critères. Ces critères sont utilisés pour décider si une axie est correcte ou non. Le critère est par exemple, dans une axie, que seules des lexies qui ont la même partie du discours sont conservées alors que des lexies qui n'ont pas la même partie du discours que les autres sont enlevées.

Un critère peut avoir besoin de données supplémentaires. Par exemple, si on utilise la stratégie consistant à fusionner les lexies de même langue qui sont synonymes selon un dictionnaire de synonymes, il faut disposer d'un ou plusieurs dictionnaires de synonymes.

Notre système permet aux utilisateurs de composer plusieurs critères dans l'étape de production et de filtrage. Le résultat de cette étape est une base d'axies. Comme nous pouvons avoir plusieurs étapes de production et de filtrage, il est possible d'avoir

plusieurs base d'axies entre chaque étape. Nous les appelons bases intermédiaires d'axies.

6.1.1.3 Évaluation de la qualité de la base d'axies produite

Une fois qu'une base d'axies est créée, le système doit pouvoir évaluer la qualité de cette base lexicale. La bonne qualité de la base signifie que la base correspond bien aux critères appliqués pour l'évaluer. Les critères possibles ont été présentés dans la section 4.2.3. En évaluant la qualité de la base, la qualité de la technique utilisée peut être jugée.

6.1.2 Types d'utilisateur

Dans le système, il existe deux ou trois types principaux d'utilisateur : 1) le lexicologue, 2) le programmeur et 3) l'administrateur de base de données.

Lexicologue

L'objectif du lexicologue dans le système est de produire une base lexicale de la meilleure qualité possible en utilisant des méthodes adaptées aux ressources disponibles. Le but est de maximiser la qualité de la base lexicale en contrôlant son homogénéité. Le lexicologue doit décider comment choisir des lexies qui ont la même signification dans une même axie. La technique choisie par le lexicologue doit bien sûr correspondre aux ressources ou aux informations lexicales dont il dispose et qu'il peut manipuler. Le lexicologue doit également décider comment représenter des acceptions monolingues et des acceptions interlingues.

Programmeur

Le rôle du programmeur dans le système est de programmer de nouveaux modules de création et de filtrage d'axies, ainsi que des modules permettant d'évaluer la base lexicale ou de modifier des modules existants. Le programmeur doit aussi programmer et maintenir l'ensemble du système.

Administrateur de base de données

Le rôle de l'administrateur de base de données est de définir l'utilisation de la base lexicale. Il ne travaille pas directement sur le système, mais il peut avoir de l'influence sur le choix du système de gestion de base de données, et il définit les droits d'accès des utilisateurs de la base de données.

6.2 Conception du système

6.2.1 Architecture globale

Jeminie a une architecture en couches composée d'un noyau, d'un interpréteur de processus, d'un interpréteur de mesure de qualité et de plusieurs modules de construction et modules de critère, comme illustré dans la figure 6.1.

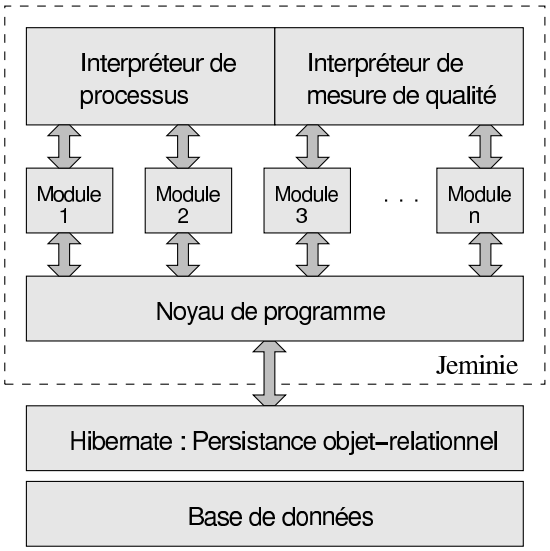

FIG. 6.1 : Architecture de Jeminie

- Le noyau est une bibliothèque de programmation Java de base qui implante tout le modèle de données, ainsi que la normalisation et l'import de dictionnaires.
- L'interpréteur de processus exécute des modules de création d'axies et de filtrage d'axies en fonction d'une spécification, écrite en langage de description de processus, fournie par un linguiste. La syntaxe que nous avons définie pour spécifier des processus est illustrée dans les exemples de « composition » de la section 5.3.1.
- L'interpréteur de mesure de qualité exécute des modules de critère d'évaluation selon une spécification définie par un linguiste. La syntaxe du langage de description de stratégies d'évaluation de qualité a été expliquée dans la section 5.3.2.
- Les modules de construction d'axies et d'évaluation de qualité sont développés en utilisant les fonctions du noyau. Jeminie peut être étendu en développant de nouveaux modules.

Les données sont représentées ici par des objets Java passés aux modules exécutés pour manipuler leurs valeurs ou pour construire de nouveaux objets. Par exemple, des objets lexies sont utilisés pour fabriquer des objets axies dans un module de construction d'axies. Le résultat produit par chaque module est sauvegardé dans une base de données à l'aide d'un intergiciel de persistance des objets et d'une base de

données relationnelle.

6.2.2 Modèle d'objets

Jeminie est développé en Java et suit le concept de programmation par objets. Il est composé de plusieurs paquets (packages) qui sont à différents niveaux d'abstraction. Le package principal est `api`. Il contient les interfaces générales qui définissent les parties de calcul du système telle que `LexieFactory` pour créer des lexies, et les interfaces qui forment un modèle des entités lexicales générales. Nous avons conçu le système pour qu'un package dépende seulement des packages plus abstraits. Par exemple, un package `api` ne dépend pas d'autres packages car il ne contient que les interfaces que les autres packages vont implémenter. Ce principe de conception nous permet de changer facilement les détails du système, qui sont implémentés dans des packages concrets, car aucune autre partie du système ne dépend d'eux. Les classes principales de modèle d'objet que nous avons définies sont :

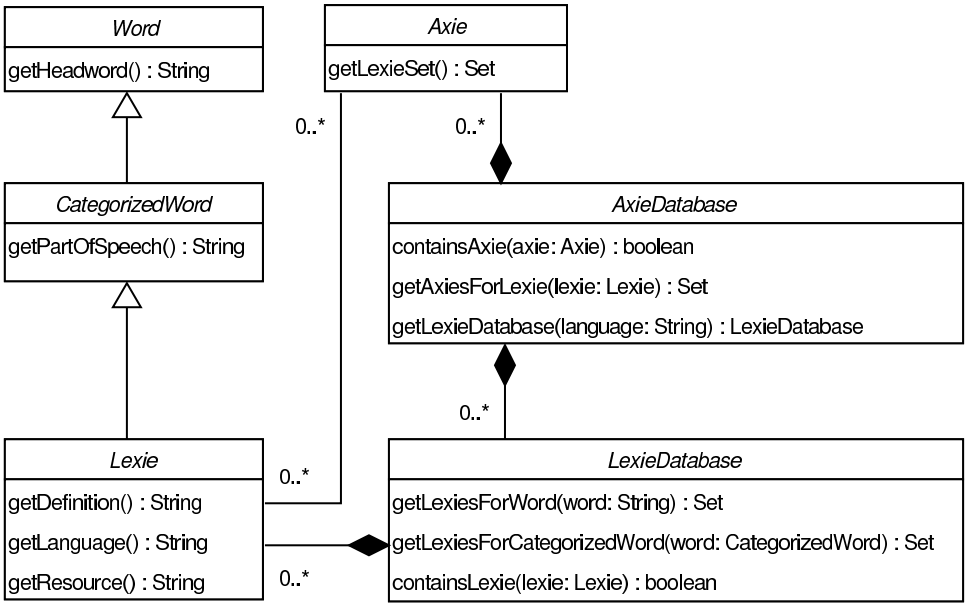

FIG. 6.2 : Exemple de modèle de données dans Jeminie

- `Word` : un objet Java contenant une chaîne de caractères qui représente un mot-vedette d'une entrée dans un dictionnaire ou un thésaurus.
- `PartOfSpeechCategory` : un objet Java contenant une chaîne de caractères qui représente une partie du discours.
- `CategorizedWord` : un objet Java qui regroupe des informations sur le mot-vedette `Word` et sur sa partie du discours `PartOfSpeechCategory`. C'est donc la représentation d'un « vocable » au sens de Mel'čuk.
- `Concept` : un `Word` qui contient un mot-vedette d'une feuille d'une structure hiérarchique de thésaurus.

- `ConceptBase` : ensemble de tous les `Concepts` contenus dans un thésaurus. Le nombre total de `concepts` sera utilisé comme la dimension des vecteurs conceptuels.
- `Lexie` : un objet Java qui regroupe des informations d'un `CategorizedWord`, une chaîne de caractères de définition, une chaîne de caractères qui représente une langue, une chaîne de caractère qui représente la ressource d'où la définition est tirée, et une axie à laquelle elle appartient (ou des axies auxquelles elle appartient temporairement).
- `ConceptualVector` : un objet Java contenant un vecteur qui a une dimension correspondant à un `ConceptBase`.
- `LexieDatabase` : un ensemble de `Lexies` d'une langue obtenues par analyse des fichiers de dictionnaires monolingues de cette langue.
- `Axie` : un objet Java contenant un ensemble de références à des `Lexies` reliées à cette axie, et eventuellement un ensemble de références à des axies (pour les problèmes contrastifs, cf. page 30).
- `AxieDatabase` : un ensemble de `Axies`.

La figure 6.2 donne un exemple de certains modèles de données. Les dépendances entre les packages sont illustrées dans la figure 6.3. Chaque package (sauf `api`) implémente un aspect du système, selon le modèle architectural défini par les interfaces du package `api`.

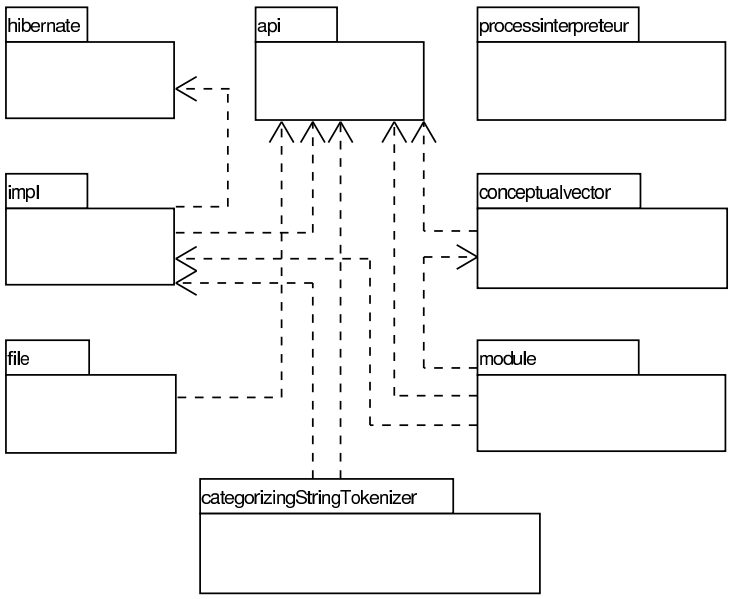

FIG. 6.3 : Dépendance des packages en Jeminie

- `impl` : implémente les interfaces de modèle de données définies dans l'`api`.
- `file` : implémente toutes les tâches qui manipulent des fichiers de données, tels que l'analyse des fichiers de lexies, des fichiers de thésaurus et des fichiers de hiérarchie de thésaurus.

- `categorizingStringTokenizer` : implémente des modules d'analyseurs morpho-syntaxique tels que l'analyseur IFSP de la plate-forme XeLDA[1] et l'analyseur XIP[2] avec l'interface d'analyseurs morpho-syntaxiques définie dans un package `api`.
- `conceptualVector` : un package contenant l'interface pour définir un vecteur conceptuel.
- `hibernate` : implémente des classes utilitaires pour la persistance d'objets dans une base lexicale. Le système Jeminie sauvegarde tous les objets lexicaux dans une base de données PostgreSQL, en utilisant Hibernate [Hib].
- `module` : implémente toutes les techniques de création d'axies, de filtrage d'axies et d'évaluation de qualité d'une base d'axies. On peut remarquer que dans la figure 6.3, le package `module` dépend du package `conceptualVector` car il implémente une technique qui utilise des vecteurs conceptuels. Il dépend aussi du package `impl` car il utilise des classes de lexies et d'axies implémentées dans le package `impl`.
- `processinterpreter` : implémente l'interpréteur de processus de création d'axies et l'interpréteur de stratégie d'évaluation de qualité.

6.2.3 Persistance transparente d'objets

Pour sauvegarder les valeurs des données lexicales calculées au cours de l'exécution d'un processus de production d'axies, nous utilisons le logiciel Hibernate intégré dans un des services de notre système. Hibernate[3] est un logiciel de persistance objet-relationnel personnalisé. Il peut s'adapter à plusieurs systèmes de gestion de bases de données. Par conséquent, nous pouvons configurer facilement Hibernate afin d'utiliser un autre système de gestion de base de données.

Pour utiliser Hibernate, nous séparons les tâches en deux parties : 1) des fichiers de configuration d'Hibernate et d'une base de données et 2) des codes source ajoutés dans les classes Java.

Dans la première partie, deux fichiers : fichier « *mapping* » et fichier « *properties* » doivent être renseignés. Le fichier mapping contient les métadonnées requises pour le mapping objet/relationnel. Les métadonnées contiennent la déclaration de toutes les classes persistantes et une correspondance entre les attributs des classes et les tables de la base de données. Pour chaque classe, tous les attributs qui vont être sauvegardés dans la base de données doivent être déclarés. Les mappings objet/relationnel sont définis dans un fichier XML. Le fichier mapping est aussi utilisé pour générer le schéma de la base de données. Un extrait de fichier mapping pour Jeminie est présenté dans la figure 6.4. `Class` et `joined-subclass` sont des classes de données. `Property` représente un attribut.

[1] « Xerox Engine for Linguistic Dependent Applications » (XeLDA) une plateform développée par XEROX – `http://www.xrce.xerox.com/`.

[2] « Xerox Incremental Parsing » (XIP) est une analyseur morpho-syntaxique de XEROX – `http://www.xrce.xerox.com/`.

[3] `http://www.hibernate.org/`

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
    "http ://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>

<!--org.papillonDictionary.jeminie.impl.BasicWord-->
 <class name="org.papillonDictionary.jeminie.impl.BasicWord"
      table="words">
  <id name="UID" type="long" column="uid" unsaved-value="-1">
    <generator class="sequence">
       <param name="sequence">uid_sequence_Word</param>
    </generator>
  </id>
    <property name="headword" type="string"/>

  <!-- org.papillonDictionary.jeminie.impl.BasicCategorizedWord-->
  <joined-subclass
    name="org.papillonDictionary.jeminie.impl.BasicCategorizedWord"
    table="categorizedWords">
    <key column="Word_uid" />
    <many-to-one name="partOfSpeechCategory"
       class="org.papillonDictionary.jeminie.impl.
              BasicPartOfSpeechCategory"
       column="PartOfSpeechCategory_uid"
       cascade="save-update" />

    <!--org.papillonDictionary.jeminie.impl.BasicLexie-->
    <joined-subclass
       name="org.papillonDictionary.jeminie.impl.BasicLexie"
       table="lexies">
       <key column="Word_uid" />
       <property name="definition" type="string" length="1024"/>
       <property name="language" type="string" length="3"/>
       <property name="resource" type="string" length="20"/>
       <set name="axieSet" table="axiesetinlexie"
          lazy="true" cascade="save-update">
          <key column="lexie_uid"/>
          <many-to-many column="linkedaxie_uid"
            class="org.papillonDictionary.jeminie.impl.BasicAxie"/>
       </set>
    ...
    </joined-subclass>
   </joined-subclass>
   ...
  </class>
</hibernate-mapping>
```

Fig. 6.4 : Extrait de fichier mapping de Hibernate pour Jeminie.

Le fichier « *properties* » contient des paramètres pour configurer Hibernate. Parmi ces paramètres de configuration, le paramètre qu'on a besoin de spécifier est le dialecte SQL où on donne le nom de la base de données à modifier et le système de gestion de bases de données. La figure 6.5 donne un exemple de fichier properties de Jeminie. Dans l'exemple, nous spécifions le système de gestion de bases de données PostgreSQL. Nous donnons également le nom de la base de données et le nom d'utilisateur. Pour le reste, nous utilisons la valeur proposée par défaut.

```
## General properties
hibernate.cglib.use_reflection_optimizer false

## Properties for the Postgresql database
hibernate.dialect
    net.sf.hibernate.dialect.PostgreSQLDialect
hibernate.connection.driver_class org.postgresql.Driver
hibernate.connection.url
    jdbc :postgresql ://localhost :5432/jeminie
hibernate.connection.username jeminie
hibernate.connection.password jeminie
hibernate.query.substitutions yes 'Y', no 'N'

## Properties for the C3P0 connection pool
hibernate.c3p0.max_size 10
hibernate.c3p0.min_size 1
hibernate.c3p0.timeout 5000
hibernate.c3p0.max_statements 100

## Properties for the transactions
hibernate.transaction.factory_class
    net.sf.hibernate.transaction.JDBCTransactionFactory
hibernate.jdbc.batch_size 10
hibernate.jdbc.use_streams_for_binary true

## print all generated SQL to the console
hibernate.show_sql false
```

FIG. 6.5 : Exemple de fichier properties de Jeminie

Dans la partie qui concerne des classes Java, on doit ajouter deux méthodes : `get` et `set` pour tous les attributs qui seront stockés dans la base de données. La méthode `get` donne à Hibernate la valeur de l'objet de l'attribut indiqué, et la méthode `set` spécifie la valeur de l'objet.

La persistance entre les objets Java et une base de données se fait à traverse une « *session* » qui est une classe de Hibernate. Avec une session, nous n'avons pas besoin de savoir comment se connecter avec la base de données pour sauvegarder et récupérer les données. D'ailleurs, avec Hibernate, l'administrateur de la base de données n'a besoin de modifier que les fichiers de mapping et de properties de Hibernate pour changer le schéma. Il n'a pas besoin de modifier le code de Jeminie. C'est un gros

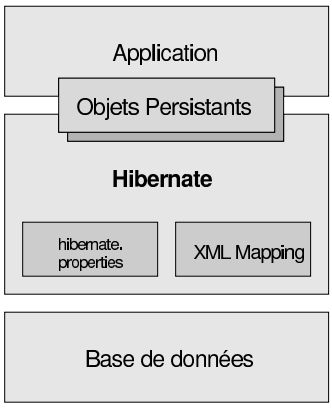

FIG. 6.6 : Architecture globale de Hibernate

avantage de Hibernate. Il permet de séparer complètement la préoccupation du code, de la préoccupation du stockage dans une base de données. Nous pouvons modifier les deux indépendamment. La figure 6.6 illustre l'architecture globale de Hibernate.

6.3 En pratique

6.3.1 Comment ajoute-t-on un nouveau module ?

Il existe deux interfaces. Une interface pour des modules de création et de filtrage d'axies, et une autre interface pour des modules d'évaluation de qualité. Pour ajouter un nouveau module, il faut simplement implémenter une classe d'une des deux interfaces.

6.3.1.1 Module de création et de filtrage d'axies

L'interface de module de création et de filtrage d'axies est illustrée dans le programme 6.1.

```
public interface AxieCreator {
    boolean checkAxieCreatorParameters(List parameters);
    void updateAxieDatabase(AxieDatabaseContext
                            context, List parameters);
}
```

PROG. 6.1 : Interface `AxieCreator`.

Cette interface a deux méthodes : `checkAxieCreatorParameters` et `updateAxieDatabase`. Pour ajouter un nouveau module, il faut implémenter un nouvel algorithme pour la méthode `updateAxieDatabase`. La méthode `checkAxieCreatorParameters` a pour but de vérifier si la liste de paramètres fournis correspond bien à la liste requise par ce

module. La méthode `updateAxieDatabase` modifie la base lexicale indiquée dans le paramètre `AxieDatabaseContext` avec une technique spécifique à ce module en utilisant des informations supplémentaires dans la liste de paramètres. `AxieDatabaseContext` contient l'information sur la base d'axies à modifier, telle que le nom de la base et la session d'Hibernate.

6.3.1.2 Module d'évaluation de qualité

Comme décrit dans la section 4.2.3, nous imposons que chaque module d'évaluation de qualité donne comme résultat d'évaluation une valeur numérique. Une interface de module d'évaluation de qualité est illustrée dans le programme 6.2. Chaque module a une méthode `getQualityValue` qui donne la valeur d'évaluation. Différentes méthodes d'évaluation peuvent être programmées dans la méthode `getQualityValue`.

```
public interface AxieEvaluation {
    boolean checkAxieEvaluationParameters(List parameters);
    float getQualityValue(AxieDatabaseContext context,
                          List parameters);
}
```

Prog. 6.2 : Interface `AxieEvaluation`.

6.3.2 Programmes utilisateurs de base

6.3.2.1 Normalisation

Dans l'étape de la normalisation, Jeminie implémente des opérations de base pour créer une base de lexies à partir du dictionnaire de définitions monolingues. Les opérations de base sont : initBase, initLexieSimple, initBaseVecteurs, initLexieCV, addLexieSimple, et addLexieCV.

Il est à noter que le format de dictionnaire monolingue utilisé est le format XML défini dans le projet Papillon.

initBase Créer une base de données vide pour stocker des lexies et des axies, avec le système de gestion de bases de données et le nom de la base de données spécifiés dans `propertiesfile`, et un schéma de base de données définit dans `mappingfile`.

Paramètres :

- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.

Résultat :

Une base de données vide.

initLexieSimple Créer des lexies simples à partir d'un dictionnaire de définitions monolingues.

Paramètres :

- `monodico` - un fichier d'un dictionnaire monolingue au format XML de Papillon.
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `language` - la langue du dictionnaire monolingue. La langue est spécifiée par une chaîne de trois caractères, comme défini dans $ISO\ 639-2/T$.

Résultat :

Une base de lexies où chaque lexie comprend un mot-vedette, une partie du discours, une définition, la ressource dont elle est tirée, et la langue de cette lexie.

initBaseVecteurs Créer une base de vecteurs conceptuels initiaux à partir d'une base lexicale contenant pour chaque entrée : un mot-vedette, une partie du discours, et les concepts correspondants (qui sont des concepts feuilles d'un thésaurus).

Paramètres :

- `listconcept` - fichier contenant une liste de concepts feuilles d'un thésaurus sur le format XML.
- `indexedfile` - fichier contenant des mot-vedettes, leur partie du discours et les concepts correspondants.
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `language` - la langue de la base des vecteurs conceptuels.

Résultat :

Une base de vecteurs conceptuels de la langue spécifiée.

initLexieCV Créer des lexies, associées à un vecteur conceptuel, à partir d'un dictionnaire de définitions monolingues.

Paramètres :

- `monodico` - un fichier d'un dictionnaire monolingue au format XML de Papillon.
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `basevecteurUID` - un ID de la base des vecteurs conceptuels initiaux.
- `language` - la langue du dictionnaire monolingue.

Résultat :

Une base de lexies où chaque lexie comprend un mot-vedette, une partie du discours, une définition, la ressource dont elle est tirée, et la langue de cette lexie. Chaque lexie est associée à un vecteur conceptuel correspondant.

addLexieSimple Créer et ajouter des lexies simples à une base de lexies existante, à partir d'un autre dictionnaire de définitions monolingues.

Paramètres :

- `monodico` - un fichier d'un dictionnaire monolingue au format XML de Papillon.

- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `baselexieUID` - un ID de la base de lexies à laquelle on ajoute de nouvelles lexies.
- `language` - la langue du dictionnaire monolingue.

Résultat :

La base de lexies `baselexieID` modifiée en ajoutant de nouvelles lexies.

addLexieCV Créer et ajouter des lexies, associées à un vecteur conceptuel, à une base de lexies existante, à partir d'un autre dictionnaire de définitions monolingues.

Paramètres :

- `monodico` - un fichier d'un dictionnaire monolingue au format XML de Papillon.
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `basevecteurUID` - un ID de la base des vecteurs conceptuels initiaux.
- `baselexieUID` - un ID de la base de lexies à laquelle on ajoute des nouvelles lexies.
- `language` - la langue du dictionnaire monolingue.

Résultat :

La base de lexies `baselexieID` modifiée en ajoutant de nouvelles lexies.

6.3.2.2 Création et filtrage d'axies

Nous avons développé plusieurs modules de création et de filtrage d'axies qui implantent l'interface de programmation `AxieCreator` donnée dans le programme 6.1. Ces modules sont décrits dans cette section.

Il est à noter que nous avons défini un nouveau format simple, basé sur XML, pour le stockage de dictionnaires bilingues. Ce format unique est utilisé par tous nos modules qui utilisent des dictionnaires bilingues. Ce format est défini par les balises XML suivantes :

- Un élément `<volume>` représente un dictionnaire entier. Il contient plusieurs éléments `<dict-entry>` ainsi que les trois attributs suivants :
  - `name` : le nom du dictionnaire ;
  - `source-language` : la langue source du dictionnaire ;
  - `target-language` : la langue cible du dictionnaire.
- Un élément `<dict-entry>` représente une entrée d'un dictionnaire bilingue. Il contient à son tour un élément `<entry>`, un élément `<pos>`, un ou plusieurs éléments `<translation>`, et optionnellement un élément `<glose>`.
- Un élément `<entry>` contient un mot-vedette de la langue source.
- Un élément `<pos>` contient l'indication de la partie du discours d'une entrée.
- Un ou plusieurs éléments `<translation>` contiennent le(s) mot(s) traduction dans la langue cible. Une entrée peut avoir un ou plusieurs mots traductions dans la langue cible, ce qui se traduit par la présence d'un ou plusieurs éléments `<translation>` dans un élément `<dict-entry>`.

– Un élément optionnel `<glose>` indique une glose d'une entrée. Cette information est optionnelle, parce que certains dictionnaires n'ont pas cette information.

Un extrait du dictionnaire bilingue *Oxford french minidictionary* français-anglais est donné dans la figure 6.7.

```
<?xml version="1.0" encoding="UTF-8"?>
<volume name="oxford_mini" source-language="fra" target-language="eng">
  <dict-entry>
     <entry>absent</entry>
     <pos>a.</pos>
     <translation>absent</translation>
  </dict-entry>
  <dict-entry>
     <entry>comprendre</entry>
     <pos>v.tr.</pos>
     <translation>understand</translation>
     <translation>comprise</translation>
  </dict-entry>
</volume>
```

Fig. 6.7 : Extrait du dictionnaire bilingue *Oxford french minidictionary* français-anglais

BlingAxieCreator (cf. section 4.1.3.1) La méthode analyse le fichier du dictionnaire bilingue et relie, dans une même axie, des lexies dont les mots-vedettes sont la traduction l'une de l'autre et sauvegarde cette axie dans la base d'axies spécifiée.

Paramètres :

– `bilingdico` - un fichier d'un dictionnaire bilingue.
– `propertiesfile` - un fichier properties de Hibernate.
– `mappingfile` - un fichier mapping de Hibernate.
– `sourcelanguage` - la langue source du dictionnaire bilingue.
– `targetlanguage` - la langue cible du dictionnaire bilingue.
– `axieDatabaseUid` - un ID de la base d'axies.

Résultat : La base d'axies `axieDatabaseUid` modifiée en ajoutant de nouvelles axies.

Remarquons qu'une lexie peut alors être reliée à plus d'une axie, ce qui n'est pas correct. Le résultat de cette méthode nécessite une amélioration ultérieure, soit par une autre méthode, soit par un lexicographe humain.

BlingTransfertAxieCreator

Paramètres :

– `bilingdico1` - un fichier d'un dictionnaire bilingue $X \longrightarrow Y$.
– `bilingdico2` - un fichier d'un dictionnaire bilingue $Y \longrightarrow Z$.
– `propertiesfile` - un fichier properties de Hibernate.
– `mappingfile` - un fichier mapping de Hibernate.
– `sourcelanguage` - la langue source X.

- `intermediatelanguage` - la langue intermédiaire Y
- `targetlanguage` - la langue cible Z.
- `axieDatabaseUid` - un ID de la base d'axies.

Résultat : La base d'axies `axieDatabaseUid` modifiée en ajoutant de nouvelles axies.

En effet, la méthode de transfert est une fusion des résultats de deux base d'axies intermédiaires obtenues par la méthode de création d'axies avec dictionnaire bilingue simple. Pour avoir des axies de trois langues, nous regroupons les couples d'axies venant de deux bases d'axies qui ont une lexie en commun. Par exemple, si nous avons deux axies :

- une axie XY_1, d'une base d'axies entre langue X et langue Y, contient deux lexies A_1 et B_1 (notée $XY_1(A_1, B_1)$),
- une axie YZ_1, d'une base d'axies entre langue Y et langue Z, contient deux lexies B_1 et C_1 (notées $YZ_1(B_1, C_1)$)

Le résultat de la méthode de transfert donne une axie $XYZ_1\,(A_1, B_1, C_1)$ comme illustré dans la figure 6.8.

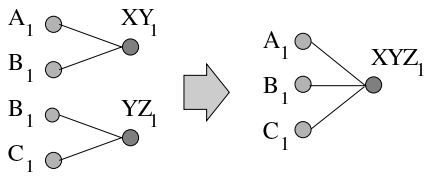

FIG. 6.8 : Création d'axies par transfert

BlingTransfertInverseAxieCreator (cf. section 4.1.3.2)

Paramètres :

- `bilingdico1` - un fichier d'un dictionnaire bilingue $X \longrightarrow Y$.
- `bilingdico2` - un fichier d'un dictionnaire bilingue $Y \longrightarrow Z$.
- `bilingdico3` - un fichier d'un dictionnaire bilingue $Z \longrightarrow Y$.
- `bilingdico4` - un fichier d'un dictionnaire bilingue $Y \longrightarrow X$.
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `sourcelanguage` - la langue source X.
- `intermediatelanguage` - la langue intermédiaire Y
- `targetlanguage` - la langue cible Z.
- `axieDatabaseUid` - un ID de la base d'axies.

Résultat : La base d'axies `axieDatabaseUid` modifiée en ajoutant de nouvelles axies.

CVAxieCreator (cf. section 4.1.4)

Paramètres :

- `bilingdico` - un fichier d'un dictionnaire bilingue
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.

- `sourcelanguage` - la langue source du dictionnaire bilingue.
- `targetlanguage` - la langue cible du dictionnaire bilingue.
- `axieDatabaseUid` - un ID de la base d'axies.
- `threshold` - la valeur de distance angulaire maximale (en degrés) qu'on utilise comme seuil pour relier des lexies à une même axie.

Résultat : La base d'axies `axieDatabaseUid` modifiée en ajoutant de nouvelles axies.

Pour utiliser ce module, il est nécessaire que chaque lexie de la langue source et de la langue cible soit associée à un vecteur conceptuel (cf. plus de détails sur les vecteurs conceptuels dans les sections 4.1.4 et 7.2.2). Dans la méthode `updateAxieDatabase`, un dictionnaire bilingue est utilisé pour trouver des paires de mots traductions.

CVAxieFilter (cf. section 5.2.2.2)

Paramètres :

- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `axieDatabaseUid` - un ID de la base d'axies.
- `threshold` - la valeur de distance angulaire maximale (en degrés) qu'on utilise comme seuil pour filtrer des lexies liées à une même axie.

Résultat : La base d'axies `axieDatabaseUid` avec des axies filtrées par ce module.

Pour utiliser ce module, comme le module précédent, il est nécessaire que chaque lexie soit associée à un vecteur conceptuel.

6.3.2.3 Évaluation de qualité

Nous avons implémenté les quatre modules Rappel, Précision, Silence, et Bruit définies dans la section 4.3, avec les noms de classes : `Rappel`, `Precision`, `Silence`, et `Bruit` respectivement. Ces quatre modules donnent comme résultat une valeur d'autant meilleure qu'elle est grande (donc on inverse les mesures de silence et de bruit) et ont les mêmes paramètres.

Paramètres :

- `reference` - un fichier d'une base de référence.
- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `axieDatabaseUid` - un ID de la base d'axies.

Chaque fichier d'une base de référence est dans un format textuel où chaque entrée est une ligne dont le format est :

```
id|mot_source(partie du discours)|définition_source|mot_cible(partie du discours)|
définition_cible
```

Voici un exemple d'entrée :

```
6|aboiement(N)|cri du chien|bark(N)|the sound made by a dog
```

Nous avons implémenté également un module évaluant le critère structural défini dans la section 4.2.3.

StructuralEval

Paramètres :

- `propertiesfile` - un fichier properties de Hibernate.
- `mappingfile` - un fichier mapping de Hibernate.
- `axieDatabaseUid` - un ID de la base d'axies.

Résultat : Une valeur mesurant la qualité structurale.

6.3.3 Utilisation du système

L'exécution des modules se fait actuellement en ligne de commande avec une commande java, en appelant soit sa classe `Main`, soit un interpréteur de processus. Par convention, la classe `Main` de chaque opération commence par `Main_` suivi par le nom d'opération défini dans la section précédente. L'exécution par la classe `Main` est par exemple.

Exemple 1 Initialisation de la base :

```
c :\>java org.papillonDictionary.jeminie.Main_initBase \
hibernate.properties \
JeminieMapping.xml
```

`Main_initBase` est une classe `Main` pour le module `initBase`, et `hibernate.properties` (cf. figure 6.5) et `JeminieMapping.xml` sont ses deux paramètres.

Exemple 2 Import d'un dictionnaire monolingue pour le français :

```
c :\>java org.papillonDictionary.jeminie.Main_initLexieSimple \
dictfr-a.xml \
hibernate.properties \
JeminieMapping.xml \
fra
```

`Main_initLexieSimple` est une classe `Main` pour le module `initLexieSimple` avec `dictfr-a.xml`, `hibernate.properties`, `JeminieMapping.xml`, et `fra` comme paramètres.

La figure 6.9 donne un extrait du fichier `dictfr-a.xml`. Le fichier utilise les mêmes balises que le format XML de Papillon mais avec moins d'informations, car c'est un fichier créé automatiquement à partir d'un dictionnaire de définitions. En résultat, on obtient une base de lexies du français. Un extrait de la base de lexies créée est donné dans la figure 6.10.

Exemple 3 Import d'un dictionnaire monolingue pour l'anglais :

```
c :\>java org.papillonDictionary.jeminie.Main_initLexieSimple \
dicten-01.xml \
hibernate.properties \
JeminieMapping.xml \
eng
```

`Main_initLexieSimple` est une classe `Main` pour le module `initLexieSimple` avec `dicten-01.xml`, `hibernate.properties`, `JeminieMapping.xml`, et `eng` comme paramètres.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--?xml-stylesheet type="text/css" href="../papillon-fr.css"? -->
<volume xmlns="http ://www-clips.imag.fr/geta/services/dml"
  xmlns :xlink="http ://www.w3.org/1999/xlink"
  xmlns :d="http ://www-clips.imag.fr/geta/services/dml"
  xmlns :xsi="http ://www.w3.org/2001/XMLSchema-instance"
  xsi :schemaLocation="http ://www-clips.imag.fr/geta/services/dml
  http ://www-clips.imag.fr/geta/services/dml/papillon_fra.xsd"
  d :history-ref="http ://www-clips.imag.fr/geta/services/dml/
  papillon-his.xml"
  source-language="fra">
<lexie>
  <headword>abaisser</headword>
  <pos>v.tr.</pos>
  <semantic-formula>faire descendre</semantic-formula>
  <government-pattern>
    <mod><actor/></mod>
  </government-pattern>
  <more-info><provenance name="LAR"></provenance></more-info>
</lexie>
<lexie>
  <headword>abandon</headword>
  <pos>n.m.</pos>
  <semantic-formula>
    le fait de cesser intentionnellement d'accomplir une obligation
  </semantic-formula>
  <government-pattern>
    <mod><actor/></mod>
  </government-pattern>
  <more-info><provenance name="LAR"></provenance></more-info>
</lexie>
<lexie>
  <headword>convertir</headword>
  <pos>v.tr.</pos>
  <semantic-formula>
    faire changer de croyance ou de religion
  </semantic-formula>
  <government-pattern>
    <mod><actor/></mod>
  </government-pattern>
  <more-info><provenance name="HDL"></provenance></more-info>
</lexie>
...
</volume>
```

FIG. 6.9 : Extrait du fichier `dictfr-a.xml`

uid	mot	définition	lang	ressrc	pos
112	abaisser	faire descendre	fra	lar	verbe
113	abandon	le fait de cesser intentionnellement d'accomplir une obligation	fra	lar	nom
114	convertir	faire changer de croyance ou de religion	fra	lar	verbe
120	central	qui est au centre	fra	hdl	adj

FIG. 6.10 : Extrait de la base de lexies françaises (en PostgreSQL)

La figure 6.11 donne un extrait du fichier `dicten-01.xml`. Un extrait de la base de lexies créée est montré dans la figure 6.12.

Exemple 4 Initialisation de la base de vecteurs conceptuels initiaux pour le français :

```
c :\>java org.papillonDictionary.jeminie.Main_initBaseVecteurs \
hierarchieLarousse.xml \
indexed_fr_01.txt \
hibernate.properties \
JeminieMapping.xml \
fra
```

`Main_initBaseVecteurs` est une classe `Main` pour le module `initLexieSimple`. Les figures 6.13 et 6.14 donnent un extrait du fichier `hierarchieLarousse.xml` et du fichier `indexed_fr_01.txt` respectivement.

Dans le fichier `hierarchieLarousse`, les balises sont :

- `name` : nom de la hiérarchie,
- `level` : délimite chaque niveau de la hiérarchie. Par exemple, le concept nommé *les_ concepts_ fondamentaux* est au niveau 2 et est fils du concept de niveau 1 *le_ monde*.
- `concept word :` le nom du concept feuille.

Dans `indexed_fr_01.txt`, un mot est indexé par un ensemble de concepts trouvés dans les balises « `conceptword` » de `hierarchieLarousse.xml`. Chaque concept est noté par un numéro correspondant au rang d'apparition du concept dans `hierarchieLarousse.xml`.

On obtient comme résultat une base de vecteurs conceptuels pour le français. Un extrait de la base de vecteurs conceptuels créée est donné dans la figure 6.15. Chaque vecteur conceptuel proprement dit (873 entrées) est codé sous forme compressé dans un autre fichier, et on ne trouve ici que son identification (ou `uid`)

Exemple 5 Initialisation de la base de vecteurs conceptuels initiaux pour l'anglais :

```
c :\>java org.papillonDictionary.jeminie.Main_initBaseVecteurs \
hierarchieLarousse.xml \
indexed_eng.txt \
hibernate.properties \
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--?xml-stylesheet type="text/css" href="../papillon-en.css"? -->
<volume xmlns="http ://www-clips.imag.fr/geta/services/dml"
  xmlns :xlink="http ://www.w3.org/1999/xlink"
  xmlns :d="http ://www-clips.imag.fr/geta/services/dml"
  xmlns :xsi="http ://www.w3.org/2001/XMLSchema-instance"
  xsi :schemaLocation="http ://www-clips.imag.fr/geta/services/dml
  http ://www-clips.imag.fr/geta/services/dml/papillon_eng.xsd"
  d :history-ref="http ://www-clips.imag.fr/geta/services/dml/
  papillon-his.xml"
  source-language="eng">
<lexie>
  <headword>pole</headword>
  <pos>n.</pos>
  <semantic-formula>
    one of the two ends of a magnet where the magnetism seems to
    be concentrated
  </semantic-formula>
  <government-pattern>
    <mod><actor/></mod>
  </government-pattern>
  <more-info><provenance name="wordnet"></provenance></more-info>
</lexie>
<lexie>
  <headword>pool</headword>
  <pos>n.</pos>
  <semantic-formula>
    an excavation that is filled with water
  </semantic-formula>
  <government-pattern>
    <mod><actor/></mod>
  </government-pattern>
  <more-info><provenance name="wordnet"></provenance></more-info>
</lexie>
<lexie>
  <headword>convert</headword>
  <pos>v.</pos>
  <semantic-formula>change religious beliefs</semantic-formula>
  <government-pattern>
    <mod><actor/></mod>
  </government-pattern>
  <more-info><provenance name="wordnet"></provenance></more-info>
</lexie>
...
</volume>
```

FIG. 6.11 : Extrait du fichier `dicten-01.xml`

```
uid |  mot  |             définition                |lang|ressrc |pos
----+-------+---------------------------------------+----+-------+-----
8503|pole   |one of the two ends of a magnet where  |eng |wordnet|nom
    |       |the magnetism seems to be concentrated |    |       |
8504|pool   |an excavation that is filled with water|eng |wordnet|nom
8505|convert|change religious beliefs               |eng |wordnet|verbe
8518|central|in or near a center                    |eng |wordnet|adj
```

FIG. 6.12 : Extrait de la base de lexies anglaises(en PostgreSQL)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<hierarchy_description>
<name>Larousse</name>
<level>omega
  <level>le_monde
    <level>les_concepts_fondamentaux
      <level>existence
        <conceptword="existence" />
        <conceptword="inexistence" />
        <conceptword="matérialité" />
        <conceptword="immatérialité" />
        <conceptword="substance" />
        <conceptword="accident" />
        <conceptword="état" />
        <conceptword="circonstance" />
        <conceptword="présence" />
        <conceptword="absence" />
        <conceptword="apparition" />
        <conceptword="disparition" />
      </level>
      <level>causalité
        <conceptword="cause" />
        <conceptword="effet" />
        <conceptword="agent" />
        <conceptword="motif" />
        <conceptword="but" />
        <conceptword="possibilité" />
        <conceptword="impossibilité" />
        <conceptword="nécessité" />
        <conceptword="éventualité" />
        <conceptword="probabilité" />
        <conceptword="hasard" />
      </level>
...
</level>
```

FIG. 6.13 : Extrait du fichier `hierarchieLarousse.xml`

```
mauvâtre | ADJ | 601
pyromètre | N | 569 244
schnorchel | N | 229
épeichette | N | 443
baladin | N | 536 275
brayer | N | 459
repasser | V | 541 825 442 490 477
hasardé | ADJ | 74 709 323 215
inemploi | N | 383 833
sécessionniste | N | 364 660 444
sécessionniste | ADJ | 364
masculin | N | 561
masculin | ADJ | 279
attaquer | V | 513 481 166 99 286 432 711 657 640 856 734 400 271 109
émousser | V | 760
repassage | N | 643
quinzomadaire | N | 454
forbannir | V | 764
```

FIG. 6.14 : Extrait du fichier `indexed_fr_01.txt`

```
mot_uid| entree  |  pos  |vecteur_uid|lang|conceptbase_uid|basecv_uid
-------+---------+-------+-----------+----+---------------+----------
 900   |mauvâtre |adverbe|   900     |fra |      1        | 1
 901   |pyromètre|nom    |   901     |fra |      1        | 1
 902   |schnorche|nom    |   902     |fra |      1        | 1
 903   |baladin  |nom    |   903     |fra |      1        | 1
```

FIG. 6.15 : Extrait de la base de vecteurs conceptuels initiaux pour le français

```
JeminieMapping.xml \
eng
```

La figure 6.16 donne un extrait du fichier `indexed_eng.txt`. Un extrait de la base de vecteurs conceptuels créée est montré dans la figure 6.17.

La base de vecteurs conceptuels pour l'anglais utilise la même base de concepts que pour le français (les concepts du thésaurus Larousse), leurs numéros de `conceptbase_uid` sont donc les mêmes.

```
absence | N | 450 226 204 615 574
absorption | N | 509 621 765 235 73
acceleration | N | 81 721 144 63
acceptance | N | 445 674 563 662 712 550
accessory | ADJ | 669 828 603
accident | N | 81 617 602 337 79 425 372 215 631 5
active | ADJ | 833 721 45 86 634
address | N | 743 14 188 297 415
adjacent | ADJ | 352 206
adventure | N | 530 425 215 45
advice | N | 209 24 849 126 331 502
age | N | 747 672 784 256 412
age | V | 747 242 637 523
```

Fig. 6.16 : Extrait du fichier `indexed_eng.txt`

```
mot_uid| entree     |pos|vecteur_uid|lang|conceptbase_uid|basecv_uid
-------+------------+---+-----------+----+---------------+----------
 51001 |absence     |nom|  51001    |eng |      1        | 2
 51002 |absorption  |nom|  51002    |eng |      1        | 2
 51003 |acceleration|nom|  51003    |eng |      1        | 2
 51004 |acceptance  |nom|  51004    |eng |      1        | 2
```

Fig. 6.17 : Extrait de la base de vecteurs conceptuels initiaux pour l'anglais

Exemple 6 Création d'une base de lexies du français associant chaque entrée à un vecteur conceptuel :

```
c :\>java org.papillonDictionary.jeminie.Main_initLexieCV \
dicten-01.xml \
hibernate.properties \
JeminieMapping.xml \
1 fra
/* 1 : id de la base des vecteurs conceptuels initiaux */
```

On crée ici une base de lexies avec les mêmes données monolingues du français que dans l'exemple 2. Mais pour exécuter `Main_initLexieCV`, la base des vecteurs conceptuels initiaux pour le français doit être déjà créée.

Dans la figure 6.15, l'ID de la base des vecteurs conceptuels initiaux est dans le champ `basecv_uid`. Un extrait de la base de lexies associée aux vecteurs conceptuels créée est donné dans la figure 6.18.

uid	mot	définition	lang	ressrc	pos	vecteur_uid
300	abaisser	faire descendre	fra	lar	verbe	71121
301	abandon	le fait de cesser	fra	lar	nom	71122
		intentionnellement d'accomplir				
		une obligation				
302	abandon	état de ce qui est délaissé	fra	hdl	adv	71123
303	abandonner	laisser entièrement et	fra	hdl	verbe	71124
		volontairement au pouvoir de				
		quelqu'un, quelque chose				

FIG. 6.18 : Extrait d'une base de lexies associées à leurs vecteurs conceptuels pour le français

Exemple 7 Création d'une structure d'axies entre deux bases lexicales monolingues, en utilisant le dictionnaire bilingue français-anglais *Oxford french minidictionary* :

```
c :\>java org.papillonDictionary.jeminie.Main_BlingAXieCreator \
mini_oxfordFE.xml \
hibernate.properties \
JeminieMapping.xml \
fra eng 1
/* 1 : id de la base d'axies */
```

La figure 6.19 donne un extrait du fichier `mini_oxfordFE.xml`. Un extrait de la base d'axies créée est montré dans la figure 6.20.

Exemple 8 Création d'une structure d'axies entre deux bases lexicales monolingues, en utilisant le module `CVAxieCreator` :

```
c :\>java org.papillonDictionary.jeminie.Main_CVAxieCreator \
mini_oxfordFE.xml \
hibernate.properties \
JeminieMapping.xml \
fra eng 1 45
/* 1 : id de la base d'axies */
/* 45 : seuil (45°)*/
```

Un extrait de la base d'axies créée est donné dans la figure 6.21. Par rapport au résultat de l'exemple 7, le lien entre la lexie 115 et 8505 n'a pas été créé car la distance angulaire entre les deux vecteurs conceptuels associés à ces deux lexies est plus grande que le seuil spécifié (45°)

```
<?xml version="1.0" encoding="UTF-8"?>
<volume name="oxford_mini" source-language="fra"
        target-language="eng">
<dict-entry> <entry>absent</entry> <pos>a.</pos>
   <translation>absent</translation>
</dict-entry>
<dict-entry> <entry>convertir</entry> <pos>v.tr.</pos>
   <translation>convert</translation>
</dict-entry>
<dict-entry> <entry>affecter</entry> <pos>v.tr.</pos>
   <translation>affect</translation>
   <translation>assign</translation>
   <translation>appoint</translation>
   <translation>post</translation>
</dict-entry>
<dict-entry> <entry>central</entry> <pos>a.</pos>
   <translation>central</translation>
</dict-entry>
<dict-entry> <entry>couloir</entry> <pos>n.m.</pos>
   <translation>corridor</translation>
   <translation>lane</translation>
   <translation>aisle</translation>
</dict-entry>
<dict-entry> <entry>vigilance</entry> <pos>n.f.</pos>
    <translation>vigilance</translation>
</dict-entry>
<dict-entry> <entry>alerte</entry> <pos>a.</pos>
   <translation>alert</translation>
</dict-entry>
</volume>
```

FIG. 6.19 : Extrait du fichier `mini_oxfordFE.xml`

```
lexies
----+---------+------------------------------+----+-------+-----
uid |   mot   |         définition           |lang|ressrc |pos
----+---------+------------------------------+----+-------+-----
114 |convertir|faire changer de croyance ou  |fra |lar    |verbe
    |         |de religion                   |    |       |
115 |convertir|changer une chose en une autre|fra |lar    |verbe
120 |central  |qui est au centre             |fra |hdl    |adj
8505|convert  |change religious beliefs      |eng |wordnet|verbe
8518|central  |in or near a center           |eng |wordnet|adj

axies
--------+----------------+---------------
axie_uid|linkedlexies_uid|linkedaxies_uid
--------+----------------+---------------
1       |    120         |
        |    8518        |
2       |    114         |
        |    8505        |
3       |    115         |
        |    8505        |
```

FIG. 6.20 : Extrait de la base d'axies de l'exemple 7

```
lexies
----+---------+------------------------------+----+-------+-----
uid |   mot   |         définition           |lang|ressrc |pos
----+---------+------------------------------+----+-------+-----
114 |convertir|faire changer de croyance ou  |fra |lar    |verbe
    |         |de religion                   |    |       |
115 |convertir|changer une chose en une autre|fra |lar    |verbe
120 |central  |qui est au centre             |fra |hdl    |adj
8505|convert  |change religious beliefs      |eng |wordnet|verbe
8518|central  |in or near a center           |eng |wordnet|adj

axies
--------+----------------+---------------
axie_uid|linkedlexies_uid|linkedaxies_uid
--------+----------------+---------------
1       |    120         |
        |    8518        |
2       |    114         |
        |    8505        |
```

FIG. 6.21 : Extrait de la base d'axies de l'exemple 8

6.3.3.1 Processus

En ce qui concerne l'exécution par un processus, nous devons d'abord écrire un processus avec le langage de description de processus, ou écrire une stratégie d'évaluation avec le langage de description de stratégies d'évaluation de qualité (cf section 5.3.1 et 5.3.2). L'interpréteur pour un processus est appelé par la commande :

`c :\>java org.papillonDictionary.jeminie.process « un fichier »`

Par exemple, on peut créer un processus pour l'initialisation des bases monolingues qui regroupe les opérations dans les exemples 1, 2 et 3. Par convention, dans un processus, le nom de chaque module est `M-` suivi par le nom d'opération défini dans la section 6.3.2.

Par exemple, l'exécution d'un processus défini dans le fichier `process1.txt` se fait par la commande :

`c :\>java org.papillonDictionary.jeminie.process process1.txt`

`process1.txt` est un fichier contenant le processus suivant :

```
M-initBase(hibernate.properties,JeminieMapping.xml);
M-initLexieSimple(dictfr-a.xml,hibernate.properties,
JeminieMapping.xml,1,fra);
M-initLexieSimple(dicten-01.xml,hibernate.properties,
JeminieMapping.xml,1,eng).
```

Les opérations seront appelées et exécutées dans l'ordre : 1) `initBase`, 2) `initLexieSimple` pour le français, et 3) `initLexieSimple` pour l'anglais. Le résultat obtenu est constitué de deux bases de lexies pour le français et pour l'anglais.

Voici un autre exemple pour le processus de création d'axies, `process2.txt`.

```
M-BlingTransfertInverseAxieCreator(mini_oxfordFE.xml,
     sethaputra_ET.xml, sethaputra_TE.xml,
     mini_oxfordEF.xml,hibernate.properties,
     JeminieMapping.xml,fra,eng,tha,1);
M-CVAxieFilter(hibernate.propeties,JeminieMapping.xml,1,45).
```

6.3.3.2 Stratégie d'évaluation de qualité

L'interpréteur pour une stratégie d'évaluation de qualité est appelé par la commande :

`c :\>java org.papillonDictionary.jeminie.eval « un fichier »`

Pour les modules d'évaluation, en plus d'une liste des paramètres nécessités par chaque module, il faut aussi donner un paramètre qui se situe dans la première position dans la liste des paramètres, pour spécifier le poids de chaque module.

Par exemple, pour évaluer globalement le rappel et la précision, le fichier contenant une stratégie d'évaluation de qualité est le suivant :

```
Q-Rappel(1,referenceFE.txt,hibernate.properties, JeminieMapping.xml,1);
Q-Precision(1,referenceFE.txt,hibernate.properties, JeminieMapping.xml,1).
```

Ici, le `1` dans le premier paramètre de `Q-Rappel` et de `Q-Precision` est le poids que l'on donne à ces modules. Un extrait du fichier `referenceFE.txt` est donné dans la figure 6.22.

Nous verrons plus de détails des expérimentations et leur évaluation dans le chapitre 7.

```
2|cageot(N)|emballage léger, à claire-voie, pour le transport des fruits, des légumes, etc|crate(N)|a rugged box ; used for shipping
4|caillou(N)|pierre petite ou moyenne ; débris de roche.|stone(N)|a lump or mass of hard consolidated mineral matter
9|calcul(N)|concrétion pierreuse qui se forme dans les réservoirs glandulaires et les canaux excréteurs.|calculus(N)|a hard lump produced by the concretion of mineral salts ; found in hollow organs or ducts of the body
10|calcul(N)|traitant de la détermination de variables qui rendent maximale ou minimale une intégrale donnée|calculus(N)|the branch of mathematics that is concerned with limits and with the differentiation and integration of functions
11|calcul(N)|concrétion solide, d'origine minérale ou organique, se formant à l'intérieur d'un canal, d'un conduit excréteur ou d'un viscère creux|calculus(N)|a hard lump produced by the concretion of mineral salts ; found in hollow organs or ducts of the body
12|calendrier(N)|système de division du temps en périodes adaptées aux besoins de la vie sociale et concordant en général avec des phénomènes astronomiques.|calendar(N)|a system of timekeeping that defines the beginning and length and divisions of the year
17|camp(N)|espace de terrain où des troupes, des forces militaires, stationnent.|camp(N)|temporary living quarters specially built by the army for soldiers
18|camp(N)|lieu aménagé pour le stationnement ou l'instruction d'une formation militaire|camp(N)|temporary living quarters specially built by the army for soldiers
20|campagnard(N)|qui vit à la campagne.|countryman(N)|a man who lives in the country and has country ways
22|campement(N)|lieu équipé d'installations, d'abris provisoires |encampment(N)|a site where people on holiday can pitch a tent
23|canard(N)|oiseaux ansériformes caractérisés par un long cou, un bec plat et des pattes courtes aux doigts palmés|duck(N)|small wild or domesticated web-footed broad-billed swimming bird usually having a depressed body and short legs
```

FIG. 6.22 : Extrait du fichier `referenceFE.txt`

6.4 Conclusion

Ce chapitre a présenté Jeminie, un système d'aide à l'automatisation de la construction de bases lexicales interlingues, conçu selon les hypothèses présentées dans la section 5.1.

Comme nous avons choisi d'utiliser l'approche de combinaison pour la production d'axies, et comme les techniques utilisables dépendent des ressources disponibles, nous avons en effet besoin d'un système qui puisse s'adapter aux techniques choisies.

Pour cela, la conception et l'implémentation du système suivent le principe de séparation des préoccupations et les techniques de programmation par objets. En plus du choix des techniques de production d'axies et des critères d'évaluation, le système

définit aussi d'autres points de paramétrage, comme le choix de l'analyseur morpho-syntaxique et le format de dictionnaire monolingue. Les interpréteurs de processus et de stratégie d'évaluation de qualité sont encore en cours de développement au moment de la rédaction de ces lignes.

Chapitre 7

Expérimentation de la structuration et de l'évaluation

L'objectif de ce chapitre est de montrer des expérimentations sur la structuration des axies bilingues et multilingues, en utilisant des méthodes différentes, et d'évaluer leurs résultats.

La section 7.1 décrit les ressources utilisées dans nos expérimentations. La section 7.2 présente l'expérimentation de création des axies bilingues avec des méthodes simples. La section 7.3 présente une expérimentation de création des axies multilingues avec différentes méthodes.

7.1 Préparation des données

Des axies seront créées à partir de ressources existantes. La première étape consiste donc à les préparer. La plupart des ressources utilisées sont récupérées via Internet et ont été transformées vers nos formats avec des programmes ad hoc. La récupération et la normalisation est différente pour chaque type de ressource. Cette section décrit en détail les ressources monolingues, les ressources bilingues, et les ressource diverses que nous avons utilisées.

7.1.1 Ressources monolingues

Dans ces expérimentations, nous utilisons des données monolingues dans trois langues : l'anglais, le français, et le thaï. Ces données ont été récupérées et normalisées.

L'anglais Les données monolingues anglaises sont récupérées à partir des données de WordNet[Pri] 1.7. Chaque entrée de WordNet est transformée en une lexie comprenant le mot-vedette, la partie du discours, et sa description. Le nombre de lexies par mot-vedette est en moyenne de 1,7.

Le français les données monolingues françaises sont téléchargées depuis un site web[1] qui regroupe plusieurs dictionnaires monolingues français, des dictionnaires de synonymes français, etc. Chaque lexie a été créée à partir d'une définition de chaque mot-vedette. Si un mot-vedette comprend plusieurs définitions, alors plusieurs lexies sont créées, une pour chaque définition. Le nombre de lexies par mot-vedette est en moyenne de 2.

Le thaï les données monolingues thaïs sont récupérées via le site web du Royal Institute de Thaïlande[2]. Comme pour le français, chaque définition d'un mot-vedette est transformée en lexie. Le nombre de lexies par mot-vedette est en moyenne de 1,7.

Les données monolingues de chaque langue sont transformées vers le format XML de Papillon avec un programme ad hoc. Le tableau 7.1 indique les volumes des données récupérées dans chaque langue.

Langue	Nombre de mots	Nombre de lexies
anglais	53 000	91 270
français	21 700	46 000
thaï	5 440	9 360

TAB. 7.1 : Taille des données récupérées et normalisées

7.1.2 Ressources bilingues

Les ressources bilingues sont des dictionnaires disponibles localement. L'étape de préparation consiste seulement à transformer les données dans notre format. Nous avons le dictionnaire *FeM* pour français-anglais, le dictionnaire *Oxford french mini-dictionary* pour français-anglais et anglais-français, et le dictionnaire *So Sethaputra* pour anglais-thaï. Par contre, le dictionnaire français-thaï est récupéré via le site web de l'université Naresuan en Thaïlande[3]. Le tableau 7.2 indique les volumes des données de chaque dictionnaire bilingue que nous avons utilisé.

7.1.3 Ressources diverses

Nous avons expérimenté la méthode de comparaison de vecteurs conceptuels (cf. section 7.2.2). Cette méthode nécessite une base lexicale pour calculer les vecteurs initiaux des lexies. Chaque entrée de cette base regroupe un mot-vedette, une partie du discours, et un ensemble de concepts feuilles dans une hiérarchie d'un thésaurus correspondant.

[1] `http://www.lirmm.fr/~lafourcade/SERVICE/semvec-service.html`
[2] `http://www.royin.go.th/`
[3] `http://dauphine.nu.ac.th/franco-thai/tafrth/dictionnaire_francais_thai.html`

Dictionnaire	Nombre de mots d'entrées
Oxford french minidictionary EN-FR	8 122
Oxford french minidictionary FR-EN	7 155
FeM	14 690
So Sethaputra EN-TH	9 900
FR-TH	18 000

Tab. 7.2 : Tailles des dictionnaires bilingues utilisés

Pour le français, une telle base lexicale est construite par l'équipe TAL du LIRMM, à Montpellier, et contient environ 49 000 mots[4]. Pour l'anglais, nous avons indexé manuellement environ 4 200 mots, et environ 2 000 mots pour le thaï.

Toutes les bases de chaque langue sont indexées avec les mêmes concepts feuilles de la hiérarchie du thésaurus Larousse. Ainsi les vecteurs ont les mêmes dimensions et les mesures de similarité peuvent être calculées entre vecteurs issus de langues différentes.

7.2 Structuration des axies bilingues

Cette section présente les expérimentations que nous avons menées pour créer une base d'axies bilingues entre le français et l'anglais. Les expérimentations sur une base d'axies multilingues sont présentées dans la section suivante (section 7.3). Toutes les expérimentations utilisent les mêmes bases lexicales (monolingues, bilingues, etc.), indiquées dans la section 7.1, sauf mention contraire.

En ce qui concerne l'évaluation, les mêmes mesures sont appliquées à chaque processus. Nous faisons une première évaluation de la qualité des méthodes en évaluant `Rappel`, `Precision`, et $lexie_{correcte-complete}$.

Pour ces trois mesures, plus la valeur mesurée est élevée, meilleure est la base d'axies. Pour la base d'axies entre le français et l'anglais, le calcul de `Rappel` et de `Precision` utilise une base d'axies de référence d'environ 3 300 axies (appelée `BDRef` dans la suite). Cette base a été créée manuellement par notre groupe de linguistes à partir des bases de lexies française et anglaise, avec l'aide du dictionnaire bilingue Oxford french minidictionary français-anglais. Faute de temps, nous n'avons pas pu créer une base de référence plus volumineuse.

7.2.1 Méthode bilingue simple

Cette méthode est appelée par la commande :

```
c :\>java org.papillonDictionary.jeminie.Main_BlingAxieCreator \
mini_oxfordFE.xml \
hibernate.propeties \
```

[4]Les données nous ont été prêtées pour l'expérimentation par D. Schwab de l'équipe TAL du LIRMM.

```
JeminieMapping.xml \
fra eng 1
/* 1 : id de la base d'axies */
```

Le dictionnaire bilingue `mini_oxfordFE.xml` comprend environ 10 760 traductions mot à mot entre le français et l'anglais. En utilisant cette méthode, nous avons créé une base de lexies françaises et une base de lexies anglaises. Avec cette méthode, tous les liens possibles sont créés. Le nombre maximum de liens (N_{total}) créés peut être calculé par la formule :

$$N_{total} = \sum_{i=0}^{n} (X_i * Y_i)$$

où

n est le nombre des traductions mot-à-mot dans le dictionnaire bilingue.

X_i est le nombre de lexies d'un mot source dans le couple de traduction i.

Y_i est le nombre de lexies d'un mot cible dans le couple de traduction i.

L'avantage de cette méthode est qu'elle est très simple à réaliser et qu'il est certain d'obtenir tous les liens possibles (le rappel est élevé). Par contre, elle crée beaucoup d'axies incorrectes (la précision est faible).

Une base d'axies entre le français et l'anglais a été créée à l'aide du dictionnaire Oxford french minidictionary français-anglais. La base d'axies obtenue compte environ 62 300 entrées. Nous avons évalué le résultat en le comparant à notre base d'axies de référence `BDRef`. En comparaison des mots-clés, il reste environ 6 820 entrées à évaluer. Le `Rappel` est 73% et la `Precision` est 15%.

Puisque la méthode crée tous les liens possibles, toutes les lexies françaises et anglaises ont plus d'un lien vers des axies. En conséquence, les valeurs de `Q-`$lexies_{correcte-complete}$ pour le français et de `Q-`$lexies_{correcte-complete}$ pour l'anglais sont très faibles (1% à 2% seulement).

Mesure	Résultat
Q-Rappel	73%
Q-Precision	15%

7.2.2 Méthode de comparaison de vecteurs conceptuels

Cette méthode est appelée par la commande :

```
c :\>java org.papillonDictionary.jeminie.Main_CVAxieCreator \
mini_oxfordFE.xml \
hibernate.propeties \
JeminieMapping.xml \
fra eng 1 45
/* 1 : id de la base d'axies */
/* 45 : seuil (45°) */
```

Dans un premier temps, nous fixons le seuil de distance angulaire maximale à $\frac{\pi}{4}$, car d'après [Laf02], deux vecteurs sont thématiquement proches si leur distance

angulaire est inférieure à $\frac{\pi}{4}$. Nous avons donc créé des axies en reliant à une même axie chaque couple de lexies qui ont la distance angulaire la plus petite parmi les lexies candidates, et sont à une distance inférieure à $\frac{\pi}{4}$.

En utilisant $\frac{\pi}{4}$ comme seuil, peu d'axies sont créées : en effet, 510 axies ont été créées à partir d'environ 10 760 traductions mot à mot. Une des raisons pour lesquelles peu d'axies sont produites est aussi le manque des lexies pour certains mots dans le dictionnaire bilingue. L'évaluation de cette base d'axies par un linguiste, donne un taux d'axies correctes d'environ 66%.

Afin d'obtenir plus d'axies, une autre procédure pour décider si deux lexies peuvent être liées à une même axie est de se limiter à choisir le couple de lexies dont la distance angulaire est la plus petite (en tout cas, moins de $\frac{\pi}{2}$) par rapport à d'autres couples candidats. Avec cette nouvelle procédure, 5 260 axies sont créées à partir des mêmes ressources.

Nous avons de nouveau évalué le résultat avec notre base d'axies de référence de 3 300 entrées. La proportion d'axies correctes est environ 44%. En évaluant la structure, parmi les axies créées, 79% des lexies françaises et 93% des lexies anglaises sont des $lexies_{correcte-complete}$.

Pour cette procédure, nous avons aussi fait une mesure vectorielle (cf. page 74), en mesurant la distance angulaire moyenne α entre chaque paire de lexies liées à une même axie (notée `Q-vcmoyen`). Plus cette distance est petite, meilleure est la sémantique. Pour obtenir une valeur d'autant plus grande que la qualité est meilleure, on peut considérer $\beta = \frac{\pi}{2} - \alpha$.

Le tableau 7.3 indique le résultat de l'évaluation de la base d'axies obtenue avec ces différentes mesures. Vu que les résultats ne sont pas très satisfaisants (le `rappel` et la `precision` sont faibles), l'amélioration contributive est donc très importante.

Mesure	Résultat
`Q-Rappel`	28%
`Q-Precision`	44%
`Q-`$lexies_{correcte-complete}$ pour le français	79%
`Q-`$lexies_{correcte-complete}$ pour l'anglais	93%
`Q-vcmoyen`	62°

TAB. 7.3 : Résultat d'évaluation d'une base d'axies

Les figures 7.1 et 7.2 montrent quelques exemples des axies produites. Dans ces exemples, une axie est extraite en forme texte :

```
distance : distance entre les deux lexies qu'elle relie (en degrés)
mot-vedette-français (partie du discours) : définition
mot-vedette-anglais (partie du discours) : définition
```

Parmi les axies incorrectes, certaines sont dues à un manque de lexies pour le sens approprié, soit dans la langue source, soit dans la langue cible. C'est le cas par exemple de la première axie dans la figure 7.2.

```
distance : 83
course (n) : faire des achats dans un magasin
racing (n) : the sport of engaging in contests of speed
```

Dans notre base de lexies monolingues du français, il en manque une pour le mot `course` représentant « épreuve de vitesse ». Il est donc impossible de relier cette lexie du mot `racing` avec une lexie du mot `course` en français.

En observant les axies correctes, nous voyons que la distance angulaire entre la plupart des couples de lexies est assez grande (parfois jusqu'à 75°). Les vecteurs conceptuels que nous avons calculés ne sont pas très satisfaisants, car ils sont souvent creux, pour deux raisons :

1. La définition de certaines lexies comprend très peu de mots, par exemple,

   ```
   mousser (V) : faire de la mousse.
   modifier (V) : changer.
   organisateur (N) : qui organise.
   rose (N) : la couleur rose.
   ```

   Si les mots dans la définition ne se trouvent ni dans la base de vecteurs initiaux, ni dans la base monolingue, alors les vecteurs de ces lexies sont creux (nombreuses valeurs à zéro), ce qui fait que la comparaison des vecteurs ne donne pas de résultat satisfaisant.

2. La base initiale pour créer des vecteurs initiaux n'est pas suffisamment grande pour l'anglais. Par conséquent, la plupart des vecteurs sont creux pour les lexies anglaises. Il faut noter que notre base lexicale pour créer des vecteurs initiaux a été réalisée à la main, ce qui prend beaucoup de temps.

7.3 Structuration des axies multilingues

Dans notre expérimentation pour le multilingue, nous appliquons nos algorithmes à trois langues : le français, l'anglais et le thaï. Dans cette expérimentation, nous réduisons la taille des dictionnaires bilingues pour obtenir une base d'axies plus petite et par là plus facile à évaluer. Nous avons choisi environ 50 mots français ayant pour la plupart plusieurs traductions possibles. Puis nous avons choisi des mots anglais et des mots thaï qui ont des liens traductionnels avec ces mots français.

7.3.1 Méthode de transfert bilingue

Cette méthode est appelée par la commande :

```
c :\>java org.papillonDictionary.jeminie.Main_BlingTransfertAxieCreator \
mini_oxfordFE.xml \
sethaputra_ET.xml \
hibernate.propeties \
JeminieMapping.xml \
fra eng tha 1
/* 1 : id de la base d'axies */
```

```
distance : 62
orteil (N) : doigt de pied
toe (N) : one of the digits of the foot

distance : 60
fumeur (N) : personne qui a l'habitude de fumer du tabac
smoker (N) : a person who smokes tobacco

distance : 30
rose (N) : la couleur rose
rose (N) : a dusty pink color

distance : 59
datte (N) : fruit du dattier
date (N) : sweet edible fruit of the date palm with a single long woody
seed

distance : 68
profil (N) : contour d'un visage vu de côté
profile (N) : a side view representation of an object

distance : 60
dos (N) : partie arrière du corps de l'homme, comprise entre la nuque et
les reins
back (N) : the posterior part of a human body from the neck to the end of
the spine

distance : 10
dividende (N) : le nombre divisé
dividend (N) : a number to be divided by another number

distance : 37
rapport (N) : relation, lien qui existe entre des choses
connection (N) : a relation between things or events

distance : 54
tenir (V) : rester à la même place, dans la même position, sans se
détacher, sans tomber
hold (V) : remain in a certain state

distance : 38
convertir (V) : faire changer de croyance ou de religion
convert (V) : change religious beliefs

distance : 29
convertir (V) : changer une chose en une autre
convert (V) : change the nature
```

Fig. 7.1 : Exemple d'axies correctes

```
distance : 83
course (n) : faire des achats dans un magasin
racing (n) : the sport of engaging in contests of speed

distance : 59
noir (n) : la couleur la plus sombre dont la surface ne réfléchit aucune
radiation visible.
dark (n) : the time after sunset and before sunrise while it is dark
outside

distance : 77
poste (n) : service d'acheminement et de distribution du courrier
post (n) : the position where someone stands or is assigned to stand

distance : 65
monter (v) : rendre utilisable en assemblant les différentes parties
climb (v) : increase in value or to a higher point

distance : 87
pouce (n) : le plus court et le plus puissant des doigts de la main,
opposable aux autres
inch (n) : a unit of measurement for advertising space

distance : 70
assurer(V) : passer un contrat d'assurance
ensure(V) : be careful or certain to do something; make certain of
something

distance : 52
cuisinier(N) : fourneau servant à cuire les aliments
cook(N) : someone who cooks food

distance : 60
milieu(N) : partie d'une chose qui se situe à égale distance de ses
extrémités
environment(N) : the totality of surrounding conditions

distance : 74
réclamation(N) : bureau où on peut réclamer
claim(N) : demand for something as rightful or due

distance : 70
froid(N) : indifférence, manque de sympathie
cold(N) : the absence of heat
```

FIG. 7.2 : Exemple d'axies non correctes

Une base d'axies entre le français et l'anglais, et une base d'axies entre l'anglais et le thaï, sont créées. Ensuite, les axies qui sont reliées à la même lexie anglaises sont regroupées.

Par exemple, voici une axie dans la base d'axies français-anglais :

```
fermer|V|1 : appliquer sur une ouverture la partie mobile destinée à la boucher
shut|V|1 : move so that an opening or passage is obstructed
```

et voici la même axie dans la base d'axies anglais-thaï :

```
shut|V|1 : move so that an opening or passage is obstructed
ปิด |V|1 : กั้น ไว้ ไม่ ให้ เผย ออก หรือ ไม่ ให้ ผ่าน ไป ได้
```

Nous avons la lexie

```
shut|V|1 : move so that an opening or passage is obstructed
```

en commun entre ces deux axies, alors comme résultat, nous avons une axie fusionnant les deux précédentes :

```
fermer|V|1 : appliquer sur une ouverture la partie mobile destinée à la boucher
shut |V|1 : move so that an opening or passage is obstructed
ปิด |V|1 : กั้น ไว้ ไม่ ให้ เผย ออก หรือ ไม่ ให้ ผ่าน ไป ได้
```

L'évaluation du résultat avec la base de référence donne une précision d'environ 20 %. Les résultats de cette méthode ont une précision assez faible, car elle crée tous les liens possibles. C'est une méthode simple qui permet de créer rapidement une base initiale avec plusieurs langues, mais elle nécessite une correction ultérieure soit manuelle, soit en utilisant une autre méthode. La figure 7.3 donne un exemple de résultat produit par cette méthode.

7.3.2 Méthode de transfert et consultation inverse de dictionnaires bilingues

La commande pour exécution de cette méthode est

```
c :\>java org.papillonDictionary.jeminie.Main_BlingTransfertInverseAxieCreator \
mini_oxfordFE.xml sethaputra_ET.xml \
sethaputra_TE.xml mini_oxfordEF.xml \
hibernate.propeties \
JeminieMapping.xml \
fra eng tha 1
/* 1 : id de la base d'axies */
```

La méthode de transfert et consultation inverse de dictionnaires bilingues [TU94] utilise une langue intermédiaire pour relier deux langues qui n'ont pas de traduction directe avec un dictionnaire bilingue.

En conservant la langue intermédiaire, nous obtenons une base d'axies de trois langues. En évaluant, le résultat de cette méthode avec une base de référence, on trouve un taux de précision de 35%.

La figure 7.4 donne un exemple des axies créées par ce processus.

Avec la consultation inverse, certaines lexies dont des mots-vedettes inappropriés sont enlevées par la consultation inverse.

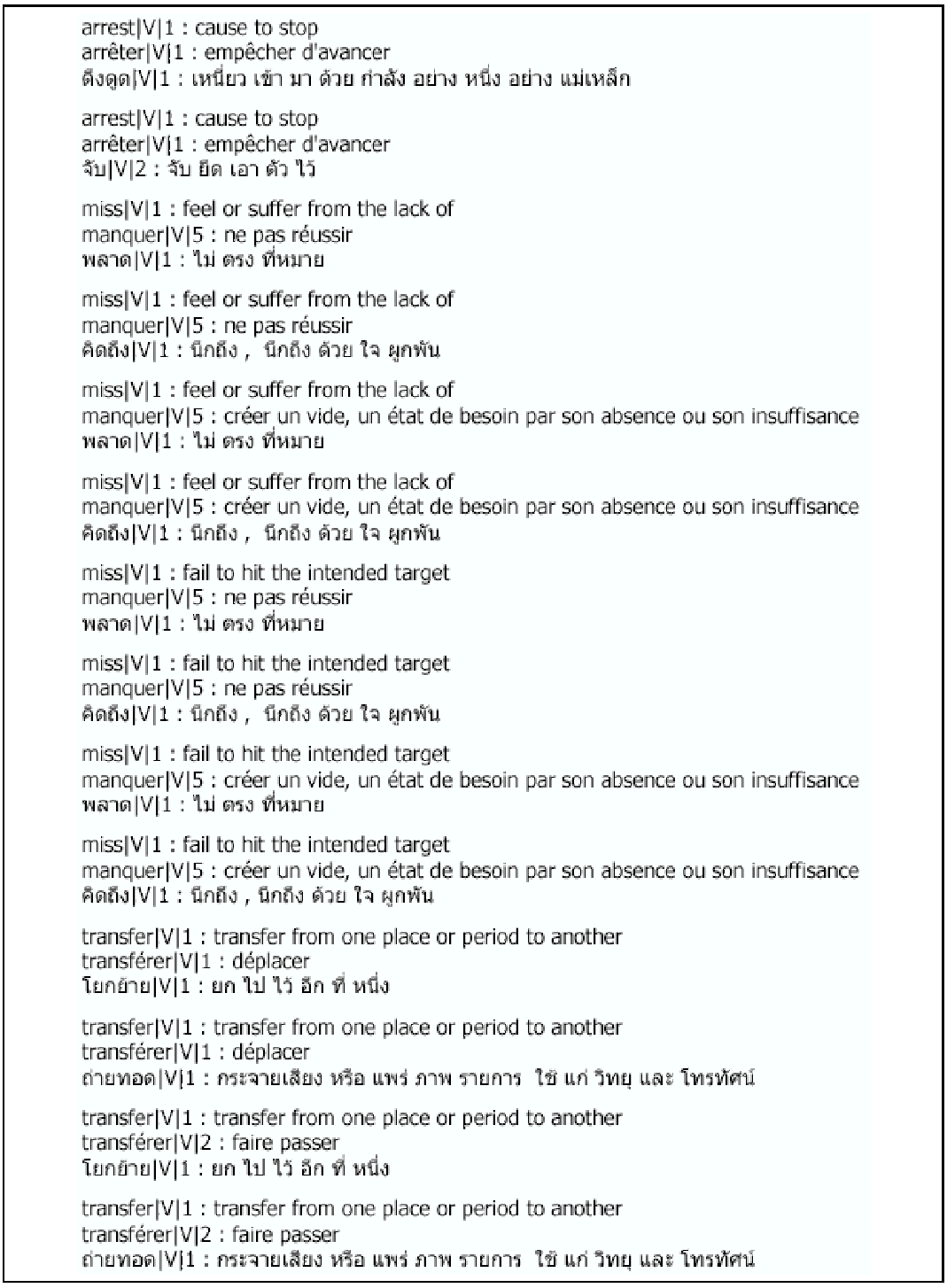

FIG. 7.3 : Exemple de résultat de la méthode de transfert bilingue

Par exemple, par rapport à l'exemple de la méthode 7.3.1 (figure 7.3), la première et la dernière axie sont enlevées. En effet, d'après cette méthode, le mot ดึงดูด n'est pas groupable avec les mots `arrest` et `arrêter`, et le mot ถ่ายทอด n'est pas groupable avec les mots `transfer` et `transférer`. Alors les axies reliées aux lexies de ces deux mots sont enlevées.

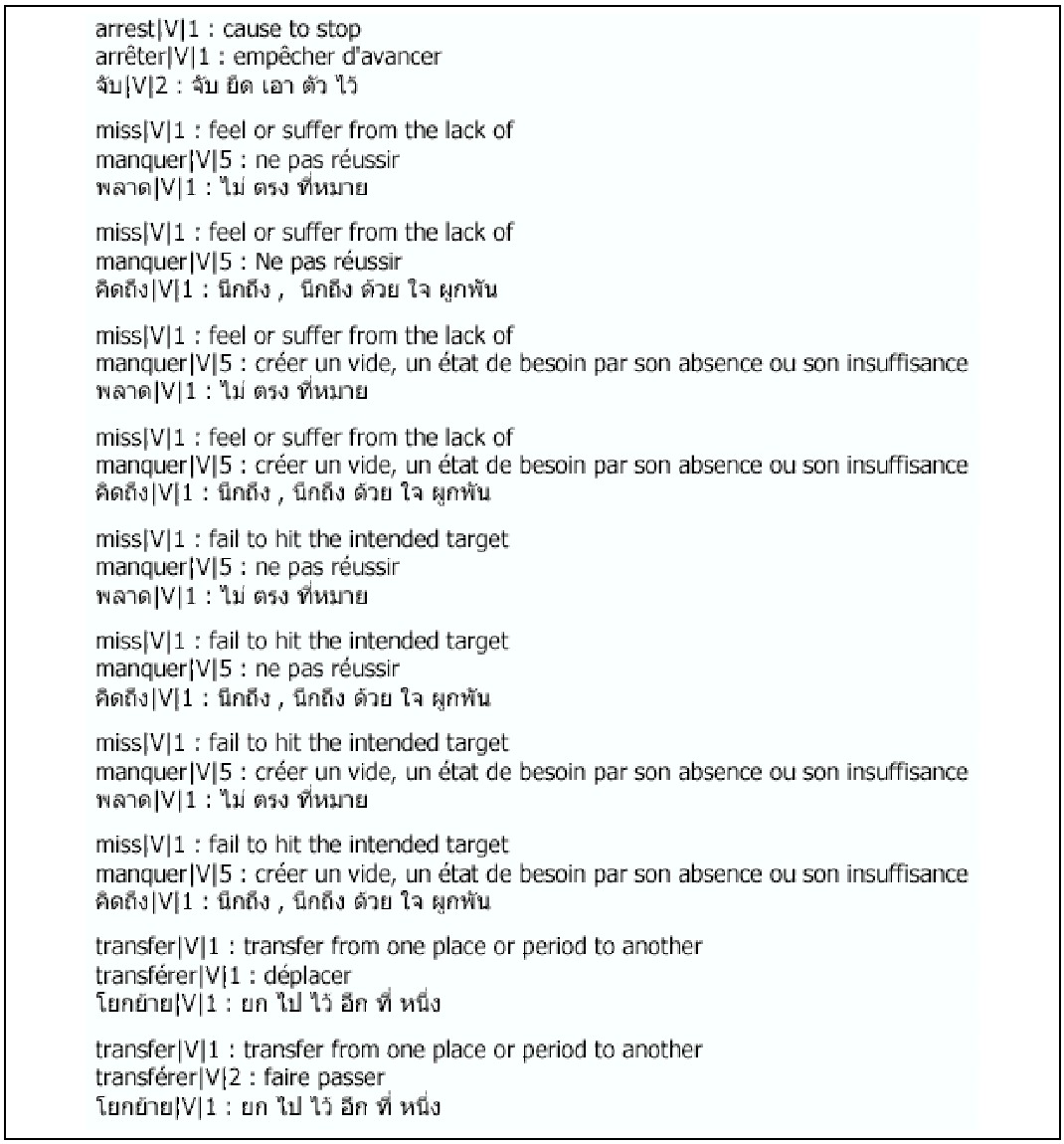

Fig. 7.4 : Exemple de résultat de la méthode de transfert et consultation inverse de dictionnaires bilingues

7.3.3 Composition d'algorithmes 1

Nous exécutons la commande :

```
c :\>java org.papillonDictionary.jeminie.Main_BlingTransfertAxieCreator \
mini_oxfordFE.xml \
sethaputra_ET.xml \
hibernate.propeties \
JeminieMapping.xml \
fra eng tha 1
/* 1 : id de la base d'axies */
```

puis la commande :

```
c :\>java org.papillonDictionary.jeminie.Main_CVAxieFilter \
hibernate.propeties \
JeminieMapping.xml \
1 45
/* 1 : id de la base d'axies */
/* 45 : seuil (45°) */
```

Avec ce processus, la technique de comparaison de vecteurs conceptuels est appliquée à la base d'axies produite par la technique de transfert bilingue (la base obtenue dans la section 7.3.1).

Avec plus de deux langues, l'algorithme de comparaison des vecteurs conceptuels calcule la moyenne des distances angulaires de chaque paire de lexies dans chaque axie. Ensuite, toutes les axies liées aux lexies ayant les mêmes mots-vedettes et parties du discours sont comparées entre elles pour trouver l'axie qui a la moyenne des distances angulaires la plus petite. Seule cette axie est conservée et les autres sont enlevées. La figure 7.5 donne un extrait des axies créées par ce processus.

arrest|V|1 : cause to stop
arrêter|V|1 : empêcher d'avancer
ดึงดูด|V|1 : เหนี่ยว เข้า มา ด้วย กำลัง อย่าง หนึ่ง อย่าง แม่เหล็ก

arrest|V|1 : cause to stop
arrêter|V|1 : empêcher d'avancer
จับ|V|2 : จับ ยึด เอา ตัว ไว้

miss|V|1 : feel or suffer from the lack of
manquer|V|5 : créer un vide, un état de besoin par son absence ou son insuffisance
คิดถึง|V|1 : นึกถึง , นึกถึง ด้วย ใจ ผูกพัน

miss|V|1 : feel or suffer from the lack of
manquer|V|5 : créer un vide, un état de besoin par son absence ou son insuffisance
พลาด|V|1 : ไม่ ตรง ที่หมาย

transfer|V|1 : transfer from one place or period to another
transférer|V|1 : déplacer
โยกย้าย|V|1 : ยก ไป ไว้ อีก ที่ หนึ่ง

transfer|V|1 : transfer from one place or period to another
transférer|V|1 : déplacer
ถ่ายทอด|V|1 : กระจายเสียง หรือ แพร่ ภาพ รายการ ใช้ แก่ วิทยุ และ โทรทัศน์

FIG. 7.5 : Exemple de résultat de la composition d'algorithmes 1

Le résultat de ce processus donne un taux de 37% d'axies correctes, en comparant avec notre base de référence. Le tableau 7.4 donne le résultat de l'évaluation.

Mesure	Résultat
taux de précision	37%
taux des $lexies_{correcte-complete}$ pour le français	50%
taux des $lexies_{correcte-complete}$ pour l'anglais	55%
taux des $lexies_{correcte-complete}$ pour le thaï	65%

Tab. 7.4 : Evaluations du processus de composition d'algorithmes 1

7.3.4 Composition d'algorithmes 2

La commande est

```
c :\>java org.papillonDictionary.jeminie.Main_BlingTransfertInverseAxieCreator \
mini_oxfordFE.xml sethaputra_ET.xml \
sethaputra_TE.xml mini_oxfordEF.xml \
hibernate.propeties \
JeminieMapping.xml \
fra eng tha 1
/* 1 : id de la base d'axies */
```

puis

```
c :\>java org.papillonDictionary.jeminie.Main_CVAxieFilter \
hibernate.propeties \
JeminieMapping.xml \
1 45
/* 1 : id de la base d'axies */
/* 45 : seuil (45°) */
```

Ce processus prend le résultat de la section 7.3.2 et applique la technique de comparaison des vecteurs conceptuels. Comme décrit dans la section 7.3.3, la technique de comparaison des vecteurs conceptuels filtre cette base d'axies de trois langues en conservant seulement les axies ayant la distance angulaire la plus petite parmi les axies candidates. La figure 7.6 donne un exemple d'axies créées par ce processus

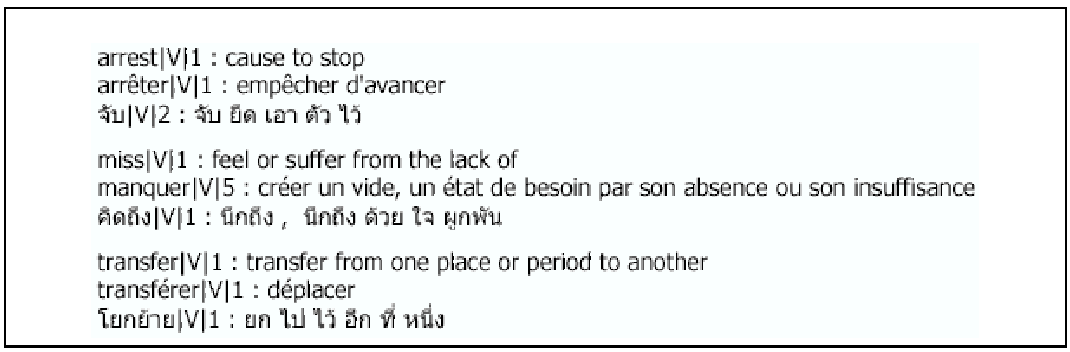

Fig. 7.6 : Exemple de résultat de la composition d'algorithmes 2

En évaluant le résultat de ce processus par comparaison avec notre base de référence, on trouve 57% d'axies correctes. Le tableau 7.5 donne le résultat d'autres évaluations.

Mesure	Résultat
Q-Precision	57%
Q-$lexies_{correcte-complete}$ pour le français	75%
Q-$lexies_{correcte-complete}$ pour l'anglais	75%
Q-$lexies_{correcte-complete}$ pour le thaï	92%

TAB. 7.5 : Evaluations du processus de composition d'algorithmes 2

7.3.5 Discussion

L'expérimentation sur une base d'axies multilingues a concerné seulement trois langues, car nous avons eu de la difficulté à trouver les ressources monolingues (lexies) dans d'autres langues. De plus, pour appliquer la méthode de comparaison de vecteurs conceptuels, il faut une base initiale pour créer un ensemble de vecteurs initiaux. Par manque de linguistes, nous ne pouvions pas non plus indexer pour d'autres langues. Pour ces deux raisons, nous nous limitons donc à trois langues.

Le tableau 7.6 résume les résultats d'évaluation de chaque algorithme que nous avons expérimenté. Dans ce tableau, le taux d'axies correctes pour le résultat de la méthode bilingue simple (ou transfert bilingue) n'est pas très différent entre une base d'axies bilingue et trilingue. Les valeurs de Q-$lexies_{correcte-complete}$ pour les méthodes mA et mB sont faibles, parce que de nombreuses lexies sont chacune reliées à plusieurs axies. Par contre, le filtrage des axies incorrectes par les méthodes mC et mD diminue le nombre d'axies auxquelles chaque lexie est reliée, ce qui se traduit par une augmentation des valeurs de Q-$lexies_{correcte-complete}$.

En outre, nous trouvons que, pour trois langues, la méthode d'inversion bilingue a éliminé plus de liens incorrects (le taux de précision passe de 20% à 35%). On obtient un résultat meilleur avec trois langues qu'avec deux langues. En combinant la méthode de transfert et consultation inverse avec la méthode de comparaison de vecteurs conceptuels, on obtient un résultat meilleur qu'avec chaque méthode seule.

7.4 Conclusion

Nos expérimentations montrent que la composition d'algorithmes permet d'améliorer la qualité des résultats. Du point de vue de la programmation, ces expérimentations permettent de vérifier que le programme Jeminie supporte l'exécution de méthodes composées. Du point de vue technique, la consultation inverse des dictionnaires bilingues et la comparaison de vecteurs conceptuels augmentent les pourcentages d'axies

Résultat	mA	mB	mC	mD
Q-Precision	20%	35%	37%	57%
Q-$lexies_{correcte-complete}$ pour le français	0%	0%	50%	75%
Q-$lexies_{correcte-complete}$ pour l'anglais	0%	0%	55%	75%
Q-$lexies_{correcte-complete}$ pour le thaï	0%	0%	65%	92%

mA : méthode de transfert bilingue
mB : méthode de création par transfert et consultation inverse de dictionnaires bilingues
mC : méthode mA + filtrage de comparaison de vecteurs conceptuels
mD : méthode mB + filtrage de comparaison de vecteurs conceptuels

TAB. 7.6 : Résultat d'évaluation d'une base d'axies multilingues

correctes en éliminant les axies incorrectes. La combinaison de ces deux méthodes donne un résultat meilleur qu'en utilisant chaque méthode seule.

En ce qui concerne l'évaluation, nous avons en particulier une base de référence pour évaluer les résultats obtenus par différentes méthodes. Le résultat d'évaluation donne un indice de l'efficacité de ces méthodes. L'évaluation par le critère structural n'est pas pertinente pour évaluer des bases d'axies obtenues par des méthodes de traduction au niveau des mots, car ces méthodes créent une axie pour chaque lexie de langue source avec toutes les lexies de langue cible qui en sont la traduction. Dans ce cas, toutes les lexies ont plusieurs liens vers des axies (et sont donc mal formées), sauf si le mot source et les mots cible sont tous monosémiques (ne sont reliés qu'à une lexie). Bien que le critère structural ne puisse pas évaluer la sémantique de la base d'axies, en appliquant ce critère à une base d'axies obtenue par la méthode de comparaison de vecteurs, nous pouvons indiquer quelles sont les lexies ou les axies mal structurées.

Conclusion

Synthèse Cette thèse aborde la problématique de la structuration d'un ensemble de dictionnaires bilingues ou multicible liant des vocables (rarement des lexies) à d'autres vocables en une BDLM reliant des lexies (monolingues) grâce à des axies (liens interlingues). La première partie a présenté les problèmes cruciaux en bases lexicales multilingues. Nous avons proposé une catégorisation globale des bases dictionnairiques et dégagé les caractéristiques « idéales » d'une base lexicale multilingue. Pour les problèmes de structuration en lexies et en axies, nous avons choisi de présenter l'effort du projet Papillon et les difficultés de construction des données dictionnairiques. Ce problème a trois volets :

- création d'un graphe de lexies et axies initiales à partir de données existantes.
- amélioration incrémentale des données initiales.
- évaluation des données produites.

Pour le premier volet, le travail réalisé au cours de cette thèse a permis de définir et de développer le système Jeminie d'aide à la structuration des bases lexicales multilingues en lexies et en axies. La structuration en lexies se fait en extrayant les informations à partir de dictionnaires monolingues de définitions. La structuration en axies se fait en combinant des techniques existantes. Jeminie permet aux utilisateurs de choisir la combinaison de techniques qui convient le mieux aux ressources disponibles. L'avantage est de pouvoir construire une base d'axies dans différentes situations, quelles que soient les ressources disponibles. De plus, le système Jeminie est ouvert à l'ajout de nouveaux algorithmes en implémentant de nouveaux modules.

D'après les expérimentations réalisées, nous pouvons affirmer que la combinaison de plusieurs techniques pour la structuration en axies augmente la précision des résultats. Par contre, dans nos expérimentations sur différents processus, avec une méthode de filtrage par vecteurs conceptuels, la structure de la base d'axies produite est correcte à plus de 75% (jusqu'à 93%), mais la précision obtenue est assez faible (jusqu'à 57% seulement) : il sera donc toujours nécessaire d'améliorer manuellement les bases lexicales produites.

Le deuxième volet est un travail manuel fait par des contributeurs et/ou des lexicographes, et on ne peut pas le traiter automatiquement. L'idée est d'offrir aux contributeurs un jeu de schémas et d'actions correspondantes de transformation locale de la BDLM, ces actions pouvant d'ailleurs être des actions élémentaires utilisées par les algorithmes globaux.

Pour le troisième volet, l'une de nos principales contributions est de proposer une catégorisation de critères d'évaluation de qualité des BDLM produites par machine. Nous avons également défini des mesures pour ces critères d'évaluation. L'approche choisie est la même que celle utilisée pour la structuration, c'est-à-dire la combinaison des différents critères pour évaluer une base lexicale. Ainsi, pour évaluer une base lexicale, soit on applique une seule mesure, soit on applique plusieurs mesures et on calcule la somme pondérée de chaque valeur mesurée.

Les trois idées-clés de la plate-forme Jeminie sont 1) séparation des préoccupations, 2) composition des techniques, et 3) similarité entre la structuration et l'évaluation. Pour la séparation des préoccupations, les tâches peuvent être séparées en deux groupes de personnes. La programmation des modules de structuration (création, filtrage) et d'évaluation est réalisée par le programmeur. La spécification des stratégies (processus) de structuration et des stratégies d'évaluation est réalisée par le lexicologue. Celui-ci décide des stratégies de structuration à exécuter (qu'il peut décrire par un langage spécialisé), en fonction des résultats d'évaluation.

Recherches futures Compte tenu des limites des ressources monolingues disponibles, nous n'avons pas pu expérimenter sur un grand ensemble de langues, mais seulement sur trois langues. Il nous semble intéressant, dans l'avenir, d'utiliser plus de langues pour observer les résultats de la combinaison.

En ce qui concerne l'implantation du système, la partie concernant les deux langages de processus n'est pas encore achevée. À court terme, il faut terminer l'implantation de cette partie. De plus, il faut améliorer les performances du système telles que la rapidité, la consommation d'espace disque, etc. Dans nos expérimentations, l'exécution de la partie du système concernant le calcul de vecteurs conceptuels est assez lente (environ 16h pour une itération du calcul pour une base de lexies d'environ 46 000 entrées). Ce problème, essentiellement technique, pourra être résolu rapidement.

Du point de vue de l'évaluation de la qualité d'une base lexicale, il est souhaitable d'ajouter de nouvelles techniques. Nous pensons notamment à celles qui fonctionnent au niveau des lexies, car nous n'avons à présent, à ce niveau, qu'une méthode utilisant des vecteurs conceptuels.

Il serait intéressant d'intégrer au système Jeminie une interface pour manipuler des données (lexies et axies) pour l'amélioration manuelle par des contributeurs ou lexicographes. Par exemple, après avoir évalué la base avec le critère structural, il serait intéressant que le système indique les endroits où se trouvent les axies mal structurées et propose aux contributeurs ou aux lexicographes une liste d'opérations de base pour corriger ces données, telles que le groupage ou le raffinement d'axies. Il serait également souhaitable d'avoir une interface simple pour visualiser les graphes des lexies et axies créées.

À long terme, une autre piste de recherche intéressante est de faire une étude sur les techniques de décision multicritères et de proposer des méthodes de décision assistant le lexicographe dans le choix des techniques de construction et d'évaluation.

Bibliographie

[ABI+03] Jurij Derenikovich Apresjan, Igor M. Boguslavsky, Leonid L. Iomdin, Alexander V. Lazursky, Valadimir Z. Sannikov, Victor G. Sizov, et Leonid L. Tsinman. ETAP-3 Linguistic Processor : a Full-Fledged NLP Implementation of the Meaning-Text Theory. Dans *Proceedings of the First International Conference on Meaning-Text Theory (MTT-2003)*, pages 279–288, Paris, France, 2003.

[ABIT03] Jurij Derenikovich Apresjan, Igor M. Boguslavsky, Leonid L. Iomdin, et Leonid L. Tsinman. Lexical Functions as a Tool of ETAP-3. Dans *Proceedings of the First International Conference on Meaning-Text Theory (MTT-2003)*, Paris, France, 2003. 17 p.

[ALJS99] Elisabeth Aimelet, Veronika Lux, Corinne Jean, et Frédérique Segond. WSD evaluation and the looking-glass. Dans *Proceedings of 6ème conférence sur le Traitement Automatique des Langues Naturelles (TALN–1999)*, Cargèse, France, 1999.

[AMHT00] Lars Ahrenberg, Magnus Merkel, Anna Sagvall Hein, et Jorg Tiedemann. Evaluation of word alignment systems. Dans *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC–2000)*, pages 1255–1261, Athens, Greece, 2000.

[Apr94] Jurij Derenikovich Apresjan. O jazyke tolkovanij i semanticheskix primitivax (On the language of explanations and semantic primitives). *Izvestija Rossiskoj Akademii Nauk*, 53(4) :466–484, 1994.

[Apr00] Jurij Derenikovich Apresjan. *Systematic Lexicography.* Oxford University Press, 2000. 320 p.

[ASU86] Alfred Aho, Ravi Sethi, et Jeffrey Ullman. *Compilers - Principles, Techniques, and Tools.* Addison-Wesley, 1986. 796 p.

[BBK05] Valérie Bellynck, Christian Boitet, et John Kenwright. ITOLDU, a Web Service to Pool Technical Lexical Terms in a Learning Environment and Contribute to Multilingual Lexical Databases. Dans *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistic (CICLing-05)*, pages 316–324, Mexico City, Mexico, 2005.

[Béd89] Chantal Bédécarrax. *Classification automatique en analyse relationnelle : la quadri-décomposition et ses applications.* Thèse, Université Paris VI, 1989. 165 p.

[BIS04] Igor Boguslavsky, Leonid L. Iomdin, et Victor Sizov. Multilinguality in ETAP-3 : Reuse of Lexical Resources. Dans *Proceedings of PostCOLING workshop on Multilingual Linguistic Resources (MLR–04)*, pages 7–14, Geneva, Switzerland, 2004.

[BK04] Valérie Bellynck et John Kenwright. ITOLDU : Accessing to Vocabulary learning in a technical English resource pooling environment. Dans *Proceedings of PAPILLON–2004*, Grenoble, France, 2004. 5 p.

[Bla95] Etienne Blanc. Une maquette de base lexicale multilingue à pivot lexical : PARAX. Dans AUPELF-UREF, éditeur, *Lexicomatique et Dictionnairique, Actes du colloque LTT*, pages 43–58, Montreal, Canada, 1995.

[Bla99] Etienne Blanc. PARAX-UNL : A large scale hypertextual multilingual lexical database. Dans *Proceedings of th 5th Natural Language Processing Pacific Rim Symposium*, pages 507–510, Beijing, 1999. Tsinghua University Press.

[BM04] Francis Brunet-Manquat. Syntactic parser combinaison for improved dependency analysis. Dans *Proceedings of PostCOLING workshop on RObust Methods in Analysis of Natural language Data (ROMAND–2004)*, Geneva, Switzerland, 2004. 7 p.

[BN86a] Christian Boitet et Nicolas Nedobejkine. Towards Integrated Dictionary for M(A)T : Motivations and Linguistic Organisation. Dans *Proceedings of the 11th International Conference on Computational Linguisttics (COLING–86)*, pages 423–428, Bonn, Germany, 1986.

[BN86b] Christian Boitet et Nicolas Nedobejkine. Vers une base lexicale intégrée pour la T(a)O : motivations et organisation linguistique. Dans *Proceedings of Journées francophones de l'informatique, bases de données et connaissances*, pages 151–169, Grenoble, France, 1986.

[Bou90] Denis Bouyssou. Building criteria : a prerequisite for MCDA. Dans Carlos António Bana e Costa, éditeur, *Readings in Multiple Criteria Decision-Aid*, pages 58–80, Berlin, Germany, 1990. Springer-Verlag.

[Brea] James William Breen. The EDICT project – `http://www.csse.monash.edu.au/~jwb/edict.html`.

[Breb] James William Breen. The JMDict Project – `http://www.csse.monash.edu.au/~jwb/j_jmdict.html`.

[Bre04] James William Breen. JMdict : a Japanese-Multilingual Dictionary. Dans *proceedings of Multilingual Linguistic Resources Workshop, post COLING-2004*, pages 71–78, Geneva, Switzerland, 2004.

[BSAG98] Andrew Borthwick, John Sterling, Eugene Agichtein, et Ralph Grishman. Exploiting Diverse Knowledge Sources via Maximum Entropy in Named

Entity Recognition. Dans *Proceedings of the Sixth Workshop on Very Large Corpora (WVLC–98)*, pages 152–160, Montreal, Canada, 1998.

[CBV+95] Ann Copestake, Ted Briscoe, Piek Vossen, Alicia Ageno, Irene Castellón, Francesc Ribas, German Rigau, Horacio Rodriguez, et Anna Samiotou. Acquisition of Lexical Translation Relations from MRDs. *Machine Translation : Special Issue on the lexicon*, 9(3) :33–69, 1995.

[Cha90] Jacques Chauché. Détermination sémantique en analyse structurelle : une expérience basée sur une définition de distance. *TAL Information*, 31(1) :17–24, 1990.

[CNF] CNFSH. Dictionnaire français d'hydrologie – `http://www.cig.ensmp.fr/~hubert/glu/indexdic.htm`.

[dA] Le dictionnaire Acrodict. Acrodict : Dictionnaire francophone des acronymes, sigles et abréviations informatiques – `http://www.teaser.fr/~spineau/acrodict/index.php`.

[dC02] Henri Bertaud du Chazaud. *Le Robert : Dictionnaire des synonymes.* 2002. 735 p.

[Des] Jean-Marc Desperrier. Le projet Dictionnaire français-japonais : `http://dico.fj.free.fr/index.php`.

[Des02] Jean-Marc Desperrier. Analysis of the results of a collaborative project for the creation of a Japanese-French dictionary. Dans *Proceedings of PAPILLON–2002*, Tokyo, Japan, 2002. 10 p.

[DJ96] Bonnie J. Dorr et Douglas Jones. Acquisition of Semantic Lexicons : Using Word Sense Disambiguation to Improve Precision. Dans *proceedings of Special Interest Group on the Lexicon (SIGLEX–96)*, pages 42–50, Santa Cruz, USA, 1996.

[EDR93] EDR. EDR Electronic Dictionary Technical Guide. Project report TR–042, Japan Electronic Dictionary Research Institute Ltd., 1993.

[Eur] Le projet EuroWordNet : `http://www.illc.uva.nl/EuroWordNet/`.

[Euz] Jean Euzéby. Dictionnaire de Bactériologie Vétérinaire – `http://www.bacterio.cict.fr/bacdico/garde.html`.

[Fel98] Christiane Fellbaum. *WordNet : An Electronic Lexical Database, Language, Speech, and Communication Series.* The MIT Press, Cambridge, MA, USA, 1998. 423 p.

[FeM] Le dictionnaire français-anglais-malais (FeM) – `http://silfide.imag.fr/`.

[FGW92] David Farwell, Louise Guthrie, et Yorick Wilks. The Automatic Creation of Lexical Entries for a Multilingual MT system. Dans *Proceedings of the 14th International Conference on Computational Linguistics (COLING–92)*, pages 532–538, Nantes, France, 1992.

[FGW93] David Farwell, Louise Guthrie, et Yorick Wilks. Automatically creating lexical entries for ULTRA, a Multilingual MT System. *Machine Translation*, 8(3) :127–146, 1993.

[FW91] David Farwell et Yorick Wilks. ULTRA : a Multilingual Machine Translator. Dans *Proceedings of the 3th Machine Translation Summit (MT Summit III)*, pages 19–24, Washington, DC, USA, 1991.

[GN91] Kenneth Goodman et Sergei Nirenburg. *The KBMT Project : A Case Study in Knowledge-Based Machine Translation.* Morgan Kaufmann, San Mateo, CA, USA, 1991. 331 p.

[Goo89] Kenneth Goodman. Special Issues on Knowledge Based MT, Parts I and II. *Machine Translation*, 4(1–2), 1989.

[Gro75] Maurice Gross. *Méthodes en syntaxe : Régime des constructions complétives*, volume 1365. 1975. 414 p.

[Hai98a] Doan Nguyen Hai. Accumulation of Lexical Sets : Acquisition of Dictionary Resources and Production of New Lexical Sets. Dans *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL–98)*, pages 330–335, Montreal, Canada, 1998.

[Hai98b] Doan Nguyen Hai. *Techniques génériques d'accumulation d'ensembles lexicaux structurés à partir de ressources dictionnairiques informatisées multilingues hétérogènes.* thèse, Institut National Polytechnique de Grenoble, 1998. 168 p.

[Hib] Hibernate. Hibernate – `http://hibernate.sourceforge.net/`.

[HKP02] Eduard Hovy, Margaret King, et Andrei Popescu-Belis. Principles of context-based machine translation evaluation. *Machine Translation*, 1 :43–75, 2002.

[HL95] Walter L. Hürsch et Cristina Videira Lopes. Separation of Concerns. Rapport Technique NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, Massachusetts, USA, 1995.

[HN98] Nabil Hathout et Fiammetta Namer. Automatic Construction and Validation of French Large Lexical Resources : Reuse of Verb Theoretical Linguistic Descriptions. Dans *Proceedings of the First International Conference on Language Resources and Evaluation (LREC–98)*, pages 627–636, Granada, Spain, 1998.

[IMS+99] Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, et Yoshihiro Hayashi. *Goi-Taikei – A Japanese Lexicon CDROM.* Iwanami Shoten, Tokyo, Japan, 1999.

[IV96] Nancy Ide et Jean Veronis. Codage TEI des dictionnaires électroniques. *Cahiers GUTenberg*, 24 :170–176, 1996.

[Jos87] Aravind K. Joshi. The relevance of tree adjoining grammar to generation. *Natural Language Generation : New Directions in Artificial Intelligence, Psychology, and Linguistics*, pages 233–252, 1987.

[Kal04] Ignacy Kaliszewski. Out of the mist – towards decision-maker-friendly multiple criteria decision making support. *European Journal of Operational Research*, 158 :293–307, October 2004.

[Kin] Irving Kind. BABEL : A Glossary of Computer Oriented Abbreviations and Acronyms – `http://pcline.epfl.ch/pc/div/babel96a.htm`.

[Laf02] Mathieu Lafourcade. Automatically Populating Acception Lexical Database through Bilingual Dictionaries and Conceptual Vectors. Dans *Proceedings of PAPILLON–2002*, Tokyo, Japan, 2002.

[Lar02] François Lareau. A Practical Guide for Writing DiCo Entries. Dans *Proceedings of PAPILLON–2002*, Tokyo, Japan, 2002. 4 p.

[LK04] Lian-Tze Lim et Tang Enya Kong. Building an Ontology-based Multilingual Lexicon for Word Sense Disambiguation in Machine Translation. Dans *Proceedings of PAPILLON–2004*, Grenoble, France, 2004. 8 p.

[LPS02] Mathieu Lafourcade, Violaine Prince, et Didier Schwab. Vecteurs conceptuels et structuration émergente de terminologies. *Traitement Automatiques des Langues*, 43(1) :43–72, 2002.

[LS99] Mathieu Lafourcade et Eugène Sandford. Analyse et désambiguïsation lexicale par vecteurs sémantiques. Dans *Proceedings of le 6ème conférence sur le Traitement Automatique des Langues Naturelles (TALN–1999)*, pages 351–356, Cargèse, France, 1999.

[LW94] Barbara H. Liskov et Jeannette M. Wing. A Behavioral Notion of Subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6) :1811–1841, 1994.

[Man] Jean-Luc Manguin. Dictionnaire des synonymes du laboratoire CRISCO à l'université de Caen – `http://elsap1.unicaen.fr/dicosyn.html`.

[MCP95] Igor Mel'cuk, André Clas, et Alain Polguère. *Introduction à la lexicologie explicative et combinatoire.* Duculot, Louvain-la-Neuve, 1995. 256 p.

[Mee04] René Meertens. *Guide anglais-français de la traduction.* Casteilla, 2004. 480 p.

[Mel81] Igor Mel'cuk. Meaning-Text models : a recent trend in Soviet linguistics. *Annual Review of Anthropology*, 10 :27–62, 1981.

[Mel84] Igor Mel'cuk. *DEC : Dictionnaire Explicatif et Combinatoire du français contemporain, recherches lexico-sémantiques I.* Presses de l'Université de Montréal, Montreal, Canada, 1984. 172 p.

[Mel88] Igor Mel'cuk. *DEC : Dictionnaire Explicatif et Combinatoire du français contemporain, recherches lexico-sémantiques II.* Presses de l'Université de Montréal, Montreal, Canada, 1988. 332 p.

[Mel92] Igor Mel'cuk. *DEC : Dictionnaire Explicatif et Combinatoire du français contemporain, recherches lexico-sémantiques III.* Presses de l'Université de Montréal, Montreal, Canada, 1992. 323 p.

[Mel99] Igor Mel'cuk. *DEC : Dictionnaire Explicatif et Combinatoire du français contemporain, recherches lexico-sémantiques IV.* Presses de l'Université de Montréal, Montreal, Canada, 1999. 347 p.

[Mey97] Bertrand Meyer. *Object-Oriented Software Construction.* Prentice-Hall, 1997. 1254 p.

[Mih03] Rada Mihalcea. Turning WordNet into an Information Retrieval Resource : Systematic Polysemy and Conversion to Hierarchical Codes. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 17(5) :689–704, 2003.

[MKFS03] Toshiki Murata, Mihoko Kitamura, Tsuyoshi Fukui, et Tatsuya Sukehiro. Implementation of Collaborative Translation Environment 'Yakushite Net'. Dans *Proceedings of the 9th Machine Translation Summit (MT Summit IX)*, pages 479–482, New Orleans, Louisiana, USA, 2003.

[ML01] Mathieu Mangeot-Lerebours. *Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue.* thèse, Université Joseph Fourier, 2001. 279 p.

[MLSL03] Mathieu Mangeot-Lerebours, Gilles Sérasset, et Mathieu Lafourcade. Construction collaborative d'une base lexicale multilingue - Le projet Papillon. *Traitement Automatique des Langues*, 44(2) :151–176, 2003.

[MNC93] Teruko Mitamura, Eric H. Nyberg, et Jaime G. Carbonell. Automated Corpus Analysis and the Acquisition of Large, Multi-Lingual Knowledge Bases for MT. Dans *Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI–93)*, Kyoto, Japan, 1993. 17 p.

[MP87] Igor Mel'cuk et Alain Polguère. A Formal Lexicon in the Meaning-Text Theory (or How to do Lexica with Words). *Computational Linguistics*, 13(3-4) :261–275, 1987.

[MSKO96] Hideo Miyoshi, Kenji Sugiyama, Masahiro Kobayashi, et Takano Ogino. An Overview of the EDR Electronic Dictionary and the Current Status of Its Utilization. Dans *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)*, pages 1090–1094, Copenhagen, Danmark, 1996.

[Nir89] Sergei Nirenburg. KBMT-89 – A Knowledge-Based MT Project at Carnegie Mellon University. Dans *Proceedings of the 2nd Machine Translation Summit (MT Summit II)*, pages 141–147, Munich, Germany, 1989.

[NMC94] Eric H. Nyberg, Teruko Mitamura, et Jaime G. Carbonell. Evaluation Metrics for Knowledge-Based Machine Translation. Dans *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, pages 95–99, Kyoto, Japan, 1994.

[Par72] David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12) :1053–1058, 1972.

[Péc92] Daniel Péchoin. *Thésaurus Larousse – Des idées aux mots – Des mots aux idées.* Larousse, Paris, 1992. 1146 p.

[Ped00] Ted Pedersen. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. Dans *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics (NAACL–2000)*, pages 63–69, Seattle, Washington, USA, 2000.

[PH97] Mari Pöyhönen et Raimo P. Hämäläinen. On the convergence of multiattribute weighting methods. Rapport technique, Systems Analysis Laboratory, Helsinki University of Technology, 1997.

[Pol98] Alain Polguère. La théorie Sens-Texte. *Dialangue*, 8–9 :9–30, 1998.

[Pol00a] Alain Polguère. Towards a theoretically motivated general public dictionary of semantic derivations and collocations for French. Dans *proceedings of the 9th EURALEX International Congress*, pages 517–527, Stuttgart, Germany, 2000.

[Pol00b] Alain Polguère. Une base de données lexicales du français et ses applications possibles en didactique. *Revue de Linguistique et de Didactique des Langues (LIDIL)*, 21 :75–95, 2000.

[Pri] Princeton University. Princeton WordNet – `http://www.cogsci.princeton.edu/~wn/index.shtml`.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, et Wei-Jing Zhu. BLEU : a Method for Automatic Evaluation of Machine Translation. Dans *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistic (ACL–2002)*, pages 311–318, Philadelphia, PA, USA, 2002.

[PS88] Carl Pollard et Ivan A. Sag. *An Information-Based Approach to Syntax and Semantics : Vol. 1 : fundamentals.* Center for the Study of Language and Information, Stanford, CA, USA, 1988. 233 p.

[PS94] Carl Pollard et Ivan A. Sag. *Head-Driven Phrase Structure Grammar.* University of Chicago Press and CSLI Publications, Chicago, Illinois, USA, 1994.

[qqc] QQCVD : Dictionnaire des acronymes, sigles et abréviations – `http://qqcvd.webiciel.com`.

[SC96] Eugène Sandford et Jacques Chauché. A semantic and syntactic analysis. Dans *International Seminar on Multimodal Interactive Disambiguation (MIDDIM–96)*, pages 59–66, Col de Porte, Grenoble, France, 1996.

[Sér94] Gilles Sérasset. *SUBLIM, un Système Universel de Bases Lexicales Multilingues et NADIA, sa spécialisation aux bases lexicales interlingues par acceptions.* thèse, Université Joseph Fourier, 1994. 194 p.

[SKSM01] Sayori Shimohata, Mihoko Kitamura, Tatsuya Sukehiro, et Toshiki Murata. Collaborative Translation Environment on the Web. Dans *Proceedings of the 8th Machine Translation Summit (MT Summit VIII)*, pages 331–334, Santialgo de Gompostela, Galicia, Spain, 2001.

[SMI+99] Sayori Shimohata, Toshiki Murata, Atsushi Ikeno, Tsuyoshi Fukui, et Hideki Yamamoto. Machine Translation System PENSEE : System Design and Implementation. Dans *Proceedings of the 7th Machine Translation Summit (MT Summit VII)*, pages 380–384, Singapore, 1999.

[Tee02] Aree Teeraparbseree. A practical guide to lexical data acquisition with recupdic. Dans *Proceedings of PAPILLON–2002*, Tokyo, Japan, 2002. 17 p.

[Tee04a] Aree Teeraparbseree. Qualitative Evaluation of Automatically Calculated Acception Based MLDB. Dans *Proceedings of PostCOLING workshop on Multilingual Linguistic Resources (MLR–04)*, pages 23–30, Geneva, Switzerland, 2004.

[Tee04b] Aree Teeraparbseree. Un systéme adaptable pour l'initialisation automatique d'une base lexicale interlingue par acceptions. Dans *Proceedings of Rencontre des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL–2004)*, Fès, Maroc, 2004.

[TMLP00] Mutsuko Tomokiyo, Mathieu Mangeot-Lerebours, et Emmanuel Planas. Papillon : a Project of Lexical Database for English, French and Japanese, using Interlingual Links. Dans *Proceedings of Journées Science et Technologie de l'ambassade de France au Japon*, Tokyo, Japan, 2000. 3 p.

[TU94] Kumiko Tanaka et Kyoji Umemura. Construction of a bilingual dictionary intermediated by a third language. Dans *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, pages 297–303, Kyoto, Japan, 1994.

[Uch01] Hiroshi Uchida. The Universal Networking Language Beyond Machine Translation. Dans *International Symposium on Language in Cyberspace*, Seoul, South Korea, 2001. 14 p.

[Vos98] Piek Vossen, éditeur. *EuroWordNet : A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, 1998. 251 p.

[Yok95] Toshio Yokoi. The EDR Electronic Dictionary. *Communications of the ACM*, 38(11) :42–44, 1995.

[YRY+96] Gut Yvan, Puteri Rashida Megat Ramli, Zaharin Yusoff, Chuah Choy Kim, Salina A. Samat, Christian Boitet, Nicolas Nédobejkine, et Mathieu Lafourcade. *Kamus Perancis Melayu Dewan : Dictionary Français-Malais*, volume 1. Dewan Bahasa Dan Pustaka, Kuala Lumpur, 1996. 667 p.

[Zhe] Dong Zhendong. HowNet Knowledge Database – `http://www.keenage.com/`.

Résumé

Cette thèse aborde le problème de la structuration de bases lexicales multilingues (BDLM) en lexise et axies, à partir de ressources existantes. Ce travail est motivé par l'inadéquation des techniques existantes utilisées isolément, pour la structuration de BDLM.

Pour résoudre ce problème, la stratégie proposée est de composer des techniques existantes de désambiguïsation pour structurer semi-automatiquement des bases lexicales multilingues à lexies et acceptions interlingues. De plus, cette thèse propose une catégorisation des critères d'évaluation de la qualité des BDLM, ainsi que les mesures correspondantes.

Cette stratégie a été implémentée dans Jeminie, un système logiciel adaptable qui permet d'implémenter à la fois des méthodes de structuration de BDLM et des mesures de qualité, sous la forme de modules logiciels réutilisables.

Des compositions arbitraires de ces modules peuvent être définies par un lexicologue dans un langage de haut niveau d'abstraction, ce qui permet d'adapter facilement la structuration et l'évaluation de qualité en fonction des objectifs du lexicologue et des ressources disponibles sans nécessiter de connaissances en programmation.

L'intérêt de cette approche a été validé expérimentalement : la qualité des BDLM obtenues est meilleure par combinaison de techniques qu'avec chaque technique antérieure utilisée seule.

Mots clés : base lexicale multilingue, construction automatique de lexies et axies, acception interlingue, évaluation de qualité, lexicographie computationnelle

Abstract

This thesis studies the problems of structuring multilingual lexical databases (MLDB) from existing resources. This work is motivated by the inadequacy of existing techniques applied separately, for the construction of MLDB.

In order solve this problem, the strategy proposed consists in composing existing disambiguation techniques, to semi-automatically build multilingual lexical databases consisting of monolingual and interlingual acceptions. Moreover, this thesis proposes a categorization of the criteria for the qualitative evaluation of MLDB, as well as corresponding measurements.

This strategy has been implemented in Jeminie, an adaptable software system which allows to implement both MLDB building methods and qualitative measurements as reusable software modules.

Arbitrary compositions of these modules can be defined by a lexicologist in a high-level language, which allows to easily adapt the structuration and evaluation processes according to the objectives of the lexicologist and to the available resources without requiring knowledge in programming.

The interest of this approach has been validated in experiments : the quality of obtained MLDB is better than those obtained with each technique separately.

Keywords : multilingual lexical database, automatic building of lexies and axies, interlingual acception, qualitative evaluation, computational lexicography