Multilinguisation d'ontologies dans le cadre de la recherche d'information translingue dans des collections d'images accompagnées de textes spontanés

David Rouquet

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade, décerné par l'Université de Grenoble, de

DOCTEUR ÈS SCIENCES

Spécialité : **Informatique**

Arrêté ministériel :

Présentée par

David Rouquet

Thèse dirigée par **Christian Boitet**
et codirigée par **Valérie Bellynck**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG), équipe GETALP**
et de **l'école doctorale MSTII**

Multilinguisation d'ontologies

dans le cadre de la recherche d'information translingue dans des collections d'images accompagnées de textes spontanés

Thèse soutenue publiquement le **6 avril 2012**,
devant le jury composé de :

Marie-Christine Fauvet
PR Université Joseph Fourrier, Présidente
Liming Chen
PR École Centrale de Lyon, Rapporteur
Marc Dymetman
DR XRCE, Rapporteur
Christophe Roche
PR Université de Savoie, Rapporteur
Jérôme Euzenat
DR INRIA Montbonnot, Examinateur
Fabien Gandon
CR INRIA Sophia Antipolis, Examinateur
Christian Boitet
PR Université Joseph Fourrier, Directeur de thèse
Valérie Bellynck
MCF Grenoble-INP, Co-Directeur de thèse

"Espace, frontière de l'infini [...] ***, avancer vers l'inconnu."***
Capitaine Spock, Star Trek.

"Au fond de l'Inconnu pour trouver du nouveau !"
Charles Baudelaire, Le voyage.

"Ne crois pas que tu t'es trompé de route
quand tu n'es pas allé assez loin."
Claude Aveline, Avec toi-même.

Remerciements

Je remercie ma codirectrice de thèse, Valérie Bellynck, de m'avoir constamment conseillé et encouragé, ainsi que mon directeur, Christian Boitet, de m'avoir donné goût à l'informatique et de m'avoir invité dans l'aventure de cette thèse.

Un grand merci aux rapporteurs Liming Chen, Marc Dymetman, et Christophe Roche de m'avoir fait l'honneur de relire ce mémoire, ainsi qu'aux examinateurs Marie-Christine Fauvet, Jérôme Euzenat et Fabien Gandon.

Mes sentiments vont à tous les membres de l'équipe GETALP qui m'ont accueilli à bras ouverts et m'ont permis d'être à l'aise dans mon travail.

Merci à ma famille, mes parents, mes grands parents pour leur soutien et pour m'avoir transmis le sens des choses.

Aux amis sans qui tout est tellement moins amusant.

A Ingrid.

Résumé

Le Web est une source proliférante d'objets multimédia, décrits dans différentes langues naturelles. Afin d'utiliser les techniques du Web sémantique pour la recherche de tels objets (images, vidéos, etc.), nous proposons une méthode d'extraction de contenu dans des collections de textes multilingues, paramétrée par une ou plusieurs ontologies. Le processus d'extraction est utilisé pour indexer les objets multimédia à partir de leur contenu textuel, ainsi que pour construire des requêtes formelles à partir d'énoncés spontanés. Il est basé sur une annotation interlingue des textes, conservant les ambiguïtés de segmentation et la polysémie dans des graphes. Cette première étape permet l'utilisation de processus de désambiguïsation "factorisés" au niveau d'un lexique pivot (de lexèmes interlingues). Le passage d'une ontologie en paramètre du système se fait en l'alignant de façon automatique avec le lexique interlingue. Il est ainsi possible d'utiliser des ontologies qui n'ont pas été conçues pour une utilisation multilingue, et aussi d'ajouter ou d'étendre l'ensemble des langues et leurs couvertures lexicales sans modifier les ontologies. Un démonstrateur pour la recherche multilingue d'images, développé pour le projet ANR OMNIA, a permis de concrétiser les approches proposées. Le passage à l'échelle et la qualité des annotations produites ont ainsi pu être évalués.

Mots-clés: Recherche d'information, extraction de contenu, ontologies, multilinguisme.

Abstract

The World Wide Web is a proliferating source of multimedia objects described using various natural languages. In order to use semantic Web techniques for retrieval of such objects (images, videos, etc.), we propose a content extraction method in multilingual text collections, using one or several ontologies as parameters. The content extraction process is used on the one hand to index multimedia objects using their textual content, and on the other to build formal requests from spontaneous user requests. The process is based on an interlingual annotation of texts, keeping ambiguities (polysemy and segmentation) in graphs. This first step allows using common desambiguation processes at th elevel of a pivot langage (interlingual lexemes). Passing an ontology as a parameter of the system is done by aligning automatically its elements with the interlingual lexemes of the pivot language. It is thus possible to use ontologies that have not been built for a specific use in a multilingual context, and to extend the set of languages and their lexical coverages without modifying the ontologies. A demonstration software for multilingual image retrieval has been built with the proposed approach in the framework of the OMNIA ANR project, allowing to implement the proposed approaches. It has thus been possible to evaluate the scalability and quality of annotations produiced during the retrieval process.

Keywords: Information retrieval, content extraction, ontologies, multilingualism

Table des matières

Table des figures

Liste des tableaux

Introduction générale

En recherche documentaire classique, l'indexation des documents se fait relativement à des ressources terminologiques plus ou moins structurées. Ces ressources fournissent un ensemble de descripteurs (ou termes préférés), utilisés pour représenter sans ambiguïté une notion contenue dans un document lors du processus d'indexation (Cacaly et al. (2008)).

L'évolution des paradigmes de représentation des connaissances dans les systèmes informatiques relevant de l'intelligence artificielle a débouché dans les années 1980 sur la notion d'ontologie. Les ontologies, au sens informatique du terme, sont des "*spécifications explicites et formelles d'un domaine de connaissance*" (Gruber (1993)) utilisées dans les systèmes d'informations. Ces structures ont un rôle fondamental dans la création du Web sémantique, un outillage qui permet aux ordinateurs d'accéder au sens des données disponibles sur le Web pour assister les utilisateurs de façon plus "intelligente" (Berners-Lee et al. (2001)).

Woods (1997) a démontré que l'utilisation de ressources sémantiques ou lexico-sémantiques (assimilables à des ontologies) pour l'indexation conceptuelle de documents améliore sensiblement les résultats et la satisfaction des utilisateurs en recherche d'information. De nombreux systèmes ou prototypes de *recherche sémantique d'information* ont été développés depuis le début des années 1990, comme l'atteste l'état de l'art présenté par Haav et Lubi (2001).

Le multilinguisme est un enjeu important pour ces outils de recherche sémantique. En effet, trois groupes de langues sont à considérer dans la création de tels systèmes : (1) les langues utilisées dans les ressources sémantiques, (2) les langues utilisées dans les requêtes et (3) les langues utilisées dans les documents ou pour décrire les documents. Afin de permettre une recherche translingue, il est logique de gérer le multilinguisme au niveau des ressources sémantiques. En effet, ces dernières jouent un rôle de pivot puisque c'est par rapport à elles que les documents sont indexés et que les requêtes sont résolues. Les ressources sémantiques utilisables pour "guider" la recherche d'information ou l'indexation de documents sont variées Weller (2007). Nous nous concentrons sur l'utilisation d'ontologies de domaine, modifiables selon les préférences des utilisateurs ou la pertinence, pour des ensembles de données spécifiques.

Le système de recherche et d'indexation doit donc s'adapter aux ressources sémantiques. Des méthodes de multilinguisation *a posteriori* d'ontologies apparaissent alors comme cruciales pour la recherche sémantique et translingue d'information. Elles permettraient en outre de considérer les ontologies comme de réels paramètres du système. Aucune solution générale et satisfaisante n'a toutefois été proposée pour la gestion multilinguisme dans les ontologies, en particulier pour la recherche d'information. Cette thèse propose une contribution à la résolution de ce problème.

Le mémoire contient 4 chapitres. Le premier introduit les enjeux de recherche d'information liés au développement du Web. Nous présentons le projet ANR OMNIA dans lequel s'est déroulé cette thèse et les défis scientifiques qu'il tentait de relever. Ce projet visait à la création d'un système, guidé par une ontologie, pour l'indexation et la recherche multimodales d'images accompagnées de textes spontanés. Le second chapitre est dédié aux ontologies. Nous proposons un ensemble de définitions, dont une notion de *description* d'un objet dans une ontologie, étendant

la notion classique de descripteur en recherche documentaire. Les contraintes de complexité liées à l'utilisation d'ontologies pour la recherche d'information et les choix faits dans le cadre du projet OMNIA sont discutés. Nous nous attaquons ensuite au problème de la multilinguisation d'ontologies. Une revue de l'état de l'art sur l'enrichissement linguistique d'ontologies, y compris en contexte monolingue, nous mène à une solution de lexicalisation externe. L'étude de plusieures architectures lexicales multilingues et les contraintes de recherche d'information liées au projet OMNIA nous ont amené à utiliser un alignement avec un lexique pivot (les UW, ou lexèmes interlingues du language UNL) pour multilingualiser une ou plusieurs ontologies. Nous présentons enfin le démonstrateur réalisé dans le cadre du projet OMNIA et implémentant certaines propositions de cette thèse.

Chapitre 1

Motivations générales et contexte applicatif

Sommaire

1 Nouveaux enjeux liés au développement du Web

Depuis sa création en 1990, Internet et le World Wide Web ont connu une cote de popularité exponentielle. Son usage par le grand public a été rendu possible la création d'interfaces graphiques pour la navigation (navigateurs) puis par le développement de moteurs de recherche. Les modalités d'accès au foisonnement d'informations bientôt disponibles (un million de sites en 1997) ont été un enjeu majeur pour la réussite du Web dès ses débuts.

1.1 Web 2.0

Internet était initialement pensé comme une infrastructure de collaboration, au sein de laquelle tous les utilisateurs consulteraient et créeraient de l'information. On a cependant observé l'émergence d'un média de diffusion plus que de collaboration, avec une séparation nette entre les usagers créateurs et consommateurs d'information. Ce clivage tend à s'estomper depuis le début des années 2000, avec l'utilisation croissante des wikis, blogs ou des réseaux sociaux. Ces technologies permettent aux utilisateurs de se réapproprier la Toile avec une activité de publication aussi importante que la simple consultation. On observe donc aujourd'hui un Web devenu réellement collaboratif, communément qualifié de Web 2.0 (Beck (2008)). Cette évolution vers le Web 2.0, tant comportementale que technologique, a eu un impact important sur la nature des ressources disponibles. Les contenus deviennent en effet de plus en plus multimédia et multilingues.

1.2 Web multimédia

Les possibilités multimédia sont apparues dès 1993 avec le navigateur NCSA Mosaic 0.10 qui permettait d'afficher des images intégrées aux pages Web. De même, l'exploitation de fichiers vidéo ou son, bien que non standardisés et non intégrés directement aux pages Web, s'est rapidement étendue. Le support de tels fichiers multimédia a même été déterminant dans la "guerre des navigateurs" qui opposa Netscape et Microsoft Explorer (Miller et al. (2009)). On trouve aujourd'hui sur le Web beaucoup de pages intégrant des éléments multimédia et de sites dédiés à la publication d'images, de sons ou de vidéos. La recherche d'objets multimédia et leur prise en compte dans l'indexation des pages Web constituent des défis importants pour la recherche d'information, avec notamment le recours à des analyses multimodales et l'exploitation de descripteurs très hétérogènes (Lew et al. (2006)). Ces descripteurs se présentent comme des listes de propriétés, c'est-à-dire des listes de couples $< attribut, valeur >$ où les attributs peuvent être de différent types.

1.3 Web multilingue

L'augmentation impressionnante du nombre d'utilisateurs du Web est allée de pair avec une diversification de ses utilisateurs, de leurs situations géographiques et de leurs langues. Dans le même temps, le nombre de pages Web disponibles a explosé et le nombre de langues représentées a augmenté. Gérer la diversité linguistique sur le Web est devenu un enjeu crucial pour permettre l'usage équitable de l'information (accès, diffusion, dialogue, etc.), comme l'attestent l'étude scientifique pilotée par l'UNESCO de 1996 à 2008 (Pimienta et al. (2009)) et le projet Multilingual Web coordonné par le W3C [1].

On estime qu'en 1998, 80% des utilisateurs du Web étaient anglophones et 75% des pages étaient rédigées en anglais. En 2007, ce pourcentage n'était plus que de 32% des utilisateurs et

1. `http://www.multilingualweb.eu`

45% des pages (Pimienta et al. (2009)). Les langues asiatiques ont notamment connu un essor considérable, mais l'UNESCO estime qu'environ 90% des 6000 langues "actives" dans le monde ne sont pas représentées de façon significative sur Internet [2].

Il est clair qu'on ne peut se satisfaire de l'anglais comme *lingua franca* d'Internet. Beaucoup d'informations peuvent ne pas être accessibles dans la langue maternelle d'un utilisateur, ni même en anglais. D'autre part, quantité de contenus multimédia (musiques, images, etc.) sont indépendants de la langue utilisée pour les décrire. Des processus de recherche d'information translingue sont donc nécessaires pour garantir l'accès aux informations et favoriser la diversité linguistique sur Internet.

1.4 Web 3.0

On parle fréquemment du Web Sémantique comme synonyme du Web 3.0. Largement promu par le W3C, le Web Sémantique est un outillage du Web qui doit permettre aux machines d'accéder au sens des informations contenues dans les pages Web, alors que le sens n'est en général accessible que par l'interprétation humaine (Breitman et al. (2007)).

Cette évolution nécessite la création d'ontologies (représentations formelles de connaissances) et l'inclusion de métadonnées faisant référence aux éléments décrits dans ces ontologies au sein des pages Web. L'existence de telles structures formelles permet aux machines d'utiliser les informations disponibles pour en créer de nouvelles par inférence. On peut alors envisager le développement de processus intelligents pour la recherche et l'exploitation des documents (Berners-Lee et al. (2001)).

La réalisation de cette extension sémantique du Web demande cependant une participation conséquente des usagers, puisque les publications sont dédoublées. Les contenus dédiés aux humains (textes en langue naturelle et autres médias) doivent être accompagnés de métadonnées décrivant formellement ces contenus pour les machines. Mais il est peu vraisemblable que les utilisateurs créent systématiquement les métadonnées sémantiques accompagnant leurs publications. D'autre part, le Web sémantique ne saurait permettre une réelle interopérabilité des systèmes sans des représentations de connaissances consensuelles, largement partagées à l'échelle du Web. Or, on observe aujourd'hui qu'il y a un grand nombre d'ontologies utilisées par des communautés relativement restreintes et dédiées à des applications spécifiques.

Ainsi, le développement de techniques automatiques pour l'annotation sémantique des documents est crucial pour le déploiement du Web Sémantique à une échelle significative. De tels processus devraient être capables d'analyser des contenus multimédia et de les annoter avec les ressources sémantiques d'une ontologie passée en paramètre.

2 Systèmes visés : l'exemple du projet OMNIA

2.1 Le projet ANR OMNIA

Le projet OMNIA [3] a été financé par l'Agence Nationale de la Recherche (ANR) de début 2008 à fin 2010. Il rassemblait des équipes du XRCE [4], du LIRIS [5] et du LIG [6]. Son but était la

2. www.unesco.org/culture
3. www.omnia-project.org
4. Centre de Recherche Européen de Xerox : www.xrce.xerox.com
5. Laboratoire d'InfoRmatique en Image et Systèmes d'information : liris.cnrs.fr
6. Laboratoire d'Informatique de Grenoble : www.liglab.fr

mise en place d'un système de recherche "sémantique" d'images, accompagnées de textes multilingues (légendes, commentaires, etc.), dans de grands entrepôts de données. Ce projet prend en compte trois aspects du Web présentés précédemment : multilinguisme, multimodalité (objets multimédia), et sémantique. Nous présentons dans cette partie les grandes lignes de ce projet au sein duquel s'est déroulée la majeure partie de cette thèse.

2.1.1 Scénario

Les applications envisagées se placent dans un scénario pré-presse, éditorial ou créatif. Dans ce scénario, les auteurs d'articles ou de brochures souhaitent illustrer leurs documents à partir de bases d'images journalistiques (AFP, BelgaNews, etc.) ou libres de droits (FlickR, Wikimedia, etc.). Ils peuvent formuler des requêtes classiques, dans leur langue maternelle, ou même soumettre au système l'intégralité du texte à illustrer. Les bases d'images dans lesquelles on cherche sont en constante évolution ; les milliers d'images ajoutées chaque jour doivent être indexées au fur et à mesure.

2.1.2 Stratégie

Les travaux de cette thèse s'intègrent dans la stratégie de recherche et d'indexation "avec ontologie" définie au début du projet OMNIA (Rouquet et al. (2010b)). Ils concernent notamment l'application de cette stratégie en contexte multilingue. Cependant, n'étant pas spécialiste en RI, il est hors de notre propos de défendre cette approche contre d'autres "sans ontologie". Nous nous plaçons simplement dans l'approche "avec ontologie", puisqu'elle est le cadre de notre projet, et de bien d'autres comme ceux présentés par Haav et Lubi (2001), même s'il n'est pas prouvé qu'elle soit opérationnellement meilleure que les approches "sans ontologie".

Dans cette stratégie, les images sont au préalable indexées par rapport à une hiérarchie de concepts faiblement contrainte par des axiomes logiques (*lightweight ontology*) comme illustré par la figure 2.1.2. Il est possible d'utiliser diverses "ontologies de domaine" disponibles sur le Web pour réaliser des indexations plus pertinentes (par exemple une ontologie de classification du vivant pour des photos d'animaux et de plantes). Dans le cadre de cette thèse, nous prévoyons de plus que les utilisateurs puissent également utiliser un schéma de classification personnalisé. Cela nous mène à la nécessité de concevoir un système permettant la recherche dans une même collection de données, à travers plusieurs ontologies.

L'indexation est réalisée de façon automatique grâce aux résultats de différentes analyses visuelles des images, ainsi que du traitement des textes multilingues accompagnant les images. Une requête soumise par un utilisateur est transformée en une requête formelle, et résolue au niveau de l'ontologie. Un ensemble d'images, classées par pertinence, est proposé. L'utilisateur peut alors interagir avec le système, pour améliorer sa recherche. Deux types d'interaction sont possible :

1. la sélection des images les plus pertinentes selon l'utilisateur (le système renvoie alors des images sémantiquement ou visuellement plus proches de celles choisies).
2. la précision du sens de certains termes de la requête (le système résout alors la requête avec cette nouvelle interprétation).

2.1.3 Thèmes scientifiques principaux d'OMNIA

Le projet OMNIA était divisé en trois grands thèmes de recherche confiés aux partenaires.

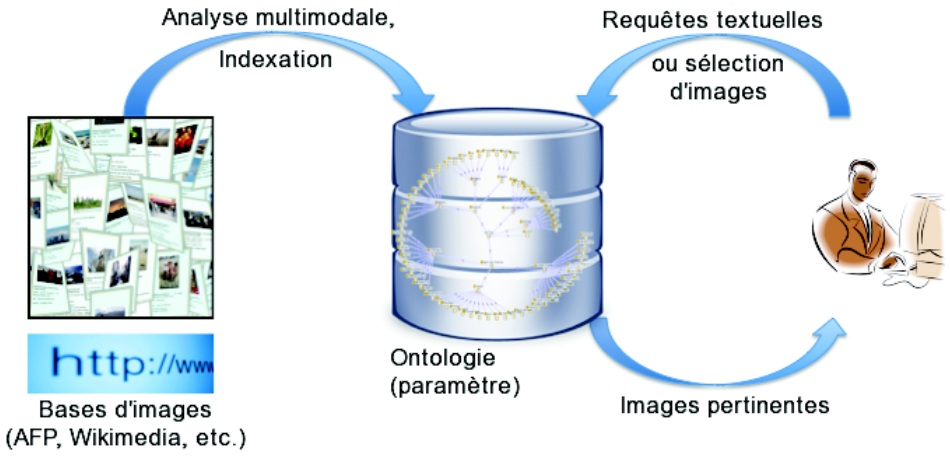


FIGURE 1 – Indexation et recherche d'images dans le projet OMNIA

Catégorisation et recherche par le contenu d'images. XRCE a travaillé sur une Catégorisation Visuelle Générique pour les images (*GVC - Generic Visual Categorization*). C'est un procédé qui catégorise automatiquement les images dans un ensemble discret de classes sémantiques (les classes d'une ontologie). On attribue à l'image des étiquettes (*labels*) correspondant aux objets ou concepts présents dans l'image. Une des spécificités de la méthode de GVC développée par XRCE est sa généricité. Elle peut être appliquée sans modification spécifique des paramètres à des catégories très variées (objets, scènes ou événements, peintures, etc.).

Le LIRIS a travaillé sur le calcul de descripteurs représentatifs des émotions suscitées par une image à partir de descripteurs de bas niveau (couleur, texture, etc.). Il s'agit de placer l'image par rapport à une représentation dimensionnelle des émotions, c'est à dire de calculer ses coordonnées dans un espace vectoriel dont les dimensions correspondent aux composantes d'une émotion (activité, appréciation, contrôle).

Analyse des textes multilingues. L'équipe GETALP [7] du LIG a travaillé sur une extraction de contenu dans les textes multilingues accompagnant les images (*textes compagnons*) afin de déterminer des éléments d'indexation pertinents pour le contenu ou le contexte des images. Le même processus est appliqué aux requêtes "libres" des utilisateurs pour identifier les éléments utilisables par le système pour la recherche.

Fusion d'analyses multimodales pour la classification de documents Web. Cette tâche, confiée au LIRIS, consiste à fusionner les descripteurs issus des différentes analyses des objets à indexer (images accompagnées de textes), pour obtenir une description unifiée de l'objet au sein de l'ontologie.

2.2 Caractéristiques essentielles des systèmes visés

Les principales caractéristiques d'OMNIA, présentées ci-après, sont :

1. le traitement de grand entrepôts d'objets multimédia, en évolution constante,

7. Groupe d'Étude pour la Traduction Automatique et le Traitement Automatisé des Langues et de la Parole

2. la prise en compte du multilinguisme dans les textes accompagant les objets,
3. l'utilisation de ressources sémantiques (*ontologiques*) pour l'indexation et la recherche.

Les travaux du projet sont applicables, en principe, à des systèmes qui partagent tout ou partie de ces caractéristiques.

2.2.1 Collections de données

Nous visons l'indexation d'objets multimédia et pas (ou peu) structurés. Dans le cas d'OMNIA, il s'agit d'images accompagnées de textes, mais des projets comme Videosense [8] utilisent une stratégie similaire pour des vidéos accompagnées de textes. D'autre part, les entrepôts de données contenant ces objets sont massifs (centaines de milliers ou millions d'objets) et en perpétuelle évolution (milliers d'objets ajoutés chaque jour). Deux jeux de données types ont été identifiés pour les expériences :

CLEF09-Belga. C'est un jeu de données mis à disposition par l'agence de presse belge *Belga News* [9] pour la campagne d'évaluation CLEF09. Il contient 500K images accompagnées de textes anglais d'une cinquantaine de mots chacun (environ 2,5M mots au total) comme illustré dans l'annexe A.1. Ce jeu de données correspond parfaitement au scénario éditorial d'OMNIA ; 8000 images sont ajoutées chaque jour sur le site de l'agence.

Wikimedia Commons. C'est une médiathèque, contenant plus de 10 millions de fichiers multimédia (sons, images, vidéos) [10], réutilisables sous diverses licences libres (GNU, Creative Commons, etc.). Les fichiers sont décrits par des textes dans plusieurs langues, parfois très courts, mais pouvant aller jusqu'à une centaine de mots. Les descriptions sont ajoutées par des contributeurs bénévoles dans une ou plusieurs langues comme illustré dans l'annexe A.2. Les images sont nettement prépondérantes dans cette base (près de 10 millions) et correspondent bien au scénario créatif d'OMNIA. Comme pour la base Belga News, près de 8000 images sont ajoutées chaque jour.

2.2.2 Possibilité d'ajout modulaire de langues

Comme nous l'avons vu avec Wikimedia Commons, les objets que l'on considère sont décrits par des textes spontanés, dans plusieurs langues. Il est donc nécessaire de développer des outils multilingues pour le traitement de ces textes. Par ailleurs, les textes étant écrits par des contributeurs du monde entier, le nombre de langues présentes dans la base est amené à augmenter. Il convient donc de pouvoir étendre le support linguistique du système à une langue supplémentaire sans interférer avec les composants concernant les autres langues.

2.2.3 Utilisation de ressources sémantiques

OMNIA est un système de Recherche d'Informations (RI) dit sémantique (ou conceptuel), utilisant des ressources sémantiques pour l'indexation et la recherche des documents. Sans être nécessairement toutes "ontologiques", ces ressources organisent un ensemble de termes ou de concepts avec des relations sémantiques.

8. `http://www.videosense.org/`
9. `http://www.belga.be`
10. `http://commons.wikimedia.org`

Ces ressources peuvent ainsi être des réseaux lexico-sémantiques comme Wordnet, des ontologies (au sens que nous préciserons au chapitre 2) généralistes comme SUMO ou YAGO, ou des ontologies de domaine. Ces approches dites sémantiques sont loin d'être consensuelles dans la communauté de RI. De nombreux travaux explorant cette voie ont cependant vu le jour depuis les années 1990, comme l'attestent Haav et Lubi (2001). Le projet OMNIA ne visait pas à démontrer une éventuelle supériorité des systèmes sémantiques sur les systèmes classiques. Il explorait les possibilités de :

1. faire passer à l'échelle ces systèmes avec inférence.
2. fusionner des analyses hétérogènes offertes par la création de descripteurs unifiés au sein des ressources sémantiques.
3. considérer les ressources sémantiques (en particulier les ontologies de domaine) comme de réels paramètres des systèmes, permettant ainsi la personnalisation des processus de recherche et d'indexation et la prise en compte des révisions de ces ressources.
4. gérer le multilinguïsme au niveau des ontologies, même si elles sont données en paramètre.

3 Problèmes

Les caractéristiques déterminées pour le système OMNIA induisent un certain nombre de problèmes et de défis. Il s'agit de problèmes non liés au TALN, comme le choix d'une stratégie de fusion de descripteurs ou le passage à l'échelle, ou bien liés au TALN comme l'extraction de contenu en contexte multilingue. Traiter l'ensemble de ces problèmes aurait largement dépassé le cadre d'une thèse. Nous terminons donc cette section en précisant ceux que nous avons traités, ce qui nous permettra de déterminer plus précisément les apports et le plan de ce mémoire.

3.1 Problèmes non liés au TALN

3.1.1 Fusion d'analyses multimodales

L'indexation de documents multimédia peut s'appuyer sur plusieurs types d'analyse. Dans le cas d'OMNIA, l'indexation des couples image-texte est réalisée à l'aide du contenu visuel des images et du contenu textuel. Des descripteurs hétérogènes sont alors obtenus : descripteurs visuels de bas niveau (couleur, netteté, saturation, etc.), descripteurs visuels de haut niveau (résultat de classification visuelle, descripteurs dimensionnels d'émotions), et descripteurs interlingues et/ou sémantiques issus des textes. La fusion de ces descripteurs est réalisée au sein d'une ontologie. Deux stratégies sont identifiées.

1. La fusion précoce consiste à unifier les descripteurs avant leur intégration dans l'ontologie.
2. La fusion tardive consiste à construire des descriptions dans l'ontologie à partir de chaque descripteur de plus bas niveau. La fusion et la résolution d'éventuels conflits se fait alors au niveau logique, sur les axiomes de l'ontologie.

Ce problème ne sera pas traité dans cette thèse, mais les travaux réalisés prennent en compte cette perspective.

3.1.2 Problèmes non fonctionnels

Nous présentons ici les principales contraintes, induites par le scénario et la stratégie choisis dans OMNIA, et qui doivent être considérées dans la plupart des choix techniques et architecturaux du projet.

Passage à l'échelle. La stratégie de recherche d'OMNIA nécessite au préalable l'indexation sémantique des documents. Comme nous visons la recherche au sein d'entrepôts de données en perpétuelle expansion, le processus d'indexation doit être capable de traiter un flux de données conséquent. Par exemple, environ 8000 nouvelles images sont ajoutées chaque jour sur Wikimedia ou sur Belga-News, soit environ 400 000 mots (ou 1600 pages standard de 250 mots) avec leurs textes compagnons.

Réactivité. Il est évident que la réactivité du système est un enjeu majeur pour le traitement des requêtes. Les moteurs de recherche sur le Web traitent les requêtes de façon quasi instantanée. Aussi, des utilisateurs habitués à cette rapidité ne se satisferont en aucun cas d'un système traitant leurs requêtes en plusieurs secondes.

Adaptabilité aux modifications et à l'ajout de ressources. Le système doit pouvoir prendre en compte facilement de nouvelles ressources ou des mises à jour de ces ressources (développement collaboratif des dictionnaires, ajout ou mise à jour d'ontologies, etc.).

3.2 Problèmes liés au TALN dans OMNIA

3.2.1 Recherche translingue d'information

Le contenu visuel des images, auquel on souhaite accéder via le système OMNIA, est indépendant de la langue utilisée dans les textes compagnons. Par exemple, une image de chien accompagnée d'une légende "dog", "perro" ou "cane" reste une image de chien (Popescu (2007)). Lors du processus de recherche, un utilisateur doit pouvoir formuler librement des requêtes, dans sa langue naturelle préférée. Il souhaite alors obtenir toutes les images pertinentes, même si les textes accompagnant ces dernières n'utilisent pas cette langue. Le traitement des textes dans OMNIA se place donc dans le cadre d'une recherche d'information translingue (*CLIR - Cross Lingual Information Retrieval*).

Position du problème. Le CLIR doit permettre de retrouver des objets (souvent des documents) dont les textes peuvent utiliser des langues différentes de celle(s) utilisée(s) pour formuler les requêtes. L'enjeu n'est cependant pas la traduction du contenu textuel pour le présenter à l'utilisateur. Dans ce domaine, la langue utilisée dans une requête est qualifiée de *langue source* et celle utilisée dans un document est dite *langue cible*.

Trois principales approches. On a identifié trois approches pour résoudre ce problème (Nasharuddin et Abdullah (2010)) :

1. traduction de la requête dans la (ou les) lange(s) cible(s).
2. traduction des documents dans la (ou les) langue(s) source(s).
3. traduction du document et de la requête dans une langue ou un formalisme commun.

Notre approche. Dans le cas d'OMNIA, le résultat de l'analyse des textes doit être fusionné avec les résultats de diverses analyses hétérogènes. Il a été choisi, dès la proposition de ce projet, de réaliser cette fusion au sein d'une ontologie partagée. Dès lors, la troisième solution est la plus viable, avec une ontologie comme formalisme commun.

Pour effectuer la "traduction" des textes dans le formalisme de l'ontologie, nous avons besoin de logiciels produisant, à partir de textes "tout venant" : (1) des descriptions dans l'ontologie

pour indexer les objets, et (2) des requêtes formelles à résoudre sur l'ontologie. Les notions de description et de requête pour une ontologie sont précisées au chapitre 2.

3.2.2 Extraction de contenu

L'analyse des textes dans OMNIA relève de l'extraction de contenu (EC). On cherche en effet à extraire, dans des textes "tout venant", des informations sémantiques structurées (descriptions ou requêtes par rapport à une ontologie).

Position du problème. L'extraction de contenu désigne les techniques permettant d'identifier et de collecter des informations pertinentes, sous forme normalisée, dans des objets pas ou peu structurés. Les travaux de cette thèse sont un peu plus larges, car ils concernent aussi l'extraction d'information dans des textes, qui part d'une EC et la complète lors du passage à une description "ontologique".

Les entrées d'un tel système sont :

1. Un modèle d'EC pouvant contenir un lexique spécialisé, une taxonomie, une ontologie, des règles d'extraction, etc.
2. Un ensemble de textes.

La sortie est un ensemble de données structurées dans une base de connaissances pouvant par exemple prendre la forme d'un tableau, de *templates*, ou plus récemment d'une ontologie. Ces données représentent en général un ensemble de concepts, de relations et d'attributs instanciés.

La notion de pertinence pour les données extraites dépend de la tâche, mais est implicitement contenue dans le modèle d'EC.

Défis. Sans aller dans le détail des modules que comportent traditionnellement les systèmes d'EC (Grishman (1997)), on peut isoler les principaux paramètres qui en régissent la conception et le fonctionnement :

1. le type des textes en entrée et leur langue (légende, journal, manuel technique, etc.),
2. le domaine des entrées (finance, tourisme, cinéma, etc.),
3. le scénario (recherche d'images, d'itinéraires, etc.),
4. le format de sortie (XML, formulaire, etc.).

Les systèmes d'EC sont en général développés pour des paramètres fixés. Leur portabilité reste délicate, malgré le développement de techniques (semi-)automatiques (Maedche et al. (2003)). Le cas d'OMNIA présente un défi, puisque plusieurs paramètres sont amenés à changer au cours de l'utilisation du système : le type, la langue, et le domaine des textes à traiter.

Notre approche. Dans la lignée de Yildiz et Miksch (2007), nous utilisons une ontologie comme paramètre pour le domaine des entrées. D'autre part, afin de générer des règles d'extraction indépendantes de la langue, nous proposons de passer par une représentation interlingue des textes. La construction d'une représentation "complète" (par exemple des arbres de dépendances pour chaque phrase) serait trop coûteuse dans ce contexte.

Nous nous contentons donc d'une annotation interlingue au niveau lexical ou sur de petits morceaux (*chunks*) de texte. Ainsi, pour développer un système d'extraction dans N langues, nous ne devons plus construire N systèmes de règles, mais un seul (agissant sur la représentation interlingue) combiné à N systèmes réutilisables pour l'analyse de surface et l'annotation des textes.

3.2.3 Désambiguïsation

Dans une approche de RI par "sacs de mots", les mots sont considérés comme de simples chaînes et sont désambiguïsés implicitement par leurs contextes. Une des spécificités de notre approche sémantique est de rendre explicites les ambiguïtés présentes dans un texte lors de l'étape d'annotation. Des procédés de désambiguïsation sont donc nécessaires pour le traitement des textes accompagnant les images et des requêtes.

Position du problème. Des problèmes de désambiguïsation interviennent à deux endroits dans le système :

1. Lors de l'annotation des textes, les lexèmes interlingues utilisés sont identifiables à des sens de mots. Trois sous-tâches sont à traiter :
   - la segmentation (recherche de multimots),
   - la désambiguïsation des sens de mots (*Word Sense Disambiguation, WSD*),
   - la recherche du meilleur chemin interprétatif possible.
2. Lors de l'extraction de contenu sur les textes annotés, il faut alors déterminer quelles acceptions (lexèmes interlinges) dénotent un concept (ou un attribut) donné.

La seconde tâche ne relève pas exactement de la WSD, mais s'y apparente fortement. En effet, la WSD permet de passer d'un mot (chaîne de caractères) à un sens de mot (entrée d'un lexique), alors qu'ici nous cherchons à passer d'un sens de mot (dans le lexique pivot) à un concept (dans l'ontologie).

Cette tâche ne s'effectue pas à la volée. On calcule un alignement lorsque l'ontologie est passée en paramètre au système. Les problèmes liés au calcul d'un tel alignement sont traités dans le chapitre 3.

État de l'art : Une revue complète des méthodes existantes pour la désambiguïsation sort du cadre de cette thèse. Le lecteur pourra se reporter à Ide et Véronis (1998) pour les travaux jusqu'en 1998 puis à Agirre et Edmonds (2006) ou Navigli (2009) pour un état de l'art plus complet. Schwab (2005) présente également une approche mixte originale (thématique et lexicale) utilisant des vecteurs conceptuels.

Défis. Lors de la phase d'annotation, toutes les interprétations possibles pour une phrase sont conservées dans une structure de graphe. On observe alors une explosion combinatoire du nombre des différents chemins interprétatifs possibles. Vu le nombre de textes à traiter lors de la phase d'indexation des images et le besoin de réactivité pour le traitement des requêtes à la volée, la rapidité et la capacité de passage à l'échelle des techniques de désambibuïsation utilisées est cruciale.

Notre approche. Nous mettons en œuvre diverses méthodes classiques de désambiguïsation lexicale à l'aide d'algorithmes à colonies de fourmis pour garantir leur passage à l'échelle (Schwab et al. (2011)).

3.3 Problèmes traités dans la thèse

Dans cette thèse, nous nous attaquons à trois sous-problèmes liés à la réalisation de systèmes d'extraction de contenu ou de recherche d'information, guidés par des ontologies, dans des textes multilingues. Un chapitre est dédié à chacun d'entre eux.

3.3.1 Utilisation d'ontologie pour l'indexation et la recherche à partir de textes spontanés

Nous commençons par définir précisément les objets que nous appelons ontologies. La définition proposée s'inspire de celles trouvées dans la littérature pour les ontologies basées sur des logiques de description. Nous nous efforçons de distinguer le niveau logique et le niveau terminologique de telles structures pour une utilisation en TALN.

À des fins de recherche et d'indexation d'objets multimédia, nous proposons une notion de description qui étend la notion de descripteur rencontrée traditionnellement en recherche documentaire.

Nous étudions par ailleurs les contraintes de complexité suffisantes pour garantir le passage à l'échelle des tâches de raisonnement sur des structures contenant les descriptions de millions d'objets. Enfin, des scores de confiance et d'intensité apparaissent naturellement lors du traitement de textes libres ; leur prise en compte dans la construction des requêtes et des descriptions est brièvement discutée.

3.3.2 Multilinguïsation d'ontologies

Dans le chapitre 3, nous nous intéressons au problème de l'utilisation d'ontologies en contexte multilingue. Nous commençons par aborder le problème de l'enrichissement d'ontologies avec des informations linguistiques. Une revue de l'état de l'art sur cette question, y compris en contexte monolingue, nous mène vers une solution de lexicalisation externe. Elle consiste à aligner des ontologies avec des ressources lexicales existantes exprimées dans des formats standard du Web sémantique.

Nous identifions ensuite des problèmes importants pour la création d'alignement entre des ressources ontologiques et lexicales. Nous isolons également un certain nombre de propriétés structurelles des alignements, utilisables pour la résolution automatique de ces problèmes.

Nous attaquons enfin le problème de la multilinguisation d'ontologies dans le cadre d'une lexicalisation externe. La solution qui nous a paru la plus pertinente, compte tenu des contraintes du projet OMNIA, est d'aligner l'ontologie considérée avec un lexique pivot (composé de lexèmes interlingues), en considérant ce dernier comme une "préontologie". Cela nous a conduit à adopter et adapter des techniques d'alignement d'ontologies.

3.3.3 Architecture multilingue pour l'annotation des textes et l'extraction de contenu guidées par une ontologie

Le dernier chapitre est essentiellement dédié à la présentation du système de démonstration réalisé dans le cadre du projet OMNIA pour l'indexation et la recherche translingue d'images accompagnées de textes. L'architecture est basée sur un processus d'extraction de contenu dans les textes accompagnant les images et les requêtes. Quelle que soit la langue des textes en entrée, la première étape est leur annotation interlingue dans une structure de graphe. L'extraction de contenu proprement dite est ainsi "factorisée" au niveau interlingue, et une ontologie peut être passée en paramètre du système par le calcul automatique d'un alignement avec le lexique interlingue.

Le démonstrateur est implémenté de façon modulaire avec une architecture à services. Il permet l'application d'un certain nombre de propositions faites dans cette thèse et offre de bonnes perspectives d'expérimentation, puisque chaque service peut être amélioré indépendamment.

Chapitre 2

Ontologies pour l'extraction de contenu et la recherche d'information

Sommaire

Introduction

La représentation des connaissances est un enjeu majeur pour les systèmes informatiques modernes relevant de l'intelligence artificielle. Des systèmes capables de raisonner à partir d'un ensemble de connaissances formalisées ont été développés depuis les années 1970. Par exemple, le système MYCIN (Shortliffe (1976)) permettait de réaliser des diagnostics médicaux à partir d'une description du domaine sous forme de règles. Au début des années 1980, le terme d'ontologie, emprunté à la philosophie, a commencé a être utilisé pour dénoter des connaissances formalisées sur le monde et utilisées dans les systèmes d'informations.

On observe alors le lancement de projets informatiques, toujours en cours à l'heure actuelle, visant à une description globale du monde, non réduite à un domaine de connaissances particulier. Ainsi, le projet CYC (Lenat et Guha (1989)) a débuté en 198O avec l'ambition de construire une base de connaissances, utilisable par des machines, et rassemblant les informations utilisées par les humains dans les raisonnements du "sens commun".

En 1985, le projet de science cognitives Wordnet (Fellbaum (1998)) a démarré avec comme objectif scientifique de produire une représentation formelle de l'organisation des mots dans l'esprit humain, et comme objectif pratique de fournir une "base lexicale" utilisable par des anglophones, comme une sorte de dictionnaire sémantique. Par contre, l'objectif de Wordnet n'était pas de servir à des applications de TALN (comme la correction, la traduction, etc.).

On peut par contraste citer le projet HowNet du Pr. Dong Zhendong [11]. Ce spécialiste de sémantique et de traduction automatique a développé HowNet (100 000 concepts, 120 000 entrées en chinois, près de 100 000 entrées en anglais) sur une base sémantique formelle, très proche d'une ontologie, pour servir de base au TALN. La base "conceptuelle" permet des applications bilingues, ainsi que l'extension à d'autres langues de façon modulaire.

Le foisonnement d'informations disponibles sur le Web rend nécessaire l'utilisation de processus "intelligents" pour y accéder de façon pertinente. Or, si les données du Web sont physiquement accessibles par des logiciels, leur organisation et leur sens sont *a priori* accessibles aux seuls êtres humains. Dans son intervention lors de la conférence WWW 94, Tim Berners-Lee, souvent considéré comme inventeur du Web, propose l'utilisation de métadonnées formalisant le contenu des documents Web pour résoudre ce problème. Le format RDF [12] fournit un standard pour l'expression de ces métadonnées et le développement de vocabulaires contrôlés pour les annotations. Dans le prolongement, RDFS [13] permet de structurer ces vocabulaires au sein de schémas partagés, dédiés à certains domaines de connaissances. Par exemple, le projet FOAF [14] fournit un vocabulaire RDF et un schéma RDFS permettant de décrire les relations entre personnes (comme "Dupont emploie Durand"). On retrouve alors, dans un contexte nouveau, le paradigme d'ontologie permettant la création d'un Web *sémantique* au sein duquel le sens et l'organisation des données sont exploitables par les agents logiciels.

Les ontologies, communément qualifiées de "*spécifications explicites et formelles d'un domaine de connaissance*" (Gruber (1993)), formalisent des schémas partagés pour la description d'informations, de données ou de services. Cette notion est très ouverte, et une réelle controverse existe quant aux systèmes de représentation de connaissances que l'on peut "enrôler" sous la bannière d'ontologie. Les recommandations du W3C [15] pour le développement du Web sémantique tendent cependant à généraliser l'utilisation d'ontologies basées sur des théories logiques, notamment les

11. www.keenage.com
12. www.w3.org/RDF/
13. www.w3.org/TR/rdf-schema/
14. www.foaf-project.org/
15. www.w3.org

logiques de description (Baader et al. (2003)).

Nous nous concentrerons sur ces ontologies "logiques" et sur leurs applications dans le cadre de la recherche d'information et de l'indexation d'objets multimédia. Woods (1997) démontre que l'utilisation de taxonomies de concepts (assimilables à des ontologies) pour l'indexation conceptuelle de documents (les manunels UNIX dans ses premières expériences) améliore sensiblement les résultats de la recherche d'informations. Les applications de recherche d'information et d'indexation qui utilisent des ontologies sont qualifiées de sémantiques. Elles sont la prolongation des méthodes basées sur des ressources terminologiques structurées. Les apports principaux de ces méthodes sémantiques sont les possibilités : (1) de réaliser des inférences, et (2) de produire des descriptions unifiées au sein de l'ontologie à partir d'analyses hétérogènes. Par ailleurs, les processus automatiques pour l'indexation ou l'annotation de documents avec des ontologies sont un apport important pour l'intégration de documents au sein du Web Sémantique (Handschuh et Staab (2003)).

Dans cette partie, nous commençons par donner les définitions utiles pour notre étude. Une notion de description dans une ontologie est proposée. Nous décrivons ensuite les outils utilisés (éditeurs, raisonneurs, etc.) et l'impact de l'expressivité des langages d'ontologies sur les systèmes. Enfin, nous présentons l'ontologie qui a été réalisée et l'usage qui en a été fait dans le cadre du projet OMNIA.

Les définitions de cette partie sont illustrées à l'aide de l'ontologie utilisée dans la tâche de détection de concepts au sein d'images lors de la campagne CLEF09 (Nowak et Dunker (2010)).

1 Définitions formelles

1.1 Ontologie abstraite, instanciation, lexique

Si un consensus informel a été atteint sur ce que doit contenir une ontologie informatique (hiérarchie de concepts, attributs, relations, axiomes logiques et possibilité d'inférence), on trouve quantité de définitions formelles concurrentes. Nous en proposons une adaptée de (Maedche et al. (2003)), et suffisamment générale pour couvrir l'ensemble des ontologies "logiques", en particulier l'ensemble des ontologies exprimées en OWL [16]. La définition est composée de trois parties :

1. *une ontologie abstraite* concerne la structure, c'est à dire l'organisation des concepts et des relations ;
2. *une instanciation* concerne les individus que l'on souhaite décrire à l'aide de l'ontologie abstraite ;
3. *un lexique* est un ensemble de symboles qui désignent les objets de l'ontologie dans une langue naturelle.

Définition 1. Une **ontologie abstraite** est une structure $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \sigma, \leq_{\mathcal{C}}, \leq_{\mathcal{R}}, \mathcal{L}, \mathcal{T})$, où :
- $\mathcal{C}$ et $\mathcal{R}$ sont respectivement des ensembles disjoints de concepts (ou classes) et de relations (ou rôles).
- $\sigma : \mathcal{R} \to \mathcal{C} \times \mathcal{C}$ est une fonction qui fournit le support d'une relation.
- $\leq_{\mathcal{C}}$ et $\leq_{\mathcal{R}}$ sont des ordres partiels sur $\mathcal{C}$ et $\mathcal{R}$, respectivement.
- $\top \in \mathcal{C}$ et $\bot \in \mathcal{C}$ sont des bornes pour $\leq_{\mathcal{C}}$, respectivement supérieure et inférieure, telles que cet ordre détermine une structure de treillis sur l'ensemble des concepts.

16. http://www.w3.org/2004/OWL/

- $\mathcal{L}$ est une théorie logique, dotée d'une sémantique formelle, dont la signature contient les constantes de $\mathcal{C}$, $\mathcal{R}$, les ordres $\leq_{\mathcal{C}}$ et $\mathcal{R}$, ainsi que σ (voir paragraphe 1.3 pour une définition de signature). $\mathcal{L}$ est un ensemble de formules, le langage de l'ontologie.
- $\mathcal{T}$ est un ensemble d'axiomes exprimés dans la logique $\mathcal{L}$, appelé T-box (*Terminological Box*).

Les concepts et relations de $\mathcal{C}$ et $\mathcal{R}$ sont dits *atomiques* et peuvent être combinés grâce aux constructeurs de la logique $\mathcal{L}$ pour obtenir des concepts composés ou des relations composées ($\mathcal{L}$-concepts et $\mathcal{L}$-relations, notés $\mathcal{L}(\mathcal{C})$ et $\mathcal{L}(\mathcal{R})$ respectivement).

L'appellation T-box (Terminological box) vient des Logiques de Description (DL en anglais). La T-box est destinée à décrire la partie terminologique de la base de connaissances, mais ne forme cependant pas nécessairement une terminologie (ensemble de termes non ambigus utilisés dans un domaine). En effet, les étiquettes utilisées pour nommer les concepts et relations relèvent souvent plus d'identifiants informatiques que de termes bien formés dans une langue.

Exemple 1. Une ontologie, organisant des concepts à retrouver dans une banque d'images, a été fournie aux participants lors de la campagne CLEF09. Cette ontologie comporte 80 concepts, organisés dans un treillis ayant jusqu'à cinq niveaux de profondeur. Une vingtaine de relations sont déclarées. Les figures 1 et 2 contiennent des extraits des treillis formés par l'ordre partiel de spécialisation (is-a) sur les concepts et les relations.

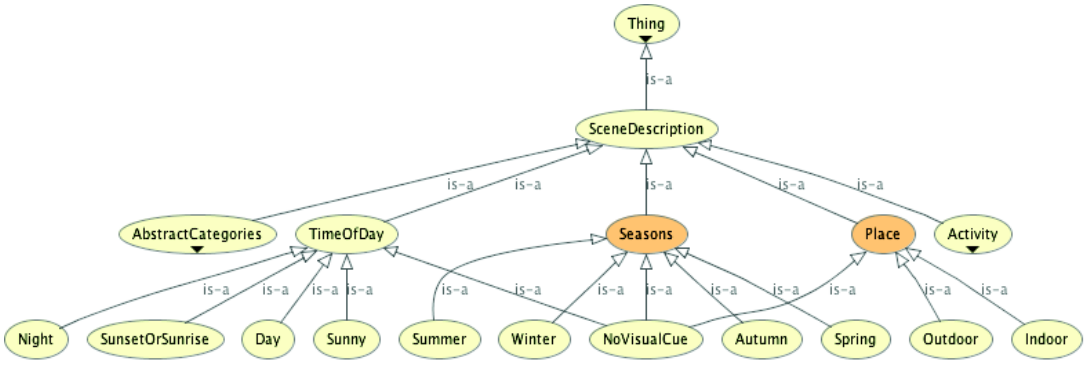


FIGURE 1 – Extrait du treillis conceptuel de l'ontologie CLEF09

La T-box comporte une centaine d'axiomes, exprimables dans une logique de description dont la sémantique sera donnée par la suite. Les axiomes logiques ont plusieurs utilités pour l'indexation ou la recherche d'information.

- Certains permettent d'enrichir les descriptions au sein de l'ontologie et améliorent le rappel lors d'une recherche. Par exemple, $Portrait \sqsubseteq \exists hasPersons \sqcup \exists hasAnimals$ [17] exprime qu'au moins une personne ou un animal figure nécessairement sur une photo de type "Portrait".
- D'autres permettent de vérifier la consistance des informations utilisées pour l'indexation et améliorent la précision des indexations. Par exemple, $Indoor \sqcap Outdoor = \bot$ exprime que les photos d'intérieur et d'extérieur forment deux ensembles disjoints.

17. Cette formule en DL peut se traduire $(\forall x)\,[Portrait(x) \Rightarrow (\exists y)\,[hasPersons(x,y) \vee hasAnimals(x,y)]]$ dans le calcul des prédicats du premier ordre (CP1).

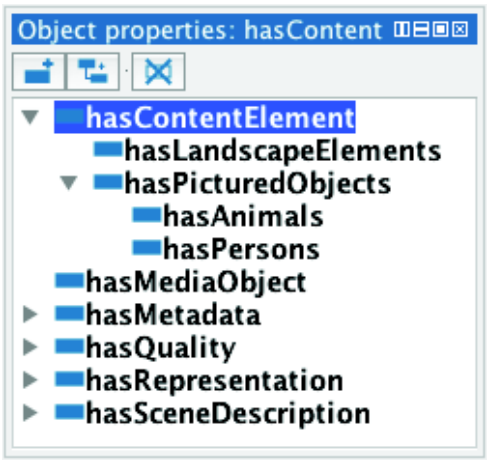


FIGURE 2 – Extrait du treillis de relations de l'ontologie CLEF09

Définition 2. Une **instanciation** d'une ontologie abstraite $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \sigma, \leq_{\mathcal{C}}, \leq_{\mathcal{R}}, \mathcal{L}, \mathcal{T})$ est une structure $\mathcal{I}nst = (\mathcal{E}, \mathcal{A})$, où :

– $\mathcal{E}$ est un ensemble d'individus (ou instances).

– $\mathcal{A}$ est un ensemble d'axiomes exprimés dans la logique $\mathcal{L}$, appelé A-box.

L'ontologie instanciée, également appelée base de connaissances, est notée :
$\mathcal{O}_{\mathcal{I}nst} = (\mathcal{C}, \mathcal{E}, \mathcal{R}, \sigma, \leq_{\mathcal{C}}, \leq_{\mathcal{R}}, \mathcal{L}, \mathcal{T}, \mathcal{A})$.

L'appellation A-box (Assertional box) vient encore des logiques de description. La A-box décrit les individus d'un domaine relativement à une ontologie abstraite. Il est possible de déclarer plusieurs instanciations d'une ontologie abstraite.

Exemple 2. Afin de décrire la photo de la figure ci-contre, nous pouvons faire les déclarations suivantes (voir figure 2) dans la A-box de l'ontologie CLEF.

FIGURE 3 – Belga1050730

```
Photo(Belga1050730.jpg)
hasID(Belga1050730.jpg,Belga1050730)
Portrait(Belga1050730.jpg)
Indoor(Belga1050730.jpg)
Person(SaddamHussein)
hasPerson(Belga1050730.jpg,SaddamHussein)
```

Ces axiomes expriment que l'objet `Belga1050730.jpg` est une photo, de type portrait, dont l'identifiant est `Belga1050730`, que cette photo a été prise en intérieur, et qu'y figure une personne identifiée comme $SaddamHussein$.

Si l'on souhaite utiliser une ontologie pour décrire et gérer une grande masse de données, la A-box doit être stockée et manipulée dans une base de données séparée de la base de connaissances terminologiques (ontologie abstraite), comme proposé par Calvanese (2005).

Le terme ontologie dénotera, selon le contexte, une ontologie abstraite ou une base de connaissances. Par souci de concision, les ontologies abstraites seront notées $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{T})$ et les bases de connaissances $\mathcal{O} = (\mathcal{C}, \mathcal{E}, \mathcal{R}, \mathcal{L}, \mathcal{T}, \mathcal{A})$.

Deux hypothèses sont faites sur la description du monde par une ontologie :

1. *l'hypothèse de nom unique*, qui signifie qu'un nom d'objet désigne toujours la même entité.
2. *l'hypothèse du monde ouvert*, qui signifie que l'on ne suppose jamais que le monde est complètement décrit.

Ce dernier postulat s'oppose à *l'hypothèse du monde clos*, utilisée dans les bases de données ou dans Prolog. Dans un monde clos, on considère que tout ce qui n'est pas explicitement spécifié ou déductible en un temps fini est faux (négation par l'échec).

Définition 3. Un **lexique** pour une ontologie $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ est une structure $\mathcal{L}ex = (S_{\mathcal{L}ex}, Ref)$ où :

- $S_{\mathcal{L}ex}$ est un ensemble de symboles représentant les entrées d'un lexique (mots ou sens de mots),
- $Ref \subseteq (\mathcal{C} \cup \mathcal{R} \cup \mathcal{E}) \times S_{\mathcal{L}ex}$ est un ensemble de liens appelés affectations lexicales.

Nous verrons au chapitre 3 que les symboles du lexique peuvent être liés par des relations (hyponymie, synonymie, flexion, etc.). Nous proposerons, dans la section 3.2.2, de faire intervenir des liens structurels entre ces relations du lexique et celles de l'ontologie.

Les affectations lexicales ne sont pas des bijections, ni même des fonctions, mais seulement des relations. Le fait que plusieurs termes puissent renvoyer au même concept permet de rendre compte de la synonymie au sein d'une langue naturelle ; le fait que plusieurs concepts puissent être désignés par le même terme permet de rendre compte de la polysémie. Pour que les symboles du lexique représentent les concepts de façon non ambiguë (et forment ainsi une terminologie), il faut imposer que les affectations lexicales soient des fonctions de l'ensemble des symboles lexicaux dans l'ensemble des concepts.

Définition 4. Un **langage de règles** pour une ontologie $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ est un ensemble de formules des la forme $antecedant \longrightarrow consequence$ telles que $antecedent \in \mathcal{L}$ et $consequence \in \mathcal{L}$.

La sémantique de telles règles est donnée au paragraphe suivant.

Exemple 3. L'axiome proposé dans l'exemple 1 peut également être exprimé à l'aide de la règle $Portrait \longrightarrow \exists hasPersons \sqcup \exists hasAnimals$.

1.2 Sémantique (interprétations et modèles)

Les ontologies (abstraites et instanciées) ne sont rien d'autre que des théories logiques un peu contraintes. On peut alors définir une sémantique formelle pour les ontologies. Cette sémantique est essentielle, car elle fournit les mécanismes d'inférence (d'implication logique) qui permettent de raisonner et de répondre à des requêtes à partir d'une ontologie. Les notions d'interprétation et de modèle sont définies suivant l'usage en logique. On pourra consulter (Doets (1996)) pour une introduction ou (Tarski (1983)) pour une version anglaise de l'article fondateur de la notion de modèle.

Définition 5. Une **interprétation** d'une ontologie $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ est un couple $\left(\Delta^{\mathcal{I}}, .^{\mathcal{I}}\right)$, où $\Delta^{\mathcal{I}}$ est un ensemble non vide appelé domaine d'interprétation, et $.^{\mathcal{I}}$ est une fonction d'interprétation qui associe :

- à chaque concept atomique $C \in \mathcal{C}$ un sous-ensemble $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, où $(C_1 \times C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \times C_2^{\mathcal{I}}$ (pour des concepts C_1 et C_2) et $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$,
- à chaque individu $i \in \mathcal{E}$ un élément $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- à chaque relation atomique $R \in \mathcal{R}$ une relation $R^{\mathcal{I}}$ sur $\Delta^{\mathcal{I}}$ conformément à son support (c'est à dire un sous-ensemble de $\sigma(R)^{\mathcal{I}}$), avec la contrainte que $\leq_{\mathcal{C}}$ et $\leq_{\mathcal{R}}$ soient interprétés comme l'inclusion ensembliste ($(\leq_{\mathcal{C}})^{\mathcal{I}} = (\leq_{\mathcal{R}})^{\mathcal{I}} = \subseteq$).

L'interprétation des concepts et relations composés, ainsi que celle des axiomes de la A-box et de la T-box, définis en utilisant les opérateurs de la logique $\mathcal{L}$, est calculée à partir des interprétations "atomiques" en utilisant la sémantique formelle de $\mathcal{L}$.

Exemple 4. Les interprétations des formules de la logique de description $\mathcal{ALC}$, utilisée dans les exemples précédents, sont données dans la table 1.

Constructeur	**Syntaxe**	**Sémantique**
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Negation	$\neg C$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$
Conjonction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjonction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Restriction existentielle	$\exists r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}}\}$
Restriction universelle	$\forall r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e \in C^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \implies e \in C^{\mathcal{I}}\}$
Subsomption de concepts	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 1 – Interprétation des constructeurs de la logique de description $\mathcal{ALC}$

La logique de description $\mathcal{ALC}$ peut être étendue avec des constructeurs comme ceux de la table 2. Le nom de la logique ainsi obtenue est formé à l'aide des identifiants usuels pour les constructeurs ajoutés. Par exemple, si l'on autorise les énumérations dans $\mathcal{ALC}$, on obtient la logique $\mathcal{ALCO}$.

Identifiant	**Constructeur**	**Syntaxe**	**Sémantique**
$\mathcal{Q}$	Restrictions quantitatives	$(\leq nrC)$ $(\geq nrC)$	$\{d \mid \#\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \leq n\}$ $\{d \mid \#\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \geq n\}$
$\mathcal{O}$	Énumérations	$\{a\}$	$\{a^{\mathcal{I}}\}$
$\mathcal{I}$	Inverse	r^-	$(r^{\mathcal{I}})^-$
$\mathcal{U}$	Rôle universel	u	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$\mathcal{H}$	Inclusion de rôles	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Table 2 – Extensions pour logiques de description

Définition 6 (Satisfaction).

- On dit qu'une interprétation $\left(\Delta^{\mathcal{I}}, .^{\mathcal{I}}\right)$ **satisfait une formule** $F \in \mathcal{L}$ ssi $F^{\mathcal{I}}$ est vraie sur $\Delta^{\mathcal{I}}$. On dit aussi que $\left(\Delta^{\mathcal{I}}, .^{\mathcal{I}}\right)$ est un *modèle* de F.

– Une interprétation $\left(\Delta^{\mathcal{I}}, .^{\mathcal{I}}\right)$ **satisfait une règle** *antecedant* $\longrightarrow$ *consequence* si elle vérifie la condition suivante :
Si $\left(\Delta^{\mathcal{I}}, .^{\mathcal{I}}\right)$ est un modèle de la formule *antecedant*, alors c'est aussi un modèle de la formule *consequence*.

Définition 7. Un **modèle** d'une ontologie $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ est une interprétation qui satisfait tous les axiomes de la T-box et de la A-box.

Définition 8 (Consistance). Une ontologie est dite **consistante** si elle admet un modèle.

Comme nous le verrons dans le paragraphe 2.3, il est d'usage de fixer un domaine d'interprétation pour une ontologie, ce qui rend sa sémantique "plus concrète". Une ontologie peut alors former une théorie logique consistante (car elle admet un modèle), mais être inconsistante au regard du domaine d'interprétation qui lui est associé.

Définition 9 (Conséquence logique). Une ontologie $\mathcal{O}$ entraîne logiquement une formule F ssi tout modèle de $\mathcal{O}$ est un modèle de F. On note alors $\mathcal{O} \models F$.

Nous n'avons défini que peu de choses au niveau syntaxique et sémantique. Une ontologie profite en effet simplement de la syntaxe et de la sémantique de sa logique associée. Nous avons juste mis en avant quelques particularités que toute théorie logique, vue comme une ontologie, doit comporter.

Ces particularités sont notamment :
– les ordres partiels de spécialisation, à interpréter comme l'inclusion ensembliste,
– la séparation entre les axiomes de la théorie selon qu'ils décrivent la connaissance au niveau intensionnel (T-box) ou extensionnel (A-box),
– l'affectation d'un lexique à l'ontologie, précisant sa valeur terminologique.

1.3 Requêtes

Un langage de requêtes détermine les moyens à disposition pour exploiter l'information contenue dans une ontologie. On peut alors voir une ontologie comme une simple boîte noire permettant de répondre à des requêtes.

Nous commençons par définir la notion de signature.

Définition 10 (Signature). Soit $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ une ontologie.
– Une signature est un ensemble de symboles $\mathcal{S} \subseteq \mathcal{C} \cup \mathcal{R} \cup \mathcal{E}$.
– La signature $sig(\varphi)$ d'une formule $\varphi \in \mathcal{L}$ est l'ensemble des symboles de $\mathcal{C} \cup \mathcal{R} \cup \mathcal{E}$ intervenant dans φ (en toute rigueur, il faudrait définir la signature de façon récursive pour tous les constructeurs de $\mathcal{L}$).

Définition 11. Un **langage de requêtes** est un ensemble $\mathcal{QL}$ (éventuellement infini) de formules ouvertes (pouvant contenir des variables libres) dans un langage logique. On note $q = \{\overrightarrow{x} | \varphi(\overrightarrow{x})\}$ une requête, où $\varphi(\overrightarrow{x}) \in \mathcal{QL}$ est une formule dont les variables libres sont rassemblées dans le vecteur $\overrightarrow{x}$.

La taille de $\overrightarrow{x}$ est appelée l'arité de la requête. La signature de q est définie comme égale à la signature de φ. Étant donné $\Delta^{\mathcal{I}}$ un domaine d'interprétation, on note $q^{\mathcal{I}}$ l'ensemble des vecteurs $\overrightarrow{a}$ d'éléments de $\Delta^{\mathcal{I}}$ tels que $\varphi^{\mathcal{I}}(\overrightarrow{a})$ soit vraie.

Définition 12 (Requête, Réponse). Soit $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ une base de connaissances.
- Une requête q sur la base de connaissances $\mathcal{O}$ est une requête telle que $sig(q) \subseteq sig(\mathcal{O})$.
- Une réponse à la requête q sur $\mathcal{O}$ est un ensemble $ans(q, \mathcal{O})$ de vecteurs $\overrightarrow{a}$ d'éléments de $\mathcal{O}$ (concepts, relations ou instances) tels que $\overrightarrow{a}^{\mathcal{M}} \in q^{\mathcal{M}}$ pour tout modèle $\mathcal{M}$ de $\mathcal{O}$, c'est à dire que $ans(q, \mathcal{O}) = \{\overrightarrow{a} | \mathcal{O} \models \varphi(\overrightarrow{a})\}$.

Exemple 5. La requête $q = \{x | Photo(x) \wedge hasPersons(x, SadamHussein)\}$ a pour réponse l'ensemble des photos représentant $SaddamHussein$.

Définition 13. Une **requête booléenne** est une requête qui ne fait pas intervenir de variables libres, c'est à dire une requête d'arité 0.

Soit $q = \{\overrightarrow{x} | \varphi(\overrightarrow{x}\}$ une requête booléenne. q est vérifiée par une base de connaissances $\mathcal{O}$ si $ans(q, \mathcal{O})$ contient (seulement) le vecteur vide, c'est à dire si $\mathcal{O} \models \varphi$. La requête n'est pas vérifiée si $ans(q, \mathcal{O}) = \emptyset$. On notera simplement $\varphi \in \mathcal{QL}$ les requêtes booléennes.

Dans un univers fini, il est possible de réduire la résolution d'une requête arbitraire à la vérification d'une requête booléenne. En effet, soit $q = \{(x_1, \ldots, x_n) | \varphi(x_1, \ldots, x_n)\}$ une requête sur une ontologie $\mathcal{O}$. Il est possible de calculer $ans(q, \mathcal{O})$ en évaluant les requêtes booléennes $\varphi(a_1, \ldots, a_n)$ pour tous les n-uplets possibles $(a_1, \ldots, a_n)$ d'éléments de $\mathcal{O}$. Cela revient au pire à $card(\mathcal{O})^n$ évaluations de requêtes booléennes où $card(\mathcal{O})$ est le nombre de concepts, relations et instances intervenant dans $\mathcal{O}$. Dans les définitions suivantes, il nous suffira donc de considérer des requêtes booléennes.

2 Descriptions dans une ontologie

À plusieurs reprises dans le chapitre 1, nous avons parlé de description (ou descripteur) d'un objet ou d'un document dans une ontologie. Cette notion mérite d'être précisée. En recherche documentaire classique, un document est caractérisé par un ensemble de descripteurs, c'est à dire un ensemble de "*termes, simples ou composés, extrait d'un langage contrôlé (tel qu'un thésaurus) pour représenter sans ambiguïté, au cours de l'opération d'indexation, une notion contenue dans un document ou dans une recherche documentaire.*" (Cacaly et al. (2008)).

L'utilisation d'ontologies prolonge cette approche basée sur l'utilisation de ressources terminologiques ; en effet, le treillis conceptuel d'une ontologie (ou plutôt le lexique associé) a une valeur terminologique et peut définir un ensemble satisfaisant de descripteurs. Cependant, si le treillis conceptuel constitue une sorte de squelette pour une ontologie, cette dernière peut contenir bien plus d'informations sous forme d'axiomes logiques. C'est pourquoi, afin d'utiliser une ontologie pour l'indexation d'objets multimédia, il est nécessaire de définir une notion de *description* qui étend celle de descripteur.

En outre, la notion de description dans une ontologie peut être appliquée pour effectuer des optimisations lors de l'utilisation d'ontologies pour l'indexation et la recherche documentaire. En effet, lors du processus d'indexation, on peut tirer profit d'une T-box de très grande taille pour réaliser une indexation riche. En revanche, lors de la recherche des documents indexés, la taille de la T-box influera de façon négative sur la réactivité du système pour fournir des résultats. Il peut donc être intéressant d'utiliser une grande T-box lors de l'indexation, et de n'en conserver que le nécessaire lors de la recherche, c'est à dire de ne conserver que les *descriptions* des objets indexés.

Nous proposons dans la suite une notion de description dans une ontologie.

2.1 Inséparabilité, extensions conservatives, modules

Certaines propriétés que nous avons identifiées pour les *descriptions* dans une ontologie reposent sur les notions logiques d'inséparabilité, d'extension conservative, et de module présentées par Konev et al. (2009). Intuitivement, deux théories logiques (ou ontologies) sont inséparables si elles donnent les mêmes réponses à un ensemble arbitraire de requêtes.

Définition 14 (QL-inséparabilité, QL-module). Soient $\mathcal{O}_1$ et $\mathcal{O}_2$ des ontologies, et $\mathcal{QL}$ un langage de requêtes.

- $\mathcal{O}_1$ et $\mathcal{O}_2$ sont ($\mathcal{QL}$)-**inséparables** ssi, pour toute requête φ de $\mathcal{QL}$, $\mathcal{O}_1 \models \varphi$ ssi $\mathcal{O}_2 \models \varphi$. On note alors $\mathcal{O}_1 \approx^{\mathcal{QL}} \mathcal{O}_2$.
- $\mathcal{O}_2$ est une extension ($\mathcal{QL}$)-**conservative** de $\mathcal{O}_1$ si $\mathcal{O}_1 \subseteq \mathcal{O}_2$ et si $\mathcal{O}_1$ et $\mathcal{O}_2$ sont $\mathcal{QL}$-inséparables. On dit alors que $\mathcal{O}_1$ est un $\mathcal{QL}$-module de $\mathcal{O}_2$.

Les définitions proposées par Konev et al. (2009) font intervenir une signature en plus du langage de requêtes. Nous considérons ici que la signature est contrainte par le language de requêtes. Plus précisément, étant donné $\mathcal{QL}$ un langage de requêtes et S une signature, nous considérons le langage de requêtes $\mathcal{QL}' = \{\varphi \in \mathcal{QL} | sig(\varphi) \subseteq S\}$.

2.2 Descriptions

Intuitivement, une description d'un objet est un ensemble d'informations qui permettent de répondre à des questions sur cet objet. La définition suivante formalise cette intuition dans le contexte des ontologies.

Définition 15 (QL-description). Soient $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{A})$ une ontologie, $\mathcal{QL}$ un langage de requêtes, et un élément $a \in sig(\mathcal{O}) \cap sig(\mathcal{QL})$.
Une *$\mathcal{QL}$-description* de a dans $\mathcal{O}$ est un ensemble d'axiomes $\mathcal{O}_a \subseteq \mathcal{T} \cup \mathcal{A}$, tel qu'il existe une requête (non tautologique) $\varphi \in \mathcal{QL}$, telle que $a \in sig(\varphi)$ et $\mathcal{O}_a \models \varphi$

Il est important de savoir si la description que l'on extrait contient toute l'information disponible et utilisable. Cela conduit à la notion de description complète. Intuitivement, une description d'un objet est complète si elle fournit autant d'information sur l'objet que l'intégralité de la base de connaissances.

Soient $\mathcal{O}$ une ontologie et $\mathcal{QL}$ un langage de requêtes.

Définition 16 (Description complète). Soit $a \in sig(\mathcal{O}) \cap sig(\mathcal{QL})$ et $\mathcal{O}_a$ une $\mathcal{QL}$-description de a.
On note $\mathcal{QL}_a = \{\varphi \in \mathcal{QL} | a \in sig(\varphi)\}$ toutes les requêtes "parlant de a".
La description $\mathcal{O}_a$ est dite **complète** si $\mathcal{O}_a$ est un $\mathcal{QL}_a$-module de $\mathcal{O}$.

On peut étendre cette définition à un ensemble d'objets.

Définition 17 (Description complète d'un ensemble d'objets). Soient $a_1, \ldots, a_n \in sig(\mathcal{O}) \cap sig(\mathcal{QL})$.
On note $\mathcal{QL}_{a_1,\ldots,a_n} = \{\varphi \in \mathcal{QL} | \{a_1, \ldots, a_n\} \cap sig(\varphi) \neq \emptyset\}$.
Une description complète de $a_1, \ldots, a_n$ est un $\mathcal{QL}_{a_1,\ldots,a_n}$-module de $\mathcal{O}$.

Si l'on souhaite extraire une description d'objets pour la réutiliser, on doit déterminer si elle ne contient pas d'information superflue (l'ontologie tout entière est toujours une description

complète de chacun de ses éléments). Cela conduit à la notion de *description minimale*. Intuitivement, une description est minimale si, dès qu'on lui retire un axiome, elle permet de répondre à moins de requêtes.

Définition 18 (Description minimale). Une $\mathcal{QL}$-description $\mathcal{O}_{a_1,\ldots,a_n}$ d'un ensemble d'objets $\{a_1,\ldots,a_n\}$ est dite **minimale** si :
pour tout ensemble d'axiomes $\mathcal{O}'_a \subsetneq \mathcal{O}_a$, il existe une requête $\varphi \in \mathcal{QL}$, telle que $\{a_1,\ldots,a_n\} \cap sig(\varphi) \neq \emptyset$ et $\mathcal{O}_a \models \varphi$ et $\mathcal{O}'_a \not\models \varphi$.

Exemple 6. Considérons l'ontologie $\mathcal{O}$ composée des axiomes suivants, dont certains ont déjà été utilisés dans les exemples précédents.

Axiomes de la T-box
T1 : $Portrait \sqsubseteq \exists hasPersons \sqcup \exists hasAnimals$
T2 : $Presidents \sqsubseteq Persons$

Axiomes de la A-box
A1 : `Portrait(Belga1050730.jpg)`
A2 : `Persons(SaddamHussein)`
A3 : `Presidents(SaddamHussein)`
A4 : `hasPersons(Belga1050730.jpg,SaddamHussein)`

Soit $\mathcal{QL}$ l'ensemble des vérifications d'instance, c'est à dire l'ensemble des requêtes de la forme $C(i)$, où C est un concept composé dans $\mathcal{O}$ et i est une instance. {T2, A1, A2, A3, A4} est une $\mathcal{QL}$-description complète de l'instance `Belga1050730.jpg`. Elle n'est cependant pas minimale, car {A1, A2, A3, A4} et {T2, A1, A3, A4} vérifient les mêmes formules, dont la signature contient `Belga1050730.jpg`, sur $\mathcal{QL}$. Ces deux ensembles d'axiomes sont des $\mathcal{QL}$-descriptions minimales de `Belga1050730.jpg`.

Cet exemple montre qu'une description minimale n'est pas unique.

2.3 Extraction de modules : calculabilité et approximations

Comme nous l'avons souligné en introduction, il peut être intéressant d'extraire un module contenant les descriptions des objets indexés pour ne conserver que la partie utile de l'ontologie lors du processus de recherche. Les définitions proposées ne peuvent cependant pas être utilisées directement pour construire des algorithmes efficaces, étant donnés les résultats de complexité pour les problèmes de décision associés. Lutz et al. (2007) montrent en effet les théorèmes suivants.

Étant donné une logique de description $\mathcal{L}$, une signature S, un langage de requêtes $\mathcal{QL} = \{C_1 \sqsubseteq C_2 | C_1, C_2 \in \mathcal{L}(S)\}$, et deux ontologies $\mathcal{O}' \subseteq \mathcal{O}$ sur $\mathcal{L}$, décider si $\mathcal{O}$ est une $\mathcal{QL}$-extension conservative de $\mathcal{O}'$ est un problème :

- 2NEXPTIME-complet pour $\mathcal{L} = \mathcal{ALC}$,
- 2NEXPTIME-complet pour $\mathcal{L} = \mathcal{ALCQI}$,
- indécidable pour $\mathcal{L} = \mathcal{ALCQIO}$, et donc pour le langage OWL-DL que nous présentons au paragraphe 3.3.

Il conviendrait donc d'utiliser des algorithmes basés sur des approximations de la notion de module, comme celles proposées par Grau et al. (2008).

3 Complexité, langages d'ontologies et de requêtes

3.1 Langages de requêtes

Nous montrons par la suite comment les principaux problèmes de raisonnement sur les ontologies et bases de connaissances s'expriment avec des langages de requêtes adéquats.

3.1.1 Pour les ontologies abstraites

Consistance. En tant que théorie logique, il est essentiel qu'une ontologie soit consistante. Le langage de requêtes $\mathcal{QL}_{\perp} = \{\top \leq_{\mathcal{C}} \perp\}$ constitué d'une seule requête non satisfaisable fournit un test de consistance. En effet, une ontologie $\mathcal{O}$ est consistante ssi $\mathcal{O} \nvDash (\top \leq_{\mathcal{C}} \perp)$.

Classification. Cette tâche consiste à calculer tous les axiomes du treillis conceptuel, c'est à dire tous les axiomes de la forme $A \leq_{\mathcal{C}} B$ où A et B sont des concepts atomiques. Le langage $\mathcal{QL}_{\leq_{\mathcal{C}}} = \{A \leq_{\mathcal{C}} B | A, B \in \mathcal{C}\}$ permet cette classification.

Subsomption. On cherche ici les relations de subsomption entre des concepts composés grâce au langage de requêtes $\mathcal{QL}_{\leq_{\mathcal{C}},\mathcal{L}(\mathcal{C})} = \{A \leq_{\mathcal{C}} B | A, B \in \mathcal{L}(\mathcal{C})\}$.

3.1.2 Pour les bases de connaissances (ontologies instanciées)

Dans le cas d'OMNIA, on utilise une ontologie comme base pour l'accès à des données. L'indexation des données est faite à l'aide de *descriptions* définies dans le paragraphe 2. Nous présentons ici deux langages de requêtes qui permettent d'accéder aux données décrites dans la A-box d'une base de connaissance. Les problèmes de complexité liés à la résolution de ces requêtes sont abordés au paragraphe 3.

Vérification d'instances. Cette tâche consiste à vérifier l'appartenance d'une instance à une classe composée de l'ontologie. Elle est réalisée à l'aide du langage $\mathcal{QL}_{\in,\mathcal{L}(\mathcal{C})} = \{a \in A | A \in \mathcal{L}(\mathcal{C}), a \in \mathcal{E}\}$.

Requêtes conjonctives. Une classe particulière de requêtes a été introduite pour les bases de données. Certains problèmes difficiles deviennent raisonnablement calculables si l'on se restreint aux *requêtes conjonctives* définies ci-après.

Définition 19. Une **requête conjonctive** est une formule de la logique du premier ordre constituée à partir de formules atomiques en utilisant seulement la conjonction $\wedge$ et la quantification existentielle $\exists$, mais pas la disjonction $\vee$, la négation $\neg$, ni la quantification universelle $\forall$.

Ces requêtes peuvent s'écrire sous la forme prénexe :
$\varphi(x_1, \ldots, x_k) = \exists x_{k+1}, \ldots x_m . A_1 \wedge \ldots \wedge A_r$
où $A_1, \ldots, A_r$ sont des formules atomiques.

3.2 Mesures de complexité

Plusieurs mesures permettent d'évaluer la complexité des problèmes de raisonnement sur les ontologies. Les mesures pertinentes dépendent de la tâche de raisonnement et du contexte applicatif de l'ontologie. On peut calculer la complexité en fonction de :

- **la taille des données** (data complexity). C'est celle de la description des instances dans l'ontologie, c'est à dire le nombre d'axiomes de la A-box.
- **la taille de la taxonomie** (taxonomic complexity). C'est celle de la description des concepts dans l'ontologie, c'est à dire le nombre d'axiomes de la T-box.
- **la taille d'une requête** (query complexity). C'est celle de la taille de l'arbre syntaxique de cette requête (et pas son nombre de caractères).
- **la combinaison des mesures précédentes**. C'est la somme de la taille des données et de la taxonomie pour les tests de consistance de satisfaisabilité, de subsomption et d'instance. On y ajoute la taille des requêtes quand c'est pertinent.

Nous avons vu au paragraphe 1.3 que répondre à des requêtes arbitraires (avec des variables libres) sur une ontologie $\mathcal{O}$ revient au pire à l'évaluation de $card(\mathcal{O})^n$ requêtes booléennes, où $card(\mathcal{O})$ est le nombre de concepts, relations et instances intervenant dans $\mathcal{O}$, et n l'arité de la requête. Étant donné un langage de requêtes, les problèmes *"répondre à une requête arbitraire"* et *"vérifier une requête booléenne"* appartiennent donc aux même classes de complexité en fonction des tailles des données (A-box) et de la taxonomie (T-box). Par exemple, si la résolution des requêtes est polynômiale, elle le reste, mais avec un polynôme de degré augmenté de n.

3.3 Le langage d'ontologies OWL

Le Web Ontology Language (OWL) [18] est un langage de représentation des connaissances, recommandé par le W3C depuis 2004, et doté d'une sémantique fondée sur les logiques de description. OWL étend l'expressivité de RDFS. Nous nous intéresserons plus particulièrement à la nouvelle version de ce langage, OWL 2 [19], qui a atteint le statut de recommandation du W3C en 2009. Plusieurs syntaxes peuvent être utilisées pour exprimer des documents OWL. Nous utilisons principalement la syntaxe RDF/XML.

Exemple 7. Nous présentons ci-dessous les axiomes de la A-box de l'exemple 2.

```
<rdf:RDF>
<!-- http://www.idmt.fhg.de/Photo_Tagging.owl#Belga1050730 -->
    <owl:Thing rdf:about="#Belga1050730"/>

<!-- http://www.idmt.fhg.de/Photo_Tagging.owl#Belga1050730.jpg -->
    <Photo rdf:about="#Belga1050730.jpg">
        <rdf:type rdf:resource="#Portrait"/>
        <rdf:type rdf:resource="&owl;Thing"/>
        <hasID rdf:resource="#Belga1050730"/>
        <hasPersons rdf:resource="#SaddamHussein"/>
    </Photo>
<!-- http://www.idmt.fhg.de/Photo_Tagging.owl#SaddamHussein -->

    <Persons rdf:about="#SaddamHussein">
        <rdf:type rdf:resource="&owl;Thing"/>
    </Persons>
</rdf:RDF>
```

FIGURE 4 – Axiomes exprimés en OWL

18. www.w3.org/2004/OWL/
19. http://www.w3.org/TR/owl2-overview/

OWL et OWL 2 permettent de représenter des ontologies conformes aux définitions de la partie 1. OWL et OWL 2 comportent des sous-langages dont une partie de l'expressivité a été sacrifiée, au profit de l'efficacité pour certaines tâches de raisonnement. Les sous-langages de OWL 2 (*OWL 2 profiles*[20]) sont :

- **OWL 2 EL**. Il est basé sur la famille de logiques de description EL++ (Baader et al. (2005)) qui n'autorise que la quantification existencielle. Ce sous-langage offre de bonnes performances de raisonnement en fonction de la taille de la T-box.
- **OWL 2 QL**. Il est basé sur la famille de logiques de description DL-Lite (Calvanese et al. (2007)) qui offre de bonnes performances pour la résolution de requêtes conjonctives en fonction de la taille de la A-box. Les raisonneurs dédiés à ce langage utilisent des bases de données relationnelles pour stocker les données de l'ontologie. Les requêtes sont résolues sur la base de données après une réécriture automatique. OWL 2 QL est le successeur du fragment OWL-lite de OWL 1.
- **OWL 2 RL**. Ce langage vise à ne pas sacrifier trop d'expressivité, tout en conservant une complexité polynômiale, en fonction de la taille de la T-box, pour les principales tâches de raisonnement. Les moteurs d'inférence pour ce fragment peuvent être développés en utilisant des langages de règles classiques.

Un récapitulatif des résultats de complexité pour ces différents langages est donné dans la table 3[21]. On se réfèrera à (Papadimitriou (1994)) pour une introduction à la complexité. Les problèmes qui apparaissent dans la table ont été définis au paragraphe 3.1. Les classes de complexité utilisées dans la table sont les suivantes :

- PTIME est la classe des problèmes résolubles par un algorithme déterministe en temps polynomial en fonction de la taille des entrées.
- LOGSPACE est la classe des problèmes résolubles par un algorithme déterministe en utilisant un espace mémoire logarithmique en fonction de la taille des entrées (la classe LOGSPACE est inclue dans PTIME).
- NP est la classe des problèmes résolubles par un algorithme non-déterministe en temps polynomial en fonction de la taille des entrées.
- un problème est dit complet pour une classe X (X-complet) s'il appartient à cette classe et si tout problème de cette classe peut lui être réduit en temps polynomial (c'est un problème parmi les plus difficiles de la classe).

3.4 Le langage de règles SWRL

Le langage SWRL (*Semantic Web Rule Langage*) est une soumission au W3C[22]. Il propose une extension d'OWL basée sur le langage de règles RuleML[23] (*Rule Markup Language*) et permet typiquement d'exprimer des *clauses de Horn* (Horn (1951)).

SWRL propose une syntaxe "humainement compréhensible" dans laquelle les variables sont préfixées par des points d'interrogation. Nous pouvons par exemple écrire une règle qui déduit une relation `oncle` à partir des relations `parent` et `frère` :

`parent(?x,?y)` $\wedge$ `frère(?y,?z)` $\Rightarrow$ `oncle(?x,?z)`

Il est possible d'ajouter des règles SWRL dans une ontologie OWL à l'aide du plugin *SWRL Tab*[24] pour l'éditeur d'ontologies *Protégé*.

20. `http://www.w3.org/TR/owl2-profiles/`
21. Pour des résultats plus précis, consulter : `http://www.w3.org/TR/owl2-profiles/`
22. `http://www.w3.org/Submission/SWRL/`
23. `http://ruleml.org/`
24. `http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab`

Langage	Raisonnement	Complexité		
		T-box	A-box	Requête
OWL 2 EL	Consistance, subsomption, vérification d'instance simple	PTIME-complet	PTIME-complet	Pas applicable
	Requêtes conjonctives (booléennes)	PTIME-complet	PTIME-complet	NP-complet
OWL 2 QL	Consistance, subsomption, vérification d'instance simple	PTIME	LOGSPACE	Pas applicable
	Requêtes conjonctives (booléennes)	PTIME	LOGSPACE	NP-complet
OWL 2 RL	Consistance, subsomption, vérification d'instance simple	PTIME-complet	PTIME-complet	Pas applicable
	Requêtes conjonctives (booléennes)	PTIME-complet	PTIME-complet	NP-complet

TABLE 3 – Récapitulatif des résultats de complexité pour les sous-langages de OWL 2, source : http ://www.w3.org/TR/2008/WD-owl2-profiles-20081202/

3.5 Le cas d'OMNIA

3.5.1 Langage d'ontologies et de requêtes pour OMNIA

On se heurte à une dualité entre, d'une part, l'expressivité logique des langages d'ontologies et de requêtes, et d'autre part la complexité des tâches de raisonnement permettant de répondre aux requêtes. Il convient donc de choisir les langages utilisés en fonction des applications.

Dans le cas d'OMNIA, nous visons des ontologies abstraites relativement simples et petites, alors que la quantité de données à gérer est colossale (centaines de milliers voire millions d'images avec leurs textes compagnons). Nous sommes donc concerné par la complexité en fonction de la taille des données. La complexité en fonction de la taille des requêtes serait également pertinente, mais on ne peut espérer tomber en dessous d'une complexité exponentielle pour résoudre des requêtes conjonctives. En effet, cette tâche présente déjà une complexité exponentielle en fonction de la taille des requêtes dans le cadre des bases de données classiques.

Par ailleurs, on souhaite retrouver des images, indexées comme instances d'une ontologie, c'est à dire accéder à la A-box de l'ontologie. Dès lors, nous sommes concerné par les requêtes de type *vérification d'instances* définies au paragraphe 3.1.2. Afin de bénéficier des algorithmes efficaces pour la vérification d'instances (en particulier pour des requêtes conjonctives) dans de grandes masses de données, nous avons utilisé le langage OWL 2 QL pour développer une ontologie de test dans le projet OMNIA.

Enfin, l'indexation des images (création de descriptions) dans OMNIA est réalisée à partir d'une extraction de contenu automatique dans les textes. Nous sommes donc confronté à des données imprécises et incertaines. L'imprécision est par exemple observée dans des expressions qui requièrent une interprétation subjective liée à l'intensité. C'est en particulier le cas lorsqu'on parle d'affect. Ainsi, la violence d'une image peut être quantifiée par un utilisateur sur une échelle de degrés, et ce degré peut être modifié au niveau linguistique par des constructions comme "une image très violente". De telles expressions linguistiques imprécises peuvent être traduites au niveau formel grâce à la logique floue (Zadeh (1965)). Des extensions floues pour les logiques de description et pour le langage OWL, telles que proposées par Bobillo et Straccia (2010), nous permettraient de rendre compte des imprécisions langagières et des processus d'intensification

au niveau des descriptions construites dans l'ontologie. Cette possibilité n'a toutefois pas été explorée plus avant dans le cadre du projet OMNIA.

En effet, nous nous sommes concentré sur le deuxième aspect : l'incertitude des informations extraites dans les textes. Pour ce faire, il nous suffit d'ajouter des scores de confiance aux axiomes, dans la A-box de l'ontologie. Cela est fait, comme c'est l'usage dans les extensions floues de OWL (Calegari et Ciucci (2007); Bobillo et Straccia (2010)), à l'aide d'une annotation (ici *Confidence*) ajoutée aux déclarations d'axiomes.

La figure 5 donne un exemple de la syntaxe utilisée pour ajouter un score de confiance à un axiome des exemples précédents. Nous verrons au chapitre 4 comment ces scores sont calculés et exploités pour l'indexation et la recherche d'images à partir de textes libres.

```
<rdf:RDF>
    <Photo rdf:about="#Belga1050730.jpg">
        <rdf:type rdf:resource="#Portrait"/>
    </Photo>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;Axiom"/>
        <Confidence rdf:datatype="&xsd;float">0,666</Confidence>
        <rdf:object rdf:resource="#Portrait"/>
        <owl:subject rdf:resource="#Belga1050730.jpg"/>
        <owl:predicate rdf:resource="&rdf;type"/>
    </rdf:Description>
</rdf:RDF>
```

FIGURE 5 – Ajout d'un score de confiance à un axiome OWL

3.5.2 Une ontologie du projet OMNIA

L'ontologie fournie dans le cadre de la campagne image CLEF09, que nous avons partiellement décrite dans les exemples précédents, nous a inspiré pour construire une ontologie de test pour le projet OMNIA. Nous n'avons pas directement utilisé l'ontologie de la campagne image CLEF09 pour deux raisons.

- Certains constructeurs de la logique $\mathcal{ALCHIQ}$ utilisés dans l'ontologie de CLEF09 ne sont pas autorisés dans le langage OWL 2 QL (inclusion de rôles, rôles fonctionnels, disjonction de classes, etc.).
- La T-box ne contient pas assez de concepts pour une extraction de contenu significative dans des textes tout venant. En effet, comme nous le verrons par la suite, nous ne cherchons dans les textes que des mots "alignés" avec des concepts de la T-box.

Pour les expériences du projet OMNIA, nous avons développé une ontologie de classification des images [25] qui comporte 732 classes dans les domaines suivants : animaux, politique, religion, armée, sports, monuments, transports, jeux, divertissements, affects, etc. Les seules instances de cette ontologie sont les images indexées. Un concept étiqueté HOPITAL doit donc être compris comme "l'ensemble des images qui évoquent/représentent un hôpital". Cette ontologie, qui se limite à une taxonomie de concepts, est utilisée dans le système de référence (*baseline*) du projet OMNIA que nous décrivons au chapitre 4.

Enfin, l'indexation des documents et la création de requêtes à partir de textes se font en contexte multilingue. Le lien entre les ontologies utilisées et les lexiques de plusieurs langues est donc un enjeu important dans le cadre d'OMNIA. De façon plus générale, il est nécessaire de disposer d'ontologies linguistiquement riches (même monolingues) pour effectuer une extraction

25. http://kaiko.getalp.org/kaiko/ontology/OMNIA/100606_OMNIA_v6.owl

de contenu satisfaisante dans des textes tout venant. Ce sont les problèmes que nous étudions maintenant dans le chapitre 3.

Chapitre 3

Multilinguisme pour l'extraction de contenu sémantique

Sommaire

Introduction

Nous avons déjà évoqué le fait que l'utilisation d'ontologies pour l'indexation documentaire est le prolongement d'approches basées sur des ressources terminologiques pour la définition de descripteurs (termes préférés). Les descripteurs utilisés pour l'indexation sont alors issus de ressources externes, et un enjeu majeur est d'identifier ces descripteurs au sein des textes, avec une couverture et une précision maximales. Disposer d'ontologies linguistiquement riches peut alors être une aide précieuse, sinon indispensable, pour réaliser une extraction de contenu satisfaisante lors du processus d'indexation. L'enjeu est la modélisation d'une ontologie à la fois linguistiquement et logiquement riche, permettant respectivement une extraction de contenu de qualité et des inférences poussées pour la résolution des requêtes.

Nous commençons par présenter une revue des principales approches permettant d'enrichir linguistiquement une ontologie OWL, c'est à dire d'associer à ses éléments (concepts, relations, instances) des termes, éventuellement dans plusieurs langues, et toute information lexicale utile (morphologie, définitions, etc.). Ayant illustré les faiblesses de ces approches dans une optique d'extraction d'information dans des textes multilingues, nous présentons une méthode basée sur des alignements entre ressources ontologiques et lexicales (lexicalisation externe). Nous verrons que l'approche proposée laisse toute liberté dans la modélisation de l'ontologie et permet d'envisager la construction de descritions pour les documents (comme définies au paragraphe 2.2). Par ailleurs, les alignements permettent de lier une ontologie à une architecure lexicale multilingue.

Quatre sous-problèmes seront étudiés dans cette partie :

1) la représentation d'ontologies linguistiquement riches,
2) l'insertion d'une ontologie dans une architecture multilingue,
3) l'implémentation des structures choisies,
4) l'exploitation des ressources obtenues dans un cadre de recherche ou d'extraction d'information.

1 Représentation d'ontologies linguistiquement riches : état de l'art

Trois principales approches ont été proposées pour obtenir des ontologies lexicalement riches (Prévot et al. (2005)) :

1) la restructuration d'un lexique à partir de principes ontologiques,
2) l'inclusion d'informations lexicales au sein d'une ontologie,
3) l'alignement de l'ontologie avec des ressources lexicales.

La première possibilité est inapplicable dans notre cas, puisque nous souhaitons utiliser des ontologies existantes, enrichies *a posteriori*, comme paramètres d'un système d'extraction. Nous nous concentrerons donc sur les deux dernières possibilités, que nous qualifierons respectivement de lexicalisations *interne* et *externe*.

Trois points doivent selon nous être analysés attentivement dans la revue de l'état de l'art qui suit :

1) les modèles logiques utilisés pour enrichir les ontologies,
2) l'implémentation de ces modèles et leur complexité calculatoire,
3) les méthodes pour instancier ces modèles.

1.1 Lexicalisation simple d'une ontologie

1.1.1 Utilisation d'annotations RDF

La façon la plus simple (et naïve) de stocker des informations complémentaires dans une ontologie est d'utiliser les annotations `rdfs:label` et `rdfs:comment`. Elles permettent en effet d'associer des chaînes de caractères aux concepts, relations et instances. L'annotation `rdf:lang` permet par ailleurs la prise en compte du multilinguisme. Il est important de remarquer que l'on parle d'annotations (`owl:AnnotationProperty`) et pas d'attributs (`owl:DatatypeProperty`), car des attributs ne pourraient être appliqués qu'aux instances, à moins d'utiliser toute l'expressivité de OWL-Full (nous reviendrons sur ce point par la suite).

C'est le choix qui a été fait pour OntoLing (Pazienza et Stellato (2006)), un plugin destiné à l'enrichissement lexical d'ontologies pour l'éditeur Protégé. La figure 1 présente un extrait de l'interface d'OntoLing.

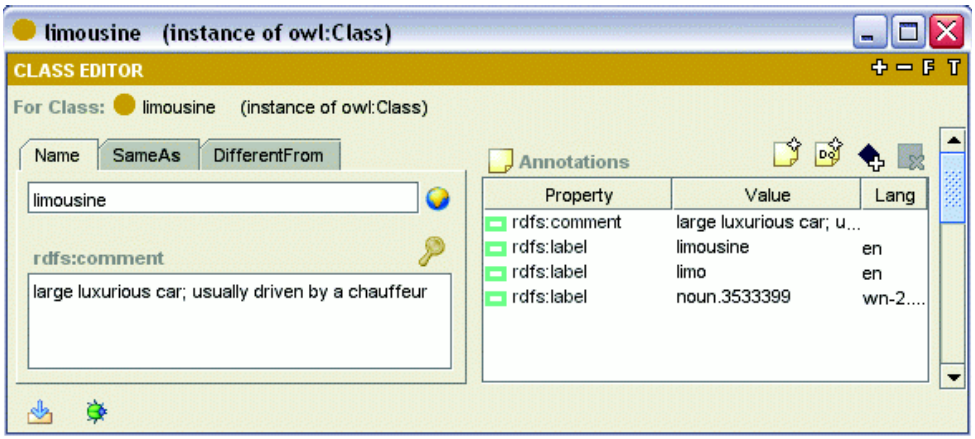


FIGURE 1 – Un extrait de l'interface d'OntoLing

Un lexique informatisé offre souvent bien plus qu'une simple liste de mots. Ses entrées peuvent être liées par des relations sémantiques, proposer des variantes morphologiques, etc. Suivant Sérasset et Mangeot (2001), nous appelons l'organisation des entrées d'un lexique sa *microstructure*. L'utilisation de simples annotations rdf est une méthode très limitée pour la lexicalisation interne, puisqu'elle permet uniquement de stocker des chaînes de caractères. On ne bénéficie donc pas des informations de la microstructure d'un lexique.

Dans une optique de lexicalisation externe, il est possible d'utiliser les annotations rdf pour stocker des URI désignant les entrées d'un lexique accessible sur le Web. Cela suppose bien entendu que de telles URI existent pour le lexique.

1.1.2 Création d'une partie dédiée dans l'ontologie

Une autre possibilité de lexicalisation interne est de dédier une partie de l'ontologie aux informations linguistiques. Nous pouvons par exemple définir une classe `Term` dans le treillis conceptuel et stocker les termes comme des instances de l'ontologie. La microstructure du lexique peut ensuite être représentée grâce à des relations sur la classe `Term`.

Cette approche a été utilisée pour développer l'ontologie du projet Biocaster (Collier et al. (2006)) décrivant un sous-domaine de la biomédecine. Elle est utilisée dans un système d'ex-

traction d'information dans des articles de presse, pour la détection d'épidémies. Un extrait de l'ontologie Biocaster est présenté dans la figure 2.

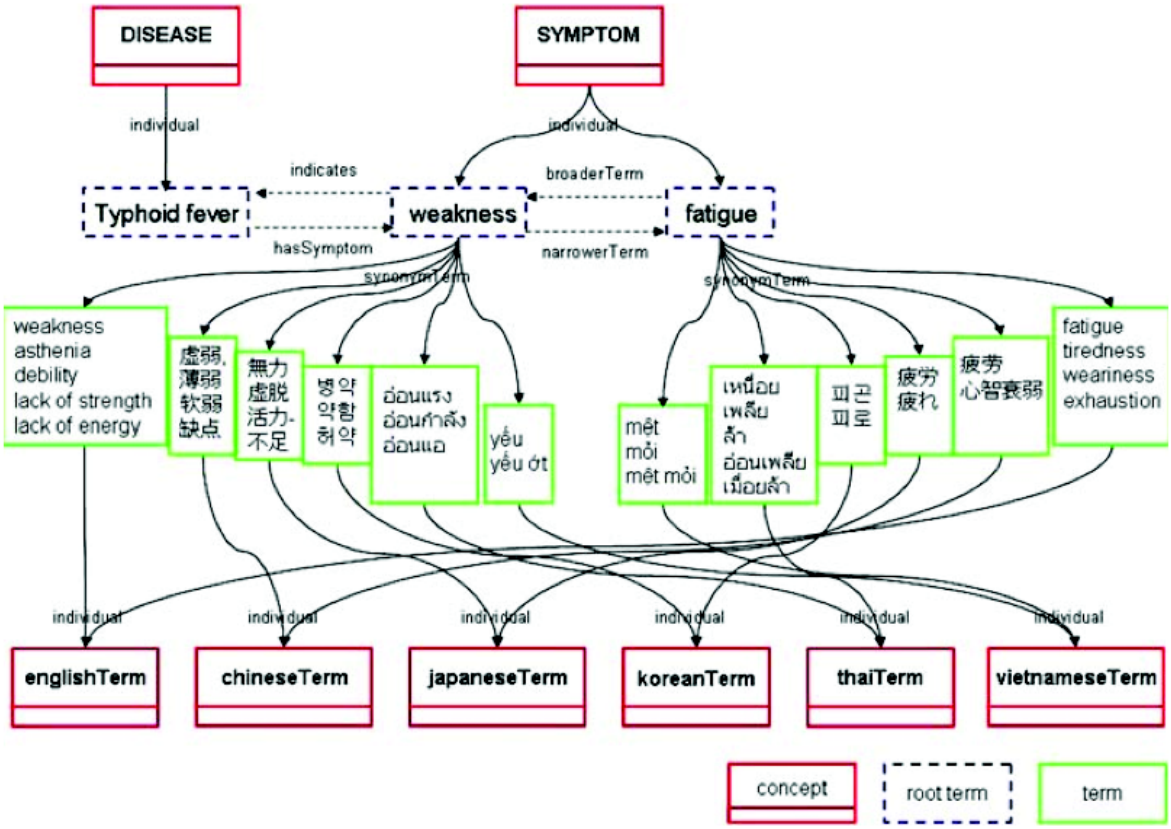


FIGURE 2 – Extrait de l'ontologie du projet Biocaster

Les informations du domaine sont organisées hiérarchiquement sous la classe `bco:Thesaurus`. Les instances de cette classe, les *termes racines* (*root terms* dans la figure 2), peuvent être liées par des relations transversales. Par exemple, une instance de la classe `bco:DISEASE` peut être liée à une instance de la classe `bco:SYMPTOM` par la relation `hasSymptom`, comme `hasSymptom(Typhoid Fever,Weakness)`.

Les termes racines sont les seuls éléments recherchés dans les textes. À cet effet, ils sont liés à des termes, si possible dans 12 langues.

Les mots sont représentées comme les instances d'une classe `bco:Term` (avec une sous-classe pour chaque langue : `bco:englishTerm`, `bco:chineseTerm`, etc.). Cette partie lexicale de l'ontologie est organisée selon une microstructure permettant de lier un terme à des synonymes (éventuellement dans plusieurs langues), à des abréviations, à sa forme scientifique, etc.

L'approche précédente ne permet une lexicalisation qu'au niveau des instances de l'ontologie. En effet, une **sous-classe** de `bco:Thesaurus` ne peut être liée à une **instance** de `bco:Term` sans utiliser de métamodèle et un langage d'ontologie du second ordre. Les problèmes relatifs à l'utilisation de métamodèles sont approfondis au paragraphe 1.3. D'autre part, la modélisation de l'ontologie est très contrainte par son aspect multilingue. En effet, les instances de l'ontologie, dans la partie de description du domaine `bco:Thesaurus`, sont exclusivement des termes (racines), ce qui réduit considérablement les possibilités d'inférence. Par exemple, la description d'une épidémie particulière (et pas seulement du terme "épidémie") comme instance de l'ontologie

permettrait un meilleur suivi de son évolution en tirant profit des possibilités d'inférence.

Nous reviendrons sur l'ontologie Biocaster dans le paragraphe 1.2.

1.2 Simple Knowledge Organisation Language (SKOS)

Le format SKOS est une recommandation du W3C pour le partage de données. Il permet de représenter des informations lexicales structurées comme des thésaurus, des taxonomies ou des schémas de classification. Le modèle de données SKOS est implémenté dans le langage OWL. Attention à la terminologie : dans SKOS, un concept est une instance de la classe `skos:Concept` qui représente une idée ou un sens de mot, alors que dans OWL, un concept désigne une classe contenant des instances.

Les concepts SKOS sont organisés en schémas (`skos:Scheme`). Dans chaque schéma, les concepts sont liés par des relations sémantiques (`skos:broader`, `skos:narrower`, etc.), et étiquetés par des mots dans différentes langues à l'aide des attributs `skos:prefLabel`, `skos:altLabel` et `skos:hiddenLabel`. On se rapproche ici de la solution proposée au paragraphe 1.1.1 avec l'utilisation des annotations rdf. Cependant les "labels" de SKOS sont implémentés comme des attributs (`owl:objectProperty`)) et ne peuvent s'appliquer qu'aux instances de la classe `skos:Concept`.

L'extension SKOS-XL (SKOS eXtension for Labels [26]) offre la possibilité d'étiqueter les concepts avec des objets plus complexes que de simples chaînes de caractères. Les étiquettes de SKOS-XL peuvent contenir des URI externes ou des instances de la classe `skosxl:label` (disjointe de `skos:Concept`). Cette classe définit une partie de la base de connaissances, dédiée aux informations lexicales, comme proposé au paragraphe précédent.

La modélisation choisie dans l'ontologie Biocaster est proche de celle proposée par le langage SKOS. Une traduction complète de l'ontologie Biocaster dans le format SKOS est disponible [27]. La classe `bco:Thesaurus` est déclarée équivalente à `skos:Concept` et `bco:Term` est déclarée équivalente à `skosxl:label`.

Un reproche fréquemment fait à SKOS est que les informations lexicales et ontologiques ne sont pas suffisamment séparées. Cela contraint fortement la modélisation des informations et pourrait entrainer des amalgames. Nous opposons à cela le fait que SKOS n'est pas conçu pour représenter n'importe quelle connaissance ontologique. En effet, le modèle SKOS à lui seul constitue déjà une ontologie OWL, qui décrit justement en partie le domaine de la représentation des informations lexicales. Par la suite, nous utiliserons SKOS pour représenter des lexiques structurés, utilisés conjointement avec des ontologies.

1.3 Métamodèles pour l'ajout d'informations linguistiques

Divers (méta-)modèles ont été proposés pour l'ajout d'informations linguistiques à des ontologies. Deux travaux sont représentatifs de cette approche :

1. LIR [28] (Linguistic Information Repository), proposé par Montiel-Ponsoda et al. (2008),
2. et LingInfo [29], proposé par Buitelaar et al. (2006).

Le métamodèle OWL décrit les types d'éléments constitutifs d'une ontologie OWL (classes, propriétés, instances, etc.) et leurs interactions possibles. Les approches que nous proposons

26. `http://www.w3.org/2008/05/skos-xl`
27. `http://sites.google.com/site/nhcollier/home/recent-activities/biocasterontologyv3released`
28. `http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/downloads/63-lir`
29. `http://olp.dfki.de/LingInfo/`

ajoutent un *schéma lexical* au métamodèle OWL, comme illustré dans la figure 3. L'extension du métamodèle permet de "brancher" un schéma d'informations lexicales à tous les objets de la métaclasse `owl:ontologyElement`.

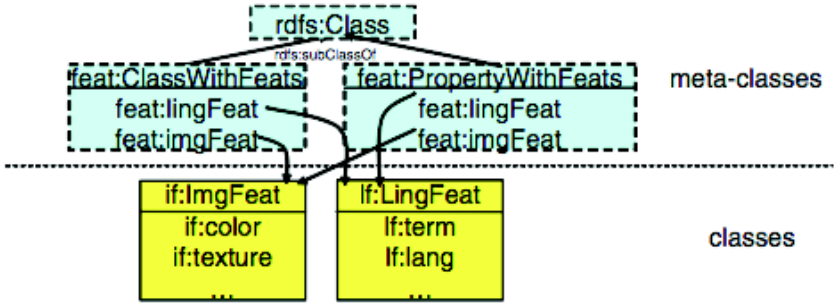


FIGURE 3 – Extrait du métamodèle LingInfo

Le schéma de LIR met l'accent sur les liens de traduction entre les sens de mots, alors que LingInfo est plus centré sur des informations morphosyntaxiques. Malgré les différences dans le contenu et l'organisation des schémas lexicaux, LIR et LingInfo partent de la même idée de métamodélisation. Dans les deux cas, les objets de l'ontologie sont liés à une "instance lexicale" (appartenant à une classe `LingInfo` ou `LexicalEntry` dans LIR) qui est le support des propriétés linguistiques. Malgré cette similarité dans l'idée initiale, nous allons voir que des méthodes très différentes sont utilisées pour implémenter ces modèles.

LingInfo a été appliqué pour associer des termes, dans plusieurs langues, aux classes de l'ontologie du projet SmartWeb [30] (SmartWeb Integrated Ontology, ou SWIntO). Le métamodèle est inclus directement dans l'ontologie de domaine. Un concept de l'ontologie est alors à la fois une sous-classe de `owl:Thing` et une instance de la métaclasse `ClassWithLingInfo`. Par exemple, nous donnons ci-dessous les déclarations qui permettent d'associer un terme français à la classe `SOCCER` (en utilisant la syntaxe XML Turtle [31]) :

```
SOCCER   rdfs:subClassOf owl:Thing ;
         rdf:type linginfo:ClassWithLingInfo ;
         linginfo:LingInfo Lex106 .

Lex106   linginfo:term ''Football'' ;
         linginfo:lang ''fr'' .
```

Cette méthode souffre d'un inconvénient majeur que nous avons déjà évoqué : elle requiert l'expressivité de OWL-Full pour incorporer le métamodèle dans l'ontologie. Ainsi, la complexité d'une ontologie de domaine est considérablement augmentée par l'ajout des informations linguistiques. Cela rend impossible l'utilisation de raisonneurs optimisés pour des sous-langages de OWL, voire l'utilisation d'un quelconque raisonneur, étant donné le caractère indécidable des tâches de raisonnement pour OWL-Full. On remarque par ailleurs que cette méthode implique des modifications importantes dans l'ontologie source, ce qui la rend peu adaptée à la lexicalisation (ou multilinguisation) d'une ontologie *a posteriori*.

Quant au modèle LIR, il est dans le plugin *LabelTanslator* pour *NeOn Toolkit*. Quand une ontologie est importée dans NeON, une ontologie LIR dédiée (et distincte de l'ontologie source)

30. `http://smartweb.dfki.de/ontology_en.html`
31. `http://www.w3.org/TeamSubmission/turtle/`

est créée. Cette nouvelle ontologie contient le schéma linguistique de LIR et des copies de tous les éléments de l'ontologie source, comme instances de la classe `lir:OntologyElement`. Il est ensuite possible d'instancier le schéma lexical de LIR pour chaque copie des éléments de l'ontologie source. Les informations lexicales d'un élément de l'ontologie source sont affichées ou exploitées en suivant le lien implicite entre l'élément source et sa copie dans l'ontologie LIR. Au delà de la séparation entre les informations linguistiques et ontologiques, un avantage de cette approche est de garder l'ontologie source inchangée. On note également l'existence de processus semi-automatiques, proposés par Espinoza et al. (2008), pour l'instanciation de l'ontologie LIR dans 3 langues.

Il est affirmé par Montiel-Ponsoda et al. (2008), et Buitelaar et al. (2006), que LIR et LingInfo sont substantiellement différents de SKOS, notamment du fait de la séparation des informations linguistiques et ontologiques. Cela est sans doute vrai si l'on cherche à utiliser SKOS comme un langage d'ontologie générique (et pas spécialisé dans la représentations d'informations lexicales). Nous avons mis en avant au paragraphe précédent que le modèle SKOS pouvait être considéré comme une ontologie du domaine de la représentation des informations lexicales. Dès lors, si l'on souhaite utiliser ce modèle pour d'autres informations, ces dernières seront de fait mélangées aux informations lexicales. De notre point de vue, SKOS s'apparente tout à fait aux schémas lexicaux de LIR et LingInfo s'il est lié à une ontologie de domaine quelconque en utilisant un métamodèle. Par exemple, dupliquer les éléments de l'ontologie source dans une ontologie SKOS, comme instances de la classe `skos:Concept`, donnerait un résultat similaire à l'ontologie LIR créée par LabelTranslator.

Bien entendu, les schémas lexicaux de LIR, LingInfo et SKOS diffèrent, mais ce n'est pas le plus important en premier lieu. En effet, les schémas lexicaux peuvent être étendus à souhait pour couvrir des besoins spécifiques. Le point essentiel est la méthode utilisée pour lier ces schémas lexicaux à une ontologie. Comme nous l'avons montré dans ce paragraphe, des idées très proches de la métamodélisation peuvent mener à des résultats très différents après implémentation. LingInfo s'apparente en effet à une lexicalisation interne, puisque le métamodèle, comme les informations linguistiques, est stocké dans l'ontologie source. L'implémentation de LIR dans LabelTranslator est plus proche de la lexicalisation externe, puisque les informations linguistiques sont stockées dans une base de connaissances distincte de l'ontologie source. Il est cependant précisé par Montiel-Ponsoda et al. (2008) que le but de LIR n'est pas de développer des lexiques puis de les relier à une ontologie, mais d'associer des informations lexicales dédiées à une ontologie. On ne peut donc pas parler exactement de lexicalisation externe, dont un exemple est donné dans le paragraphe suivant.

1.4 Alignement de l'ontologie SUMO avec Wordnet

Niles et Pease (2003) présentent la réalisation d'une correspondance entre les concepts de SUMO (Suggested Upper Merged Ontology, Pease et al. (2002)) et les synsets de Wordnet (Fellbaum (1998)). Trois types de liens sont considérés : synonymie, hyponymie et instance. Le choix de ces relations nous semble peu approprié, car les deux premières sont généralement appliquées entre des sens de mots. Considérant que les synsets sont des objets assez proches de concepts, des relations comme l'équivalence et la subsomption auraient pu être utilisées. Bien qu'aucune différence pratique ne soit observée, le choix de la terminologie nous semble important dans ce problème transdisciplinaire. Nous présenterons au paragraphe 2.2 des relations possibles entre concepts et sens de mots.

Cette initiative n'entre pas complètement dans le cadre de notre étude. Les auteurs ont en effet souhaité évaluer (et compléter) la couverture de SUMO en cherchant à lier tous les synsets

de Wordnet 1.6 à l'ontologie. On remarque également que les correspondances ont été réalisées de façon manuelle et que leur manipulation n'est pas outillée. Cet alignement SUMO-Wordnet est un excellent exemple de lexicalisation externe, c'est à dire d'enrichissement linguistique d'une ontologie à l'aide de ressources lexicales existantes. Il ne constitue cependant pas un cadre général pour la lexicalisation d'ontologies.

1.5 KYOTO

Le projet KYOTO (Vossen et al. (2008)) a pour but principal la création d'un environnement de type wiki (Wikyoto[32]) pour la création d'ontologies liées aux Wordnets de différentes langues (actuellement sept langues sont supportées : basque, chinois, allemand, anglais, italien, japonais et espagnol). À partir d'entrées des Wordnets ou de termes extraits automatiquement de documents, des experts de chaque langue peuvent proposer des liens vers les concepts existants, ou créer de nouveaux concepts dans une ontologie. Il est également possible d'étendre le Wordnet d'une langue avec un terme désignant un concept de l'ontologie ou traduisant un terme dans un autre Wordnet.

La base de connaissances multilingue obtenue grâce à Wikyoto est stockée à l'aide de la plate-forme DebVisDic[33] (Horak et al. (2006)). Un format de données dédié, KYOTO-LMF (ou Wordnet-LMF), a été développé à partir de la norme `ISO 24613:2008` (*Lexical Markup Framework*[34], Francopoulo et al. (2006)). Il permet de représenter de façon unifiée les Wordnets de différentes langues et de les relier à des ressources externes (par exemple aux concepts d'une ontologie) en utilisant l'attribut `MonolingualExternalRef`. Un exemple de ce format est donné dans la figure 4.

Les Wordnets sont également organisés dans une architecture multilingue à pivot en utilisant la notion d'acception interlingue (*axie* ou `SenseAxis` dans LMF) empruntée au projet Papillon (Sérasset et Mangeot (2001)). Une axie lie des synsets de différentes langues partageant une relation de traduction (synonymie, quasi-synonymie, etc.) à un synset pivot (par convention un synset du Wordnet anglais). Une axie peut être associée à une ressource externe (en particulier les concepts d'une ontologie) avec l'attribut `InterlingualExternalRef`, comme présenté dans l'exemple 5. On obtient alors une architecture multilingue, incluant une ontologie, proche de celle que nous avons proposée dans Rouquet et Nguyen (2009) et que nous détaillerons au paragraphe 3.3. Nous reviendrons également sur la notion d'axie au paragraphe 3.3.1.

32. `http://www.wikyoto.net/`
33. `http://deb.fi.muni.cz/clients-debvisdic.php`
34. `http://www.lexicalmarkupframework.org/`

```
<Synset id="ENG-16-06060223-n" baseConcept="1">
    <meta author="piek" date="2008-05-12"/>
        <Definition gloss="bla bla">
            <Statement example="bla bla"/>
    </Definition>
    <SynsetRelations>
        <SynsetRelation targets="EU-16-06056130-n" relType="has_hyperonym">
            <meta author="german" date="2008-05-12" status="yes"
            source="whatsoever" confidenceScore="99"/>
        </SynsetRelation>
    </SynsetRelations>
    <MonolingualExternalRefs>
        <MonolingualExternalRef externalSystem="SUMO"
        externalReference="PoliticalProcess" relType="at">
            <meta author="monica" date="2008-05-27"/>
        </MonolingualExternalRef>
    </MonolingualExternalRefs>
</Synset>
```

FIGURE 4 – Exemple du format KYOTO-LMF

```
<SenseAxes>
    <SenseAxis id="sa_en16-en30_001" relType="equal_synonym">
        <Meta author="monica" date="2008-05-27"/>
        <Target ID="EN-16-06060223-n"/>
        <Target ID="EN-30-08135342-n"/>
        <InterlingualExternalRefs>
            <InterlingualExternalRef externalSystem="SUMO"
            externalReference="PoliticalProcess" relType="at">
                <Meta author="claudia" date="06-06-2008"/>
            </InterlingualExternalRef>
        </InterlingualExternalRefs>
    </SenseAxis>
</SenseAxes>
```

FIGURE 5 – Exemple d'axie dans KYOTO-LMF

La base de connaissance multilingue KYOTO exploite une architecture de multilinguisation externe intéressante. La plate-forme wiki pour le développement collaboratif de cette ressource est remarquable ainsi que son exploitation pour l'extraction de faits dans des textes, qui se rapproche également de la méthode utilisée dans OMNIA.

Conclusion

Nous n'avons pas trouvé, dans les approches existantes, de cadre formel satisfaisant pour ajouter des informations linguistiques à une ontologie existante. En effet, les approches de lexicalisation interne sont :

- soit non adaptées pour représenter la microstructure d'un lexique (utilisation de commentaires dans l'ontologie),
- soit contraignantes pour la modélisation de l'ontologie (comme dans le projet Biocaster),
- soit gourmandes en expressivité pour inclure un métamodèle dans l'ontologie, ce qui augmente la complexité des tâches de raisonnement sur l'ontologie.

Les approches existantes de lexicalisation externe, plus satisfaisantes pour notre objectif, ne proposent pas, selon nous, un cadre suffisamment général. Dans le cas de LIR, la microstructure du lexique dédié est contrainte, et les liens entre les éléments de l'ontologie source et leurs copies dans l'ontologie LIR restent implicites. Par ailleurs, l'alignement entre SUMO et Wordnet est un résultat intéressant (comme ressource), mais ne propose pas de cadre général applicable à d'autres ontologies. La base de connaissances multilingue du projet KYOTO a une architecture pertinente de notre point de vue, mais est dédiée à l'utilisation et au développement de Wordnets.

Aussi, nous proposerons par la suite un cadre, formel et technologique, pour l'ajout d'informations linguistiques à des ontologies. Cette approche de lexicalisation externe s'appuie sur des correspondances entre l'ontologie source et des ressources lexicales existantes.

2 Lexicalisation externe d'ontologies

2.1 Principe et hypothèses

Nous proposons dans cette partie un cadre général de lexicalisation externe pour l'ajout d'informations linguistiques aux ontologies. Notre proposition permet de représenter les approches de lexicalisation externe présentées précédemment et de transformer des ontologies utilisant une lexicalisation interne (comme Biocaster) pour les représenter avec notre formalisme. Nous ne cherchons pas à ajouter les informations linguistiques au sein de l'ontologie source mais à l'aligner avec des lexiques externes, existants ou dédiés. En procédant de la sorte, on ajoute les informations linguistiques sans modifier l'ontologie et en laissant toute liberté dans sa modélisation et son utilisation pour des tâches de raisonnement.

Sauf mention contraire, nous ne considérons que des ontologies exprimées en OWL.

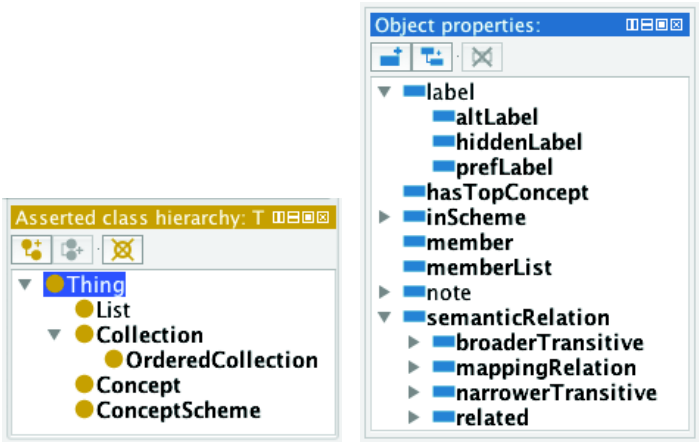
Nous considérons une ressource lexicale comme un ensemble de lexies, avec la définition proposée par Polguère (2002) :

Définition 1. *"Une **lexie**, aussi appelée unité lexicale, est un regroupement 1) de mots-formes ou 2) de constructions linguistiques qui ne se distinguent que par la flexion. Dans le premier cas, il s'agit de lexèmes et dans le second cas, de locutions. Chaque lexie (lexème ou locution) est associée à un sens donné, que l'on retrouve dans le signifié de chacun des signes (mots-formes ou constructions linguistiques) auxquels elle correspond."* Nous supposons par ailleurs que les ressources lexicales sont partagées sur le Web. Leurs lexies sont identifiées par des URI et éventuellemnt liées par des relations lexico-sémantiques (lexiques sémantiques). Typiquement les lexiques exprimés dans le format SKOS conviennent parfaitement à notre approche. Un lexique sera noté $\mathcal{D} = (\mathcal{C}_\mathcal{D}, \mathcal{R}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathcal{L}_\mathcal{D}, \mathcal{T}_\mathcal{D}, \mathcal{A}_\mathcal{D})$, comme une ontologie dont les instances sont des lexies. Voyons sur l'exemple d'un lexique représenté en SKOS à quoi correspondent les éléments du sextuplet $\mathcal{D}$.

Exemple 1. Le modèle SKOS, présenté au paragraphe 1.2, est décrit par une ontologie abstraite OWL que nous notons $\mathcal{D} = (\mathcal{C}_\mathcal{D}, \mathcal{R}_\mathcal{D}, \mathcal{L}_\mathcal{D}, \mathcal{T}_\mathcal{D})$. Il est téléchargeable à l'adresse `http://www.w3.org/2004/02/skos/core`. Seulement 5 classes sont définies dans $\mathcal{C}_\mathcal{D}$ comme illustré dans la figure 3.6(a). Les instances de la classe `skos:Concept` seront ici des sens de mots (c'est à dire les entrées d'un dictionnaire). 35 relations sont définies dans $\mathcal{R}_\mathcal{D}$, dont un certain nombre sont représentées dans la figure 3.6(b). Elles permettent entre autre de lier les concepts par des relations sémantiques (`skos:narrower`, `skos:related`, etc.) ou de leur associer des définitions ou des exemples. Un petit nombre d'axiomes sont exprimés dans la T-box $\mathcal{T}_\mathcal{D}$

pour définir les treillis des classes et des relations, ainsi que quelques propriétés comme la disjonction des classes `skos:Concept` et `skos:ConceptScheme`, ou la symétrie de la relation `skos:related`. La logique $\mathcal{T}_\mathcal{D}$ nécessaire pour exprimer les axiomes du modèle SKOS est la logique de description $\mathcal{SHIF}$.

Une instanciation $\mathcal{D} = (\mathcal{C}_\mathcal{D}, \mathcal{R}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathcal{L}_\mathcal{D}, \mathcal{T}_\mathcal{D}, \mathcal{A}_\mathcal{D})$ du modèle SKOS permet de représenter un dictionnaire sémantique. Les instances dans $\mathcal{E}_\mathcal{D}$ sont des sens de mots ou des collections de sens de mots. La A-box $\mathcal{A}_\mathcal{D}$ contient notamment des déclarations reliant des sens de mots par des relations sémantiques.



(a) Classes (b) Relations

FIGURE 6 – Hiérarchies des classes et des relations déclarées dans la T-box du modèle SKOS

2.2 Alignement entre une ontologie et un lexique sémantique

Nous proposons une définition pour les alignements ontologie-lexique basée sur celle des alignements entre ontologies.

2.2.1 Définitions

La notion de lexique pour une ontologie et d'affectation lexicale, inspirée de Maedche et al. (2003), et présentée au chapitre 2 section 1, considère le lexique comme une liste non structurée de mots ou de sens de mots. Nous présentons maintenant une notion d'alignement, plus complète que la notion d'affectation lexicale, et qui prend en compte la structure du lexique.

Intuitivement, un alignement est un ensemble de liens étiquetés (correspondances élémentaires) entre les éléments de l'ontologie (concepts, individus, relations) et un ensemble de lexies ou de relations lexico-sémantiques entre ces lexies.

Définition 2 (Lien étiqueté). Étant donnés une ontologie $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O}, \mathcal{E}_\mathcal{O}, \mathcal{L}_\mathcal{O}, \mathcal{T}_\mathcal{O}, \mathcal{A}_\mathcal{O})$ et un lexique $\mathcal{D} = (\mathcal{C}_\mathcal{D}, \mathcal{R}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \mathcal{L}_\mathcal{D}, \mathcal{T}_\mathcal{D}, \mathcal{A}_\mathcal{D})$, un lien étiqueté est un quadruplet : $\langle e, e', r, \gamma \rangle$ où $e \in \mathcal{C}_\mathcal{O} \cup \mathcal{R}_\mathcal{O} \cup \mathcal{E}_\mathcal{O}$ est un élément de l'ontologie et $e' \in \mathcal{E}_\mathcal{D}, \mathcal{R}_\mathcal{D}$, est une entrée ou une relation du lexique ; r est la relation entre e et e', parmi un ensemble de relations d'alignement (par exemple, $\equiv$, $\sqsubseteq$, ou $\sqsupseteq$) ; et $\gamma \in [0\ 1]$ est le degré de confiance associé à la relation.

Définition 3(Alignement). Un alignement A est un ensemble de liens étiquetés.

On distingue deux types de liens : lexicaux et structurels. Les liens lexicaux procurent une lexicalisation aux objets de l'ontologie (leur associent des entrées du lexique). Ils peuvent par exemple porter une des relations suivantes :

1. $\equiv$ pour une acception qui dénote exactement un concept ou une relation,
2. $\sqsubseteq$ ou $\sqsupseteq$ pour des acceptions dénotant respectivement un concept plus restreint ou plus général,
3. $\in$ pour des acceptions dénotant une instance d'un concept.

Une règle de bonne conduite est d'éviter d'avoir des informations redondantes entre l'alignement et les ressources (lexique et ontologie). Typiquement, on évitera de déclarer *Saddam Hussein* $\equiv$ *SADDAM_HUSSEIN* et *Saddam Hussein* $\in$ *PRESIDENT* si *SADDAM_HUSSEIN* est déjà déclaré comme instance de la classe *PRESIDENT* dans l'ontologie.

Les liens structurels associent des relations du lexique à des relations de l'ontologie pour lesquelles une lexicalisation n'est pas nécessairement pertinente. Une relation du lexique peut être déclarée équivalente, plus générale, ou moins générale qu'une relation de l'ontologie (on utilisera encore les symboles $\equiv$, $\sqsupseteq$ et $\sqsubseteq$). Par exemple, la relation `skos:narrower` d'un lexique SKOS peut être déclarée équivalente à la relation `rdfs:subClassOf` d'une ontologie OWL. La relation `owl:sameAs` peut quant à elle être déclarée plus générale qu'une relation de synonymie. En effet, des lexies synonymes réfèrent toujours au même concept, mais des lexies peuvent référer au même concept sans être synonymes ("chien" et "cabot" dénotent le même animal avec un niveau de langue différent). Comme nous le verrons au paragraphe 2.3, ces correspondances fournissent des contraintes d'intégrité structurelle exploitables pour la création ou la vérification automatique d'alignements.

La définition d'alignement entre une ontologie et un lexique est presque identique à celle utilisée pour des alignements entre ontologies. Cependant, il y a des différences pragmatiques importantes.

- On cherche usuellement à aligner des ontologies du même domaine (ou dont les domaines ont une intersection non vide). Ici, on cherche à aligner une ontologie de domaine quelconque avec un lexique. Nous l'avons vu avec les SKOS, un lexique peut être considéré comme une ontologie d'un domaine de la linguistique, par exemple la morphologie du français.
- Les alignements entre ontologies sont généralement réalisées de façon homogène, c'est à dire avec des liens entre une classe et une autre classe, une instance et une autre instance, etc. Nos liens lexicaux peuvent associer un élément quelconque de l'ontologie source avec une instance du lexique. Une conséquence est l'impossibilité d'utiliser certaines techniques classiques qui comparent les extensions (instances) des classes et relations pour les aligner.
- Les relations standard du langage OWL ne sont pas équivalentes dans l'ontologie source et le lexique si ce dernier est représenté dans un dialecte OWL. Typiquement, si le lexique est représenté avec SKOS, nous avons vu que la relation `rdfs:subClassOf` de l'ontologie peut correspondre à la relation `skos:narrower`, mais en aucun cas à la relation `rdfs:subClassOf` dans le schéma SKOS.
- Enfin, nous pouvons remarquer une différence probablement importante dans la taille des structures à aligner. Les ontologies de domaine ont typiquement quelques centaines ou milliers d'éléments [35] alors que des dictionnaires informatisés comportent des centaines de

35. Par exemple, les 207 ontologies disponibles au 18/01/2012 sur le dépot d'ontologies TONES (`owl.cs.manchester.ac.uk/repository/`) comportent en moyenne 3 196 éléments. Seulement 9 d'entre elles com-

milliers, voire des millions d'entrées [36].

Ces différences empêchent en général l'utilisation de processus d'alignement entre ontologies dans le cas ontologie-lexique. Nous pouvons néanmoins utiliser les formats communément utilisés pour les alignements.

2.2.2 Implémentation : le format d'alignement

La définition précédente est implémentée dans un format RFD, manipulable à l'aide d'une API d'alignement dédiée (David et al. (2011)).

Chaque *cellule* de l'alignement représente un lien étiqueté et contient un élément de l'ontologie et du lexique (identifiés par des URI), la relation qui lie ces deux éléments et un score de confiance. Une cellule peut être étendue avec toute métadonnée complémentaire utile (par exemple le contributeur ou le logiciel qui a créé le lien).

```
<map>
    <cell>
        <entity1 rdf:resource=''URI1''>
        <entity2 rdf:resource=''URI2''>
        <relation>=</relation>
        <measure>0.8</measure>
    </cell>
</map>
```

FIGURE 7 – Exemple de cellule d'un alignement

Cet outillage offre des facilités considérables pour développer des processus d'alignement automatique entre ontologies et lexiques. Par ailleurs, des alignements représentés dans ce format peuvent être utilisés dans des plates-formes de gestion de bases lexicales multilingues supportant XML, comme Jibiki que nous présenterons au paragraphe 3.3.3.

2.3 Sémantique, intégrité structurelle

L'alignement entre ontologies est en premier lieu une notion syntaxique. En particulier, aucune sémantique n'est imposée pour le format d'alignement que nous utilisons. Trois sémantiques possibles sont proposées par Zimmermann et al. (2006).

Dans le cas d'alignements ontologie-lexique, nous ne sommes pas nécessairement intéressés par une sémantique formelle pour les alignements et les lexiques. En effet, il est communément admis qu'un dictionnaire n'est jamais complet ni consistant. D'autre part, nous utilisons les alignements pour passer d'un espace lexical à l'espace conceptuel de l'ontologie (ou inversement), mais pas pour effectuer des inférence sur un système distribué. Sans définir une sémantique formelle, nous pouvons interpréter les correspondances structurelles entre une ontologie et un lexique de la façon suivante :

Soient R_L une relation entre lexies, R_O une relation de l'ontologie, l_1 et l_2 des lexies, e_1 et e_2 des éléments de l'ontologie.

– Si $R_O \sqsupseteq R_L$, $e_1 \equiv l_1$ et $e_2 \equiv l_2$, alors $R_L(l_1, l_2) \Rightarrow R_O(e_1, e_2)$.

portent plus de 10 000 éléments, et le nombre d'éléments médian pour cet ensemble d'ontologies est de 123 seulement.

36. Par exemple 200 000 entrées en moyenne pour chaque dictionaire de langue européenne utilisés dans les systèmes de traduction automatique de SYSTRAN (Senellart et al. (2001)) et près de 7 millions d'entrées en 2008 pour les dictionnaires du système de traduction automatique ATLAS-II de Uchida (1989).

– Si $R_O \sqsubseteq R_L$, $e_1 \equiv l_1$ et $e_2 \equiv l_2$, alors $R_O(e_1, e_2) \Rightarrow R_L(l_1, l_2)$.
– Si $R_O \equiv R_L$, $e_1 \equiv l_1$ et $e_2 \equiv l_2$, alors $R_L(l_1, l_2) \Leftrightarrow R_O(e_1, e_2)$.
– Si $R_O \equiv R_L$, $e_1 \sqsubseteq l_1$ et $e_2 \equiv l_2$, alors $R_L(l_1, l_2) \Rightarrow R_O(e_1, e_2)$.
– etc.

Idéalement, un alignement devrait donc être un morphisme entre le lexique et l'ontologie, avec des propriétés d'héritage d'une structure à l'autre, et la "commutativité des diagrammes". Il a cependant été montré par Roche (2007) qu'en général, les structures lexicales et conceptuelles d'un domaine ne se superpose pas, ou, en d'autres termes, qu'il n'existe pas nécessairement de morphisme (projection) de l'une vers l'autre.

Exemple 2. Reprenons l'exemple donné par Roche (2007), illustré dans la figure 8, et dans lequel nous considérons que la relation "is-a" de l'ontologie est équivalente à la relation "hyperonymie" du lexique. Alors, les liens 2 et 3 ne respectent pas la contrainte "de morphisme" numéro 3 énoncée précédemment. En effet, dans la langue de spécialité du domaine (ici le *contrôle-commande* pour EDF) le terme "relais de tension" est employé pour désigner une sorte de "relais à seuil" (le concept de l'ontologie) pour lequel la valeur seuillée est la tension. Cette subtilité dans la conceptualisation du domaine n'est ici pas reflétée dans les informations disponibles sur le lexique de spécialité.

On remarque que l'alignement de la figure 8 n'induit pas de projection de l'ontologie sur le lexique [37], mais induit une projection du lexique sur l'ontologie, si l'on considère le fait que la relation "is-a" est transitive (projection modulo l'axiome de transitivité pour la relation "is-a").

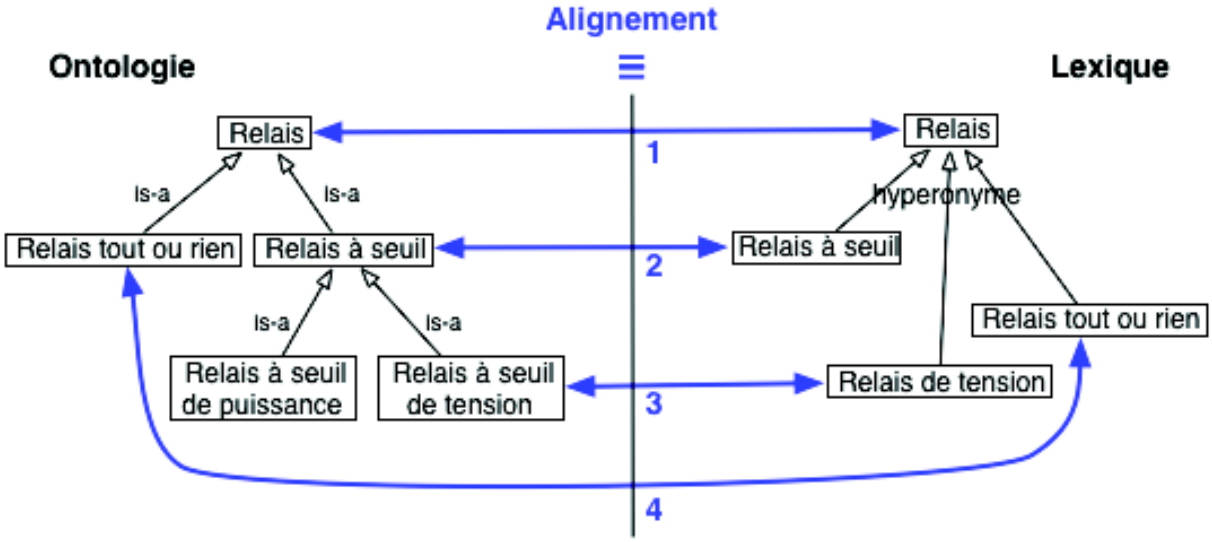


FIGURE 8 – Exemple d'alignement

Les contraintes précédentes peuvent toutefois être utilisées localement pour :

37. En effet, "relais à seuil" et "relais à seuil de tension" sont liés par la relation "is-a" dans l'ontologie, mais leurs images "relais à seuil" et "relais de tension" dans le lexique ne sont pas liées par la relation d'hyperonymie.

– déduire des relations dans l'ontologie à partir de celles du lexique, et inversement, si l'on dispose d'un alignement de bonne qualité (par exemple une relation d'hyperonymie entre "relais à seuil" et "relais de tension dans le lexique de la figure 8),
– identifier des problèmes structurels dans l'ontologie ou le lexique,
– renforcer des liens s'ils vérifient des contraintes (par exemple les liens 1 et 2 de la figure 8),
– proposer des liens probables qui vérifient des contraintes,

2.4 Création d'un alignement

Un alignement ontologie-lexique peut être créé de façon manuelle ou collaborative, en utilisant une plate-forme dédiée (par exemple Jibiki que nous présenterons après). Cependant, les objets considérés étant de taille importante, faisant éventuellement intervenir un grand nombre de langues et sujets à des modifications fréquentes, le recours à des processus automatiques pour créer des alignements est nécessaire. Trois principaux problèmes sont identifiés : la découverte de correspondances, le traitement des multimots ne figurant pas dans le lexique, et la vérification de correspondances.

2.4.1 Découverte de liens étiquetés

Ce problème peut être considéré comme la phase d'initialisation d'un alignement. Il consiste à identifier toutes les liens plausibles entre l'ontologie et le lexique. L'enjeu est d'augmenter le rappel des applications utilisant l'alignement. Plusieurs stratégies peuvent être utilisées :
– la comparaison des chaînes de caractères des étiquettes de l'ontologie et des entrées du lexique si elles sont dans la même langue (un lien est proposée lorsqu'elles sont identiques ou suffisament proches),
– l'utilisation de classes de synonymes ou quasi-synonymes, si elles sont disponibles dans le lexique (par exemple les synsets de Wordnet),
– l'utilisation des contraintes sur les correspondances structurelles pour proposer des correspondances probables à partir de celles existantes.

2.4.2 Traitement des multimots

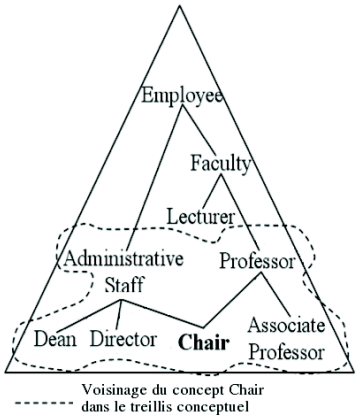
Il est fréquent que des concepts non lexicalisés interviennent dans des ontologies. Par exemple, l'ontologie Biocaster, dont nous avons parlé au paragraphe 1.1.2, comprend des concepts comme "HumanDesease" ou "BiologicallyActiveSubstance" qui ne correspondent sans doute à aucune entrée simple d'un lexique. Le problème de l'identification et du traitement des multimots, déjà important en traduction automatique, mérite d'être soulevé dans notre contexte. Les possibilités pour le résoudre n'ont cependant pas été explorées dans le cadre de cette thèse.

2.4.3 Vérification des liens

Cette tâche consiste à modifier le score de confiance d'un lien en fonction de sa vraisemblance. Le cas le plus fréquent, lorsque l'alignement est initialisé avec des comparaisons de chaînes, est qu'un élément de l'ontologie soit lié à plusieurs sens du même mot. Le cas contraire est également possible. En effet, les éléments d'une ontologie peuvent être vus comme les nœuds ou

arcs étiquetés (id, λ) d'un graphe (nœuds pour les concepts et instances, arcs pour les relations). On peut alors avoir deux éléments distincts (id_1, λ) et (id_2, λ) portant la même étiquette et donc initialement liés à un même sens du mot λ.

Si les étiquettes ne sont pas toujours suffisantes pour préciser le sens porté par un élément dans une ontologie, ce sens est toujours précisé par le voisinage de l'élément dans l'ontologie. Le problème de la vérification des liens entre ontologie et lexique se rapproche alors de la désambiguïsation en sens de mots du TALN (*Word Sense Disambiguation, WSD*). Ici, le contexte pour désambiguïser un élément de l'ontologie est l'ensemble des lexies (au sens de la définition 1 de la section 2.1) alignées avec le "voisinage" de l'élément dans l'ontologie. Une notion de contexte est définie par Espinoza et al. (2008). La figure 2.4.3 illustre les éléments d'une ontologie que l'on considère dans le voisinage pour une classe, une relation ou une instance. Il est maintenant possible d'adapter nombre de techniques, utilisées pour la désambiguïsation "classique" en sens de mots, à la vérification automatique de liens ontologie-lexique. Nous présentons dans la section 4.4 du chapitre 4 un algorithme inspiré de Lesk (1986) qui montre la validité de cette approche.



(a)

Voisinage	Élément de l'ontologie		
	Classe	Relation	Instance
Sous-classes et super-classes	✓		✓
Sous-relations et super-relation		✓	✓
Domaine/but		✓	
Frères	✓	✓	✓
Instances	✓	✓	
Attributs	✓	✓	✓

(b)

FIGURE 9 – Notion de voisinage pour les éléments d'une ontologie

3 Multilinguisation d'ontologies

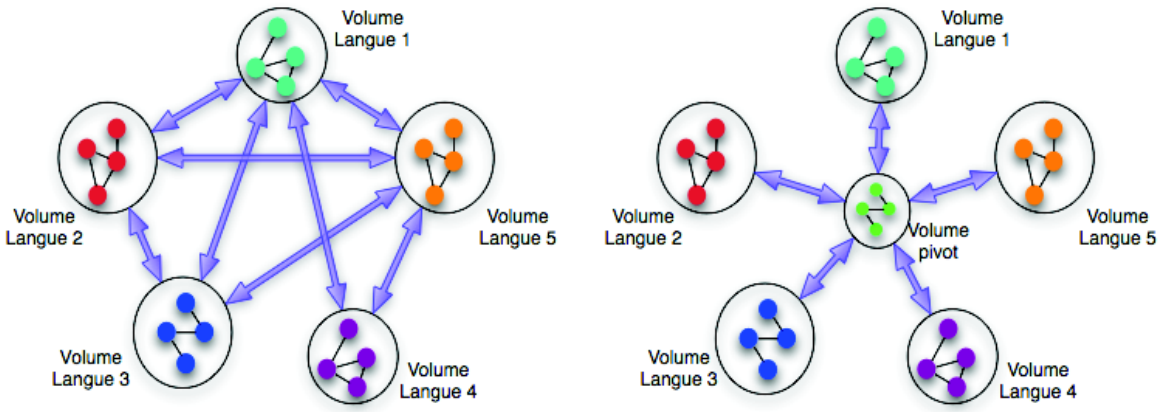
3.1 Principe

Nous utilisons une lexicalisation externe pour ajouter des informations multilingues à une ontologie. Il s'agit alors d'insérer l'ontologie source au sein d'une architecture lexicale multilingue. En format papier, un dictionnaire bilingue constitue un volume à part entière. En format informatique, il est préférable de considérer un dictionnaire bilingue comme deux volumes monolingues dont les entrées sont liées par des relations de synonymie. C'est pourquoi on s'intéresse non pas aux entrées classiques, mais aux lexies qui correspondent aux différents sens des mots.

Un ensemble de dictionnaires bilingues forme une architecture lexicale multilingue où chaque volume monolingue est lié aux autres, comme illustré dans la figure 3.10(a).

Une autre approche, illustrée par la figure 3.10(b), consiste à utiliser un lexique pivot comme "porte d'accès" vers les autres volumes. La traduction d'une langue à l'autre se fait alors via le lexique pivot et on fait une économie importante dans le nombre de correspondances à maintenir entre les volumes [38]. Le schéma utilisé pour relier les lexiques entre eux est communément appelé la *macrostructure*. Ce terme fait écho à la notion de *microstructure* qui dénote l'organisation interne des lexiques. Le pivot peut être :

1. un ensemble abstrait d'axies (*acceptions interlingues*), indexant simplement les liens entre les langues, comme dans le projet Papillon (Sérasset et Mangeot (2001)),
2. le lexique d'un interlingua, comme dans les systèmes de traduction automatique ATLAS-II (Uchida (1989)), PIVOT (Muraki (1989)), ou plus récemment UNL (Uchida et al. (1999)),
3. les concepts d'une base de connaissances ou d'une ontologie comme dans les systèmes de traduction KBMT (Nirenburg (1989)), KANT (Nyberg et Mitamura (1992)), ou Mikrokosmos (Beale et al. (1995)).



(a) Ensemble de dictionnaires bilingues

(b) Architecture multilingue à pivot

FIGURE 10 – Architectures de ressources lexicales multilingues

Nous présentons maintenant des exemples de ressources et d'architectures multilingues illustrant les différents types de pivot.

3.2 Bases et architectures lexicales multilingues

3.2.1 Le projet Papillon

Lancé en 2000, le projet Papillon (Mangeot et al. (2003)) était initialement dédié à la construction d'une base lexicale multilingue pour l'allemand, l'anglais, le français, le japonais, le lao, le malais, le thaï, le vietnamien et le chinois. Ce projet s'appuyait sur le principe de construction collaborative.

38. pour n volumes, $O(n^2)$ avec des dictionnaires bilingues et $O(n)$ avec un lexique pivot.

La macrostructure de la base par acceptions interlingues Papillon-NADIA a été initialement définie par Sérasset (1994). Les lexies composant les dictionnaires monolingues sont reliées par le biais d'un volume pivot composé d'axies. Ces axies peuvent être vues comme la réification de liens n-aires entre lexies de différentes langues. Les problèmes d'équivalence lexicale entre les langues (par exemple les différences de spécification entre les langues) sont traités au niveau des axies comme illustré dans la figure 11. Ces liens entre axies, tout comme les liens lexies-axies, ne doivent pas être considérés comme des liens ontologiques. Ils ne structurent pas le volume d'axies et indiquent simplement que deux acceptions reliées peuvent être utilisées comme traductions l'une de l'autre.

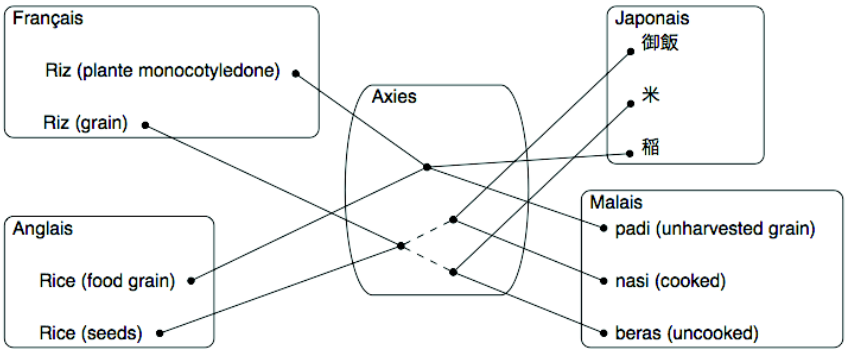


FIGURE 11 – Liens entre axies dans la base Papillon-NADIA

Les données sont toujours accessibles et modifiables sur le site du projet [39], mais le nombre de contributeurs est resté décevant. Le manque d'implication des contributeurs a selon nous deux causes principales :

1. La contribution à des articles de dictionnaires est moins valorisante que la contribution à des articles d'encyclopédie comme dans Wikipedia (du fait de leur taille et de leur visibilité).
2. Les contributeurs doivent intervenir directement au niveau des axies, ce qui rend la tâche plus complexe (nécessité de comprendre le concept, sensation de travail sur n langues).

Le projet Mot à Mot lancé en 2009 (Mangeot et Nguyen (2009)) reprend l'architecture de Papillon pour la création d'une base lexicale pour des langues asiatiques peu dotées (Khmer, Vietnamien, Lao). Deux solutions sont proposées pour réduire les freins à la contribution :

1. le recours à des jeux sérieux pour motiver les contributeurs,
2. la manipulation (et création) transparente des axies à partir de liens bilingues qui permet aux contributeurs de travailler sans se soucier de l'architecture à pivot.

Les bases lexicales et les interfaces des projets Papillon et Mot à Mot sont supportées par la plate-forme générique Jibiki que nous présenterons au paragraphe 3.3.3.

3.2.2 Wordnets et EuroWordnet

Wordnet (Fellbaum (1998)) est une base lexicale de l'anglais développée à l'université de Princeton depuis 1985 et toujours maintenue aujourd'hui. Les composants essentiels de Wordnet sont les synsets, des ensembles de mots "synonymes" qui peuvent porter le même sens et

39. `http://www.papillon-dictionary.org/`

être substitués dans au moins un contexte. Il ne s'agit donc pas de vrais synonymes, ni même souvent de quasi-synonymes, mais d'ensembles de mots (lemmes) qui partagent une ou plusieurs composantes sémantiques (par exemple : evaluate, judge, weight, estimate) sans même avoir nécessairement le même cadre de valence sémantique.

La version 3 de Wordnet, disponible depuis décembre 2006, rassemble 206 941 sens de mots dans 117 659 synsets, pour environ 90 000 mots-lemmes. Les synsets sont accompagnés de définitions, d'exemples, et organisés au sein d'un réseau lexical par des relations linguistiques ou sémantiques (hyponymie, antonymie, meronymie, etc.).

Plusieurs projets sont en cours pour développer des ressources similaires dans d'autres langues (on les appelle également des Wordnets). L'association Global Wordnet [40] recense des projets pour le développement de ressources dans plus de cinquante langues. Les Wordnets sont souvent développés à partir du Princeton Wordnet et reliés à une version de ce dernier. En particulier, le projet EuroWordnet (Ellman (2003)), achevé en 1999, propose des Wordnets reliés dans une base lexicale multilingue pour 8 langues européennes (allemand, espagnol, italien, anglais, français, néerlandais, tchèque et estonien). Les données de cette ressource sont payantes. Le nombre de mots, de synsets, de liens translingues et le prix de chaque volume sont donnés dans l'annexe D. Le Wordnet du néerlandais, le plus étendu développé lors du projet, rassemble 70 201 sens de mots dans 44 015 synsets.

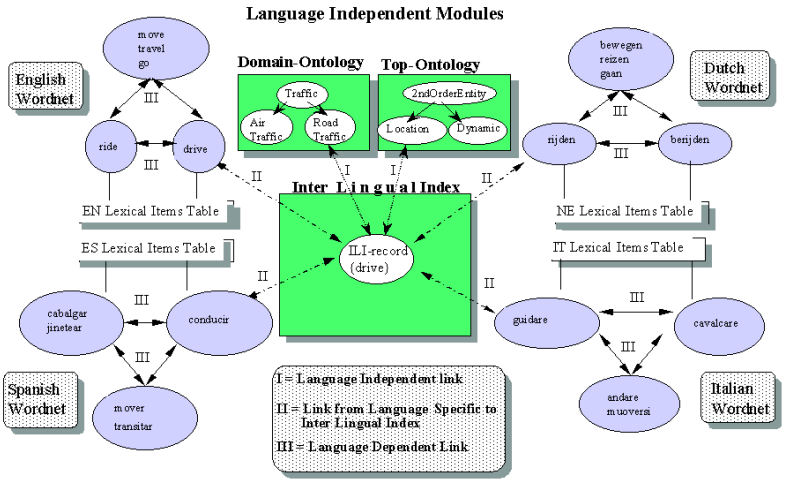
L'architecture de la base multilingue est présentée dans la figure 3.12(a). Les synsets des différentes langues sont liés par un index interlingue (*ILI*). L'ILI est un vocabulaire non structuré composé de synsets. Il a été initialisé avec les synsets du Princeton Wordnet 1.5 et étendu au besoin lorsque des langues contenaient des mots sans équivalent anglais. Les synsets de chaque langue sont liés aux synsets de l'ILI grâce à des relations présentes dans Wordnet, distinguées par le préfixe "`EQ_`" (`EQ_NEAR_SYNONYM`, `EQ_HYPONYM`, `EQ_MERONYM`, etc.). Ainsi, les problèmes d'équivalence lexicale entre les langues ne sont pas traités au niveau du pivot, mais au niveau des liens entre les volumes monolingues et le pivot, comme illustré dans la figure 3.12(b). L'ILI est également lié à une ontologie de haut niveau (*top-ontology*) rassemblant 63 concepts considérés comme indépendants de la langue (objet, substance, etc.) et à une ontologie rassemblant des classes thématiques (sports, sports aquatiques, armée, etc.). Le but ici est de structurer le lexique et pas d'enrichir les ontologies, mais d'autres ontologies que l'on souhaiterait multilingualiser peuvent prendre place de la même façon dans la base lexicale. C'est la stratégie qui été adoptée dans le projet KYOTO (présenté au paragraphe 1.5) où les éléments de l'ontologie sont liés aux synsets de différentes langues via un ensemble d'axies, équivalent à l'ILI.

3.2.3 UNL (Universal Networking Language)

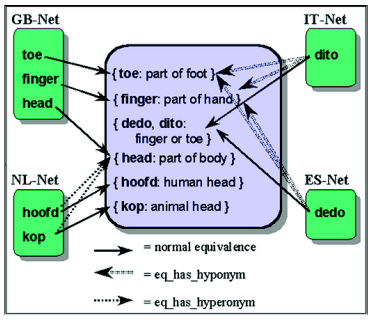
Le terme UNL recouvre trois choses différentes (Uchida et al. (1999), Boguslavsky et al. (2005), Boitet et al. (2007)) :

- le projet international UNL, lancé en novembre 1996 par l'UNU (Université des Nations Unies) à Tokyo [41] ;
- le langage UNL, qui est un langage pivot "anglo-sémantique" d'hypergraphes, et pas une langue humaine naturelle ou construite (comme l'espéranto) ;
- un format de documents multilingues, alignés au niveau de la phrase, intégré à HTML (avec des variantes XML). Les "fichiers" UNL sont utilisés depuis 2008 (projet Unesco-EOLSS-UNL) comme des fichiers "compagnons" de documents XML, contenant donc les graphes

40. `http://www.globalwordnet.org`
41. http ://www.undl.org/



(a) Architecture multilingue



(b) L'index interlingue (ILI)

FIGURE 12 – Composants d'EuroWordnet

UNL comme des annotations "externes".

Le langage UNL représente le sens d'un énoncé dans une langue par une structure sémantique abstraite (un hyper-graphe) d'un énoncé anglais équivalent (d'où la qualification de langage "anglo-sémantique"). Un arc d'un hypergraphe UNL porte une relation sémantique prise dans un ensemble de 41 relations (agt, obj, aoj, pos, pls, mea, cag...). Un nœud d'un hyper-graphe UNL porte : (1) soit un lexème UNL (appelé UW pour *Universal Word*) et des attributs sémantiques interlingues qui suivent le symbole @ (temps abstrait, aspect, nombre, détermination, thème, emphase, etc.), (2) soit un sous-graphe UNL (d'où le nom d'hypergraphe) appelé *scope* et défini comme l'ensemble des arcs étiquetés par le numéro du dit scope, et les nœuds afférents à ces arcs. Un scope doit être connexe par arcs (en négligeant l'orientation des arcs). Chaque graphe UNL, et chacun de ses scopes éventuels, possède un unique nœud d'entrée, marqué de l'attribut `@entry`. Les relations et attributs disponibles dans UNL sont détaillés dans l'annexe E. La figure 13 donne un exemple de graphe UNL simple (sans sous-graphe) tiré de (Boitet (2003)). Un exemple plus complexe est donné dans la section 5 de l'annexe E.

Le vocabulaire du langage UNL est constitué de lexies interlingues appelées UW (*Universal Words*). Idéalement, elles déterminent de manière non ambiguë une acception existant dans au moins une langue. Les UW sont construits pour représenter les acceptions des mots d'un ensemble de langues (et pas seulement les sens de mots). On trouve par exemple des UW distincts pour "affection" et "desease" qui portent le même concept en anglais.

La grammaire des UW est donnée en annexe E.3. Une UW est composée de :

- un mot-vedette, si possible dérivé de l'anglais, qui peut être un mot, une expression ou encore un groupe syntagmatique,
- une liste de restrictions, utilisant les relations sémantiques d'UNL, et servant à délimiter le sens de mot précis porté par l'UW.

Exemples :

- `book(icl>thing)` et `book(icl>do,agt>human,obj>thing)` pour lesquels le sens de l'UW est précisé par des restrictions ;
- `ikebana(icl>flower_arrangement)` dont le mot-vedette a été importé du japonais ;

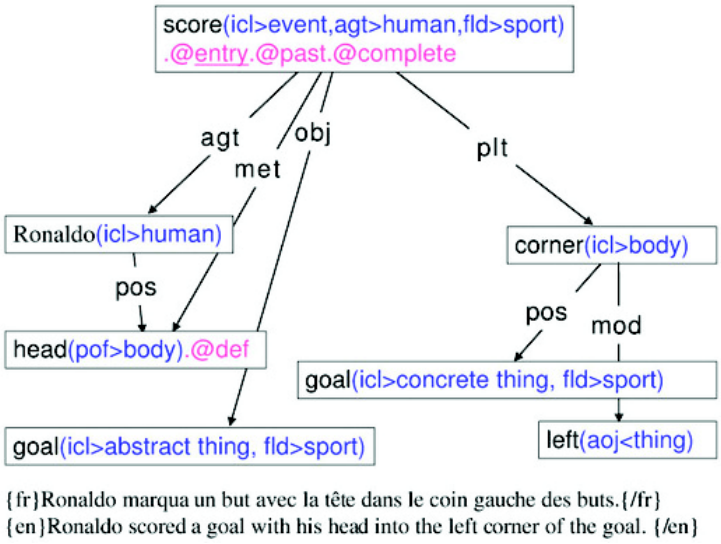


FIGURE 13 – Exemple de graphe UNL simple

– `go_down` dont le mot-vedette est une expression et dont le sens n'a pas besoin d'être précisé par des restrictions.

Les UW sont organisés dans un réseau sémantique nommé UNLKB (UNL Knowledge Base). Ce réseau utilise les 41 relations sémantiques disponibles dans UNL pour établir des liens pondérés entre les UW. Les relations `icl` (*included in*), `iof` (*instance of*) et `equ` (*equivalent to*) structurent UNLKB en une taxonomie. De plus, UNLKB possède des propriétés d'héritage et de remplacement : une UW hérite des propriétés d'UW plus générales et une UW dans une phrase peut être remplacée par une UW plus générale pour résoudre des problèmes de sous-spécification entre langues.

Par commodité, des schémas d'UW (*templates*) sont utilisés pour représenter tous les UW qui ont les mêmes restrictions. Par exemple `uw(iof>person)` représente `John(iof>person)` et toutes les autres personnes de la base. UNLKB fournit la sémantique et la connaissance linguistique sur les UW. Elle permet en outre de vérifier les graphes UNL car les relations utilisées doivent être conformes aux relations possibles déclarées dans UNLKB. UNLKB peut faire penser à une ontologie, mais elle n'est pas dotée de sémantique formelle et ne fournit aucune garantie de consistance logique. Nous la qualifions donc de base de connaissances "pré-ontologique". Les quatre premier niveaux de UNLKB sont présentés dans la table de l'annexe E.4.

Dans un système de TA basé sur UNL, les UW sont liés aux lexies des langues naturelles via des dictionnaires bilingues ou au sein d'une base lexicale multilingue comme celle présentée dans le paragraphe 3.3.3. Contrairement à l'ILI d'EuroWordnet, le lexique UNL (l'ensemble des UW) est structuré (notamment par des liens de raffinement). Cela permet de relier les UW aux langues en utilisant une seule sorte de correspondance (une équivalence traductionnelle), et de traiter les problèmes d'équivalence lexicale entre les langues au niveau du pivot. Ainsi, la sous-spécification du mot espagnol "dedo" par rapport aux mots anglais "finger" et "toe", illustrée dans la figure 3.12(b) pour EuroWordnet, pourrait être résolue comme dans la figure 14 pour UNL.

Il est possible de créer un UW lorsqu'aucun ne convient pour représenter un sens de mot d'une langue. L'ensemble d'UW maintenu par la fondation UNL est accessible par l'interface

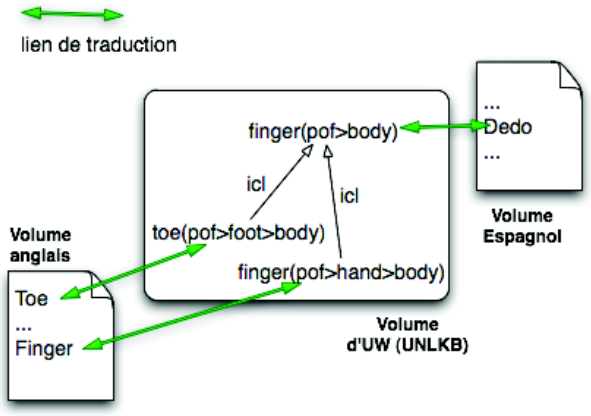


FIGURE 14 – Liens de raffinement entre UW

UWgate [42]. Au 16 août 2011, l'interface annonce des liens entre les UW et 19 langues (table 1).

Langue	Liens		Langue	Liens
arabe	297 032		arménien	7326
bengali	12 477		chinois simplifié	260 157
allemand	258 333		anglais	459 360
espagnol	279 198		français	121 537
hindi	221 239		indonésien	268 393
italien	1634		japonais	320 200
coréen	240 310		letton	7936
mongole	19 181		portugais	278 122
russe	150 776		thai	52 895
vietnamien	251 016			

TABLE 1 – Liens entre UW et sens de mots disponibles en décembre 2012 sur UWgate

La qualité de ces ressources mérite cependant d'être étudiée. Par exemple, le dictionnaire Français-UW semble avoir été construit par simple traduction des mots vedettes des UW. Ainsi, le mot "banque" est lié à toutes les UW dont le mot vedette est "bank".

Un autre ensemble de 207 000 UW a été créé par le consortium U++ [43] à partir des sens de mots de Wordnet 2.1 (Boguslavsky et al. (2007)). Ces lexies, dites UW++, sont consultables en ligne [44] et peuvent bénéficier des ressources multilingues basées sur Wordnet.

La plate-forme PIVAX (Nguyen et al. (2007)) a été développée pour accueillir les données du consortium U++. Elle est utilisable pour toute base lexicale à pivot. Sa macrostructure, illustrée par la figure 15, permet la coexistence de plusieurs dictionnaires par langue. Une couche d'acceptions monolingues (*axèmes*) a été ajoutée pour rendre compte de l'équivalence entre lexies de volumes différents au sein d'une même langue. Techniquement, les axèmes ont la même structure que les axies. Les UW constituent bien un lexique interlingue, mais n'occupent pas une place centrale (de pivot) dans l'architecture d'une base PIVAX. Le choix a été fait de passer par

42. http://www.undl.org/uwgate/
43. http://www.unl.fi.upm.es/consorcio/index.php
44. http://www.unl.fi.upm.es/dicweb

des axies pour représenter les liens entre les langues et de considérer les volumes d'UW comme n'importe quels volumes monolingues. Cela permet la présence de plusieurs volumes d'UW, facilite le versionnage des dictionnaires, et permet éventuellement de changer d'interlingua en conservant les liens de traduction entre les langues.

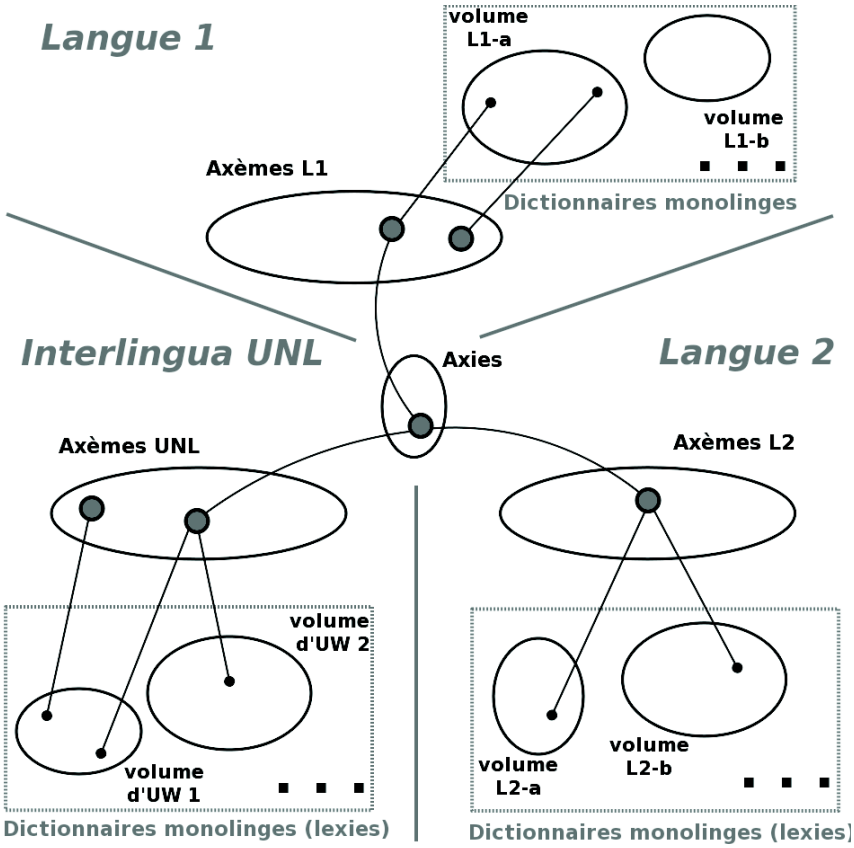


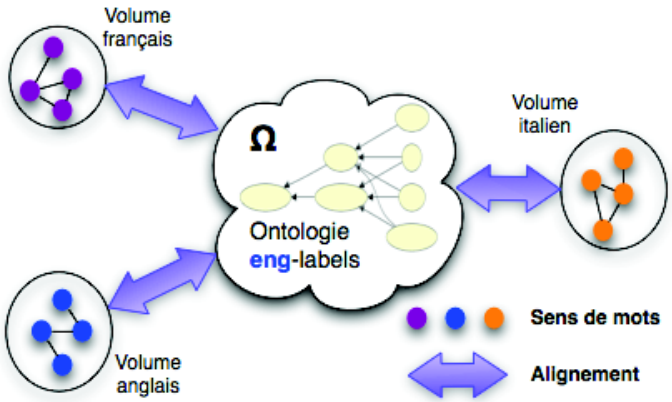
FIGURE 15 – Macrostructure de la plate-forme PIVAX

3.3 Insertion d'une ontologie dans une base lexicale multilingue

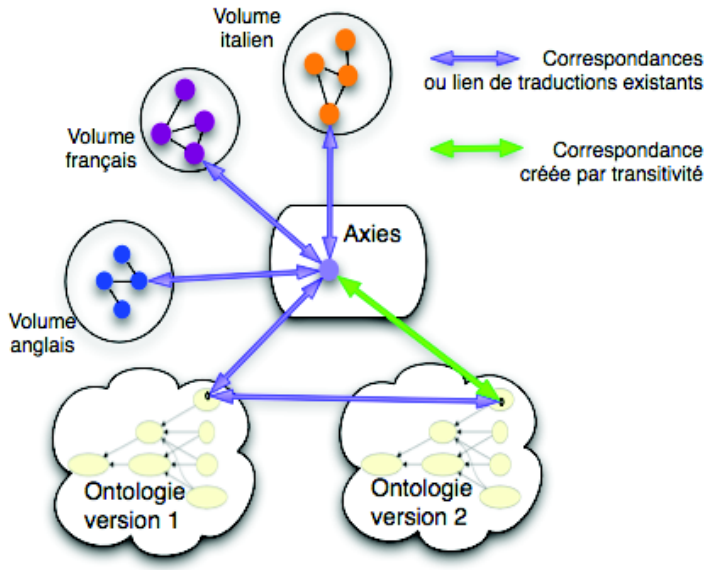
3.3.1 Principe utilisé dans OMNIA

Lorsqu'une ontologie est alignée avec des volumes de différentes langues, elle est *de facto* considérée comme un pivot conceptuel. Dans le cas d'OMNIA, l'ontologie est utilisée comme pivot pour stocker les informations extraites de textes dans différentes langues, et éventuellement les restituer dans d'autres langues. Cependant, comme dans le cas des UW sur la plate-forme PIVAX, il n'est pas pertinent de placer une ontologie que l'on souhaite multilingualiser au centre d'une architecture lexicale multilingue (figure 3.16(a)). En effet, les ontologies sont sujettes à de fréquentes révisions et extensions, et changer la version d'une ontologie utilisée comme pivot imposerait une nouvelle version de toute l'architecture multilingue.

Nous avons donc choisi de considérer l'ontologie comme un volume (contenant des symboles de concepts, d'instances et de relations) et de le lier aux autres langues via des axies. Ainsi, une nouvelle version de l'ontologie peut être insérée dans l'architecture multilingue en utilisant un alignement avec l'ancienne version (figure 3.16(b)). On conserve donc l'architecture existante, avec la création de liens étant nécessaire seulement pour les concepts et relations ajoutés dans la nouvelle ontologie, et la possibilité d'utiliser n'importe quelle version de l'ontologie dans les applications. De plus, cela permet l'utilisation conjointe de plusieurs ontologies, correspondant à des modèles (ou "ponts de vue") différents.



(a) Ontologie comme pivot



(b) Ontologie alignée via des axies

FIGURE 16 – Ontologie dans une architecture multilingue

Pour le projet OMNIA, nous avons choisi d'utiliser les ressources multilingues du projet UNL qui présentent selon nous les avantages suivants.

– Un lexique pivot d'UW constitue un vocabulaire désambiguïsé à l'aide de relations pertinentes au niveau interlingue et structuré dans une base de connaissance "pré-ontologique"

qui peut faciliter son alignement automatique avec une ontologie.
- Le projet UNL rassemble des équipes dans le monde entier pour la création de ressources et d'applications multilingues. Des ontologies alignées avec un lexique d'UW peuvent bénéficier de toutes les avancées du projet.
- UNL ne fournit pas seulement un vocabulaire interlingue, mais un interlingua formel permettant de représenter le sens des énoncés de toute langue naturelle. Lier une ontologie à un ensemble d'UW de grande taille ouvre donc des perspectives pour l'alignement de constructions linguistiques (à un niveau interlingue) et de constructions logiques dans l'ontologie. Cela permettra ultérieurement d'aller vers une représentation logique du sens des textes et la construction de descriptions riches dans une ontologie, en s'inspirant par exemple de Cardeñosa et al. (2008).

Nous verrons également que l'utilisation d'un vocabulaire interlingue permet de considérer une ontologie comme un paramètre lors d'un processus d'extraction de contenu dans des textes multilingues. On passe alors par une phase d'annotation interlingue des textes, décrite dans le chapitre 4 de cette thèse.

Pour une ontologie dont les étiquettes sont basées sur une langue naturelle L1, notre démarche consiste donc à utiliser :

1) un dictionnaire UW-L1 pour créer des correspondances lexicales possibles entre UW et éléments de l'ontologie,
2) la structure du lexique d'UW et celle de l'ontologie (liées par des correspondances structurelles) pour désambiguïser les correspondances lexicales,
3) le lexique d'UW (et possiblement l'interlingua UNL) comme passerelle vers les langues naturelles.

Ici encore, le lexique d'UW n'est pas placé au centre de l'architecture multilingue. Tous les volumes sont liés par un ensemble d'axies.

Nous présentons maintenant deux implémentations possibles permettant d'aligner ontologies et ressources lexicales multilingues. La première est une ontologie OWL représentant le schéma de l'architecture multilingue, mise à jour à l'aide d'un ensemble de règles SWRL. La seconde est la plate-forme Jibiki, qui permet d'importer tout volume représenté en XML, en particulier des ontologies OWL.

3.3.2 Implémentation avec une ontologie dédiée

Une expérience menée pour l'implémentation de bases lexicales multilingues est l'utilisation d'une ontologie dédiée. On retrouve ici l'idée du schéma linguistique de LIR exprimé en OWL (paragraphe 1.3), mais à une autre échelle. En effet, l'ontologie linguistique de LIR a une taille de l'ordre de celle de l'ontologie de domaine qui lui est associée (quelques centaines, voire milliers d'entrées). On souhaite ici représenter et utiliser toute une base lexicale multilingue dans une ontologie OWL (centaines de milliers, voire millions d'entrées). Les premières expériences pour le passage à l'échelle ont été menées avec des données multilingues du consortium U++ (79 656 UW liées à 44 173 entrées anglaises, 23 598 entrées françaises et 21 998 entrées russes, soit 169 425 entrées au total) et l'éditeur d'ontologies Protégé 3.4.

L'ontologie lexicale est composée de classes représentant les volumes monolingues et d'une classe d'axies. Afin de simplifier les expériences préliminaires, nous ne considérons qu'une relation entre les entrées de dictionnaires (l'équivalence lexicale *LexEq*) et aucune entre les axies. Les instances des volumes monolingues peuvent être liées entre elles par la relation `LexEq` et liées

aux axies par la relation `AxLexEq`. La microstructure des volumes monolingues est représentée par les propriétés définies sur la classe associée.

Le grand nombre d'instances de l'ontologie rend nécessaire l'utilisation de bases de données pour les stocker. Protégé 3.4 offre cette possibilité (*database backend*) avec des performances relativement convaincantes. À partir d'un fichier OWL représentant les UW et leurs liens vers les entrées des autres langues, la base de données est créée en une dizaine de minutes sur un ordinateur de bureau (CPUCore 2 Duo 2,53GHz). Malgré quelques ralentissements, la navigation et la modification des données avec l'interface graphique restent utilisables. La figure 17 montre une capture d'écran de l'éditeur Protégé pour l'édition d'UW.

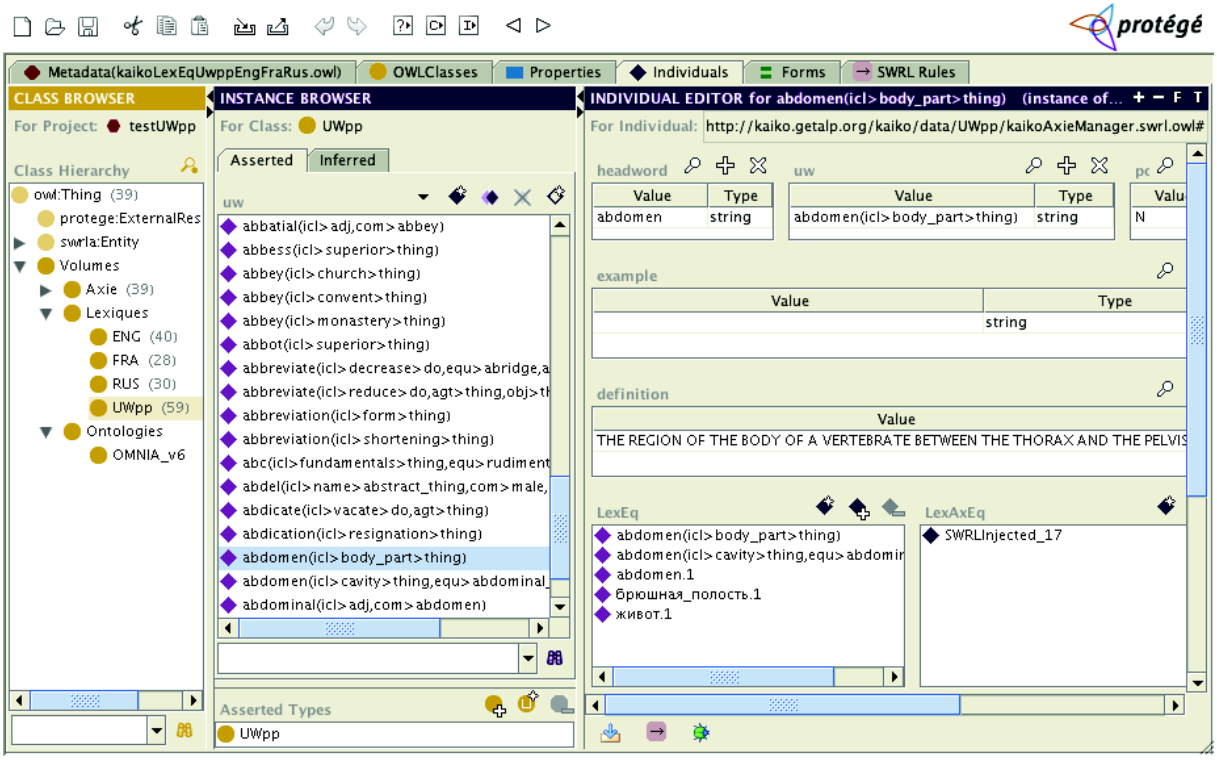


FIGURE 17 – Interface de Protégé pour la modification d'un UW

Dans la lignée du projet Mot à Mot, nous avons souhaité permettre l'édition des données sans nous soucier de l'architecture à pivot. Cela est rendu possible grâce à un ensemble de règles SWRL qui créent les liens avec les axies (et au besoin les axies elles-mêmes) à partir des liens entre volumes monolingues. Les règles utilisées sont les suivantes :

1. La relation d'équivalence lexicale est symétrique :
   `LexEq(?lex1,?lex2)` → `LexEq(?lex2,?lex1)`
2. Si deux lexies sont liées à la même axie, elles sont équivalentes :
   `LexAxEq(?lex1,?ax)` ∧ `LexAxEq(?lex2,?ax)` → `LexEq(?lex1,?lex2)`
3. Si deux lexies sont équivalentes et si l'une est liée à une axie, alors la seconde l'est aussi :
   `LexEq(?lex1,?lex2)` ∧ `LexAxEq(?lex1,?ax)` → `LexAxEq(?lex2,?ax)`
4. Si deux lexies sont équivalentes, et si aucune d'elles n'est liée à une axie, on crée une axie pour une des lexies (ici celle dont l'URI est la première dans l'ordre lexicographique), et on lie les deux lexies à cette nouvelle axie :

```
LexEq(?lex1, ?lex2) ∧ abox:hasNumberOfPropertyValues(0,?lex1,LexAxEq) ∧
abox:hasNumberOfPropertyValues(0,?lex2,LexAxEq)
∧ abox:hasURI(?lex1,?uri1) ∧ abox:hasURI(?lex2, ?uri2) ∧ swrlb:lessThan(?uri1,?uri2)
∧ swrlx:createOWLThing(?ax, ?lex1)
→ Axie(?ax) ∧ LexAxEq(?lex1,?ax) ∧ LexAxEq(?lex2, ?ax)
```

5. Si une lexie est liée à deux axies différentes, ces axies sont candidates pour être fusionnées :
   `LexAxEq(?lex,?ax1) ∧ LexAxEq(?lex,?ax2) → sameAx(?ax1,?ax2)`
6. Si deux axies sont candidates pour être fusionnées, on en place une arbitraire (celle dont l'URI est la première dans l'ordre lexicographique) dans une classe d'axies potentiellement redondantes. Si l'on fait l'hypothèse forte qu'une lexie est liée à au plus une axie, on peut systématiquement supprimer toutes les instances de la classe `AxieRedondante` :
   ```
   sameAx(?ax1,?ax2) ∧ abox:hasURI(?ax1,?uri1) ∧ abox:hasURI(?ax2,?uri2)
   ∧ swrlb:lessThan(?uri1,?uri2)
   → AxieRedondante(?ax2)
   ```

Certaines règles pourraient être exprimées directement comme des axiomes OWL. Nous préférons cependant garder un schéma OWL le plus simple possible et exécuter les règles chaque fois que nécessaire pour les mises à jour. Nous avons choisi d'exécuter les règles SWRL à l'aide du moteur Jess, car c'est aujourd'hui le seul qui permet de créer des instances dans l'ontologie comme dans la règle 4 ci-dessus. Le pont entre les langages SWRL et Jess, ainsi que son intégration à Protégé, sont décrits par O'connor et al. (2005). La figure 18 montre l'onglet SWRLtab de Protégé qui permet d'éditer des règles et d'interagir avec Jess.

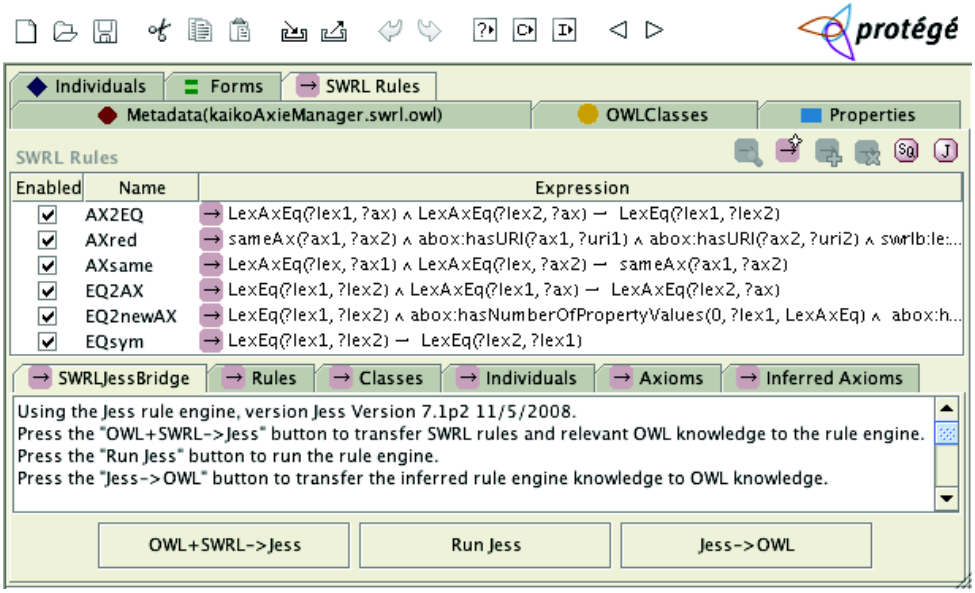


FIGURE 18 – SWRLtab avec les règles nécessaires au maintien de l'architecture lexicale à pivot

L'exécution des règles à partir de l'interface graphique de Protégé fonctionne correctement sur un petit nombre d'instances (par exemple dans le cas de lexiques dédiés à l'ontologie), mais ne termine pas si les 169 425 entrées de nos lexiques test sont chargées. Cette solution a donc été écartée dans le cadre du projet OMNIA. Des expériences complémentaires sont en cours pour tester si un passage à l'échelle est possible en utilisant l'API de Protégé.

Nous présentons maintenant la plate-forme de gestion de ressources lexicales Jibiki que nous avons décidé d'utiliser pour le projet OMNIA.

3.3.3 Implémentation dans la plate-forme Jibiki

La plate-forme Web Jibiki (Mangeot et Chalvin (2006), Sérasset et al. (2006)) permet d'importer, de consulter et d'éditer collaborativement des ressources lexicales de types et de structures très différents : lexiques monolingues, dictionnaires bilingues, bases lexicales multilingues ou bases terminologiques, à partir du moment où elles sont encodées en XML. Une instance de la plate-forme Jibiki implémente la macrostructure d'une architecture multilingue (ensemble de dictionnaires bilingues, architecture à pivot, etc.). Typiquement, la plate-forme PIVAX destinée à accueillir les données du consortium U++ a été construite à partir de Jibiki.

L'environnement permet de manipuler les ressources dans leur format d'origine sans modification, grâce à un système de pointeurs communs (*CDM, Common Dictionary Markup*, Mangeot (2002)). Les volumes monolingues sont importés grâce à un fichier de métadonnées XML qui décrit la microstructure du volume. Le fichier de métadonnées fournit les chemins Xpath correspondant aux pointeurs communs. Des pointeurs spécifiques peuvent être ajoutés au besoin dans les métadonnées. Une interface générique de consultation et d'édition est paramétrée à partir d'un schéma XML décrivant la structure des entrées (Mangeot et Thévenin (2004)).

Comme dit au paragraphe précédent, dans le but d'inclure une ontologie dans la plate-forme, nous la considérons essentiellement comme un volume contenant des symboles de concepts, d'instances et de relations. La figure 19 présente le fichier de métadonnées utilisé dans OMNIA pour importer les concepts de l'ontologie sur la plate-forme Jibiki.

```
<volume-metadata>
<comments>PIVAX OWL - OMNIA Ontology for OMNIA project</comments>
  <cdm-elements>
    <cdm-volume    xpath="/rdf_RDF"/>
    <cdm-entry    xpath="/rdf_RDF/owl_Class"/>
    <cdm-entry-id    xpath="/rdf_RDF/owl_Class/@rdf_ID"/>
    <cdm-headword    d:lang="owl"
xpath="/rdf_RDF/owl_Class/text()"/>
  </cdm-elements>
  <administrators>
    <user-ref    name="nguyenht"/>
  </administrators>
  <system    name="omnia">
    <copyright>Copyright by OMNIA project</copyright>
    <description>OWL Ontology dictionary</description>
    <organization>GETALP-LIG</organization>
    <adress/>
    <responsability>Christian.Boitet@imag.fr</responsability>
    <url_link>http://www.ellemme.org/</url_link>
  </system>
  <xmlschema-ref    xlink:href="pivax_local.xsd"/>
  <volume-ref    xlink:href="Pivax_owl-omnia.xml"/>
  <template-entry-ref    xlink:href="Pivax_owl-omnia-
template.xml"/>
</volume-metadata>
```

FIGURE 19 – Métadonnées Jibiki pour l'import des concepts d'une ontologie OWL

Les ressources peuvent être construites, corrigées ou modifiées en ligne par des communautés de contributeurs, bénévoles ou non. Il existe un système de gestion de différents statuts et niveaux de qualité des entrées (et des liens) ainsi que des contributeurs, de façon à indiquer la qualité de chaque information. Les entrées (ou liens) ont un score de 1 à 20 et un niveau de 1 à 5 étoiles, 5 étant une qualité certifiée par un spécialiste reconnu du domaine ou traducteur certifié. Les

contributeurs ont également un niveau de compétence noté de 1 à 5 étoiles et un "score par défaut" de 0 à 20. Cette idée d'associer un niveau (de une à cinq étoiles) et un score (de 0 à 20) à chaque objet est reprise du système d'exploitation de corpus de traductions SECTra_w (Huynh et al. (2008)).

Dans le cas d'OMNIA, Jibiki a été testée comme serveur de dictionnaire via son API REST. Il est intéressant d'utiliser la plate-forme directement comme serveur, car on bénéficie ainsi des versions les plus à jour des données. Des résultats concluants sur l'utilisation de cette plate-forme dans ce contexte sont donnés dans le chapitre 4.

Conclusion

Nous avons montré les avantages d'une approche de lexicalisation externe pour connecter des informations linguistiques à une ontologie. De la même façon, dans le cas multilingue, nous ne cherchons pas à ajouter les informations directement au sein de l'ontologie, mais à lier l'ontologie à une base lexicale multilingue existante. À cet effet, nous nous sommes concentré sur l'utilisation de bases lexicales multilingues à pivot, déjà mise en œuvre dans des systèmes de traduction automatique.

Dans la famille des ressources multilingues à pivot, notre choix s'est tourné vers l'interlingua UNL et son lexique d'UW, que l'on peut considérer comme un ensemble structuré de lexèmes interlingues constituant une "préontologie".

Nous utilisons un volume d'axies (acceptions interlingues) qui représentent des liens de traduction entre lexies de différentes langues. Pour lier une ontologie à cette architecture multilingue, nous considérons l'ensemble des URI de ses éléments comme un volume que nous relions aux autres langues (en particulier aux UW) via les axies. Ainsi, nous considérons le lexique formé d'UW++ comme un pivot linguistico-sémantique, l'ontologie comme un pivot conceptuel, et le volume d'axies comme un pivot abstrait, placé au centre de l'architecture multilingue. Le language d'ontologie OWL est commode pour représenter des lexiques et leurs microstructures, ainsi que la macrostructure d'une base lexicale multilingue. Les outils associés à OWL, que nous avons testés jusqu'à présent, présentent cependant des limites au passage à l'échelle, qui ont empêché leur utilisation directe pour la gestion de lexiques de grande taille. Nous nous sommes donc tourné vers la plate-forme de gestion de ressources lexicales Jibiki, qui permet l'import d'ontologies OWL et leur intégration dans l'architecture lexicale multilingue.

Nous présentons dans le chapitre suivant un processus d'annotation de textes par des UW. C'est l'étape préliminaire pour une extraction de contenu guidée par une ontologie alignée avec un lexique d'UW. Nous décrivons également le démonstrateur du projet OMNIA, qui met en œuvre un certain nombre de propositions de cette thèse.

Chapitre 4

Architecture pour la recherche translingue et l'extraction de contenu dans des objets multimédia

Sommaire

1 Architecture générale

1.1 Scénarios typiques d'utilisation

De nombreux entrepôts d'images accompagnées par des textes spontanés sont accessibles sur le Web. On pense notamment aux sites permettant aux internautes de partager leurs photos (FlickR, Picassa, Panimages, etc.) ou aux sites d'agences de presse (AFP, Belga News, etc.). Le contenu des images peut être considéré comme indépendant de la langue utilisée pour les décrire. Nous souhaitons permettre à un utilisateur d'effectuer des recherches parmi ces images, sous forme de requêtes libres dans sa langue préférée, et d'accéder à des images éventuellement décrites dans d'autres langues. Nous souhaitons également permettre à un utilisateur d'indexer ces images relativement à une hiérarchie de concepts personnelle ou pertinente pour des recherches sur un domaine précis (ontologie de domaine).

XRCE, le porteur projet OMNIA, avait retenu un scénario d'utilisation pré-presse, éditorial ou créatif. Dans ce scénario des auteurs d'articles ou de brochures souhaitent illustrer leurs documents à partir de bases d'images journalistiques ou libres de droits. Ils peuvent formuler leurs requêtes de façon classique, par mots-clés ou par un fragment de texte, ou même soumettre au système l'intégralité du texte à illustrer, ce qui pose des problèmes de passage à l'échelle. Le scénario proposé par XRCE était le suivant.

> " *Tony est l'unique employé du département de marketing pour une PME d'eau minérale appelée "Dundrum Waters" et basée en Irlande.* [...] *il doit être extrêmement organisé, et assumer seul les responsabilités d'une équipe de marketing complète. Il partage son temps entre la recherche, la planification, l'explication de ses projets à ses supérieurs, et la réalisation. S'il peut gagner du temps sur la recherche et la création, l'ensemble de son travail processus s'accélère, et il peut être fier quand les ventes commencent à grimper. Chaque fois qu'un nouveau produit prometteur est lancé, Tony doit concevoir et déployer une campagne en commençant pratiquement de zéro. Parfois, les employés du pôle de production lui laissent un grand degré de liberté, mais la plupart du temps, ils ont déjà une idée claire de ce qu'ils veulent. Il est actuellement en train de développer une campagne pour une nouvelle eau minérale qui sera distribuée partout en Irlande. Il doit créer rapidement un projet de brochure à faire valider par le pôle de production.* "

1.2 Utilisateurs

La principale catégorie d'utilisateurs visée par le système est constituée par les personnes recherchant des images pour illustrer des documents. De leur point de vue, le service proposé relève de la recherche d'information translingue, puisque la langue utilisée pour la requête peut différer de celle utilisée pour décrire les images proposées.

D'autres utilisateurs pourraient souhaiter disposer d'un processus d'indexation personnalisé, dont ils bénéficieraient lors de la phase de recherche. Ces utilisateurs pourraient soumettre leurs propres termes d'indexation au système, organisés sous forme d'ontologie, comme paramètre des processus d'indexation et de recherche.

Enfin, les utilisateurs qui partagent des images (de façon gratuite ou mercantile) ne sont pas nécessairement des "clients" directs du système. Nous supposons qu'ils décrivent les images dans des légendes, en utilisant leur langue préférée, sans aucune restriction (mots-clés, texte libre, etc.).

1.3 Schéma général

La figure 1 récapitule de façon simple l'utilisation du système.

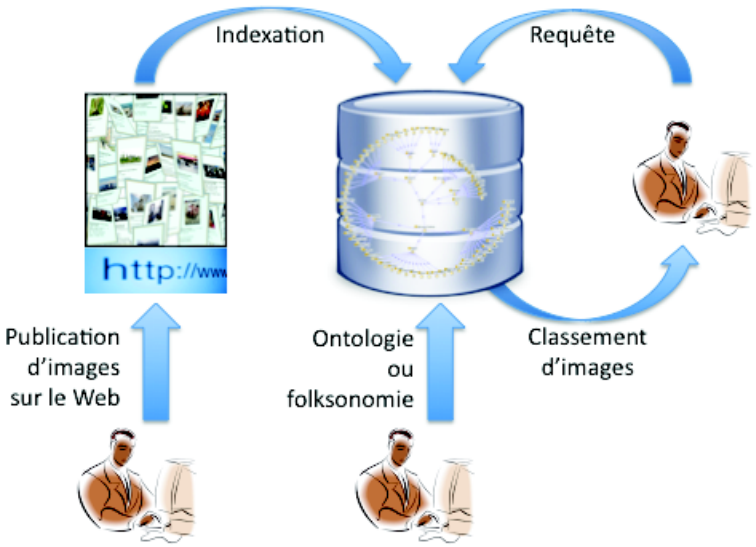


FIGURE 1 – Schéma simple d'utilisation

2 Annotation des textes pour l'extraction de contenu sémantique

2.1 Principe

Le développement de processus sémantiques d'extraction et de recherche d'information passe par la description des documents à l'aide d'ontologies partagées (indexation). Pour rendre significatif le nombre de documents décrits, il est nécessaire d'avoir recours à des processus automatisés. Dans ce cadre, l'annotation sémantique des textes, avec les éléments d'une ontologie, est une étape facilitant la création de descriptions des documents au sein de l'ontologie. Buitelaar et Declerck (2003) présentent les étapes de traitement linguistique pour l'annotation sémantique de textes libres :

1) analyse morphologique,

2) étiquetage des catégories grammaticales,

3) segmentation en groupes élémentaires (*chunking*),

4) analyse en structures de dépendances,

5) annotation en sens de mots.

L'annotation sémantique passe par une segmentation et une annotation linguistique des textes. Comme décrit dans le chapitre 3, nous utilisons des ontologies dont les aspects multilingues sont gérés via des alignements avec un lexique pivot d'UW. Nous proposons donc d'effectuer les annotations linguistiques des textes à un niveau interlingue, avec des UW.

Le but initial du projet UNL est de produire un (hyper)graphe "anglosémantique" pour représenter le sens d'un énoncé, potentiellement exprimé dans n'importe quelle langue naturelle.

Le projet prévoit donc la création d'enconvertiseurs et de déconvertisseurs, traduisant respectivement les textes d'une langue naturelle vers UNL et inversement, mais pas de processus d'annotation interlingue des textes au niveau des lexèmes ou de morceaux de phrases. Un tel processus, que l'on peut qualifier de lemmatisation interlingue ou d'annotation pré-sémantique des textes, peut constituer un préambule commun pour toute annotation sémantique ou extraction de contenu, quelle que soit la langue ou le domaine des textes considérés. Ainsi, la phase de désambiguisation en sens de mots, qui est la plus couteuse en calculs dans le processus d'annotation sémantique, est factorisée entre les langues et ne dois pas être renouvelée si l'ontologie utilisée pour l'annotation conceptuelle (considérée comme un paramètre du domaine) vient à être modifiée.

En outre, si l'on dispose d'un texte annoté par des UW ou de petits graphes (représentant par exemple des structures de dépendances), une ontologie arbitraire peut être passée en paramètre du système d'annotation ou d'extraction par le biais d'un alignement avec le volume d'UW.

Nous présentons maintenant le langage-Q qui fournit un formalisme pour la représentation des textes annotés par des graphes-Q et un moteur de règles de réécriture (constituant des systèmes-Q) permettant éventuellement d'implémenter les étapes du processus d'annotation.

2.2 Représentation de textes annotés : le langage-Q

Le langage-Q a été développé en 1967 par Alain Colmerauer à l'Université de Montréal. Colmerauer (1970) donne la description suivante :

"Un système-q est un ensemble de règles permettant de faire certaines transformations sur des graphes orientés. Chaque flèche d'un tel graphe est surmontée d'une expression parenthésée. Les transformations peuvent correspondre à une analyse, à une synthèse de phrase, ou à une manipulation formelle de ce genre. Il est possible d'utiliser le même système-Q pour décrire à la fois un processus et le processus inverse, comme par exemple l'analyse et la synthèse d'une même phrase."

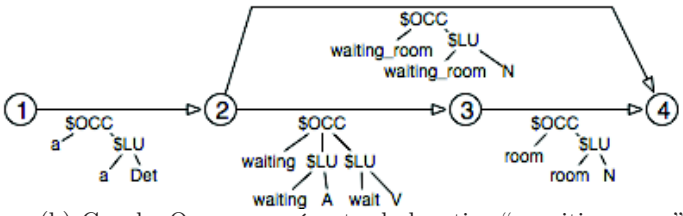
Dans ce formalisme, on appelle chaîne une suite d'arbres (expressions parenthésées). Le symbole + est utilisé comme séparateur comme dans l'exemple de la figure 2.a page 67. Les sommets d'un arbre portent chacun une étiquette formée d'un caractère arbitraire suivit d'une suite éventuellement vides de lettres et de chiffres. Un graphe-Q est un graphe de chaînes orienté, sans cycle, possédant un unique sommet d'entrée (sans arc entrant) et de sortie (sans arc sortant), dont les arcs sont étiquetés par des arbres. Les figures 2.b et 2.c donnent un exemple de tel graphe et de la syntaxe utilisée pour le décrire. Une chaîne ou un graphe-Q peuvent contenir des variables notées `<LETTRE>*<Chiffre>`. Les lettres A à F sont utilisées pour des variables représentant des étiquettes, I à N pour des arbres et U à Z pour des listes d'arbres (forêts).

Une règle-Q est une règle de réécriture composée d'une condition d'exécution et de deux chaînes (avec variables et de longueurs connues), une en partie gauche et une en partie droite. L'application d'une règle-Q à un chemin du graphe-Q (nécessairement de même longueur que la partie gauche de la règle) consiste à ajouter au graphe, entre les nœuds extrèmes du chemin, et pour chaque instanciation des variables vérifiant la condition, l'instanciation correspondante du chemin en partie droite. La figure 2.d donne un exemple de règle correspondant à une entrée de dictionnaire Anglais-UW. Un système-Q est un ensemble de règles-Q. L'application d'un système-Q à un graphe se déroule en deux phases. On commence par appliquer toutes les règles du système sur tous les chemins du graphe sur lesquels elles peuvent s'appliquer, y compris à ceux contenant des arcs ajoutés lors de cette phase. Si le processus converge, le résultat est indépendant de l'ordre dans lequel on applique les règles. On supprime ensuite les arcs du graphe qui ont été utilisés pour appliquer une règle et les chemins orphelins qui ne sont plus reliés à l'entrée ou à la sortie du graphe. Si la phase d'ajout ne converge pas, cela provoque une erreur

de débordement de mémoire, et on s'arrête en produisant l'entrée. Un traitement-Q est une suite de systèmes-Q.

```
$OCC(a)+$OCC(waiting)+$OCC(room)
```

(a) Chaîne pour la locution "a waiting room"



(b) Graphe-Q pour représenter la locution "a waiting room" lemmatisée

```
-01-$OCC(a,$LU(a,Det))-02-
-02-$OCC(waiting room,$LU(waiting_room,N))-04-
-02-$OCC(waiting,$LU(waiting,N),$LU(wait,V))-03-
-03-$OCC(room,$LU(room,N),$LU(room,V))-04-
```

(c) Code correspondant

```
$OCC(A*,U*,$LU(room,N),V*)
==
$OCC(A*,U*,$LU(room,N,
$UW(room[icl>position>thing],room[icl>area>thing],room[icl>gathering>thing])),V*)
```

(d) Règle-Q ajoutant les UW correspondant au nom room dans le graphe de la figure 2.b

```
-01-$OCC(a,$LU(a,Det))-02-
-02-$OCC(waiting_room,$LU(waiting_room,N))-04-
-02-$OCC(waiting,$LU(waiting,N),$LU(wait,V))-03-
-03-$OCC(room,$LU(room,N,$UW(room[icl>position>thing],
room[icl>area>thing],room[icl>gathering>thing])),$LU(room,V))-04-
```

(e) Résultat de l'application du système-Q composé uniquement de la règle (d) sur le graphe (e)

FIGURE 2 – Exemples du formalisme des systèmes-Q

Le langage-Q a été réimplémenté au GETALP. Il est possible de le tester en ligne [45].

De notre point de vue, l'utilisation du langage-Q a trois avantages principaux :

- il fournit une structure de représentation formelle pour les textes, qui facilite le portage linguistique (Hajlaoui et Boitet (2007)) ;
- les traitements sur les textes peuvent être unifiés grâce à un puissant système de règles de réécriture ;
- les textes représentés sont facilement interprétables et manipulables par des non-informaticiens (linguistes, etc.).

2.3 Annotation des textes grâce à des systèmes-Q

Il est possible d'écrire des systèmes-Q pour développer des systèmes de traduction automatique complets et opérationnels. Deux exemples sont marquants :

- METEO (Chandioux et Guéraud (1981)) qui a traité 11K mots par jour pour la traduction de bulletins météo au Canada de 1977 à 1985,

45. http://unldeco.imag.fr/unldeco/SystemsQ.po

– TAUM-Aviation (Isabelle et Bourbeau (1985)), prototype développé 1977 à 1981 pour la traduction de manuels techniques en aéronautique. Le langage-Q y était utilisé seulement pour les phases d'analyse et de génération morphologique. Pour les autres, Steward (1978) avait développé REZO, transducteur de graphes-Q, dont le moteur était dans la famille des ATN (*augmented transition network*, Woods (1970)).

Dans notre cas, la représentation des textes sous forme de graphes-Q, dont chaque chemin correspond à une interprétation possible, permet de conserver toutes les hypothèses à chaque étape du processus d'annotation. En effet, pour des applications de recherche d'information, il est toujours préférable de conserver une ambiguïté que de mal la résoudre. De plus, le formalisme des graphes-Q permet de représenter des ambiguïtés de segmentation imbriquées, comme pour l'interprétation de "*white paper wall*" illustrée dans la figure 3.

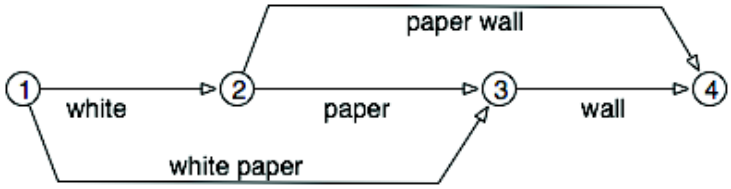


FIGURE 3 – Segmentation de la chaîne "white paper wall"

Enfin, il est en principe possible d'utiliser le langage-Q pour l'implémentation de chaque étape du processus d'annotation.

Nous avons utilisé une version du langage-Q réimplémentée en 2007 par Hong-Thai Nguyen lors de sa thèse dans l'équipe GETALP (Nguyen (2009)), et finalisée par David Cattanéo dans son projet de TER de master (février-juin 2010). L'annexe H (un article pas encore publié) présente en détail le langage-Q est sa nouvelle implémentation. Une restriction importante des systèmes-Q, dans leur version originale, est le nombre limité de caractères utilisables dans les étiquettes (essentiellement les caractères alphanumériques). Le support d'Unicode ne pose pas de problème théorique mais n'est pas implémenté pour l'instant. Aussi, des expériences préliminaires ont été menées avec le langage-Q uniquement pour l'anglais, qui ne comporte pas (ou très peu) de caractères accentués. Le langage-Q a toutefois été utilisé pour traiter de nombreuses langues grâce à des "transcriptions minimales" basées sur les lettres majuscules, les chiffres et les autres caractères de l'ISO-646 (Phan et Geta (1992), Boitet et Tchéou (1990)).

2.4 Désambiguisation

Notre tâche s'apparente à la désambiguïsation de sens de mots. Schématiquement, il s'agit de trouver, pour chaque mot du texte, le sens le plus plausible parmi la liste des sens possibles. Les sens sont dans notre cas représentés par des UW. Par exemple, si l'on a la phrase "L'avocat plaide devant les juges.", le but de la désambiguïsation est de choisir si le mot avocat correspond au fruit (*avocado(icl>fruit>thing)*) ou au juriste (*lawyer(icl>professional>thing)*). Dans ce cadre, il existe un certain nombre de méthodes que nous ne détaillons pas ici ; leur étude dépasse l'objet de cette thèse. Le lecteur pourra se reporter à Ide et Véronis (1998) pour les travaux jusqu'en 1998 puis à Agirre et Edmonds (2006) ou Navigli (2009) pour un état de l'art des travaux plus récents.

Dans le cadre d'OMNIA, nous utilisons, des algorithmes locaux (pour la comparaison d'un sens de mot avec un autre ou avec son contexte) inspirés de *Lesk adapté* (Banerjee et Pedersen

(2002)). Ces algorithmes locaux sont mis en œuvre de façon globale, sur le graphe-Q représentant un texte, à l'aide d'algorithmes à colonies de fourmis (Schwab (2005)). Les implémentations sont dues à Didier Schwab. Nous sommes intervenu avec Achille Falaise sur les expérimentations présentées au paragraphe 4.3.

Nous insistons sur le fait que la désambiguïsation est réalisée sur l'annotation du texte au niveau interlingue (avec des UW). Le processus est donc commun à toutes les langues considérées.

2.4.1 Algorithme local

Méthodes de similarité sémantique. Ces méthodes consistent à donner un score censé refléter la proximité des objets linguistiques (généralement des mots ou des sens de mots) comparés. Ces scores peuvent être des similarités (et donc avoir une valeur entre 0 et 1), des distances (et donc respecter les trois propriétés : $[d(x, y) = 0] \Rightarrow [x = y]$), symétrie et inégalité du triangle) ou une valeur positive non bornée. On pourra consulter Pedersen et al. (2005), Cramer et al. (2010) ou Navigli (2009) pour un panorama assez complet.

En désambiguïsation lexicale, ces méthodes sont utilisées de façon locale pour comparer deux sens de mots, et sont ensuite appliquées à un niveau global (phrase, paragraphe ou document). Dans le cadre d'OMNIA, nous nous sommes peu intéressé à ces algorithmes locaux, préférant nous concentrer sur l'algorithme global. Nous avons donc choisi une méthode rapide et simple à mettre en œuvre avec les ressources dont nous disposions, et surtout efficace en temps de calcul, pour permettre une utilisation dans des conditions réelles.

Algorithme choisi : une variante de Lesk. L'algorithme que nous avons utilisé est une variante de celui proposé par Lesk (1986). Cet algorithme, inventé il y a plus de 25 ans, se caractérise par sa simplicité. Il ne nécessite qu'un dictionnaire et aucun apprentissage. Le score donné à une paire de sens est le nombre de mots (ici, simplement le texte entre deux délimiteurs) communs dans leurs définitions, sans tenir compte ni de leur ordre, ni de sous-séquences communes (approche sac de mots), ni d'information syntaxiques ou morphologiques. Les variantes de cet algorithme sont encore aujourd'hui parmi les meilleures en précision pour la désambiguïsation de l'anglais.

Dans notre cas, les sens de mots sont représentés par des UW, quelle que soit la langue des textes en entrée. Cela permet d'utiliser le même algorithme de désambiguisation pour toutes les langues supportées. Les UW du consortium U++, que nous utilisons, sont reliés à Wordnet. Nous bénéficions donc non seulement de ses définitions, mais aussi des liens entre synsets que nous exploitons dans la variante utilisée dans le cadre d'OMNIA. Notre variante fonctionne comme la version originale, mais, au lieu d'utiliser uniquement la définition des sens, elle utilise également les définitions des différents synsets qui ont des liens avec l'UW considéré. Cette idée est similaire à celle de Banerjee et Pedersen (2002).

Efficacité de l'algorithme L'algorithme de base pour comparer le nombre de mots communs à deux définitions a une complexité en $O(n \times m)$ sans même considérer le coût de comparaision entre chaînes de caractères [46]. On pourrait penser qu'il suffirait de précalculer la matrice de similarité avec l'ensemble des définitions. Cette idée est utopique non seulement à cause de la taille que peuvent atteindre les dictionnaires (jusqu'à plusieurs millions de définitions) [47], mais aussi parce qu'on a toujours besoin de faire des calculs sur de nouvelles données, puisque (1) les

46. Si n et m sont les longueur en mots des définitions.
47. une forme de cache peut en partie régler ce problème.

dictionnaires d'UW évoluent, et (2) on peut créer des définitions à la volée comme avec notre algorithme à colonies de fourmis.

Nous avons amélioré ce calcul en utilisant un prétraitement qui se déroule en deux étapes. Nous commençons par affecter un nombre entier à chacun des mots trouvés dans le dictionnaire, puis nous convertissons chacune des définitions en un vecteur de nombres correspondant aux mots qu'elle contient, triés du plus petit au plus grand. Nous appelons ces vecteurs *vecteurs de définitions étendues*.

Par exemple, si notre première étape a donné ‹‹kind››= 1 ; ‹‹of››= 2 ; ‹‹evergreen››= 3 ; ‹‹tree››= 4 ; ‹‹with››= 5 ‹‹needle-shaped››= 6 ; ‹‹leaves››= 7 ; ‹‹fruit››= 8 ; ‹‹certain››= 9 avec la définition A, « *kind of evergreen tree with needle-shaped leaves* », nous obtenons le vecteur $[1, 2, 3, 4, 5, 6, 7]$ et avec B, « *fruit of certain evergreen tree* », nous obtenons $[2, 3, 4, 8, 9]$.

Cette conversion a deux avantages : (1) la comparaison de nombres est bien plus efficace que la comparaison de chaînes de caractères, (2) ordonner ces nombres permet d'éviter des comparaisons inutiles et de gagner en efficacité. Ainsi, avec ce prétraitement, la complexité passe de $O(n \times m)$ à $O(n)$, où n et m ($n > m$) sont les longueurs (en mots) des définitions.

Avec nos exemples, calculer le score pour les définitions A et B se fait en $7 \times 5 = 35$ opérations (sur des chaînes de caractères) avec l'algorithme classique, tandis que si les définitions sont converties en vecteurs, nous n'avons plus que 7 opérations (sur des entiers).

2.4.2 Mise en œuvre globale

Il s'agit de mettre en place une méthode utilisant un algorithme local pour trouver les sens les plus vraisemblables pour chaque mot d'un texte. La plus directe est la méthode de calcul brut utilisée par exemple par Banerjee et Pedersen (2002). Elle consiste à considérer les combinaisons de l'ensemble des sens des mots dans le même contexte, à donner un score à chacune de ces combinaison, et à choisir celle qui a le meilleur score.

Le principal problème de cette méthode est la rapide explosion combinatoire qu'elle engendre. Considérons la phrase « *The pictures they painted were flat, not round as a figure should be, and very often the feet did not look as if they were standing on the ground at all, but pointed downwards as if they were hanging in the air.* » qui est composée de 17 mots. Si l'on se réfère au dictionnaire anglais-UW++, ‹*picture*› a 9 sens, ‹*paint*› 4, ‹*be*› 13, ‹*flat*› 17, ‹*figure*› 13, ‹*very*› 2, ‹*often*› 2, ‹*foot*› 11, ‹*look*› 10, ‹*stand*› 12, ‹*ground*› 11, ‹*at all*› 1, ‹*point*› 13, ‹*downwards*› 1, ‹*hang*› 15 et ‹*air*› 9. Il y a alors 137 051 946 345 600 combinaisons de sens possibles à analyser, ce qui est énorme mais dans l'ordre de grandeur de ce que peuvent traiter les grands calculateurs actuels. Le calcul brut est donc très compliqué à réaliser dans des conditions réelles et, surtout, rend impossible l'utilisation d'un contexte d'analyse plus important.

Pour contourner ce problème, plusieurs solutions ont été proposées. Par exemple, des approches utilisant un corpus pour diminuer le nombre de combinaisons à examiner comme la recherche des chaînes lexicales compatibles (Gale et al. (1992), Vasilescu et al. (2004)), des approches issues de l'intelligence artificielle comme le recuit simulé (Cowie et al. (1992)), et des algorithmes génétiques (Gelbukh et al. (2003)).

Ces méthodes ne permettent toutefois pas d'exploiter notre représentation des textes sous forme de graphe. Pour cette raison, nous avons choisi d'utiliser une méthode à colonies de fourmis inspirée de Lafourcade et Schwab (2005).

Algorithmes à colonies de fourmis Les algorithmes à fourmis ont pour origine la biologie et les observations réalisées sur le comportement social des fourmis. En effet, ces insectes ont collectivement la capacité de trouver le plus court chemin entre leur fourmilière et une source

de nourriture. Il a pu être démontré que la coopération au sein de la colonie est auto-organisée et résulte d'interactions entre individus autonomes. Ces interactions, souvent très simples, permettent à la colonie de résoudre des problèmes compliqués. Ce phénomène est appelé *intelligence en essaim* (Bonabeau et Théraulaz (2000)).

En 1989, Jean-Louis Deneubourg étudie le comportement des fourmis biologiques dans le but de comprendre la méthode avec laquelle elles choisissent le plus court chemin et le retrouvent en cas d'obstacle. Il élabore ainsi le modèle stochastique dit *modèle de Deneubourg* (Deneubourg et al. (1989)), conforme à ce qui est observé statistiquement sur les fourmis réelles quant à leur partage entre les chemins. Ce modèle stochastique est à l'origine des travaux sur les algorithmes à fourmis.

Le concept principal de l'intelligence en essaim est la *stygmergie*, c'est-à-dire l'interaction entre agents par modification de l'environnement. Une des premières méthodes que l'on peut apparenter aux algorithmes à fourmis est l'*écorésolution* qui a montré la puissance d'une heuristique de résolution collective basée sur la perception locale, évitant tout parcours explicite de graphe d'états (Drogoul (1995)). Informellement on peut dire qu'on cherche seulement à être dans une bonne situation, et à éviter les mauvaises.

En 1992, Marco Dorigo conçoit le premier algorithme basé sur ce paradigme pour le célèbre problème combinatoire du voyageur de commerce (Dorigo et Gambardella (1997)). Dans les algorithmes à base de fourmis artificielles, l'environnement est généralement représenté par un graphe et les fourmis virtuelles utilisent l'information accumulée sous la forme de phéromone déposée sur les arcs du graphe, définissant ainsi des chemins privilégiés. De façon simple, une fourmi se contente de suivre les traces de phéromones déposées précédemment ou explore au hasard dans le but de trouver un chemin optimal dans le graphe. L'heuristique repose sur la quantité de phéromone (Hao et al. (1999)).

Ces algorithmes offrent une bonne alternative aux algorithmes "gloutons" classiques pour la résolution de problèmes modélisables sous forme d'un graphe. Ils permettent un parcours rapide et efficace en offrant une qualité de résultat comparable à ceux obtenus avec d'autres méthodes plus coûteuses. Leur grand intérêt réside dans leur capacité à s'adapter à un changement de l'environnement. Dorigo et Stützle (2004) donne un bon état de l'art sur la question.

Analyse sémantique par algorithme à fourmis sur graphes-Q À cette étape, nous partons d'un graphe-Q tel que présenté dans la section précédente. Suivant les idées développées par Schwab (2005) puis Guinand et Lafourcade (2009), dans un système d'analyse sémantique par colonie de fourmis (ACFAS), chaque sens possible d'un mot, donc ici chaque UW, est associé à une fourmilière. Les fourmilières produisent des fourmis qui partent à la recherche de nourriture. Leurs déplacements se déroulent en fonction des scores locaux (cf. section 2.4.1), de la présence de nourriture, et du passage des autres fourmis. Une fois arrivée à la fourmilière d'un autre UW, une fourmi peut choisir de revenir directement à sa fourmilière mère. Elle établit alors un pont que les autres fourmis sont, à leur tour, susceptibles d'emprunter et de renforcer grâce à leurs phéromones. Ce renforcement a lieu si les informations lexicales (issues du graphe-Q et des définitions) conduisent les autres fourmis à emprunter le pont et disparaît dans le cas inverse. Ces ponts forment des interprétations que nous pouvons faire de la phrase. Qui plus est, une fourmi qui arrive à une autre fourmilière donne une partie de son sucre. Ainsi, les fourmis établissent de nombreux liens entre fourmilières compatibles, qui se comportent comme des métafourmilières pour mutualiser les ressources et ainsi se renforcer par rapport à celles qui n'arrivent pas à s'allier et qui finissent par épuiser les ressources.

Conclusion

Nous avons présenté les grands principes utilisés pour une annotation interlingue de textes. C'est une étape préliminaire à l'extraction de contenu dans les textes (compagnons et requêtes) réalisée dans OMNIA. Le formalisme de représentation choisi permet de conserver dans un graphe toutes les ambiguïtés explicitées lors de l'annotation des textes. D'autre part, l'utilisation d'algorithmes à colonies de fourmis pour la mise en œuvre d'un processus de désambiguïsation automatique permet l'exploitation efficace de notre structure de graphe.

Nous présentons maintenant la réalisation d'un système de recherche d'images accompagnées de textes multilingues basé sur l'annotation des textes par des UW et l'alignement d'une ontologie avec des UW.

3 Analyse fonctionnelle

3.1 Fonctions principales

3.1.1 Rechercher des images

La fonction fondamentale du système est de permettre de rechercher des images décrites par des textes en langue naturelle. Comme expliqué précédemment, cette tâche relève d'une recherche d'information translingue.

Entrée : Une requête libre formulée dans la langue préférée de l'utilisateur. La requête peut être un texte long, comme l'intégralité d'un article à illustrer.

Sortie : Un ensemble d'images ordonnées selon la vraisemblance qu'elles satisfassent la requête.

3.1.2 Passer une ontologie en paramètre

Entrées : (1) Une ontologie au format OWL dont les étiquettes (`rdfs:label`) sont des mots simples ou composés de l'anglais (on pourra ajouter des langues de façon modulaire). Sortie : Un système d'indexation guidé par l'ontologie.

Un utilisateur peut passer une ontologie en paramètre du système. Cette ontologie définit et organise des termes d'indexation pertinents du point de vue de l'utilisateur ou du domaine des images recherchées. Les termes définis au sein de l'ontologie permettent de guider l'extraction de contenu réalisée dans les textes compagnons, d'une part pour la construction de descriptions dans l'ontologie, et d'autre part dans les requêtes des utilisateurs pour la constructions de requêtes formelles résolubles par un raisonneur. La structure de l'ontologie permet d'améliorer le résultat des recherches par inférence.

La stratégie que nous utilisons pour intégrer une ontologie dans un processus de recherche translingue a été décrite au chapitre 3. Cette fonction revient en fait à créer un alignement entre les éléments de l'ontologie et un lexique d'UW. Une fois l'ontologie passée en paramètre, il faut (ré-)indexer les images par rapport à l'ontologie passée en paramètre, c'est à dire construire des descriptions pour les images dans l'ontologie.

3.2 Fonctions contraintes

La stratégie de recherche "sémantique" choisie pour le projet OMNIA et les conclusions du chapitre 3 entrainent un certain nombre de contraintes pour la réalisation des fonctions principales. A l'aide de leurs textes compagnons, les images sont au préalable indexées par des descriptions

dans l'ontologie. Ensuite, chaque requête est traitée à la volée, en utilisant les mêmes traitements que pour l'indexation, pour produire une requête formelle pour l'ontologie. La résolution de la requête est réalisée grâce à un moteur d'inférence.

Dans la mesure où les possibilités d'inférence restent limitées à l'utilisation de la taxonomie de concepts, il est possible de "propager" les descriptions suivant la taxonomie (voir paragraphe 4.6.1) et de les stocker dans une base de donnée relationnelle. Les requêtes spontanées des utilisateurs doivent alors être transformées en requêtes SQL.

Les requêtes et les textes compagnons subissent un processus d'extraction de contenu commun jusqu'à la dernière étape, où sont produites respectivement des requêtes formelles ou des descriptions. L'extraction de contenu est réalisée à partir d'une annotation interlingue, puis conceptuelle, des textes (Falaise et al. (2010)).

3.2.1 Aligner une ontologie et un lexique d'UW

Entrées : une ontologie au format OWL et un lexique d'UW.

Sortie : un alignement entre l'ontologie et le lexique d'UW.

La création automatique d'un alignement ontologie-UW est la fonction essentielle qui permet de passer une ontologie en paramètre du système. L'alignement créé est utilisé pour passer de l'annotation interlingue à l'annotation conceptuelle des textes.

3.2.2 Annoter les textes avec des UW (annotation interlingue)

Entrée : Un texte compagnon ou une requête d'un utilisateur.

Sortie : Le texte annoté par des UW sous forme de graphe-Q.

Ce processus est le préambule commun à l'annotation conceptuelle des textes, quelle que soit la langue considérée. Il rassemble tous les traitements de surface nécessaire au traitement du texte. L'utilisation de procédés de désambiguïsation automatique permet d'augmenter la qualité de l'annotation. Toutefois, pour le traitement des requêtes à la volée, le temps d'exécution de ces procédés est critique pour assurer une bonne réactivité du système.

3.2.3 Annoter un texte avec les éléments d'une ontologie (annotation conceptuelle)

Entrée : Un texte annoté par des UW sous forme de graphe-Q et un alignement ontologie-UW.

Sortie : Un texte annoté par les éléments de l'ontologie sous forme de graphe-Q.

Cette étape consiste essentiellement à suivre les correspondances de l'alignement pour passer des UW aux éléments de l'ontologie.

3.2.4 Extraire du contenu formalisé

Entrée : Une ontologie, un texte annoté par les éléments de l'ontologie sous forme de graphe-Q.

Sortie : Une description dans l'ontologie si le texte est un texte compagnon, une requête pour l'ontologie si le texte est la requête d'un utilisateur.

Concrètement, on récupère les annotations conceptuelles et on les organise sous formes d'axiomes OWL ou de requêtes. A cette étape, il semble indispensable de guider l'extraction de contenu par une connaissance "basique" sur la structure des groupes syntagmatiques élémentaires, et de l'ordre des groupes en général, dans la langue du texte ou de la requête.

3.2.5 Résoudre les requêtes sur l'ontologie

Entrée : Une ontologie (T-box et A-box) contenant des descriptions d'images et une requête SPARQL ou SQL.

Sortie : Un ensemble ordonné d'images.

La sélection des images (instances de l'ontologie) est déléguée à un moteur d'inférence ou à un système de gestion de bases de données relationnelles si l'ontologie et les descriptions des images sont stockées dans une base de données. Le classement des images par ordre de vraisemblance est effectué *a posteriori* à l'aide des scores de désambiguïsation des UW et des poids de l'alignement entre ontologie et UW.

3.3 Fonctions complémentaires

3.3.1 Rechercher à partir des UW

Entrée : Les textes compagnons d'une collection d'image et la requête d'un utilisateur, annotés par des UW, sous forme de graphe-Q.

Sortie : Un ensemble ordonné d'images.

L'ontologie passée en paramètre du système permet d'améliorer la recherche avec des inférences. Toutefois, une recherche basée uniquement sur l'ontologie peut mener à de nombreux silences si cette dernière ne contient pas les éléments pour annoter un texte compagnon ou une requête. C'est le cas si le domaine de l'ontologie est trop restreint par rapport aux domaines des recherches et des images. Une recherche translingue par "sacs d'UW" permet d'éviter ces silences.

3.3.2 Interagir avec l'utilisateur

Le besoin de réactivité lors du traitement des requêtes peut empêcher d'avoir recours à des processus de désambiguïsation automatique poussés. Il est cependant possible d'interagir avec l'utilisateur pour qu'il précise le sens de certains mots ambigus dans sa requête. Deux modalités sont envisagées pour cette interaction : le choix d'un sens de mots parmi une liste (désambiguïsation directe) et le choix d'un ensemble d'images pertinentes parmi celles proposées en première réponse à la requête (désambiguïsation indirecte).

Désambiguïsation directe

Entrée : Une requête annotée par des UW pondérées, une liste d'UW choisie par l'utilisateur pour certaines annotations ambiguës.

Sortie : La requête annotée par des UW et une nouvelle pondération (on renforce le poids des UW choisies).

Le système présente à l'utilisateur les mots ambigus dans sa requête. Chaque mot est accompagné d'une liste contenant les UW associées lors de l'annotation. L'utilisateur peut alors désambiguïser le mot en choisissant une ou plusieurs UW.

Désambiguïsation indirecte

Entrée : Une requête annotée par des UW pondérées, un ensemble d'images choisies par l'utilisateur et leurs textes compagnons annotés par des UW.

Sortie : La requête annotée par des UW et une nouvelle pondération (on renforce le poids des UW qui interviennent dans la requête ET dans les textes compagnons des images choisies).

L'utilisateur peut choisir les images qui correspondent le mieux à sa requête parmi celles proposées par le système. Une nouvelle requête est alors calculée et résolue pour proposer un nouvel ensemble d'images.

3.3.3 Fusionner avec les résultats d'analyses multimodales

Il est possible de réaliser une fusion tardive des résultats de nos analyses textuelles avec les résultats d'analyses portant sur d'autres modalités (notamment des analyses des propriétés visuelles des images). Cette possibilité, bien que non développée dans cette thèse, a été envisagée dès le début du projet OMNIA.

Entrée : Un ensemble de descriptions d'un couple image/texte dans une ontologie.

Sortie : Une description unique et cohérente du couple image/texte.

4 Architecture logicielle, implémentation et évaluations

4.1 Principe et vue d'ensemble

La chaîne de traitements textuels est implémentée suivant une architecture à services dans laquelle chaque fonction correspond à un service Web. Les données passent d'un service à l'autre et les résultats intermédiaires peuvent être consultés.

Nous pouvons ainsi utiliser des ressources existantes, appelées par des interfaces REST (Fielding (2000)) ou de simples formulaires HTML, et les changer au besoin, de façon modulaire. Achille Falaise a développé un "superviseur" pour gérer ces interfaces Web hétérogènes et les problèmes de normalisation des données (encodages, *cookies*, etc.).

Cette architecture est capable de gérer plusieurs tâches en parallèle, ce qui permet par exemple de traiter les requêtes des utilisateurs à la volée avec une forte priorité, alors que le traitement des textes compagnons pour l'indexation des images se fait en tâche de fond avec un faible priorité.

Le passage des données textuelles d'un service à l'autre pour réaliser l'extraction de contenu est récapitulé dans la figure 4.

4.2 Annotation interlingue

4.2.1 Lemmatisation

La première étape du traitement est la lemmatisation des textes (c'est à dire l'annotation de chaque occurrence avec les lemmes possibles). En recherche d'information, il vaut mieux conserver une ambiguïté que mal la résoudre. Le lemmatiseur doit donc conserver les ambiguïtés dans un graphe représentant aussi bien les ambiguïtés de segmentation en mots (composés, voire simples pour les langues écrites sans séparateurs de mots typographiques) que les ambiguïtés morpho-syntaxiques et sémantiques (ces dernières donnant lieu à des ensembles d'UW). Un étiqueteur (tagger) classique ne convient donc pas. Par contre, plusieurs logiciels peuvent être utilisés pour couvrir les langues souhaitées. Leurs sorties devront être présentées sous forme de graphes-Q.

Les premières expériences ont été réalisées avec le logiciel NooJ de Max Silberztein (2009) qui propose une licence gratuite à des fins de recherche. NooJ présente ses résultats sous une forme proche d'un graphe-Q. Le résultat de l'analyse morphosyntaxique d'un énoncé est représenté par un graphe sans cycle (figure 5) dont les nœuds sont étiquetés par la position d'un caractère dans l'énoncé, et les arcs par un lemme accompagné de ses propriétés morphosyntaxiques possibles. NooJ propose une sortie au format XML. Deux méthodes ont été testées pour traduire cette

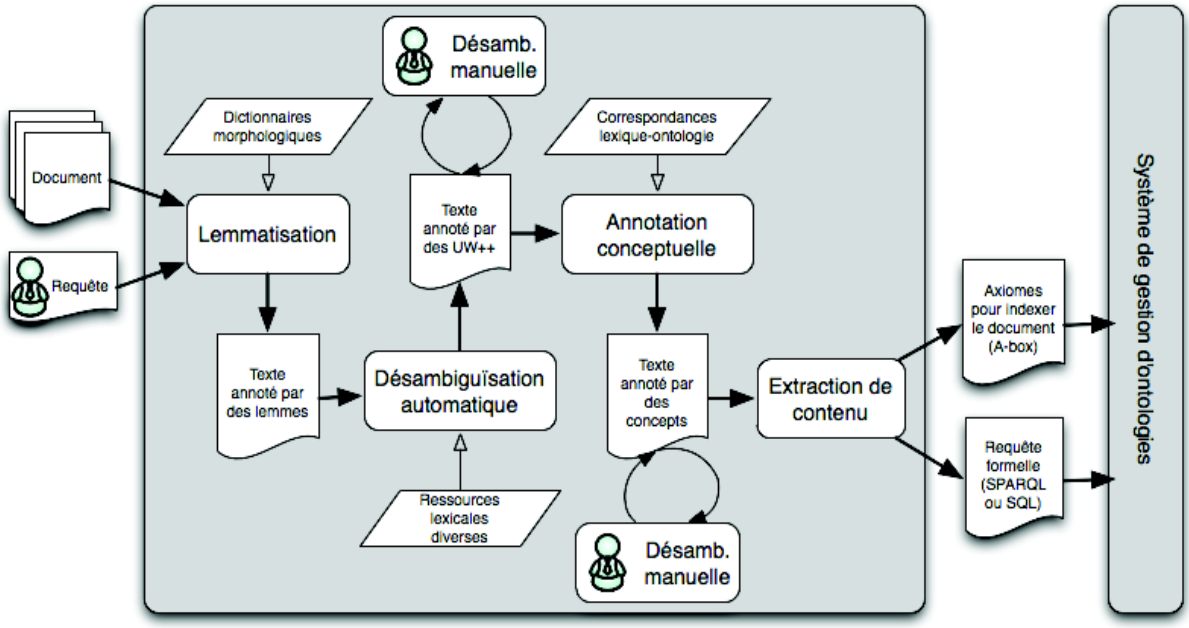


FIGURE 4 – Flot de données

sortie sous forme de graphe-Q : une transformation XSLT (récursive) et un code C généré à l'aide d'ANTLR [48]. La transformation XSLT, complète et commentée est fournie en annexe F.

Sur un ordinateur de bureau (CPU Intel Core 2 Duo 2,53GHz), la lemmatisation d'un texte de 50 mots prend 40ms en moyenne. La traduction du résultat sous forme de graphe-Q prend environ une seconde avec le code C généré par ANTLR et près de 4 secondes avec la transformation XSLT exécutée avec saxon9he (et le chargement de la machine virtuelle Java).

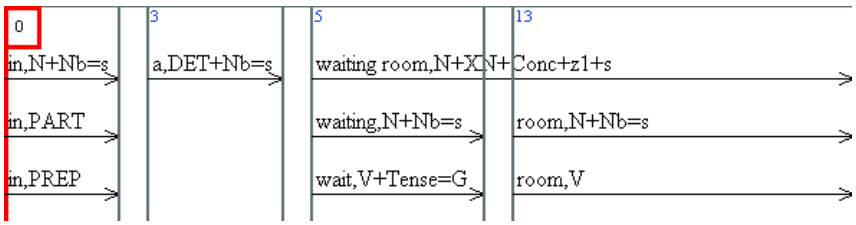


FIGURE 5 – Sortie du logiciel NooJ pour la chaîne "in a waiting room"

Pour le français et l'anglais, nous avons également développé des lemmatiseurs *ad hoc*, dont les sorties sont des graphes-Q. Ils sont basés sur les dictionnaires morphologiques DELAF [49], disponibles sous licence LGPL.

L'algorithme de lemmatisation peut être résumé comme suit :

1. Le texte est d'abord segmenté au maximum. Par exemple, pour l'anglais et le français, les segments sont des mots typographiques, c'est à dire des chaînes de caractères séparées par des espaces et des signes de ponctuation ; mais pour un système d'écriture sans séparateur, on peut considérer chaque caractère comme un segment.

48. `http://www.antlr.org`
49. `http://infolingu.univ-mlv.fr/DonneesLinguistiques/Dictionnaires/telechargement.html`

2. On initialise le graphe en créant un nœud par séparateur ainsi trouvé.
3. Ensuite, les arcs sont construits. Tous les regroupements de segments contigus possibles, dans les limites d'une fenêtre de taille paramétrable (3 segments par défaut), sont testés. S'ils sont présents dans le dictionnaire, l'arc correspondant est ajouté au graphe.

La lemmatisation sous forme de graphe-Q d'un texte de 50 mots prend 300ms en moyenne. C'est la solution que nous avons intégrée dans le premier démonstrateur.

Exemple 1. Nous donnons les résultats des étapes successives pour le fragment de texte "*guerre civile opposant catholiques et protestants*", tels que renvoyés par les services du démonstrateur présenté au paragraphe 5.

Dans le texte lemmatisé sous forme de graphe-Q, chaque arc est étiqueté avec des informations sur l'occurrence (`$OCC`) qu'il couvre dans le texte : sa forme (`$FORME`), eu une unité lexicale possible (`$LU`) composée d'un lemme (`$LEMMA`) et de sa catégorie grammaticale (`$CAT`).

La figure 6 donne le code correspondant au graphe-Q. La figure 7 présente une forme graphique et simplifiée d'une partie du graphe-Q. Pour plus de lisibilité, les nœuds `$LEMMA` et `$CAT` ont été omis dans les arbres étiquetant le graphe-Q. De même, les catégories grammaticales (NOUN, ADJ, VERB, etc.) ont été remplacées par leurs initiales (N, A, V, etc.).

```
-0-$OCC($FORME(guerre),$LU($LEMMA(guerre),$CAT(NOUN)))-1-
-0-$OCC($FORME(#Uguerre#20#civile),$LU($LEMMA(#Uguerre#20#civile),$CAT(NOUN)))-3-
-1-$OCC($FORME(#U#20#))-2-
-2-$OCC($FORME(civile),$LU($LEMMA(civil),$CAT(ADJ)))-3-
-2-$OCC($FORME(civile),$LU($LEMMA(civil),$CAT(NOUN)))-3-
-3-$OCC($FORME(#U#20#))-4-
-4-$OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(ADJ)))-5-
-4-$OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(NOUN)))-5-
-4-$OCC($FORME(opposant),$LU($LEMMA(opposer),$CAT(VERB)))-5-
-5-$OCC($FORME(#U#20#))-6-
-6-$OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(ADJ)))-7-
-6-$OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(NOUN)))-7-
-7-$OCC($FORME(#U#20#))-8-
-8-$OCC($FORME(et),$LU($LEMMA(et),$CAT(CONJ)))-9-
-9-$OCC($FORME(#U#20#))-10-
-10-$OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(ADJ)))-11-
-10-$OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(NOUN)))-11-
```

FIGURE 6 – Texte lemmatisé sous forme de graphe-Q (code)

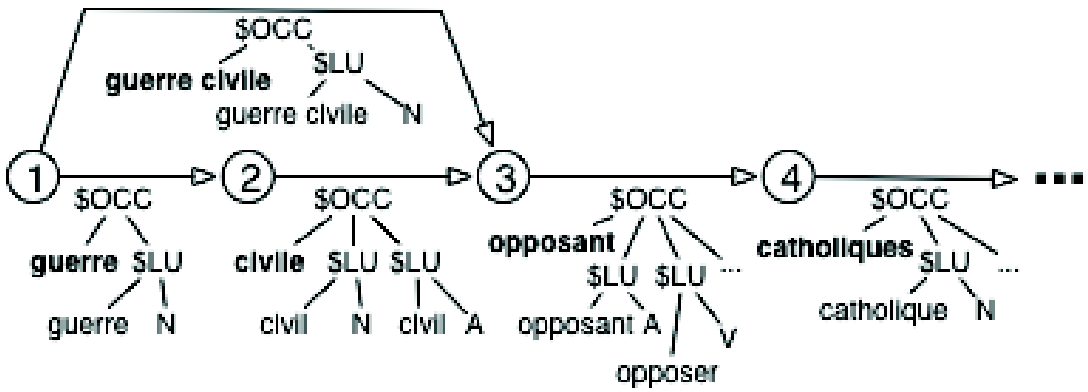


FIGURE 7 – Texte lemmatisé sous forme de graphe-Q (graphique simplifié)

4.2.2 Ajout des UW au graphe-Q

Cette phase consiste à ajouter des UW au graphe-Q en consultant un dictionnaire bilingue liant les lemmes de la langue aux UW. La base Jibiki (cf. section 3.3.3 du chapitre 3) est utilisée comme serveur de dictionnaires à travers son interface de programmation (API) REST [50].

100 requêtes http, utilisant l'API REST pour trouver les UW associées à un lemme, lancées à la suite sur le même sous-réseau, sont résolues en 8,5 secondes soit en moyenne 85 ms par requête. Lorsque les requêtes sont lancées depuis une machine d'un autre pays (Estonie), le temps total pour 100 requêtes passe à 35 s, soit en moyenne 350 ms par requête.

Les mots qui ne sont pas trouvés dans le dictionnaire sont stockés en mémoire sur la plate-forme. Ils sont ensuite proposés en priorité aux contributeurs bénévoles, travaillant collaborativement à l'édition des ressources, pour qu'ils en proposent des traductions. Si une entrée de dictionnaire est modifiée par un contributeur, la modification est directement prise en compte dans le processus d'annotation, car les ressources ne sont pas compilées mais utilisées directement lors de chaque requête.

Le nombre d'ambiguïtés présentes dans les textes annotés est conséquent : jusqu'à 12 UW pour une occurrence, auxquelles s'ajoutent les ambiguïtés de segmentation. Nous utilisons les procédés de désambiguïsation automatique, décrits dans la section 2.4, pour affecter des scores aux interprétations possibles d'un mot, suivant leur vraisemblance dans le contexte.

Exemple 2. Pour chaque unité lexicale identifiée dans le graphe-Q, nous ajoutons les UW possibles (`$UW`), avec pour chacune son identifiant dans la plate-forme Jibiki (`$ID`), son mot vedette (`$HW`), est ses restrictions (`$REST`).

La figure 8 donne le code correspondant au graphe-Q enrichi par des UW. La figure 9 présente une forme graphique et simplifiée d'une partie de ce graphe-Q. Pour plus de lisibilité, les UW sont présentés directement sous un nœud `$UW` au lieu d'une décomposition avec les nœuds intermédiaires `$ID`, `$HW`, `$REST`.

```
-0-$OCC($FORME(guerre),$LU($LEMMA(guerre),$CAT(NOUN),$UW($ID(unl.upp.war.201047),$HW(war),$
    REST(#Uicl#3e#hostility#3e#thing#2c##20#ant#3e#peace)),$UW($ID(unl.upp.warfare.201122),$HW
    (warfare),$REST(#Uicl#3e#military#5f#action#3e#thing#2c##20#equ#3e#war))))-1-
-0-$OCC($FORME(#Uguerre#20#civile),$LU($LEMMA(#Uguerre#20#civile),$CAT(NOUN),$UW($ID(unl.upp.
    civil_war.34984),$HW(civil_war),$REST(#Uicl#3e#war#3e#thing))))-3-
-1-$OCC($FORME(#U#20#))-2-
-2-$OCC($FORME(civile),$LU($LEMMA(civil),$CAT(ADJ)))-3-
-2-$OCC($FORME(civile),$LU($LEMMA(civil),$CAT(NOUN)))-3-
-3-$OCC($FORME(#U#20#))-4-
-4-$OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(ADJ)))-5-
-4-$OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(NOUN)))-5-
-4-$OCC($FORME(opposant),$LU($LEMMA(opposer),$CAT(VERB),$UW($ID(unl.upp.play_off.132128),$HW(
    play_off),$REST(#Uicl#3e#confront#3e#do#2c##20#equ#3e#pit#2c##20#agt#3e#thing#2c##20#obj#3
    e#thing)),$UW($ID(unl.upp.oppose.126171),$HW(oppose),$REST(#Uicl#3e#refute#3e#do#2c##20#
    agt#3e#thing#2c##20#obj#3e#thing))))-5-
-5-$OCC($FORME(#U#20#))-6-
-6-$OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(ADJ)))-7-
-6-$OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(NOUN),$UW($ID(unl.upp.Catholic.31414)
    ,$HW(Catholic),$REST(#Uicl#3e#Christian#3e#thing))))-7-
-7-$OCC($FORME(#U#20#))-8-
-8-$OCC($FORME(et),$LU($LEMMA(et),$CAT(CONJ)))-9-
-9-$OCC($FORME(#U#20#))-10-
-10-$OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(ADJ)))-11-
-10-$OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(NOUN)))-11-
```

FIGURE 8 – Annotation du graphe-Q avec des UW (code)

50. papillon.imag.fr/papillon/Api.po

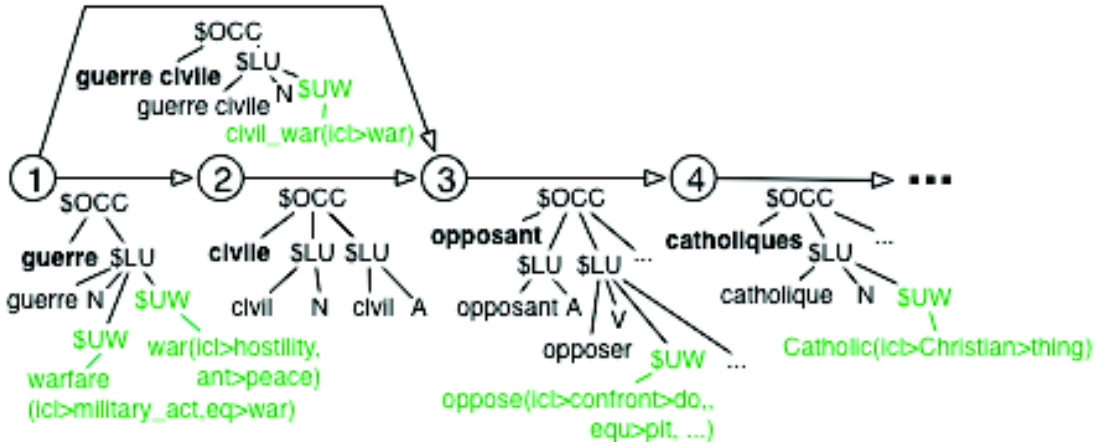


FIGURE 9 – Annotation du graphe-Q avec des UW (graphique simplifié)

4.3 Désambiguïsation

Nous avons testé notre méthode de désambiguïsation sur le corpus de la tâche *grain grossier* (*coarse grained*) de la campagne d'évaluation Semeval 2007 (Navigli et al. (2007)). Ce corpus est composé de 5 textes de genres divers (journalisme, critique littéraire, voyage, informatique, biographies). La tâche consiste à annoter 2269 mots avec l'un de leur sens dans Wordnet (ce qui est équivalent à annoter avec un UW++). L'évaluation du résultat fourni se fait en grain grossier, c'est à dire considérant que des sens proches sont équivalents. On peut citer, comme exemples, « *neige/précipitation* » et « *neige/couverture* » ou « *porc/animal* » et « *porc/viande* ». Cette tâche nous paraît plus proche de notre cadre applicatif qu'une tâche à grain fin, puisque, suivant la granularité de l'ontologie considérée pour l'extraction, des sens proches pourront désigner le même concept.

Nous avons testé les services utilisés dans OMNIA après la phase de lemmatisation (le corpus nous donnant déjà le lemme et sa catégorie grammaticale), jusqu'à la fin de la phase de désambiguïsation. Les textes d'évaluation lemmatisés sont transformés en graphes-Q, puis l'algorithme à fourmis est lancé. Pour chaque mot du texte, un score, proportionnel à la quantité de sucre déposée sur la fourmilière correspondante, est donné à chaque UW. Dans une phase d'exploitation classique d'OMNIA, ces scores sont fournis à l'extracteur de concepts. Ici, nous choisissons l'UW ayant le meilleur score et remplissons le fichier destiné à l'évaluation avec son sens équivalent dans Wordnet.

Dans les expériences présentées ici, deux algorithmes locaux sont testés. Celui noté *Lesk* ne tient compte que des définitions associées aux synsets considérés, tandis que *Lesk adapté* utilise en plus les définitions des synsets liés par une relation sémantique dans Wordnet (cf. partie 2.4.1).

4.3.1 Algorithme global brutal

Le tableau 1 présente les résultats obtenus par l'algorithme global brutal de Banerjee et Pedersen (2002), pour établir un point de comparaison avec notre approche. Rappelons que cet algorithme calcule un score pour un sens de mot, à partir de toutes les combinaisons de sens possibles pour les autres mots de la phrase. Nous avons choisi comme contexte la phrase, excluant *de facto* les phrases d'un mot. Pour des raisons calculatoires, nous avons également exclu les phrases dont l'interprétation mène à plus de 10 milliards de combinaisons.

Algorithme local	% de mots étiquetés	Précision	Rappel	F-mesure	Temps
Lesk	77,30%	69,21%	53,50%	60,35%	≈ 40h
Lesk adapté	77,30%	77,82%	60,16%	67,86%	≈ 300h

TABLE 1 – Résultats des algorithmes locaux avec l'algorithme global brut

On peut remarquer que seulement 77,3% du corpus est étiqueté, à cause de l'explosion combinatoire produite dans certaines phrases. La précision est bien meilleure dans le cas de Lesk adapté, mais au prix d'une augmentation importante du temps de calcul. Dans les deux cas, ce temps de calcul, constaté sur des processeurs Intel Xeon X5550, 4 cœurs, 2.66Ghz, et ramené à un temps monoprocesseur, est incompatible avec l'efficacité requise dans le cadre du scénario OMNIA.

4.3.2 Algorithme global à base de colonies de fourmis

Notre algorithme garantit le choix d'un sens parmi les différentes possibilités pour un mot. La couverture est donc de 100%, soit 22,7% de mieux que le calcul brut. Le tableau 2 présente les résultats obtenus lors d'une exécution de cet algorithme sur le corpus de la tâche *grain grossier* de la campagne d'évaluation Semeval 2007 (2269 mots à annoter).

Algorithme local	% de mots étiquetés	Précision	Rappel	F-mesure	Temps
Lesk	100,0%	67,05%	67,05%	67,05%	≈ 3mn
Lesk adapté	100,0%	74,35%	74,35%	74,35%	≈ 8mn

TABLE 2 – Résultats des algorithmes locaux avec l'algorithme à fourmis

Dans les deux cas, la F-mesure est nettement supérieure à celle obtenue par le même algorithme local dans un temps très inférieur, et cela sans même exploiter les possibilités de parallélisation des calculs.

L'exécution d'un algorithme à fourmis n'est pas déterministe. Pour donner une idée de la variation des résultats sur un grand nombre d'exécutions, nous avons répété notre expérience 100 fois avec l'algorithme adapté. Le tableau 3 montre une différence de 2,16% entre le meilleur et le pire résultat, soit une soixantaine d'erreurs en plus. Dans tous les cas, la F-mesure reste nettement supérieure à celle du calcul brut.

	minimum	maximum	moyenne	médiane	étendue	Écart-type
Lesk adapté	73,20	75,36	74,25	74,24	2,16	0,51

TABLE 3 – 100 exécutions de Lesk adapté complet à 100 cycles

Pour conclure l'évaluation de la désambiguïsation dans le cadre d'OMNIA, nous avons comparé nos résultats avec les résultats obtenus par les différents systèmes qui participaient à la campagne Semeval 2007. Notre système serait arrivé 3ème sur 7 [51], ce qui très encourageant vu les temps de calcul et les possibilités d'extension qu'offrent les algorithmes à fourmis.

51. Nous excluons ici les systèmes qui utilisent les sens grossiers *a priori*, car cela n'est pas en adéquation avec notre modèle.

Exemple 3. La désambiguïsation ajoute des scores de vraisemblance pour chaque UW dans le graphe. `$SCORE` est normalisé, tel que la somme des scores des UW qui annotent un arc soit égale à 1. `$OverallScore` est normalisé, tel que la somme des scores de tous les UW du graphe-Q soit égale à 1.

A partir de cette exemple, nous ne donnons plus de version graphique du résultat car elle serait trop chargée pour être lisible.

```
-0- $OCC($FORME(#Uguerre#20#civile),$LU($LEMMA(#Uguerre#20#civile),$CAT(NOUN),$UW($ID(unl.upp.
    civil_war.34984),$HW(civil_war),$REST(#Uicl#3e#war#3e#thing),$SCORE(1.0),$OverallScore
    (0.011235955056179775)))) -3-
-0- $OCC($FORME(guerre),$LU($LEMMA(guerre),$CAT(NOUN),$UW($ID(unl.upp.war.201047),$HW(war),$
    REST(#Uicl#3e#hostility#3e#thing#2c##20#ant#3e#peace),$SCORE(0.45614035087719296),$
    OverallScore(0.011235955056179775)),$UW($ID(unl.upp.warfare.201122),$HW(warfare),$REST(#
    Uicl#3e#military#5f#action#3e#thing#2c##20#equ#3e#war),$SCORE(0.543859649122807),$
    OverallScore(0.06741573033707865)))) -1-
-1- $OCC($FORME(#U#20#)) -2-
-2- $OCC($FORME(civile),$LU($LEMMA(civil),$CAT(ADJ))) -3-
-2- $OCC($FORME(civile),$LU($LEMMA(civil),$CAT(NOUN))) -3-
-3- $OCC($FORME(#U#20#)) -4-
-4- $OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(ADJ))) -5-
-4- $OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(NOUN))) -5-
-4- $OCC($FORME(opposant),$LU($LEMMA(opposer),$CAT(VERB),$UW($ID(unl.upp.play_off.132128),$HW(
    play_off),$REST(#Uicl#3e#confront#3e#do#2c##20#equ#3e#pit#2c##20#agt#3e#thing#2c##20#obj#3
    e#thing),$SCORE(0.4528301886792453),$OverallScore(0.011235955056179775)),$UW($ID(unl.upp.
    oppose.126171),$HW(oppose),$REST(#Uicl#3e#refute#3e#do#2c##20#agt#3e#thing#2c##20#obj#3e#
    thing),$SCORE(0.5471698113207547),$OverallScore(0.06741573033707865)))) -5-
-5- $OCC($FORME(#U#20#)) -6-
-6- $OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(ADJ))) -7-
-6- $OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(NOUN),$UW($ID(unl.upp.Catholic.31414)
    ,$HW(Catholic),$REST(#Uicl#3e#Christian#3e#thing),$SCORE(1.0),$OverallScore
    (0.011235955056179775)))) -7-
-7- $OCC($FORME(#U#20#)) -8-
-8- $OCC($FORME(et),$LU($LEMMA(et),$CAT(CONJ))) -9-
-9- $OCC($FORME(#U#20#)) -10-
-10- $OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(NOUN))) -11-
-10- $OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(ADJ))) -11-
```

FIGURE 10 – Résultat de la désambiguïsation sur graphe-Q

4.4 Alignement ontologie UW

Cette étape permet de passer une ontologie en paramètre du système et de bénéficier de sa structure lors de l'extraction de contenu et de la recherche. Dans les expériences préliminaires, nous avons utilisé deux méthodes, développées à l'aide de l'API d'alignement décrite dans Euzenat (2004), pour initialiser un alignement entre l'ontologie et un lexique d'UW. Elles sont basées sur des méthodes simples de comparaison de chaînes ; les alignements sont créés à l'aide d'un dictionnaire bilingue "espace lexical de l'ontologie"-UW stocké dans Jibiki. Une méthode de désambiguïsation des alignements implémentée avec le langage PERL est ensuite présentée.

Les ressources suivantes ont été considérées :

– l'ontologie OMNIA (732 concepts) [52] ;
– le dictionnaire d'UW du consortium U++ (207 023 entrées) [53].

Nous avons montré au paragraphe 2.3 du chapitre 3 que les structures ontologiques et lexicales ne sont en générale pas superposable. La figure 11 illustre ce fait avec les données concrètes manipulées dans le cadre du projet OMNIA ("TOOL" est un frère de "VEHICLE poir la relation

52. http://kaiko.getalp.org/kaiko/ontology/OMNIA/OMNIA_current.owl
53. http://kaiko.getalp.org/kaiko/volume/Kaiko_UWpp_OWLapi.rdf

"IS-A" dans l'ontologie mais "tool" est un parent de "vehicle" pour la relation "icl" dans le lexique d'UW). Nous verrons toutefois que la méthode de désambiguïsation des alignements que nous avons implémenté donne de bons résultats malgré les différences de structures entre les objets à aligner.

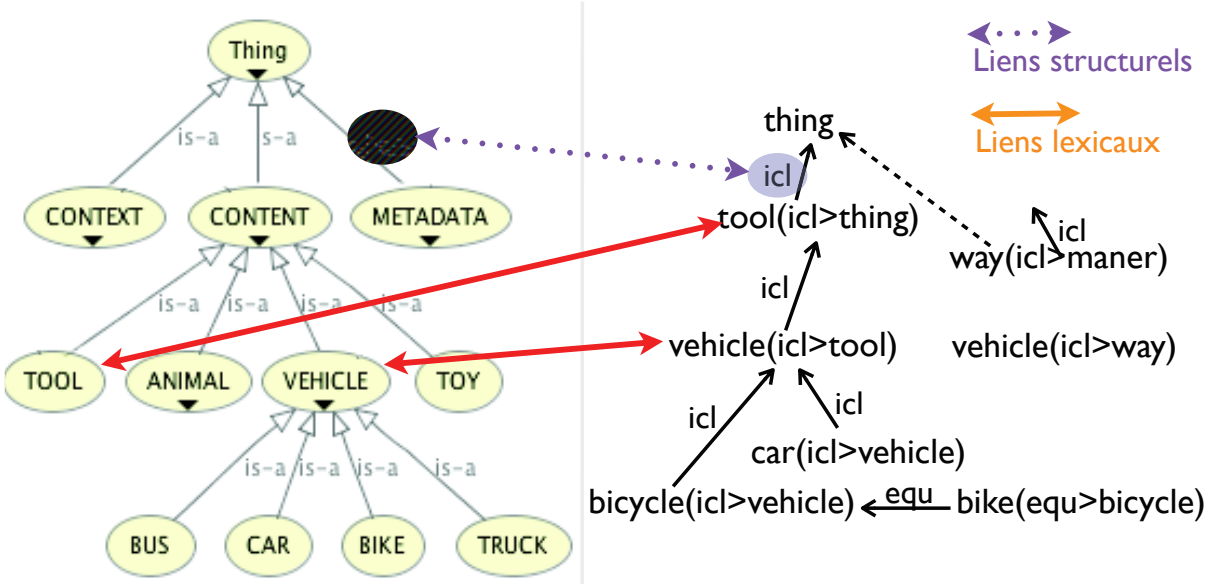


FIGURE 11 – Les structures de l'ontologie OMNIA et du lexique d'UW ne sont pas superposables.

L'alignement basé sur l'égalité des chaînes de caractères a été calculé en 15 minutes sur une machine dotée d'un processeur de 2GHz. Des statistiques sur cet alignement sont données dans la table 4. 1 979 liens concept-UW ont été proposés par l'algorithme [54], soit 2,7 UW par concept en moyenne. 592 concepts ont été alignés avec au moins un UW (ont été *lexicalisés*), soit un "silence" pour 140 concepts. Le concept le plus ambigu de l'ontologie est LIGHT, aligné avec 46 UW++, suivi de GO (le jeu), aligné avec 33 UW++ (notamment des verbes) alors qu'une seule est correcte.

Nb. d'UW++	207023
Nb. de concepts	732
Nb. de liens concept-UW	1979
Nb. moyen d'UW (par concept)	2,7
Nb. de concepts liés à au moins un UW	592 (81%)
Nb. moyen d'UW (par concept lié à au moins une UW)	3,3
Nb. max. d'UW liées à un concept	46

TABLE 4 – Statistiques pour l'initialisation de l'alignement entre l'ontologie OMNIA et les UW++

54. http://kaiko.getalp.org/kaiko/link/Kaiko_align_UWpp-OMNIAv6_StringEq.rdf

Cet alignement, dans lequel chaque lien à été pondéré à 0,5 par défaut, a été utilisé dans les premières expériences d'extraction de contenu présentées ci-après. Il doit cependant être considéré comme une initialisation du processus d'alignement.

Comme nous l'avons vu au paragrapge 2.4 du chapitre 3, le raffinement des correspondances proposées est proche d'une tâche de désambiguïsation lexicale. Il s'agit en effet de trouver quels sens de mots correspondent à un concept, alors qu'en TALN classique il s'agit du sens porté par un mot dans un contexte. Nous avons donc travaillé sur l'adaptation de techniques classiques de désambiguïsation pour la pondération d'un alignement ontologie-lexique. Pour prouver la validité de cette approche, une méthode inspirée de l'algorithme de désambiguïsation lexicale en sens de mots présentée dans Lesk (1986) a été implémentée en PERL.

Le principe est le suivant :

1. Pour chaque concept intervenant dans un alignement, on calcule son voisinage dans le treillis conceptuel (parents, frères et enfants).
2. Nous cherchons ensuite chaque concept du voisinage (en fait son étiquette) parmi les restrictions, la définition et l'exemple de l'UW correspondante.
3. Si une association est trouvée, le score de l'alignement est augmenté de 0,5. Si l'association porte sur un parent du concept et une restriction icl de l'UW, le score est augmenté de 1.

L'algorithme, complet et commenté, est donné en annexe G. Il a été exécuté en 45 min sur une machine dotée d'un processeur de 2GHz. Quelques optimisation peuvent toutefois ramener le temps d'exécution à quelques minutes. Notamment, nous chargeons le dictionnaire d'UW (au format SKOS) à l'aide du module PERL XML SIMPLE et la structure de donnée obtenue n'est pas indexée de façon pertinente pour notre tâche. Cela conduit à des parcours superflus de l'ensemble du dictionnaire. La table 5 présente des statistiques sur le résultat de la désambiguïsation. La précision a été évaluée manuellement en vérifiant que, pour un concept donné, l'alignement portant le meilleur score pointait vers un UW valide pour ce concept. En particulier, les concepts LIGHT et GO, dont l'ambiguïté était problématique, ont été désambiguïsés avec succès.

Nb. d'UW++	207023
Nb. de concepts	732
Nb. de liens concept-UW	1979
Nb. de concepts liés à au moins un UW	592
Nb. de concepts désambiguïsés	360 (61%)
Précision pour l'alignement ayant le meilleur score	95%

Table 5 – Statistiques pour la désambiguïsation de l'alignement entre l'ontologie OMNIA et les UW++

4.5 Annotation conceptuelle

À cette étape, nous disposons d'une part d'un graphe-Q annoté par des UW, et d'autre part d'un alignement entre le lexique des UW et les concepts de l'ontologie. Dans un premier temps, nous enrichissons l'annotation des textes avec les éléments de l'ontologie correspondant aux UW dans l'alignement. Toutes les UW ne sont pas alignées avec un concept de l'ontologie. C'est pourquoi, afin d'augmenter la couverture, nous utilisons pour ces UW « orphelines », les

restrictions de type *icl*, *iof* et *equ*, qui correspondent respectivement aux relations *subclass-of*, *instance-of* et *same-class-as* côté ontologie.

C'est en particulier utile pour les entités nommées. Par exemple, si l'on a identifié l'UW *Lyon(iof>city>thing)* dans un texte, et si la ville *Lyon* ne figure pas dans l'ontologie, la recherche dans l'alignement UW-Ontologie ne donnera rien. Toutefois, si l'ontologie contient un concept pour *city*, il pourra être identifié grâce à la restriction *iof>city*.

Nous rencontrons des incertitudes quant aux informations à extraire des textes. Ces incertitudes sont quantifiées à deux niveaux :

– par les scores de désambiguïsation affectés aux UW dans les graphes-Q,

– par les scores de confiance des liens entre UW et concepts.

Aussi, nous associons des scores de confiance aux annotations conceptuelles dans le graphe-Q. Le score associé à un concept est défini comme le produit du score de l'UW qui a permis d'identifier le concept par le score du lien entre le concept et cette UW. D'autres méthodes de calcul pourraient bien entendu être utilisées.

Exemple 4. Les concepts sont ajoutés au graphe-Q sous l'étiquette `$CEPT`. Pour chacun, des scores de vraisemblance sont calculés à partir des scores des UW et de l'alignement entre l'ontologie et les UW.

```
-0- $OCC($FORME(#Uguerre#20#civile),$LU($LEMMA(#Uguerre#20#civile),$CAT(NOUN),$UW($ID(unl.upp.
    civil_war.34984),$CEPT(CIVIL_WAR),$Link2CeptAbsScore(1),$HW(civil_war),$REST(#Uicl#3e#war
    #3e#thing),$SCORE(1.0),$OverallScore(0.011235955056179775)))) -3-
-0- $OCC($FORME(guerre),$LU($LEMMA(guerre),$CAT(NOUN),$UW($ID(unl.upp.war.201047),$CEPT(WAR),$
    Link2CeptAbsScore(1),$HW(war),$REST(#Uicl#3e#hostility#3e#thing#2c##20#ant#3e#peace),$
    SCORE(0.45614035087719296),$OverallScore(0.011235955056179775)),$UW($ID(unl.upp.warfare
    .201122),$HW(warfare),$REST(#Uicl#3e#military#5f#action#3e#thing#2c##20#equ#3e#war),$SCORE
    (0.543859649122807),$OverallScore(0.06741573033707865)))) -1-
-1- $OCC($FORME(#U#20#)) -2-
-2- $OCC($FORME(civile),$LU($LEMMA(civil),$CAT(ADJ))) -3-
-2- $OCC($FORME(civile),$LU($LEMMA(civil),$CAT(NOUN))) -3-
-3- $OCC($FORME(#U#20#)) -4-
-4- $OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(ADJ))) -5-
-4- $OCC($FORME(opposant),$LU($LEMMA(opposant),$CAT(NOUN))) -5-
-4- $OCC($FORME(opposant),$LU($LEMMA(opposer),$CAT(VERB),$UW($ID(unl.upp.play_off.132128),$HW(
    play_off),$REST(#Uicl#3e#confront#3e#do#2c##20#equ#3e#pit#2c##20#agt#3e#thing#2c##20#obj#3
    e#thing),$SCORE(0.4528301886792453),$OverallScore(0.011235955056179775)),$UW($ID(unl.upp.
    oppose.126171),$HW(oppose),$REST(#Uicl#3e#refute#3e#do#2c##20#agt#3e#thing#2c##20#obj#3e#
    thing),$SCORE(0.5471698113207547),$OverallScore(0.06741573033707865)))) -5-
-5- $OCC($FORME(#U#20#)) -6-
-6- $OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(ADJ))) -7-
-6- $OCC($FORME(catholiques),$LU($LEMMA(catholique),$CAT(NOUN),$UW($ID(unl.upp.Catholic.31414)
    ,$CEPT(CATHOLIC),$Link2CeptAbsScore(1),$HW(Catholic),$REST(#Uicl#3e#Christian#3e#thing),$
    SCORE(1.0),$OverallScore(0.011235955056179775)))) -7-
-7- $OCC($FORME(#U#20#)) -8-
-8- $OCC($FORME(et),$LU($LEMMA(et),$CAT(CONJ))) -9-
-9- $OCC($FORME(#U#20#)) -10-
-10- $OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(NOUN))) -11-
-10- $OCC($FORME(protestants),$LU($LEMMA(protestant),$CAT(ADJ))) -11-
```

FIGURE 12 – Ajout des concepts au graphe-Q

4.6 Extraction du contenu relatif à l'ontologie

Cette dernière étape du processus diffère selon que l'on traite un texte accompagnant une image ou la requête d'un utilisateur.

4.6.1 Extraction dans les textes compagnons

Bien que nos travaux aient été réalisés dans la perspective de construire des descriptions complexes pour les documents, nous nous sommes limité à une tâche de classification pour le projet OMNIA. Le processus est donc fort simple : pour chaque concept présent dans l'annotation du texte accompagnant une image, nous créons un axiome du type *image* $\in$ *concept* (*axiome de classification*). Les scores associés aux concepts sont stockés sous forme de commentaires (éléments d'une sous-classe *confidence* de `rdfs:comment`) suivant la syntaxe de la figure 5, section 3.5, chapitre 2. Si un concept apparait deux fois dans l'annotation (avec les scores a et b), le score de l'axiome de classification associé est $a+b-a.b$ (qui est plus grand que $max(a,b)$). Nous avons choisi cet opérateur pour renforcer la confiance dans un concept s'il apparait plusieurs fois. Si le concept apparait plus de deux fois avec les scores $c_1, \cdots, c_n$, on itère l'opération pour obtenir un score égal à :

$$\sum_{p=1}^{n} (-1)^{p+1} \sum_{1 \leq i_1 < \cdots < i_p \leq n} c_{i_1} c_{i_2} \cdots c_{i_p} \text{ (formule du crible de Poincaré).}$$

Les scores que nous associons aux axiomes ne sont pas utilisables par des moteurs d'inférence classiques comme Pellet (Sirin et al. (2007)) ou Racer (Haarslev et Müller (2001)). Nous les interprétons comme des degrés d'appartenance floue d'une image à un concept. De fait, une extension floue du langage OWL et des raisonneurs flous permettraient d'exploiter ces scores comme souhaité. Bobillo et Straccia (2010) mentionnent trois raisonneurs disponibles pour des extensions floues de OWL : FuzzyDL (Bobillo et Straccia (2008)), DeLorean (Bobillo et al. (2008)), et Fire (Stoilos et al. (2006)). Toutefois, ces raisonneurs ne sont pas optimisés pour le traitement de grandes masses de données et n'ont pas été utilisés pour OMNIA à cause des impératifs de passage à l'échelle.

Nous propageons ensuite les axiomes de classification et les scores extraits en suivant le treillis conceptuel de l'ontologie. Ainsi, tous les axiomes déductibles sont explicités et stockés dans une base de données relationnelle classique. Les scores des axiomes inférés sont calculés grâce à des opérateurs de logique floue qui permettent de calculer les degrés d'appartenance d'un élément pour des unions ou des intersections d'ensembles flous.

Plusieurs familles d'opérateurs permettent de définir de façon satisfaisante une théorie des ensembles flous. Nous avons choisi les opérateurs min/max, qui semblent les plus communément admis.

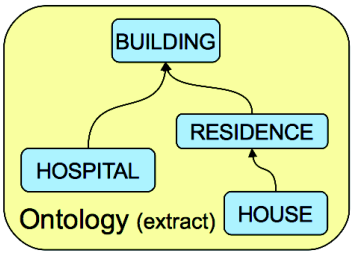


FIGURE 13 – Extrait de l'ontologie

Définition 1. On note $M_E(x)$ le **degré d'appartenance** d'un élément x à un ensemble E.
1. $A \subseteq B \iff (\forall x)\,[M_A(x) \leq M_B(x)]$
2. $M_{\overline{A}}(x) = 1 - M_A(x)$
3. $M_{A \cap B}(x) = min(M_A(x), M_B(x))$
4. $M_{A \cup B}(x) = max(M_A(x), M_B(x))$

Ainsi, en considérant l'extrait du treillis conceptuel de l'ontologie ci-dessus, si pour une image i on extrait (à partir des annotations) les scores :

$M_{HOUSE}(i) = 0,5$ et $M_{HOSPITAL}(i) = 0.7$, on peut déduire :
$M_{RESIDENCE}(i) \geq M_{HOUSE}(i) = 0,5$
et $M_{BUILDING}(i) \geq max(M_{HOSPITAL}(i), M_{RESIDENCE}(i)) = 0,7$
car $HOUSE \subseteq RESIDENCE$ et $(HOSPITAL \cup RESIDENCE) \subseteq BUILDING$

On remarque que les scores propagés et stockés sont des bornes inférieures pour les degrés d'appartenance.

Les résultats sont exportés en XML comme illustré dans l'exemple 5. Pour assurer le passage à l'échelle dans le cadre du projet OMNIA, ils sont stockés dans une base de données relationnelle MySQL.

Exemple 5. On remarque que la propagation dans le treillis conceptuel permet d'identifier plus de concepts que ceux directement contenus dans le texte.

```
<contentExtraction>
<comment>Version 0.4.1</comment>
<concept name="POLITICS" likelihood="0.0549314139295081 1"/>
<concept name="CHRISTIAN" likelihood="0.011235955056179775"/>
<concept name="CIVIL_WAR" likelihood="0.011235955056179775"/>
<concept name="CATHOLIC" likelihood="0.011235955056179775"/>
<concept name="PEOPLE" likelihood="0.07604959846864168"/>
<concept name="WAR" likelihood="0.02234566342633506"/>
<concept name="CHRISTIANISM" likelihood="0.02234566342633506"/>
<concept name="RELIGION" likelihood="0.3034291500629581"/>
<concept name="MONOTHEISM" likelihood="0.07604959846864168"/>
<concept name="PERSON" likelihood="0.02234566342633506"/>
</contentExtraction>
```

FIGURE 14 – Concepts extraits à partir du texte compagnon "*guerre civile opposant catholiques et protestants*"

4.6.2 Extraction dans les requêtes

Le calcul du score associé à un concept présent dans la requête se fait de la même façon que pour un texte compagnon. Toutefois, ayant réalisé une propagation des descriptions des images en suivant le treillis conceptuel de l'ontologie (de façon ascendante), il n'est pas pertinent de faire de même (de façon descendante) pour les requêtes. Nous réalisons la propagation au niveau des descriptions pour gagner du temps sur le traitement des requêtes. Les concepts extraits (en XML) sont utilisés pour créer une requête conjonctive SQL.

Exemple 6. Si le texte est une requête, on se contente d'extraire les concepts directement contenus dans le texte.

```
<contentExtraction>
<comment>Version 0.4.1</comment>
<concept name="CIVIL_WAR" likelihood="0.011235955056179775"/>
<concept name="CATHOLIC" likelihood="0.011235955056179775"/>
<concept name="WAR" likelihood="0.02234566342633506"/>
</contentExtraction>
```

FIGURE 15 – Concepts extraits à partir du texte "*guerre civile opposant catholiques et protestants*" considéré comme une requête.

5 Démonstrateur pour la recherche d'images accompagnées de textes

Les services développés ont été intégrés par Achille Falaise dans une interface graphique en ligne. Ils sont également accessibles séparément via leurs interfaces REST. La description de ces implémentations est décrite par Rouquet et al. (2010a).

5.1 Appel des composants

Les composants, développés sous forme de services Web, sont aisément accessibles. La documentation de chaque service (et notamment la liste de ses paramètres) est accessible simplement en y accédant sans paramètre à l'aide d'un navigateur Web.

Service	URL
Lemmatisation	`http://aiakide.net/omnia/lemmatix`
Annotation interlingue	`http://aiakide.net/omnia/uwix`
Désambiguïsation	`http://aiakide.net/omnia/desamb`
Annotation conceptuelle	`http://aiakide.net/omnia/wnid`
Scores et propagation	`http://aiakide.net:8180/Omnia3-web/OmniaLexicalExtractorServlet`

TABLE 6 – Adresses et paramètres des différents services.

Ils peuvent par exemple être appelés à l'aide du logiciel Curl [55]. Nous donnons ici des exemples de commandes sur un système Unix, pour l'analyse d'un texte anglais nommé *test.txt*, lemmatisé avec une fenêtre de 5 mots, et dont les concepts sont issus d'une ontologie *onto-6* intégrée au système. Les résultats intermédiaires sont placés dans les fichiers *1.tmp*, *2.tmp*, etc. et le résultat final est dans *res.xml*. Pour des raisons de place et de lisibilité, chaque commande est décomposée sur plusieurs lignes.

```
curl http://aiakide.net/omnia/lemmatix/
  --data-urlencode text@test.txt -d lang=eng -d window=5
  > 1.tmp
```

Cette commande renvoie sous forme d'un graphe-Q la lemmatisation du texte contenu dans le fichier `test.txt`. Le paramètre `lang` peut prendre la valeur `fra` pour traiter des textes en français. Le paramètre `window` détermine la taille de la fenêtre utilisée pour la recherche des mots composés lors de la lemmatisation.

```
curl http://aiakide.net/omnia/uwix/
  --data-urlencode gq@1.tmp -d lang=eng
    > 2.tmp
```

Cette commande ajoute les UW++ possibles dans le graphe-Q du texte lemmatisé contenu dans le fichier `1.tmp`. Cette commande est paramétrée par la langue, car elle utilise un dictionnaire "langue du texte"-UW.

```
curl http://aiakide.net/omnia/desamb/
  --data-urlencode gq@2.tmp
  > 3.tmp
```

Cette commande ajoute des scores de désambiguïsation aux UW++ annotant le texte contenu dans le fichier `2.tmp`. Cette commande est indépendante de la langue source du texte, puisque la désambiguïsation se fait sur les UW, à un niveau interlingue.

55. `http://curl.haxx.se/`

```
curl http://aiakide.net/omnia/wnid/
  --data-urlencode gq@3.tmp --data-urlencode onto=onto-6
  > 4.tmp
```

Cette commande renvoie l'annotation conceptuelle du texte sous forme d'un graph-Q, annoté par des UW++ pondérées, contenu dans le fichier `2.tmp`. Cette commande est paramétrée par une ontologie alignée au préalable avec un ensemble d'UW++.

```
curl http://aiakide.net:8180/Omnia3-web/OmniaLexicalExtractorServlet
  --data-urlencode onto=onto-6 -d gqType=UW
  --data-urlencode gq@4.tmp -d outFormat=XML
  > res.xml
```

Cette commande renvoie l'ensemble des concepts contenus dans le texte et les scores associés, en format XML. Cette forme intermédiaire permet ensuite de construire les descriptions dans l'ontologie si l'entrée est un texte compagnon et des requêtes SQL si le texte en entrée est une requête de l'utilisateur.

5.2 Interface d'évaluation

Une interface Web (figure 16) a été développée par Achille Falaise (Falaise et al. (2010), Rouquet et al. (2010a)) en vue d'évaluer notre système en situation, à travers le scénario d'utilisation établi pour le projet : une recherche d'images dans un cadre prépresse. Cette interface permet de soumettre une requête, qui est analysée comme décrit précédemment, et affiche les images correspondantes, dont les textes compagnons ont été analysés au préalable (un échantillon de 61393 images issues de la base BELGA news).

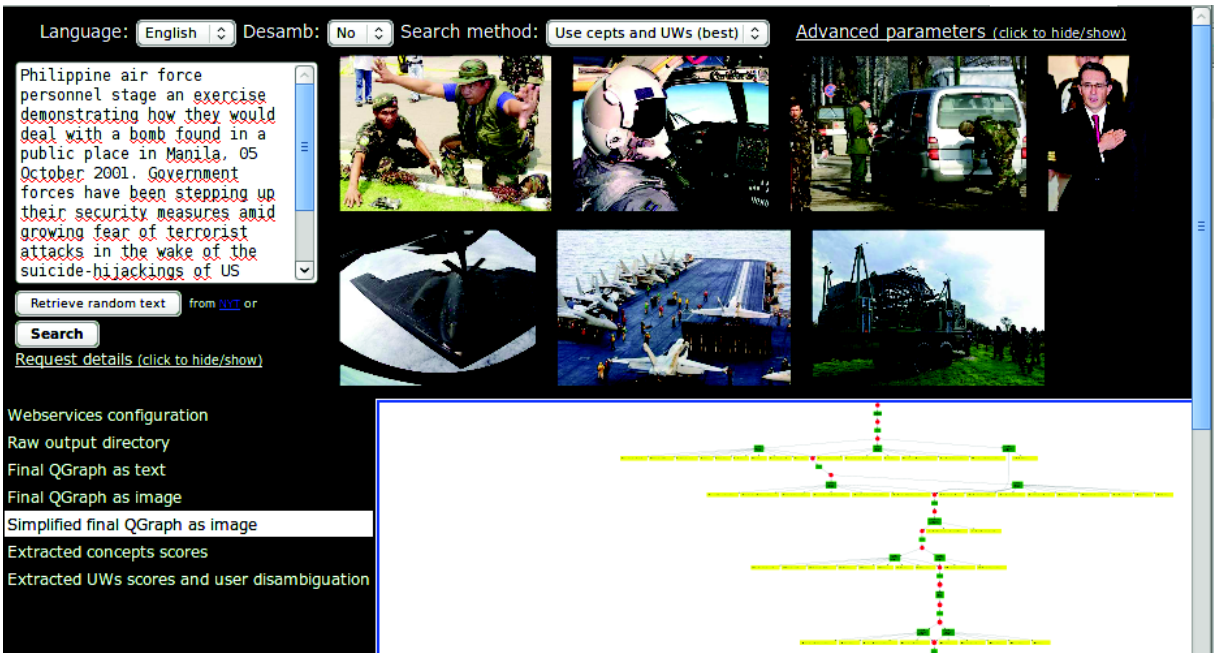


FIGURE 16 – Interface de test d'A. Falaise. Exemple de résultat pour le texte "*Philippine air force personnel stage an exercise demonstrating how they would deal with a bomb found in a public place in Manila, 05 October 2001. Government forces have been stepping up their security measures amid growing fear of terrorist attacks in the wake of the suicide-hijackings of US aircraft that claimed thousands of lives on September 11*".

Il est possible de changer la langue du système (le français et l'anglais sont disponibles), d'activer la désambiguïsation et de choisir le mode de recherche : (1) utiliser uniquement les concepts trouvés par le système, (2) utiliser uniquement les UW trouvées, ou bien (3) combiner les deux résultats.

Les images sont classées en fonction de leur score d'adéquation avec la requête, dont le calcul diffère suivant le mode de recherche. Les scores d'adéquation sont calculés de manière assez simple, puisque notre travail portait sur l'indexation et l'annotation des documents et pas sur les algorithmes de recherche.

Tous les résultats d'analyse sont consultables dans l'interface. Pour la requête, il s'agit d'une représentation textuelle ou graphique du graphe-Q final, des concepts et des UW extraits. Pour les images de la base, il est possible de consulter le texte compagnon, le graphe-Q résultant de son analyse, les concepts et les UW extraits, ainsi qu'une quantification de leur participation au score global de l'image.

L'utilisateur peut corriger les scores des UW de sa requête attribués par la désambiguïsation automatique (figure 17). Il peut le faire, soit directement à partir du résultat d'analyse de sa requête, soit à partir du résultat d'analyse d'une image. Dans ce dernier cas, il ne peut intervenir que sur les scores des UW de l'image qui sont aussi présents dans la requête (ceux qui participent à l'adéquation entre l'image et de la requête). Il peut ainsi "faire le tri" dans les images retournées par le système, en corrigeant la désambiguïsation sémantique uniquement sur les UW qui participent à des résultats erronés.

FIGURE 17 – Possibilité de désambiguïsation manuelle des UW dans l'interface de test d'A. Falaise.

Conclusion générale et perspectives

Dans ce mémoire, nous avons présenté un certain nombre de travaux réalisés lors du projet ANR OMNIA de 2008 à 2011. Nous nous sommes placé dans un contexte de recherche sémantique d'information (guidée par une ontologie), exploré par de nombreux projets depuis les années 1990. Nous avons contribué au développement de l'aspect multilingue, en permettant de considérer une ou plusieurs ontologies comme des paramètres du système.

Nous avons d'abord précisé les définitions liées aux ontologies "logiques" adoptées dans le cadre du projet OMNIA. Nous nous sommes efforcé de séparer clairement les niveaux logiques et terminologiques de telles structures pour faciliter leur utilisation en TALN, en particulier pour l'extraction de contenu dans des textes. Nous avons par ailleurs proposé une notion de *description d'un objet dans une ontologie*, prolongeant la notion de descripteur utilisée en recherche documentaire classique. Le caractère opérationnel des définitions liées à cette notion (description complète, minimale, etc.), par exemple pour réaliser des optimisations lors de l'utilisation d'ontologies très grandes ou complexes, n'a toutefois pas été évalué durant cette thèse.

Afin de faciliter l'échange et la réutilisation des données, nous nous sommes reposé sur les standards du W3C pour le Web sémantique, en particulier le langage OWL. Dans un souci de passage à l'échelle, pour la résolution de requêtes sur une ontologie dont la A-box contient un grand nombre d'instances (centaines de milliers voire millions), nous nous sommes restreint à des sous-langages de OWL basés sur la famille de logiques de description DL-lite (Calvanese (2005)).

Un frein majeur quand on veut utiliser des moteurs d'inférence actuels pour résoudre des requêtes sur l'ontologie est de gérer les scores de confiance obtenus lors de l'indexation automatique des documents (couples image-texte), de façon à classer les images proposées à l'utilisateur. L'utilisation de moteurs intégrant des modèles d'ontologie floue pour interroger de très grandes A-box n'a pas été concluante dans le cadre d'OMNIA et nous nous sommes replié sur l'utilisation de bases de données relationnelles classiques. C'est pourquoi nous souhaitons dans le futur renforcer le modèle flou dans lequel peut être formalisée notre méthode d'extraction de contenu, et expérimenter le passage à l'échelle avec des moteurs d'inférence flous.

Un aspect important que nous avons exploré avec le projet OMNIA est celui de la multilinguisation de systèmes de recherche sémantiques d'information. Trois groupes de langues sont à considérer dans un tel système : (1) les langues utilisées dans les ressources sémantiques (ontologies), (2) les langues utilisées dans les requêtes et (3) les langues utilisées dans les documents ou pour décrire les documents. Afin de permettre une recherche translingue, il est logique de gérer le multilinguisme au niveau des ontologies vu le rôle de pivot qu'elles occupent (elles supportent l'indexation et la résolution des requêtes).

Nous nous sommes donc intéressé au problème de la multilinguisation d'ontologies. Une contrainte importante est qu'un utilisateur puisse mettre à jour ou changer les ontologies utilisées par le système. La multilinguisation doit donc être faite *a posteriori*, et en respectant la complexité initiale de l'ontologie. Une revue de l'état de l'art sur l'enrichissement linguistique (plus particulièrement lexical) d'ontologies, y compris en contexte monolingue, nous a amené à

proposer une solution de *lexicalisation externe*.

L'ontologie est reliée à des ressources lexicales externes exprimées dans un dialecte OWL (typiquement dans le formalisme SKOS recommandé par le W3C). L'alignement ontologie-lexique s'apparente alors à un alignement entre ontologies avec un certain nombre de différences que nous avons identifiées. Nos alignements sont représentés dans le format proposé par David et al. (2011) et manipulables à l'aide de l'API dédiée.

Les liens entre les éléments de l'ontologie et les entrées du lexique sont de deux types. (1) Les liens lexicaux procurent une lexicalisation pour les objets de l'ontologie. (2) Les liens structurels associent des relations du lexique à des relations de l'ontologie pour lesquelles une lexicalisation n'est pas pertinente (par exemple l'hyponymie à la relation `rdfs:subClassOf`). Les liens structurels fournissent des contraintes d'intégrité pour l'alignement. Ce dernier est idéalement un morphisme entre les les structures du lexique et de l'ontologie. Une possibilité pour vérifier un alignement serait donc de s'assurer que c'est bien un morphisme. Toutefois, les structures lexicales et ontologiques d'un domaine ne sont en général pas superposables (Roche (2007)). Il convient donc d'utiliser les contraintes structurelles sur un alignement de façon "souple", pour renforcer des liens entre l'ontologie et le lexique, sans rendre l'alignement formellement invalide.

Une perspective intéressante est la construction d'espaces métriques, à partir des structures de l'ontologie et du lexique, permettant d'approcher la notion de morphisme. Cette idée s'inspire de techniques d'approximation pour la résolution de requêtes (Gandon et al. (2008)). Une autre possibilité est de plonger l'ontologie et le lexique dans un même espace métrique pour évaluer leur proximité. Nous sommes déjà en train d'explorer cette voie à l'aide de vecteurs conceptuels, associés automatiquement aux éléments de l'ontologie et du lexique, et pour lesquels la distance angulaire peut être interprétée comme une distance thématique (Schwab (2005)). Enfin, l'utilisation d'espaces métriques permettrait de formaliser des méthodes comme celle de Espinoza et al. (2008) pour désambiguïser la traduction des étiquettes d'une ontologie. En effet, la notion de voisinage utilisée peut être mieux définie (et manipulée) en utilisant un espace métrique adéquat.

Dans le cadre d'une lexicalisation externe d'ontologies, nous avons choisi de gérer le multilinguisme en alignant l'ontologie avec un lexique pivot, les UW (lexèmes interlingues) du langage UNL, qui sont organisées en une "préontologie". Dans notre architecture, les liens de traduction entre l'ontologie, le langage pivot, et les langues naturelles sont tous représentés de façon abstraite sous forme d'axies (acceptions interlingues). Nous considérons le lexique formé d'UW comme un pivot linguistico-sémantique, l'ontologie comme un pivot conceptuel, et le volume d'axies comme un pivot abstrait, placé au centre de l'architecture multilingue. En outre, cette architecture facilite grandement le versionnage des ressources, tant ontologiques que lexicales. Dans le cadre du projet OMNIA, nous avons utilisé la plate-forme de gestion de ressources lexicales Jibiki (Mangeot et Chalvin (2006), Sérasset et al. (2006)) pour le support et l'exploitation de l'architecture multilingue.

L'utilisation des UW d'UNL comme espace lexical pivot à large couverture offre plusieurs avantages pour une extraction de contenu sémantique dans des textes multilingues. Elle permet notamment d'appuyer l'extraction de contenu (ou l'annotation) sémantique sur une annotation interlingue et pré-sémantique des textes, avec des UW. Le processus d'extraction de contenu est en quelque sorte factorisé, puisque, pour n langues considérées, nous n'avons plus besoin de n extracteurs de contenu mais d'un seul (opérant sur l'annotation interlingue des textes), associé à n analyseurs morphologiques pour produire les annotations interlingues. La phase de désambiguïsation, la plus couteuse en calcul pour l'annotation sémantique, est alors réalisée une fois pour toute lors de l'annotation interlingue et il est possible de passer une ontologie en paramètre du système en calculant un alignement avec le langage pivot. Enfin, les possibilités d'UNL pour représenter formellement le sens d'un énoncé permettent d'envisager, dans un futur

plus lointain, une extraction de contenu riche et structurée, à un niveau interlingue, en alignant des constructions linguistico-sémantiques d'UNL avec des constructions logiques dans l'ontologie (Cardeñosa et al. (2008)).

L'annotation des textes est réalisée en utilisant le langage-Q de Colmerauer (1970). Les graphes de chaînes (graphes-Q) manipulés par ce langage sont bien adaptés à la représentation des nombreuses ambiguïtés rencontrées lors de l'annotation par des UW, puis par des concepts. De plus, les systèmes-Q, des règles de réécriture sur les graphes-Q, permettent d'envisager l'unification des traitements de surface effectués sur les textes et leur manipulation par des experts non-informaticiens.

Lors de l'annotation des textes, les ambiguïtés sont rendues explicites. Des procédés de désambiguïsation automatique permettent d'améliorer la précision des annotations. Ils sont réalisés sur l'annotation des textes par des UW et ne dépendent donc pas de la langue originale des textes. Nous avons utilisé une méthode inspirée de Lesk (1986) pour la désambiguïsation en sens de mots. L'utilisation d'algorithmes à colonies de fourmis pour la mise en œuvre globale de cette méthodes sur des graphes-Q a permis le passage à l'échelle (Schwab et al. (2011)). Les algorithmes à colonies de fourmis sont efficaces pour la résolution de problèmes modélisables sous forme de graphes. Leur utilisation pour la désambiguïsation d'alignements entre une ontologie et un lexique (qui est en fait un graphe biparti) fera l'objet de futurs travaux.

Enfin, nous avons présenté le démonstrateur du projet OMNIA pour la recherche d'images accompagnées de textes de la base Belga News (Falaise et al. (2010)). Les requêtes des utilisateurs et les textes accompagnant les images subissent le même traitement jusqu'à la dernière étape de l'extraction de contenu. L'implémentation du démonstrateur est réalisée de façon modulaire, suivant une architecture à services, de sorte que chaque composant puisse être amélioré indépendamment.

Bibliographie

[Agirre et Edmonds (2006)]. Eneko Agirre et Philip Edmonds. *Word Sense Disambiguation : Algorithms and Applications (Text, Speech and Language Technology)*. Springer-Verlag, Secaucus, NJ, USA, 2006. ISBN 1402048084. 364p.

[Baader et al. (2003)]. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, et Peter F. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003. ISBN 0-521-78176-0. 555p.

[Baader et al. (2005)]. Franz Baader, Sebastian Brand, et Carsten Lutz. *Pushing the EL envelope*. Proc. IJCAI 2005, pp. 364–369. Morgan-Kaufmann Publishers, 2005.

[Banerjee et Pedersen (2002)]. Satanjeev Banerjee et Ted Pedersen. *An adapted lesk algorithm for word sense disambiguation using wordnet*. Proc. Third International Conference on Computational Linguistics and Intelligent Text Processing, CICLing '02, pp. 136–145, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43219-1.

[Beale et al. (1995)]. Stephen Beale, Sergei Nirenburg, et Kavi Mahesh. *Semantic analysis in the Mikrokosmos machine translation project*. Proc. Second Symposium on Natural Language Processing (SNLP- 95), pp. 297–307, Bangkok, Thailand, 1995.

[Beck (2008)]. Timo Beck. *Web 2.0 : user-generated content in online communities : a theoretical and empirical investigation of its determinants*. Diplomica Verlag, 2008. ISBN 9783836654920. 124p.

[Berners-Lee et al. (2001)]. Tim Berners-Lee, James Hendler, et Ora Lassila. *A new form of web content that is meaningful to computers will unleash a revolution of new possibilities*. Scientific American, 284(5) pp. 34–43, May 2001. ISSN 00368733.

[Bobillo et Straccia (2008)]. F. Bobillo et U. Straccia. *fuzzyDL : an expressive fuzzy description logic reasoner*. Proc. IEEE International Conference on Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence), pp. 923–930. IEEE, June 2008. ISBN 978-1-4244-1818-3. doi : 10.1109/FUZZY.2008.4630480.

[Bobillo et Straccia (2010)]. F. Bobillo et U. Straccia. *Representing fuzzy ontologies in OWL 2*. Proc. IEEE International Conference on Fuzzy Systems (FUZZ), pp. 1–6. IEEE, July 2010. ISBN 978-1-4244-6919-2. doi : 10.1109/FUZZY.2010.5584661.

[Bobillo et al. (2008)]. Fernando Bobillo, Miguel Delgado, et Juan Gómez-Romero. *Delorean : A reasoner for fuzzy owl 1.1*. Proc. 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008), Karlsruhe, 2008.

[Boguslavsky et al. (2005)]. I. Boguslavsky, J. Cardenosa, C. Gallardo, et L. Iraola. *The UNL initiative : an overview.* Alexander Gelbukh, editor, Computational Linguistics and Intelligent Text Processing, volume 3406, pp. 377–387. Springer, Berlin, Heidelberg, 2005. ISBN 978-3-540-24523-0, 978-3-540-30586-6.

[Boguslavsky et al. (2007)]. I. Boguslavsky, J. Bekios, J. Cardenosa, et C. Gallardo. *Using Wordnet for building an interlingual dictionary.* Proc. Fifth International Conference "Information Research and Applications" (TECH'2007), pp. 39–45, Varna, Bulgaria, 2007. ISBN 1313-1109.

[Boitet (2003)]. Christian Boitet. *Automated translation.* Revue française de linguistique appliquée, VIII(2) pp. 99–121, 2003. ISSN 1386-1204.

[Boitet et Tchéou (1990)]. Christian Boitet et François Tchéou. *On a phonetic and structural encoding of chinese characters in chinese texts.* Proc. ROCLING III, pp. 73–80, Taipeh, 1990.

[Boitet et al. (2007)]. Christian Boitet, Igor M. Boguslavsky, et Jesus Cardeñosa. *An evaluation of unl usability for high quality multilingualization and projections for a future unl++ language.* Proc. International Conference on Computational Linguistics and Intelligent Text Processing, CICLing '07, pp. 361–373. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-70938-1. doi : http://dx.doi.org/10.1007/978-3-540-70939-8_32.

[Bonabeau et Théraulaz (2000)]. Éric Bonabeau et Guy Théraulaz. *L'intelligence en essaim.* Pour la science, (271) pp. 66–73, 2000.

[Breitman et al. (2007)]. Karin K. Breitman, Marco Antonio Casanova, Walt Truszkowski, United States. National Aeronautics Administration, et Space. *Semantic Web : concepts, technologies and applications.* Springer, 2007. ISBN 9781846285813. 64p.

[Buitelaar et Declerck (2003)]. Paul Buitelaar et Thierry Declerck. *Linguistic annotation for the semantic web.* Annotation for the semantic web, pp. 193–111, 2003. ISSN 158603345X.

[Buitelaar et al. (2006)]. Paul Buitelaar, Thierry Declerck, Anette Frank, Malte Kiesel, Michael Sintek, Massimo Romanelli, Daniel Sonntag, Berenike Loos, Vanessa Micelli, et Robert Porzel. *LingInfo : design and applications of a model for the integration of linguistic information in ontologies.* Proc. OntoLex workshop 2006 (LREC), pp. 28–32, 2006.

[Cacaly et al. (2008)]. S. Cacaly, E. Sutter, P.D. Pomart, et Y.F.L. Coadic. *Dictionnaire de l'information.* Dictionnaire (Paris. 2000). Armand Colin, 2008. ISBN 9782200351328. 295p.

[Calegari et Ciucci (2007)]. Silvia Calegari et Davide Ciucci. *Fuzzy ontology, fuzzy description logics and Fuzzy-OWL.* Applications of Fuzzy Sets Theory, volume 4578, pp. 118–126. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73399-7, 978-3-540-73400-0.

[Calvanese (2005)]. Diego Calvanese. *DL-Lite : tractable description logics for ontologies.* Proc. Twentieth National Conference on Artificial Intelligence (AAAI 2005), pp. 602–607, Pittsburgh, Pennsylvania, 2005.

[Calvanese et al. (2007)]. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, et Riccardo Rosati. *Tractable reasoning and efficient query answering in description logics : the DL-Lite family.* Journal of Automated Reasoning, 39(3) pp. 385–429, October 2007. doi : 10.1007/s10817-007-9078-x.

[Cardeñosa et al. (2008)]. Jesús Cardeñosa, Carolina Gallardo, Luis Iraola, et Miguel Angel de la Villa. *A new knowledge representation model to support multilingual ontologies - a case study.* Proc. International Conference on Semantic Web and Web Services, pp. 313–319, Las Vegas, 2008. ISBN 1-60132-089-2.

[Chandioux et Guéraud (1981)]. John Chandioux et Marie-France Guéraud. *METEO : un système à l'épreuve du temps.* Meta, 26(1) pp. 18–22, 1981. ISSN 0026-0452.

[Collier et al. (2006)]. Nigel Collier, Ai Kawazoe, Lihua Jin, Mika Shigematsu, Dinh Dien, Roberto Barrero, Koichi Takeuchi, et Asanee Kawtrakul. *A multilingual ontology for infectious disease surveillance : rationale, design and challenges.* Language Resources and Evaluation, 40(3) pp. 405–413, December 2006. doi : 10.1007/s10579-007-9019-7.

[Colmerauer (1970)]. Alain Colmerauer. *Les systèmes-q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur.* Publication interne du département d'informatique de l'Université de Montréal, (43), 1970.

[Cowie et al. (1992)]. Jim Cowie, Joe Guthrie, et Louise Guthrie. *Lexical disambiguation using simulated annealing.* COLING 1992, International Conference on Computational Linguistics, volume 1, pp. 359–365, Nantes, France, août 1992.

[Cramer et al. (2010)]. Irene Cramer, Tonio Wandmacher, et Uli Waltinger. *WordNet : An electronic lexical database*, chapter Modeling, Learning and Processing of Text Technological Data Structures. Springer, 2010.

[David et al. (2011)]. Jérôme David, Jérôme Euzenat, François Scharffe, et Cássia Trojahn dos Santos. *The alignment API 4.0.* Semantic Web, 2(1) pp. 3–10, January 2011. doi : 10.3233/SW-2011-0028.

[Deneubourg et al. (1989)]. J.L. Deneubourg, S. Gross, N. Franks, et J.M. Pasteels. *The blind leading the blind : Modeling chemically mediated army ant raid patterns.* Journal of Insect Behavior, 2 pp. 719–725, 1989.

[Doets (1996)]. K. Doets. *Basic model theory.* Studies in logic, language, and information. CLSI Publications, 1996. ISBN 9781575860480. 130p.

[Dorigo et Stützle (2004)]. Dorigo et Stützle. *Ant colony optimization.* MIT Press, 2004. ISBN 9780262042192. 324p.

[Dorigo et Gambardella (1997)]. Marco Dorigo et Luca Gambardella. *Ant colony system : A cooperative learning approach to the traveling salesman problem.* IEEE Transactions on Evolutionary Computation, 1 pp. 53–66, 1997.

[Drogoul (1995)]. Alexis Drogoul. *When ants play chess (or can strategies emerge from tactical behaviors ?).* Proc. Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 93), pp. 13–27. Springer, 1995.

[Ellman (2003)]. Jeremy Ellman. *EuroWordNet : a multilingual database with lexical semantic network.* Nat. Lang. Eng., 9 pp. 427–430, December 2003. ISSN 1351-3249. doi : 10.1017/S1351324903223299.

[Espinoza et al. (2008)]. Mauricio Espinoza, Asuncion Gomez-Pérez, et Eduardo Mena. *Enriching an ontology with multilingual information.* The Semantic Web : Research and Applications, volume 4011 of Lecture Notes in Computer Science, pp. 333–347. Springer, 2008. ISBN 978-3-540-34544-2.

[Euzenat (2004)]. Jérôme Euzenat. *An API for ontology alignment.* Proc. 3rd International Semantic Web Conference, pp. 698–7112, Hiroshima, Japan, 2004.

[Falaise et al. (2010)]. Achille Falaise, David Rouquet, Didier Schwab, Hervé Blanchon, et Christian Boitet. *Ontology driven content extraction using interlingual annotation of texts in the OMNIA project.* Proc. 4th Workshop on Cross Lingual Information Access (Coling 2010), pp. 52–60, Pekin, Chine, 2010.

[Fellbaum (1998)]. Christiane Fellbaum. *WordNet : An Electronic Lexical Database.* The MIT Press, May 1998. ISBN 026206197X. 423p.

[Fielding (2000)]. Roy T. Fielding. *Architectural styles and the design of network-based software architectures.* Thèse de doctorat, University of California, 2000p.

[Francopoulo et al. (2006)]. Gil Francopoulo, Nuria Bel, Monte George, Nicoletta Calzolari, Monica Monachini, Mandy Pet, et Claudia Soria. *Lexical markup framework (LMF) for nlp multilingual resources.* Proc. Workshop on Multilingual Language Resources and Interoperability, MLRI '06, pp. 1–8, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. ISBN 1-932432-82-5.

[Gale et al. (1992)]. William Gale, Kenneth Church, et David Yarowsky. *One sense per discourse.* Proc. Fifth DARPA Speech and Natural Language Workshop, pp. 233–237, Harriman, New-York, États-Unis, 1992.

[Gandon et al. (2008)]. Fabien Gandon, Olivier Corby, Ibrahima Diop, et Moussa Lo. *Distances sémantiques dans des applications de gestion d'information utilisant le web sémantique.* Proc. Workshop Mesures de similarités sémantique, EGC, INRIA Sophia Antipolis, 2008.

[Gelbukh et al. (2003)]. Alexander Gelbukh, Grigori Sidorov, et Sang Yong Han. *Evolutionary approach to natural language word sense disambiguation through global coherence optimization.* WSEAS Transactions on Communications, 2(1) pp. 11–19, 2003.

[Grau et al. (2008)]. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, et Ulrike Sattler. *Modular reuse of ontologies : theory and practice.* JAIR, 31 pp. 273–318, 2008.

[Grishman (1997)]. Ralph Grishman. *Information extraction : techniques and challenges.* International Summer School on Information Extraction : a Multidisciplinary Approach to an Emerging Information Technology, pp. 10–27. Springer-Verlag, 1997. ISBN 3-540-63438-X.

[Gruber (1993)]. Thomas R. Gruber. *A translation approach to portable ontology specifications.* Knowledge Acquisition, 5(2) pp. 199–220, 1993. ISSN 1042-8143.

[Guinand et Lafourcade (2009)]. Frédéric Guinand et Mathieu Lafourcade. *Fourmis artificielles et traitement de la langue naturelle.* Fourmis Artificielles 2, pp. 225–267. Lavoisier, 2009.

[Haarslev et Müller (2001)]. Volker Haarslev et Ralf Müller. *RACER system description.* Automated Reasoning, volume 2083, pp. 701–705. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-42254-9.

[Haav et Lubi (2001)]. Hele-Mai Haav et Tanel-Lauri Lubi. *A survey of concept-based information retrieval tools on the web.* Proc. 5th East-European Conference ADBIS, volume 2, pp. 29–41, 2001.

[Hajlaoui et Boitet (2007)]. Najeh Hajlaoui et Christian Boitet. *Portage linguistique d'applications de gestion de contenu.* Proc. TOTh07, Annecy, 2007.

[Handschuh et Staab (2003)]. Siegfried Handschuh et Steffen Staab. *Annotation for the semantic web.* IOS Press, January 2003. ISBN 9781586033453p.

[Hao et al. (1999)]. Jin-Kao Hao, Philippe Galinier, et Michel Habib. *Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes.* Revue d'Intelligence Artificielle, 2(13) pp. 263–324, 1999.

[Horak et al. (2006)]. Ales Horak, Karel Pala, Adam Rambousek, et Martin Povolny. *DEBVisDic - first version of new Client-Server wordnet browsing and editing tool.* Proc. Third International WordNet Conference, pp. 325–328, Brno, République Tchèque, 2006. ISBN 80-210-3915-9.

[Horn (1951)]. Alfred Horn. *On Sentences Which are True of Direct Unions of Algebras.* The Journal of Symbolic Logic, 16(1) pp. 14–21, 1951. ISSN 00224812.

[Huynh et al. (2008)]. Cong-Phap Huynh, Christian Boitet, et Hervé Blanchon. *SECTra_w.1 : an online collaborative system for evaluating, post-editing and presenting mt translation corpora.* Proc. Sixth International Language Resources and Evaluation Conference (LREC'08), Marrakech, Morocco, may 2008.

[Ide et Véronis (1998)]. Nancy Ide et Jean Véronis. *Word sense disambiguation : the state of the art.* Computational Linguistics, 28(1) pp. 1–41, 1998.

[Isabelle et Bourbeau (1985)]. Pierre Isabelle et Laurent Bourbeau. *TAUM-AVIATION : its technical features and some experimental results.* Computational Linguistics, 11(1) pp. 18–27, 1985. ISSN 08912017.

[Konev et al. (2009)]. Boris Konev, Carsten Lutz, Dirk Walther, et Frank Wolter. *Formal properties of modularisation.* Ontology Modularisation, LNCS, pp. 25–66. Springer-Verlag, 2009. ISBN 978-3-642-01906-7.

[Lafourcade et Schwab (2005)]. Mathieu Lafourcade et Didier Schwab. *Multi-caste ants algorithms for holistic semantic text analysis.* Proc. SNLP'2005 : the 6th Symposium on Natural Language Processing, Chiang Rai, Thailand, 2005.

[Lenat et Guha (1989)]. Douglas Lenat et R.V. Guha. *Building Large Knowledge-Based Systems : representation and Inference in the CYC Project.* Addison-Wesley Longman Publishing Co., 1989. ISBN 0201517523. 372p.

[Lesk (1986)]. Michael Lesk. *Automatic sense disambiguation using machine readable dictionaries : how to tell a pine cone from an ice cream cone.* Proc. 5th annual international conference on Systems documentation, SIGDOC '86, pp. 24–26, New York, NY, USA, 1986. ACM. ISBN 0-89791-224-1. doi : http://doi.acm.org.gate6.inist.fr/10.1145/318723.318728.

[Lew et al. (2006)]. Michael S. Lew, Nicu Sebe, Chabane Djeraba, et Ramesh Jain. *Content-based multimedia information retrieval : State of the art and challenges.* ACM Trans. Multimedia

Comput. Commun. Appl., 2 pp. 1–19, February 2006. ISSN 1551-6857. doi : http://doi.acm.org/10.1145/1126004.1126005.

[Lutz et al. (2007)]. Carsten Lutz, Dirk Walther, et Frank Wolter. *Conservative extensions in expressive description logics.* Proc. IJCAI-2007, pp. 453–459. AAAI Press, 2007.

[Maedche et al. (2003)]. Alexander Maedche, Günter Neumann, et Steffen Staab. *Bootstrapping an ontology-based information extraction system.* Intelligent exploration of the web, pp. 345–359. Physica-Verlag GmbH, 2003. ISBN 3-7908-1529-2.

[Mangeot (2002)]. Mathieu Mangeot. *An XML markup language framework for lexical databases environments : the common dictionary markup language (CDM).* Proc. LREC Workshop on International Standards of Terminology and Language Resources Management, pp. 37–44, Las Palmas, 28 May 2002.

[Mangeot et Chalvin (2006)]. Mathieu Mangeot et Antoine Chalvin. *Dictionary building with the Jibiki platform : the GDEF case.* Proc. LREC 2006, pp. 1666–1669, Genova, Italie, 21-26 May 2006.

[Mangeot et Nguyen (2009)]. Mathieu Mangeot et Hong-Thai Nguyen. *Building lexical resources : towards programmable contributive platforms.* Proc. International Conference on Computing and Telecommunication Technologies (IEEE-RIVF 09), pp. 84–92, Danang, Vietnam, July 2009.

[Mangeot et Thévenin (2004)]. Mathieu Mangeot et David Thévenin. *Online generic editing of heterogeneous dictionary entries in Papillon project.* Proc. COLING-2004, volume 2, pp. 1029–1035, Genève, Suisse, 2004.

[Mangeot et al. (2003)]. Mathieu Mangeot, Gilles Sérasset, et Mathieu Lafourcade. *Construction collaborative d'une base lexicale multilingue, le projet Papillon.* TAL, 44(2) pp. 151–176, 2003.

[Max Silberztein (2009)]. *Max Silberztein.* NooJ linguistic software. http ://www.nooj4nlp.net/pages/nooj.html, September 2009.

[Miller et al. (2009)]. F.P. Miller, A.F. Vandome, et J. McBrewster. *Browser Wars.* Alphascript Publishing, 2009. ISBN 9786130226015. 204p.

[Montiel-Ponsoda et al. (2008)]. Elena Montiel-Ponsoda, Guadalupe Aguado de Cea, Asunción Gómez-Pérez, et Wim Peters. *Modelling multilinguality in ontologies.* Proc. Coling-2008 companion volume : Posters, pp. 67–70, Manchester, UK, August 2008.

[Muraki (1989)]. K. Muraki. *PIVOT : Two-phase machine translation system.* Proc. Second Machine Translation Summit (MTS-1989), pp. 113–115, Tokyo, Japan, 1989. ISBN 9784274074455.

[Nasharuddin et Abdullah (2010)]. *Nurul Amelina Nasharuddin et Muhamad Taufik Abdullah.* Cross-lingual information retrieval : State-of-the-Art. http ://ejcsit.uniten.edu.my/index.php/eJCSIT/article/viewArticle/22, January 2010.

[Navigli (2009)]. Roberto Navigli. *Word sense disambiguation : a survey.* ACM Computing Surveys, 41(2) pp. 1–69, 2009.

[Navigli et al. (2007)]. Roberto Navigli, Kenneth C. Litkowski, et Orin Hargraves. *Semeval-2007 task 07 : Coarse-grained English all-words task*. Proc. Fourth International Workshop on Semantic Evaluations (SemEval-2007), pp. 30–35, Prague, 2007.

[Nguyen (2009)]. Hong Thai Nguyen. *Des systèmes de TA homogènes aux systèmes de TA hétérogènes*. Thèse de doctorat, Université de Grenoble, 2009. 373p.

[Nguyen et al. (2007)]. Hong-Thai Nguyen, Christian Boitet, et Gilles Sérasset. *PIVAX, an online contributive lexical data-base for heterogeneous MT systems using a lexical pivot*. Proc. Seventh Symposium on Natural Language Processing (SNLP-2007), Bangkok, Thailand, 2007.

[Niles et Pease (2003)]. Ian Niles et Adam Pease. *Linking lexicons and ontologies : Mapping Wordnet to the Suggested Upper Merged Ontology*. Proc. 2003 International Conference on Information and Knowledge Engineering, pp. 23—26, Las Vegas, 2003. doi : 10.1.1.10.4486.

[Nirenburg (1989)]. Sergei Nirenburg. *Knowledge-based machine translation*. Machine Translation, 4(1) pp. 5–24, 1989. ISSN 0922-6567, 1573-0573. doi : 10.1007/BF00367750.

[Nowak et Dunker (2010)]. Stefanie Nowak et Peter Dunker. *Overview of the CLEF 2009 large-scale visual concept detection and annotation task*. Proc. 10th international conference on Cross-language evaluation forum : multimedia experiments, CLEF'09, pp. 94–109, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15750-5, 978-3-642-15750-9.

[Nyberg et Mitamura (1992)]. Eric H. Nyberg et Teruko Mitamura. *The KANT system : fast, accurate, high-quality translation in practical domains*. Proc. 14th International Conference on Computational Linguistics (COLING '92), volume 4, pp. 1254–1258, Nantes, France, 1992.

[O'connor et al. (2005)]. Martin O'connor, Holger Knublauch, Samson Tu, et Mark Musen. *Writing rules for the semantic web using SWRL and Jess*. Proc. 8th International Protege Conference, Protege with Rules Workshop, 2005.

[Papadimitriou (1994)]. C.H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. ISBN 9780201530827. 523p.

[Pazienza et Stellato (2006)]. Maria Teresa Pazienza et O Stellato. *Exploiting linguistic resources for building linguistically motivated ontologies*. Proc. second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006), Gênes, 2006.

[Pease et al. (2002)]. Adam Pease, Ian Niles, et John Li. *The suggested upper merged ontology : A large ontology for the semantic web and its applications*. Proc. AAAI-2002 Workshop on Ontologies and the Semantic Web, 2002.

[Pedersen et al. (2005)]. T. Pedersen, S. Banerjee, et S. Patwardhan. *Maximizing Semantic Relatedness to Perform Word Sense Disambiguation*. Research Report UMSI 2005/25, University of Minnesota Supercomputing Institute, March 2005.

[Phan et Geta (1992)]. Huy Khánh Phan et Christian Boitet Geta. *Multilinguisation d'un éditeur de documents structurés : application à un dictionnaire trilingue*. Proc. 14th conference on Computational linguistics (COLING) - Volume 3, COLING '92, pp. 966–970, Stroudsburg, PA, USA, 1992. doi : http://dx.doi.org/10.3115/992383.992412.

[Pimienta et al. (2009)]. Daniel Pimienta, Daniel Prado, et Alvaro Blanco. *Douze années de mesure de la diversité linguistique sur l'Internet : bilan et perspectives.* Publications de l'UNESCO pour le Sommet mondial sur la société de l'information, 2009.

[Polguère (2002)]. Alain Polguère. *Notions de base en lexicologie.* OLST-Département de linguistique et de traduction, Université de Montréal, 2002. 218p.

[Popescu (2007)]. Adrian Popescu. *Image retrieval using a multilingual ontology.* Large Scale Semantic Access to Content (Text, Image, Video, and Sound), RIAO '07, pp. 461–474, 2007.

[Prévot et al. (2005)]. L. Prévot, S. Borgo, et A. Oltramari. *Interfacing ontologies and lexical resources.* Proc. workshop on Ontologies and Lexical resources (OntoLex2005), pp. 185–200, 2005.

[Roche (2007)]. Christophe Roche. *Dire n'est pas concevoir.* Proc. 18ème journées francophones Ingénierie/es Connaissances, pp. 157–168, Grenoble, 2007.

[Rouquet et Nguyen (2009)]. David Rouquet et Hong-Thai Nguyen. *Multilinguïsation d'une ontologie par des corespondances avec un lexique pivot.* Proc. TOTh-2009, pp. 143–160, Annecy, France, 2009.

[Rouquet et al.(2010a)]. David Rouquet, Achille Falaise, Didier Schwab, Christian Boitet, Valérie Bellynck, Hong-Thai Nguyen, Mathieu Mangeot, et Jean-Philippe Guilbaud. *Rapport final de synthèse du projet ANR OMNIA, passage à l'échelle et implémentation : Extraction de contenu sémantique dans des masses de données textuelles multilingues.* Technical report, Grenoble, France, 2010a.

[Rouquet et al.(2010b)]. David Rouquet, Achile Fallaise, Didier Schwab, Hervé Blanchon, Vallérie Bellynck, Christian Boitet, Emmanuel Dellandréa, Ningning Liu, Alexandre Saidi, Sandra Skaff, Luca Marchesotti, et Gabriela Csurka. *Classifiation multilingue et multimédia pour la recherche d'images dans le projet OMNIA.* Proc. RISE'2010 : second atelier Recherche d'Information SEmantique, pp. 52–70, Marseille, France, 2010b.

[Schwab (2005)]. Didier Schwab. *Approche hybride - lexicale et thématique - pour la modélisation, la détection et l'exploitation des fonctions lexicales en vue de l'analyse sémantique de texte.* Thèse de doctorat, Université Montpellier 2, 2005. 363p.

[Schwab et al. (2011)]. Didier Schwab, Jérôme Goulian, et Nathan Guillaume. *Désambiguïsation lexicale par propagation de mesures sémantiques locales par algorithmes à colonies de fourmis.* Proc. TALN'2011 : Traitement Automatique des Langues Naturelles, pp. 185–196, Montpellier, France, juin-juillet 2011 2011.

[Senellart et al. (2001)]. Jean Senellart, Péter Dienes, et Tamás Váradi. *New generation SYSTRAN translation system.* Proc. MT Summit IX, 2001.

[Sérasset (1994)]. Gilles Sérasset. *SUBLIM : un système universel de bases lexicales multilingues et NADIA : sa spécialisation aux bases lexicales interlingues par acceptions.* Thèse de doctorat, Université Joseph Fourier-Grenoble 1, 1994. 194p.

[Sérasset et Mangeot (2001)]. Gilles Sérasset et Mathieu Mangeot. *The Papillon lexical database project : Monolingual dictionaries and interlingual links.* Proc. Natural Language Processing Pacific Rim Symposium (NLPRS '01), pp. 119–125, 2001.

[Sérasset et al. (2006)]. Gilles Sérasset, Francis Brunet-Manquat, et Elena Chiocchetti. *Multilingual legal terminology on the Jibiki platform : The LexALP project.* Proc. 21st International Conference on Computational Linguistics, pp. 937–944, Sydney, Australia, 2006. doi : 10.3115/1220175.1220293.

[Shortliffe (1976)]. Edward Hance Shortliffe. *Computer-based medical consultations, MYCIN.* Elsevier, New York, 1976. ISBN 0444001794. 264p.

[Sirin et al. (2007)]. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, et Yarden Katz. *Pellet : A practical owl-dl reasoner.* Web Semantics, 5 pp. 51–53, June 2007. ISSN 1570-8268. doi : 10.1016/j.websem.2007.03.004.

[Steward (1978)]. Gilles Steward. *Spécialisation et compilation des ATN : REZO.* Proc. COLING-78, pp. 14–18, Bergen, Norvège, 1978.

[Stoilos et al. (2006)]. G. Stoilos, N. Simou, G. Stamou, et S. Kollias. *Uncertainty and the semantic web.* IEEE Intelligent Systems, 21(5) pp. 84–87, 2006.

[Tarski (1983)]. Alfred Tarski. *The concept of truth in formalized languages.* Logic, semantics, metamathematics : papers from 1923 to 1938, pp. 152–268. Hackett Publishing, 1983. ISBN 9780915144761.

[Uchida (1989)]. Hirochi Uchida. *ATLAS-II : A machine translation system using conceptual structure as an interlingua.* Proc. Second Machine Translation Summit, pp. 93–100, Tokyo, Japon, 1989. ISBN 9784274074455.

[Uchida et al. (1999)]. Hiroshi Uchida, Meiying Zhu, et Tarcisio Della Senta. *The UNL, A Gift for a Millennium.* UNU/IAS, 1999. ISBN 4-906686-06-0p.

[Vasilescu et al. (2004)]. Florentina Vasilescu, Philippe Langlais, et Guy Lapalme. *Evaluating variants of the Lesk approach for disambiguating words.* Proc. LREC 2004, the 4th International Conference On Language Resources And Evaluation, pp. 633–636, Lisbone, Portugal, May 2004.

[Vossen et al. (2008)]. P. Vossen, E. Agirre, N. Calzolari, et C. Fellbaum. *KYOTO : a system for mining, structuring and distributing knowledge across languages and cultures.* Proc. LREC 2008, Marrakech, Morocco, May 2008.

[Weller (2007)]. Katrin Weller. *Folksonomies and ontologies : Two new players in indexing and knowledge representation.* Proc. Applying Web 2.0. Innovation, Impact and Implementation : Online Information 2007, pp. 108–115, London, 2007.

[Woods (1970)]. W. A. Woods. *Transition network grammars for natural language analysis.* Commun. ACM, 13 pp. 591–606, 1970. ISSN 0001-0782. doi : http://doi.acm.org/10.1145/355598.362773.

[Woods (1997)]. William A. Woods. *Conceptual indexing : A better way to organize knowledge.* Technical Report TR-99-83, Mountain View, CA, USA, 1997.

[Yildiz et Miksch (2007)]. Burcu Yildiz et Silvia Miksch. *OntoX - a method for Ontology-Driven information extraction.* Computational Science and Its Applications - ICCSA 2007, pp. 660–673. 2007.

[Zadeh (1965)]. L.A. Zadeh. *Fuzzy sets*. Information and Control, 8(3) pp. 338–353, June 1965. ISSN 0019-9958. doi : 10.1016/S0019-9958(65)90241-X.

[Zimmermann et al. (2006)]. Antoine Zimmermann, J Euzenat, et Isabel Cruz. *Three semantics for distributed systems and their relations with alignment composition*. The semantic Web - ISWC 2006, volume 4273, pp. 16–29. Springer, Berlin, 2006.

Annexe A

Exemples de couples image-texte traités dans le projet OMNIA

1012—Australian Open champion Mary Pierce of France volleys the ball during the second-round match against Kyoko Nagatsuka of Japan in the Toray Pan Pacific Open women's tennis tournament in Tokyo 02 Feburary. Pierce defeated Nagatsuka 6-4, 6-0.

1050730—AWA05 - 20020924 - BAGHDAD, IRAQ : Iraqi women sit under a portrait of Iraqi President Saddam Hussein in a waiting room in Baghdad's al-Mansur hospital 24 September 2002. Saddam Hussein is doggedly pursuing the development of weapons of mass destruction and will do his best to hide them from UN inspectors, the British government claimed in a 55-page dossier made public just hours before a special House of Commons debate on Iraq. Iraqi Culture Minister Hamad Yussef Hammadi called the British allegations "baseless." EPA PHOTO AFPI AWAD AWAD

FIGURE 1 – Images et textes de la base CLEF09-Belga

English : A full moon and 25 second exposure allowed sufficient light into this photo taken at Amundsen-Scott South Pole Station during the long Antarctic night. The new station can be seen at far left, power plant in the center and the old mechanic's garage in the lower right. Red lights are used outside during the winter darkness as their spectrum does not pollute the sky, allowing scientists to conduct astrophysical studies without artificial light interference. There is a background of green light. This is the Aurora Australis, which dances thorugh the sky virtually all the time during the long Antarctic night (winter).The photo's surreal appearance makes the station look like a futuristic Mars Station.

Français : Vue de la station américaine Amundsen-Scott au pôle sud, éclairée par la pleine lune, durant la longue nuit polaire. La nouvelle station est visible tout á gauche, la centrale électrique au centre et l'ancien atelier de mécanique en bas á droite. Durant la nuit polaire, seule la lumiere rouge est utilisée pour limiter la pollution lumineuse et ne pas perturber les expériences d'astrophysique menées dans la base. La lumiere verte est due aux aurores australes qui sont quasi-permanentes durant la nuit polaire. L'apparence surréaliste de la photo fait penser Ã une base futuriste sur Mars.

FIGURE 2 – Image et descriptions de la base Wikimedia Commons

Annexe B

L'ontologie de la campagne d'évaluation Image CLEF09

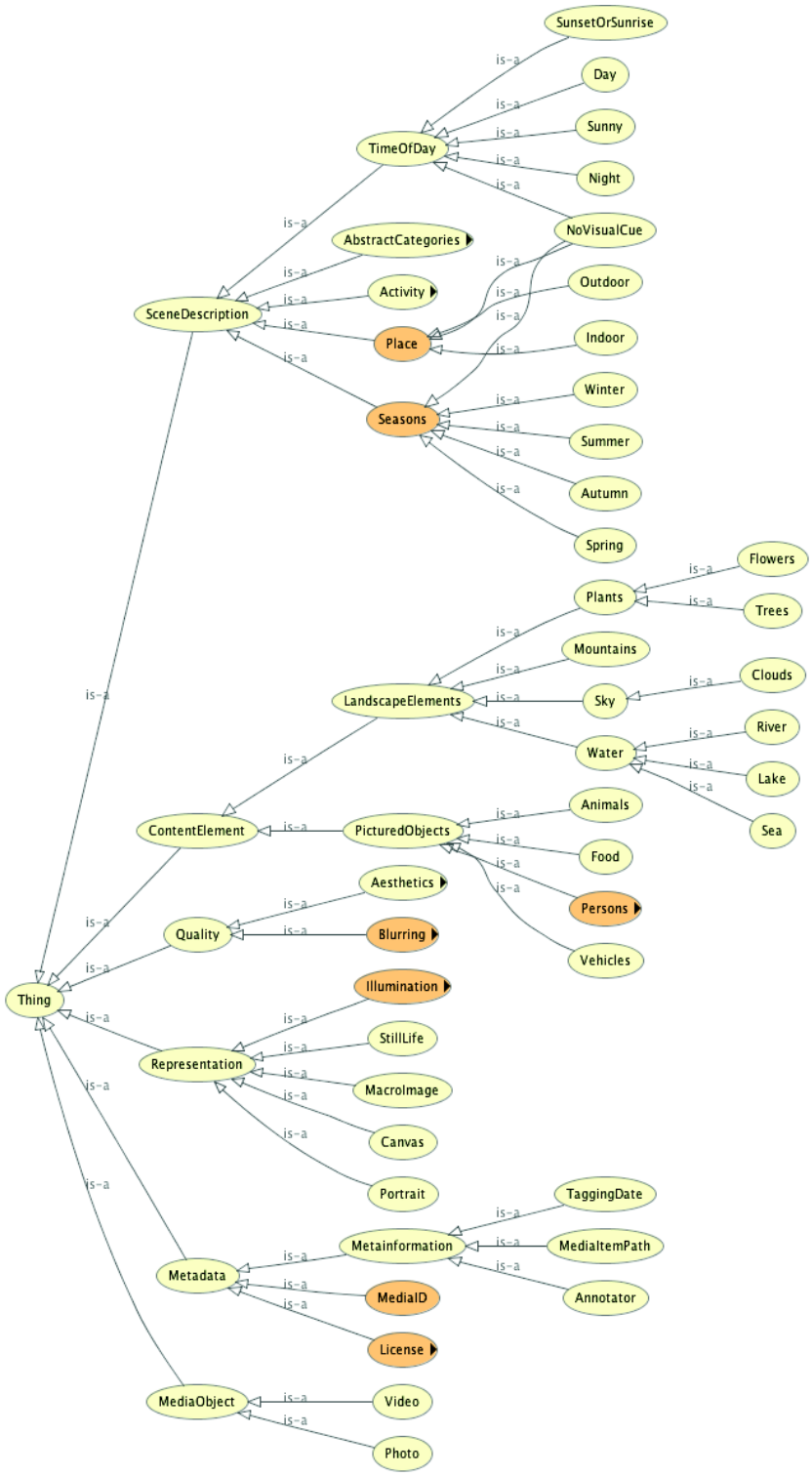


FIGURE 1 – Extrait du treillis conceptuel de l'ontologie image CLEF09

Annexe C

Treillis de concepts de l'ontologie du projet OMNIA

La relation IS-A forme un treillis sur les concepts, et non un arbre. Le nom d'un concept peut donc apparaitre plusieurs fois dans la liste s'il a plus d'un parent pour la relation IS-A (représenté avec des tabulations dans la liste).

```
owl#Thing
AESTHETICS
        COLOR
                BLACK_AND_WHITE
        COMPOSITION
                CLARITY
                COMPOSITION_DEPTH
                FORCES
                        BALANCE
                        DIRECTION
                        INTENSITY
                        MAGNETISM
                        RYTHM
                        SIMMETRY
                DEPTH
                FORM
                        CONVEXITY
                        FORM_TYPE
                        TEXTURENESS
        LIGHT
                DYNAMIC_RANGE
                EXPOSURE
                LIGHT_TYPE
                        ARTIFICIAL_LIGHT
                        DAY_LIGHT
                        NIGHT
                        SUNRISE
                        SUNSET
        NOISE
CONTENT
        ANIMAL
                AARDVARK
                AFRICAN_BUFFALO
                ALLIGATOR
                ANT
                ANTEATER
                ANTELOPE
                APE
                ARMADILLO
                BABOON
                BADGER
                BAT
                BEAR
                BEAVER
                BEE
                BIRD
                BISON
                BOAR
                BUFFALO
                BUSH_BABY
                BUTTERFLY
                CAMEL
                CARIBOU
                CAT
                CATTLE
                CHAMOIS
                CHEETAH
                CHICKEN
                CHIMPANZEE
                COBRA
                COCKROACH
                CORMORANT
                COYOTE
                CRAB
                CRANE
                CROCODILE
                CROW
                DEER
                DOG
                DOGFISH
                DOLPHIN
                DONKEY
                DOVE
                DRAGONFLY
                DUCK
                DUGONG
                EAGLE
                ECHIDNA
                EEL
                ELAND
                ELEPHANT
                ELEPHANT_SEAL
                FALCON
                FARM_ANIMAL
                FERRET
                FINCH
                FISH
                FLY
                FOX
                FROG
                GAUR
                GAZELLE
                GERBIL
                GIANT_PANDA
                GIRAFFE
                GNU
                GOAT
                GOOSE
                GORILLA
                GUANACO
                GUINEA_FOWL
                GUINEA_PIG
                GULL
                HAMSTER
                HARE
                HAWK
                HEDGEHOG
                HERON
                HIPPOPOTAMUS
                HORNET
                HORSE
                HUMAN
```

```
        HYENA
        INSECT
        JACKAL
        JAGUAR
        JELLYFISH
        KANGAROO
        KOMODO_DRAGON
        KOUPREY
        KUDU
        LARK
        LEMUR
        LEOPARD
        LION
        LLAMA
        LOBSTER
        LORIS
        LOUSE
        LYREBIRD
        MAGPIE
        MALLARD
        MANATEE
        MEERKAT
        MINK
        MOLE
        MONKEY
        MOOSE
        MOSQUITO
        MOUSE
        MULE
        NIGHTINGALE
        OKAPI
        ORYX
        OSTRICH
        OTTER
        OWL
        OX
        OYSTER
        PANTHER
        PARTRIDGE
        PEAFOWL
        PELICAN
        PET
        PIG
        PIGEON
        PONY
        PORCUPINE
        QUELEA
        RABBIT
        RACCOON
        RAIL
        RAM
        RAT
        RAVEN
        RED_DEER
        RED_PANDA
        RHINOCEROS
        SALAMANDER
        SEA_LION
        SEAL
        SEASTAR
        SHARK
        SHEEP
        SHREW
        SILVERBACK
        SKUNK
        SNAIL
        SNAKE
        SPIDER
        SQUID
        SQUIRREL
        SWAN
        TAPIR
        TARSIER
        TIGER
        TOAD
        TURKEY
        TURTLE
        VICUNA
        WALRUS
        WAPITI
        WASP
        WATER_BUFFALO
        WEASEL
        WHALE
        WILD_ANIMAL
        WOLF
        WORM
        YAK
        ZEBRA
ART
        CRAFTS
        PAINTING
        SCULPURE
BUILDING
        AGRICULTURAL_BUILDING
                BARN
                CHICKENHOUSE
                FARMHOUSE
                GRAINERY
                GREENHOUSE
                HAYLOFT
                HORSE_MILL
                PIGPEN
                ROOT_CELLAR
                SHED
                SILO
                STABLE
                STORM_CELLAR
                TIDE_MILL
                WATERMILL
                WELL_HOUSE
                WIND_MILL
        COMMERCIAL_BUILDING
                BANK
                BAR
                        PUB
                BROTHEL
                CASINO
                COFFEE_HOUSE
                CONVENTION_CENTER
                FORUM
                GAS_STATION
                GROCERY_STORE
                HOTEL
                        MOTEL
                MARKET
                NIGHTCLUB
                        JAZZ_CLUB
                OFFICE_BUILDING
                RESTAURANT
                RETAIL_STORE
                SHOP
                SHOPPING_MALL
                SKYSCRAPER
                STOCK_EXCHANGE
                SUPERMARKET
                WAREHOUSE
        EDUCATIONAL_BUILDING
                CLASSROOM_BUILDING
                COLLEGE
                DORMITORY
                GYMNASIUM
                LIBRARY
                MUSEUM
                        ART_GALLERY
                SCHOOL
                STUDENTS_UNION
                THEATER
                        AMPHITHEATER
                        CINEMA
                        CONCERT_HALL
                        OPERA_HOUSE
                        SYMPHONY
                UNIVERSITY
        GOVERNMENT_BUILDING
                CAPITOL
                CITY_HALL
                CONSULATE
                COURTHOUSE
                EMBASSY
                FIRE_STATION
                PALACE
                PARLIAMENT
                POLICE_STATION
                POST_OFFICE
                PRISON
        INDUSTRIAL_BUILDING
                BREWERY
                FACTORY
                FOUNDRY
                MILL
                MINING
                OIL_RIG
                POWER_PLANT
                REFINERY
        MILITARY_BUILDING
                BARRACKS
                BUNKER
                        BLOCKHOUSE
                CASTLE
                CITADEL
                CITY_GATE
                DEFENSIVE_WALL
                FORT
```

```
        FORTIFICATION
        TOWER
OTHER_BUILDING
        AQUEDUCT
        BATHHOUSE
        BRIDGE
        FOLLY
        HOSPITAL
        LOW-ENERGY_BUILDINGS
        MARINA
        ROAD
        STADIUM
                ARENA
        TRIUMPHAL_ARCH
RELIGIOUS_BUILDING
        CHURCH
                BASILICA
                CATHEDRAL
                        DUOMO
                CHAPEL
                MARTYRIUM
                ORATORY
        FIRE_TEMPLE
        GURDWARA
        IMAMBARGAH
        MITHRAEUM
        MONASTERY
        MOSQUE
                MIHRAB
        PAGODA
        PYRAMID
        SHRINE
        SYNAGOGUE
        TEMPLE
RESIDENTIAL_BUILDING
        APARTMENT_BLOCK
        ASYLUM
        CONDOMINIUM
        DORMITORY
        DUPLEX
        HOUSE
                APARTMENT
                AUL
                BARNDOMINIUM
                BARRACKS
                BASEMENT
                BEDSIT
                BROWNSTONE
                BUNGALOW
                CASTLE
                CHATTEL_HOUSE
                CHOULTRY
                COLONIAL_HOUSE
                CONDOMINIUM
                COTTAGE
                CRACKER_HOUSE
                DUPLEX
                FLAT
                GARALOW
                GEODESIC_DOME
                IGLOO
                IZBA
                KONAK
                KSAR
                LOG_CABIN
                LUSTRON_HOUSE
                MAISONETTE
                MANOR
                MANSION
                MESS
                MUDHIF
                PATIO
                PENTHOUSE
                PLATTENBAU
                PREFABRICATED
                QUEENSLANDER
                RANCH
                ROWHOUSE
                SHACK
                SHOPHOUSE
                SHOTGUN_HOUSE
                STUDIO
                TIPI
                TOWNHOUSE
                TREE_HOUSE
                TRULLO
                VILLA
                YAODONG
        NURSING_HOME
STORAGE_BUILDING
        BARN
        BOATHOUSE
                CARPORT
                GARAGE
                HANGAR
                SHED
                STORAGE_SILO
                WAREHOUSE
        TRANSIT_STATION
                AIRPORT
                BUS_STATION
                FERRY_SLIP
                METRO_STATION
                TERMINAL
                TRAIN_STATION
FOOD
FURNITURE
MUSICAL_INSTRUMENT
NATURAL
        CLOUDS
        FLOWERS
        GRASS
        ROCKS
        SAND
        SKY
        SNOW
        TREE
        WATER
        PEOPLE
        CROWD
        GROUP
PERSON
        ADMIRAL
        ANARCHIST
        ANIMIST
        BUDDHIST
        CAMMANDANT
        CAPITALIST
        CAPTAIN
        CHANCELLOR
        CHRISTIAN
                CATHOLIC
                ANGLICAN
                ORTHODOX
                PROTESTANT
                BAPTIST
                LUTHERAN
                METHODIST
                PRESBYTERIAN
        COLONEL
                MARSHAL
        COMMANDER
        COMMODORE
        COMMUNIST
        CORPORAL
        DEMOCRAT
        DICTATOR
        FASCIST
        FEMINIST
        GENERAL
        HINDUIST
        ISLAMIST
        JEWISH
        KING
        LIBERTARIAN
        LIEUTENANT
        LOBBY
                LOBBYIST
        LOBBYIST
        MAJOR
        MAN
                KING
        MARXIST
        METROSEXUAL
        MILITANT
        MINISTER
                PRIME-MINISTER
        MONOTHEIST
        NAZI
        OFFICER
        PARTISAN
        PIRATE
        PRESIDENT
        QUEEN
        RACIST
        RASTAFARI
        REPUBLICAN
        REVISIONIST
        SECRETARY
        SERGEANT
        SOCIALIST
        SYNDICALIST
        TERRORIST
        VEGAN
```

```
                WOMAN
                        QUEEN
                ZIONIST
                POLYTHEIST
        TOYS
        VEHICLE
                AIRCRAFT
                AUTOMOBILE
                BICYCLE
                BIKE
                BOAT
                BUS
                CAR
                HOVERCRAFT
                MOTORCYCLE
                ROCKET
                SHIP
                SIDECAR
                SPACECRAFT
                SUBMARINE
                SUBMERSIBLE
                TRACTOR
                TRAIN
                TRAM
                TRICYCLE
                TRUCK
                VAN
CONTEXT
        BACKGROUND
                INDOOR
                OUTDOOR
                        AERIAL
                        ARCHEOLOGY
                        BEACH
                        CAVE
                        COAST
                        COUNTRYSIDE
                        DESERT
                        MOUNTAIN
                        NATURAL_PARKS
                        RIVER
                        UNDERWATER
                        URBAN
                        WATERFALL
        EMOTION
                NEGATIVE
                        NEGATIVE_ACTIVE
                                ANGER
                                CONSTERNATION
                                DISAPPROVAL
                                DISGUST
                                DISTURBANCE
                                FEAR
                                HATE
                                INDIGNATION
                                IRRITATION
                                OBSESSION
                        NEGATIVE_NEUTRAL_AROUSAL
                                BITTERNESS
                                DISDAIN
                                INSENSIBILITY
                                OFFENCE
                        NEGATIVE_PASSIVE
                                BOREDOM
                                DEPRESSION
                                DISAPPOINTMENT
                                EMBARRASSMENT
                                INHIBITION
                                PITY
                                SADNESS
                NEUTRAL_APPRAISAL
                        NEUTRAL_APPRAISAL_ACTIVE
                                DESIRE
                                INTEREST
                                SURPRISE
                        NEUTRAL_APPRAISAL_NEUTRAL_AROUSAL
                                NEUTRAL
                        NEUTRAL_APPRAISAL_PASSIVE
                                INDIFFERENCE
                POSITIVE
                        POSITIVE_ACTIVE
                                FASCINATION
                                FUN
                                JOY
                                JUBILATION
                                LOVE
                                PRIDE
                                STIMULATION
                                WONDER
                        POSITIVE_NEUTRAL_AROUSAL
                                PASSION
                                POSITIVE_PASSIVE
                                APPEASEMENT
                                RELIEF
                                SATISFACTION
THEMATIC
        ARTISTIC
        ENTERTAINMENT
                AMUSEMENT_PARK
                CIRCUS
                COMEDY
                CONCERTS
                DISCOTHEQUE
                FASHION
                        READY-TO-WEAR
                FILM
                FUNFAIRS
                GAMBLING
                GAME
                        BACKGAMMON
                        BATTLESHIP
                        CHECKERS
                        CHESS
                        CHINESE_CHECKERS
                        CONNECT_FOUR
                        DIPLOMACY
                        DRAUGHTS
                        GO
                        GO-MOKU
                        JACQUET
                        MAHJONG
                        MANCALA
                        OTHELLO
                        REVERSI
                        SOGO
                        STRATEGO
                MUSIC
                        LIVE
                RADIO
                SHOW_BUSINESS
                SPORT
                        ARCHERY
                        ATHLETICS
                        BADMINTON
                        BASKETBALL
                        BIATHLON
                        BOBSLEIGH
                        BOXING
                        CANOEING
                        CLIMBING
                        CURLING
                        CYCLING
                        DECATHLON
                        DIVING
                        DUATHLON
                        EQUITATION
                        FENCING
                        FOOTBALL
                        GOLF
                        GYMNASTIC
                        HANDBALL
                        HEPTATHLON
                        HOCKEY
                        ICE_HOCKEY
                        JUDO
                        LUGE
                        MOTOCROSS
                        NORDIC_COMBINED
                        PENTATHLON
                        RACING
                        ROWING
                        RUGBY
                        SAILING
                        SHOOTING
                        SKATEBOARDING
                        SKATING
                        SKELETON
                        SKIING
                        SNOWBOARDING
                        SOCCER
                        SWIMMING
                        TABLE_TENNIS
                        TAEKWONDO
                        TENNIS
                        TRIATHLON
                        VOLLEYBALL
                        WEIGHTLIFTING
                        WRESTLING
                TELEVISION
                THEME_PARKS
                TRADE_SHOW
                VIDEO_GAMES
        FAMILY
        POLITICS
```

```
ANARCHISM
ANARCHIST
AUTOCRACY
BUDGET
CABINET
CAMPAIGN
CAPITALISM
        CAPITALIST
CHANCELLOR
CIA
COLLECTIVISM
COMMUNISM
        COMMUNIST
CONGRESS
CONSERVATIVE
CONSPIRACY
CONSTITUTION
DEMOCRACY
DEMOCRAT
DICTATOR
ENVIRONMENTALISM
        ENVIRONMENTALIST
FASCISM
        FASCIST
FBI
FEMINISM
FEMINIST
HOMOPHOBIA
INFLATION
JUSTICE
KING
LIBERAL
        LIBERTARIAN
LOBBY
        LOBBYIST
MARXISM
        MARXIST
METROSEXUAL
MILITANT
MINISTER
        PRIME-MINISTER
MONARCHY
NATIONALISM
NAZI
NEW_WORLD_ORDER
PARLIAMENT
PARTISAN
PARTY
PHILOSOPHY
PIRACY
PIRATE
PLURALISM
POSITIVISM
PRESIDENT
PROPAGANDA
PROPERTY
QUEEN
QUOTA
RACISM
RACIST
RECESSION
REPUBLIC
REPUBLICAN
REVISIONIST
SECRETARY
SOCIALISM
SOCIALIST
SUICIDE_BOMBER
SYNDICALISM
SYNDICALIST
TERRORISM
TERRORIST
TOTALITARIANISM
TYRANNY
UNITED_NATIONS
UTILITARIANISM
VEGAN
WAR
        ADMIRAL
        CAMMANDANT
        CAPTAIN
        CIVIL_WAR
        COLONEL
        MARSHAL
        COMMANDER
        COMMODORE
        CORPORAL
        GENERAL
        LIEUTENANT
        MAJOR
        MILITARY
        MILITIA
        OFFICER
        SERGEANT
WATERGATE
WEAPONS
        WEAPONS_OF_MASS_DESTRUCTION
ZIONISM
        ZIONIST
RELIGION
        MONOTHEISM
        CHRISTIANISM
                CHRISTIAN
                CATHOLIC
                ANGLICAN
                ORTHODOX
        PROTESTANT
                BAPTIST
                LUTHERAN
                METHODIST
                PRESBYTERIAN
        GNOSTICISM
                GNOSTIC
        ISLAM
                ISLAMIST
        JUDAISM
                JEWISH
        MONOTHEIST
        RASTAFARISM
                RASTAFARI
        POLYTHEISM
                ANIMISM
                        ANIMIST
                BUDDHISM
                        BUDDHIST
                HINDUISM
                HINDUIST
                POLYTHEIST
                VODOU
TRADITIONAL
METADATA
        DATE
        FORMAT
        LOCATION
        CITY
        NATIONAL_CAPITAL
        CONTINENT
        COUNTRY
        STATE
        MATERIAL
        PHOTOGRAPHER
        RESOLUTION
```

Annexe D

Statistiques sur EuroWordnet

Langue	**Synsets**	**Sens de mots**	**Relations monolingues**	**Relations translingues**	**Fourchettes de prix en Euros**
Néerlandais	44015	70201	111639	53448	11003-440
Espagnol	23370	50526	55163	21236	5842-233
Italien	40428	48499	117068	71789	10107-404
Allemand	15132	20453	34818	16347	3783-151
Français	22745	32809	49494	22730	5686-227
Tchèque	12824	19949	26259	12824	3206-128
Estonien	7678	13839	16318	9004	1919-76
Anglais (entention)	16361	40588	42140	0	4090-163
Anglais (WordNet1.5)	94515	187602	211375	0	0

Source : `http://www.illc.uva.nl/EuroWordNet/finalresults-ewn.html`

Annexe E

Extraits des spécifications d'UNL

Les extraits de cette annnexe sont issus des spécifications UNL du 7 juin 2005.

Ces documents étaient accessibles le 14 août 2011 sur le Web :
`http://www.undl.org/unlsys/unl/unl2005/`.

1 Les relations d'UNL

agt	agent	
	indicates a thing in focus that initiates an action	
Ex.	agt(break(agt>thing,obj>thing), John(iof>person))	John breaks ...
	agt(run(icl>act(agt>volitional thing)), car(icl>vehicle))	car runs ...
	agt(destroy(agt>thing,obj>thing), explosion(icl>event))	explosion destroys ...
and	conjunction	
	indicates a partner to have conjunctive relation to	
Ex.	and(quickly, easily)	... easily and quickly
	and(dance(agt>person), sing(agt>person))	... singing and dancing
	and(Mary(iof>person), John(iof>person))	... John and Mary
aoj	thing with attribute	
	indicates a thing that is in s state or has an attribute	
Ex.	aoj(red(aoj>thing), leaf(pof>plant))	... leaf is red.
	aoj(available(aoj>thing,obj<thing), information)	This information is available for ...
	aoj(nice, ski(agt>person))	Skiiing is nice.
bas	basis	
	indicates a thing used as the basis (standard) of comparison	
Ex.	bas(more(aoj>thing,bas>thing), 7)	Ten is three more than seven.
	bas(more(icl>how,bas>thing), Jack(iof>person))	Betty weighs more than Jack (does).
	man(beautiful, more(icl>how,bas>thing))	A tulip is more beautiful than a rose
ben	beneficiary	
	indicates an indirectly related beneficiary or victim of an event or state	
Ex.	ben(give(agt>thing,gol>thing,obj>thing), country(icl>region))	To give one's life for one's country.
	ben(good(aoj>thing), John(iof>person))	It is good for John to ...
cag	co-agent	
	indicates a thing not in focus that initiates an implicit event that is done in parallel	
Ex.	cag(walk(agt>volitional thing), John(iof>person))	To walk with John
	cag(live(agt>volitional thing), aunt(icl>person))	To live with ... aunt
cao	co-thing with attribute	
	indicates a thing not in focus that is in a parallel state	
	Examples and readings	
	cao(exist(aoj>thing), you)	be with you
cnt	content	
	indicates the content of a concept	
Ex.	cnt(Internet(icl>communication network),amalgamation(icl>harmony))	The Internet : an amalgamation
	cnt(language generator, deconverter.@double_quote)	a language generator "deconverter"...

	cnt(risk(icl>danger), :01)	the risk of losing money
cob	affected co-thing	
	indicates a thing that is directly affected by an implicit event done in parallel or an implicit state in parallel	
Ex.	cob(die(obj>living thing), Mary(iof>person))	... dead with Mary
con	condition	
	indicates a non-focused event or state that conditions a focused event or state	
Ex.	aoj :01(tired(aoj>thing), you)	If you are tired, we will go straight home
	con(go(icl>move(agt>thing,gol>place,src>place)), :01)	
coo	effected co-thing	
	indicates a co-occurrent event or state for a focused event or state	
Ex.	coo(cry(icl>weep(agt>volitional thing)), run(icl>act(agt>volitional thing)))	... was crying while running
	coo(red(aoj>thing), hot(aoj>thing))	... is red while ... is hot
dur	duration	
	indicates a period of time during which an event occurs or a state exists	
Ex.	dur(work(agt>person), hour(icl>period))	... work nine hours (a day)
	qua(hour(icl>period), 9)	
	dur(talk(icl>express(agt>thing,gol>person,obj>thing), meeting(icl>event))	... talk ... during meeting
	dur(come(icl>move(agt>thing,gol>place,src>place), absence(icl>state))	... come during (my) absence
equ	effected co-thing	
	indicates an equivalent concept	
Ex.	equ(deconverter, language generator.@parenthesis)	the deconverter (a language generator)
fmt	range/from-to	
	indicates a range between two things	
Ex.	fmt(z(icl>letter), a(icl>letter))	the alphabets from a to z
	fmt(New York(iof>city), Osaka(iof>city))	the distance from Osaka to New York
	fmt(Friday(icl>day), Monday(icl>day))	the weekdays from Monday to Friday
frm	origin	
	indicates an initial state of a thing or a thing initially associated with the focused thing	
Ex.	frm(visitor(icl>person), Japan(iof>country)	a visitor from Japan
gol	goal/final state	
	indicates a final state of object or a thing finally associated with the object of an event	
Ex.	gol(change(gol>thing,obj>thing,src>thing), red(aoj>thing))	the lights changed from green to red
	gol(deposit(agt>thing,gol>thing,obj>thing), account(icl>record))	millions were deposited in a bank account
icl	included/a kind of	
	indicates an upper concept or a more general concept	
Ex.	icl(bird(icl>animal), animal(icl>living thing))	a bird is a (kind of) animal
ins	instrument	
	indicates an instrument to carry out an event	
Ex.	ins(look(agt>thing,obj>thing), telescope(icl>optical instrument))	look at stars through a telescope
	ins(write(agt>thing,obj>thing), pencil(icl>stationery))	write [draw] with a pencil
	ins(cut(agt>thing,obj>thing,opl>thing), scissors(icl>cutley))	He cut the string with a pair of scissors
int	intersection	
	indicates all common instances with a partner concept	
Ex.	int(tableware(icl>tool), cookware(icl>tool)))	an intersection of tableware and cookware
iof	an instance of	
	indicates a class concept that an instance belongs to	
Ex.	iof(Tokyo(iof>city), city in Japan)	Tokyo is a city in Japan
man	manner	
	indicates a way to carry out an event or the characteristics of a state	
Ex.	man(move(agt<thing,gol>place,src>place), quickly)	move quickly
	man(visit(agt>thing,obj>thing), often)	I often visit him.
	man(beautiful, very(icl>how))	it is very beautiful.
met	method/means	
	indicates a means to carry out an event	
Ex.	met(solve(icl>resolve(agt>thing,obj>thing)), dynamics(icl>science))	... solve ... with dynamics
	met(solve(icl> resolve(agt>thing,obj>thing)), algorithm(icl>method))	... solve ... using ... algorithm
mod	modification	
	indicates a thing that restricts a focused thing	
Ex.	mod(story(icl>tale), whole(mod<thing))	the whole story
	mod(plan(icl>idea), master(mod<thing))	a master plan
	mod(part(pof>thing), main(mod<thing))	the main part
nam	name	
	indicates a name of a thing	
	Examples and readings	

	nam(son(icl>relative), Hikari)	his son "Hikari"
obj	affected thing	
	indicates a thing in focus that is directly affected by an event or state	
Ex.	obj(move(gol>place,obj>thing,src>place), table(icl>furniture))	the table moved.
	obj(melt(gol>thing,obj>thing), sugar(icl>seasoning))	the sugar melts into ...
	obj(cure(agt>thing,obj>thing), patient(icl>person))	to cure the patient.
opl	affected place	
	indicates a place in focus affected by an event	
Ex.	opl(pat(icl>touch(agt>thing,obj>thing,opl>thing)), shoulder(pof>trunk))	... pat ... on shoulder
	opl(cut(agt>thing,obj>thing,opl>thing), middle(icl>place))	... cut ... in middle
or	disjunction	
	indicates a partner to have disjunctive relation to	
Ex.	or(leave(agt>thing,obj>place), stay(icl>remain(agt>thing)))	Will you stay or leave ?
	or(blue(icl>color), red(icl>color))	Is it red or blue ?
	or(Jack(iof>person), John(iof>person))	Who is going to do it, John or Jack ?
per	proportion/rate/distribution	
	indicates a basis or unit of proportion, rate or distribution	
	Examples and readings	
	per(hour(icl>period), day(icl>period))	eitgh hours a day
	per(time(icl>frequency), week(icl>period))	... twice a week
plc	place	
	indicates a place where an event occurs, or a state that is true, or a thing that exists	
Ex.	plc(cook(agt>thing), kitchen(pof>building))	... cook ... in the kitchen
	plc(sit(agt>thing), beside(icl>place))	... sit beside me
	plc(cool(icl>cold), here(icl>place))	It's cool here.
plf	initial place	
	indicates a place where an event begins or a state that becomes true	
Ex.	plf(travel(agt>volutional thing), Tokyo(icl>city))	traveling from Tokyo
	plf(deep(aoj>thing), there(icl>place))	The sea is deep from there to here
plt	final place	
	indicates a place where an event ends or a state that becomes false	
	Examples and readings	
	plt(travel(agt>volitional thing), Boston(iof>city))	to travel to Boston
	plt(deep(aoj>thing), here(icl>place))	The sea is deep from there to here
pof	part of	
	indicate a concept of which a focused thing is a part	
Ex.	pof(preamble(icl>information), document(icl>information))	the preamble of a document
	pof(initial(icl>letter), machine translation)	the initials of Machine Translation
pos	possessor	
	indicates the possessor of a thing	
	Examples and readings	
	pos(dog(icl>aminal), John(iof>person))	John's dog
	pos(book(icl>document), I)	my book
ptn	partner	
	indicates an indispensable non-focused initiator of an action	
Ex.	ptn(compete(agt>thing,ptn>thing), John(iof>person))	... compete with John
	ptn(share(icl>divide(agt>thing,obj>thing)), poor(icl>person))	... share ... with the poor
	ptn(collaborate(agt>thing,ptn>person), he)	... collaborate with him ...
pur	purpose	
	indicates the purpose or objective of an agent of an event or the purpose of a thing that exists	
Ex.	pur(come(icl>move(agt>thing,gol>place,src>place)),	... come to see you
	pur(work(agt>person), money(icl>mark))	... work for money
	pur(budget(icl>expense), research(icl>study))	our budget for research
qua	quantity	
	indicates the quantity of a thing or unit	
Ex.	qua(cup(icl>tabelware), 2))	Two cups of coffee
	qua(kilogram(icl>unit), many(qua<thing))	many kilograms
	qua(dog(icl>animal), 2)	two dogs
rsn	reason	
	indicates a reason why an event or a state happens	
Ex.	rsn(go(icl>move(agt>thing,gol>place,src>place)).@not, rain(icl>weather))	... didn't go because of the rain
	rsn(known(aoj>thing), beauty(icl>abstract thing))	a city known for its beauty
scn	scene	
	indicates a scene where an event occurs, or state is true, or a thing exists	

Ex.	scn(win(agt>thing), contest(icl>event))	... win a prize in a contest
	scn(appear(gol>thing,obj>thing), program(icl>plan))	... appear on a TV program
	scn(play(agt>thing,obj>thing), movie(icl>cinema))	... play in movie
seq	sequence	
	indicates a prior event or state of a focused event or state	
Ex.	seq(leap(icl>jump(agt>thing)), look(agt>thing,obj>thing))	Look before you leap.
	seq(red(aoj>thing), green(aoj>thing))	It was green and then red.
	seq(take off(agt>thing,obj>thing), come in(agt>thing))	She came in and took her coat off
shd	sentence head	
	indicates a number, a mark or a thing that shows the position of a sentence, a paragraph or a chapter in a document	
	Examples and readings	
	shd(relation(icl>sate), chapter(pof>book))	Chapter 2 Relation
	mod(chapter(pof>book), 2)	
src	source/initial state	
	indicates the initial state of an object or thing initially associated with the object of an event	
	Examples and readings	
	src(change(obj>thing), red(aoj>thing))	The lights changed from green to red.
	src(withdraw(agt>thing,obj>thing), stove(icl>furniture))	I quickly withdrew my hand from the stove
tim	time	
	indicates the time an event occurs or a state is true	
Ex.	tim(leave(agt>thing,obj>place), Tuesday(icl>day))	... leave on Tuesday
	tim(do(agt>thing,obj>thing), o'clock(icl>time))	... do ... at ... o'clock
tmf	initial time	
	indicates the time an event starts or a state becomes true	
Ex.	tmf(work(agt>person), morning(icl>time))	... work from morning to [till] night
	tmf(change(obj>thing), live(agt>volitional thing))	... has changed ... since I have lived here.
tmt	final time	
	indicates a time an event ends or a state becomes false	
Ex.	tmt(work(agt>person), night(icl>time))	... work from morning to [till] night
	tmt(full(aoj>thing), tomorrow(icl>time))	... be full till tomorrow
to	destination	
	indicates a final state of a thing or a final thing (destination) associated with the focused thing	
Ex.	to(train(icl>vehicle), London(iof>city))	a train for London
	to(letter(icl>document), you)	a letter to you
via	an intermediate place or state	
	indicates an intermediate place or state of an event	
Ex.	via(go(icl>move(agt>thing,gol>place,src>place)), New York(iof>city))	... go ... via New York
	via(bike(agt>thing), Alps(iof>mountain))	... bike ... through the Alps
	via(drive(agt>thing), tunnel(icl> facilities))	... drive ... by way of the tunnel

2 Les attributs d'UNL

Attributes describing logicality of UW		
@transitive	attached to an UW that has transitivity	@transitive' can be attached to UW 'ancestor(icl>kindred)'. Because if "A is ancestor of B" and "B is ancestor of C" are true, "A is ancestor of C" will be true.
@symmetric	attached to an UW that has symmetricity	'@symmetric' can be attached to UW 'partner(icl>....)'. Because if "A is a partner of B" is true, "B is a partner of A" will be true also.
@identifiable	attached to an UW that can identify the subject	'@identifiable' can be attached to (compound) UW 'national health insurance id' as the content of the UW can identify the person who hold the ID.
@disjointed	attached to an UW or a group of UWs to show that all element concept do not hold common instance. All connected UWs do not share instances.	'@disjointed' can be attached to a scope consists of "men and women" as these two element concepts do not share common instances.
Attributes describing times with respect to the speaker		
@past	happened in the past	ex) It was snowing yesterday
@present	happening at present	ex) It is raining hard.
@future	will happen in future	ex) He will arrive tomorrow
Attributes describing speaker's view on aspects of event		
@begin	beginning of an event or a state	ex) It began to work again

@complete	finishing/completion of a (whole) event	ex) I've looked through the script. look.@entry.@complete
@continue	continuation of an event	ex) He went on talking. talk.@continue.@past
@custom	customary or repetitious action	ex) I used to visit [I would often go] there when I was a boy. visit.@custom.@past
@end	end/termination of an event or a state	ex) I have done it. do.@end.@present
@experience	experience	ex) Have you ever visited Japan ? visit.@experience.@interrogation ex) I have been there. visit.@exterience
@progress	an event is in progress	ex) I am working now. work.@progress.@present
@repeat	repetition of an event	ex) It is so windy that the tree branches are knocking against the roof. knock.@entry.@present.@repeat
@state	final state or the existence of the object on which an action has been taken	ex) It is broken. break.@state
Attributes modifiing speaker's view on aspects of event		
@just	Expresses an event or a state that has just begun or ended/completed	Ex) He has just come. come.@complete.@just
@soon	Expresses an event or a state that is about to begin or end/completed	Ex) The train is about to leave. leave.@begin.@soon
@yet	Expresses an event or a state that has not yet started or ended/completed, together with @not	Ex) I have not yet done it. do.@complete.@not.@yet
Attributes describing speaker's view of reference to concepts		
@generic	generic concept	ex) The dog is a faithful animal.
@def	already referred	ex) the book you lost
@indef	non-specific class	ex) There is a book on the desk.
@not	complement set	ex) Don't be late !
@ordinal	ordinal number	ex) the 2nd door
Attributes describing speaker's view of emphasis, focus and topic		
@contrast	Contrasted UW	For instance, "but" in the examples below is used to introduce a word or phrase that contrasts with what was said before. ex) It wasn't the red one but the blue one. ex) He's poor but happy
@emphasis	Emphasized UW	ex) I do like it.
@entry	Entry or main UW of a sentence or a scope	ex) He promised (entry of the sentence) that he would come (entry of the scope)
@qfocus	Focused UW of a question	ex) Are you painting the bathroom blue ? To this question, the answer will be "No, I'm painting the LIVING-ROOM blue"
@theme	Instantiates an object from a different class	
@title	Title	
@topic	Topic	ex) He(@topic) was killed by her. ex) The girl(@topic) was given a doll. ex) This doll(@topic) was given to the girl.
Attributes describing speaker's attitudes		
@affirmative	Affirmation	
@confirmation	Confirmation	ex) You won't say that, will you ? ex) It's red, isn't it ? ex) Then you won't come, right ?
@exclamation	Exclamation	ex) kirei na ! ("How beautiful (it is) !" in Japanese) ex) Oh, look out !
@humility	In a humility manner	ex) That is quite impossible for the likes of me.@humility.
@imperative	Imperative	ex) Get up ! ex) You will please leave the room.
@interrogative	Interrogation	ex) Who is it ?
@invitation	Inducement	ex) Will / Won't you have some tea ? ex) Let's go, shall we ?
@polite	polite wayÂ	ex) Could you (please)... ex) If you could ... I would ...
@request	Request	ex) Please don't forget...
@respect	Respectful way	ex) o taku ("(your) house" in Japanese) ex) Good morning, sir.
@vocative	Vocative	ex) Boys, be ambitious !
Attributes describing speaker's feelings and judgments		
@ability	Ability, capability of doing something	ex) The child can 't walk yet. ex) He can speak English but he can't write it very well.

@get-benefit	Speaker's feeling of receiving benefits through the fact or result of something (to be) done by somebody else	ex) I'll have my secretary type the letter. *In Japanese the auxiliary verb of " te morau" is used to express the getting benefits feeling. For instance it is frequently used in a sentence in the sense of "to have somebody do something" in Japanese.
@give-benefit	Speaker's feeling of giving benefits by doing something for somebody else	ex) Be kind to old people. *In Japanese the auxiliary verb of " te ageru" is used to express the giving benefits feeling. For instance it is frequently used in a sentence in the sense of "Be kind to old people" in Japanese.
@conclusion	Logical conclusion due to a certain condition	ex) He is her husband ; she is his wife.
@consequence	Logical consequence	ex) He was angry, wherefore I left him alone.
@sufficient	Sufficient condition	ex) only have to
@consent	Consent feeling of the speaker about something	
@dissent	Dissent feeling of the speaker about something	ex) But that's not true.
@grant	To give/get consent/permission to do something	ex) Can I smoke in here ? ex) You may borrow my car if you like.
@grant-not	Not to give consent to do something	ex) You {mustn't/are not allowed to/may not} borrow my car.
@although	Something follows against [contrary to] or beyond expectation	ex) Although he didn't speak, I felt a certain warmth in his manner.
@discontented	Discontented feeling of the speaker about something	ex) (I'll tip you 10 pence.) But that's not enough !
@expectation	Expectation of something	ex) Children ought to be able to read by the age of 7. ex) If you leave now, you should get there by five o'clock.
@wish	Wishful feeling, to wish something is true or has happened	ex) If only I could remember his name ! (I do wish I could remember his name !) ex) You might have just let me know.
@insistence	Strong determination to do something	ex) He will do it, whatever you say.
@intention	Intention about something or to do something	ex) He shall get this money. (Speaker's intention) ex) We shall let you know our decision.
@want	Desire to do something	ex) I want to go France.
@will	Determination to do something	ex) I'll write as soon as I can. ex) We won't stay longer than two hours.
@need	Necessity to do something	ex) You need to finish this work today. ex) I must be going now. ex) I always have to work hard.
@obligation	Obligation to do something according to (quasi-) law, contract, or ...	ex) The vendor shall maintain the equipment in good repair. ex) You must come by nine.
@obligation-not	Obligation not to do something, forbid to do something according to (quasi-) law, contract or ...	ex) Cars must not park in front of the entrance. ex) No smoking
@should	To do something as a matter of course	ex) You should do as he says. ex) You ought to start at once.
@unavoidable	Unavoidable feeling of the speaker about doing something	ex) I could not help speaking the truth.
@certain	Certainty that something is true or happens	ex) If Peter had the money, he would have bought a car. ex) They should be home by now.
@inevitable	Logical inevitability that something is true or happens	ex) All living things must die.
@may	Practical possibility that something is true or happens	ex) It may be true. ex) It could be.
@possible	Logical possibility that something is true or happens	ex) Anybody can make mistakes. ex) If Peter had the money, he would buy a car.
@probable	(Practical) probability that something is true or happens	ex) That would be his mother. ex) He must be lying.
@rare	Rare logical possibility that something is true or happens	ex) If such a thing should happen, what shall we do ? ex) If I should fail, I will [would] try again.
@unreal	Unreality that something is true or happens	ex) If we had enough money, we could buy a car. ex) If Peter had the money, he could buy a car.
@admire	Admiring feeling of the speaker about something	
@blame	Blameful feeling of the speaker about something	ex) A sailor, and afraid of the sea !
@contempt	Contemptuous feeling of the speaker about something	ex) You never could do it *In Japanese the post-positional particles of "nado", "nanka" or "nante" as in "kimi nado niha.." can be used to express the contemptuous feeling of the speaker about the target, mainly in a negative sentence

@regret	Regretful feeling of the speaker about something	ex) It's a pity that he should miss such a golden opportunity.
@surprised	Surprised feeling of the speaker about something	ex) (He has succeeded!) But that's great!
@troublesome	Troublesome feeling of the speaker about the occurrence of something	ex) My house was [I had my house] broken into.@troublesome yesterday. *There is a troublesome feeling of the speaker when using a passive form of the verb in this case in Japanese.

3 Grammaire des UW (BNF)

```
<UW> ::= <Headword> [<Constraint List>]
<Headword> ::= <character>...
<Constraint List> ::= ``('' <Constraint> [ ``,'' <Constraint>]... ``)''
<Constraint> ::= <Relation Label> { ``>'' | ``<'' }
       <Headword> [<Constraint List>] | <Constraint without relation>]
<Constraint without relation> ::= { ``>'' | ``<Headword>'' }...
<Relation Label> ::= ``agt'' | ``and'' | ... | ``via'' |
       ``equ'' | ``icl'' | ``iof''
<character> ::= ``A'' | ... | ``Z'' | ``a'' | ... | ``z'' |
       0 | 1 | 2 | ... | 9 | ``_'' | `` '' | ``#'' | ``!'' |
       ``$'' | ``%'' | ``='' | ``^'' | ``~'' | ``|'' | ``@'' |
       ``+'' | ``-'' | ``<'' | ``>'' | ``?''
```

Exemples :

- `book(icl>thing)` et `book(icl>do,agt>human,obj>thing)`
- `ikebana(icl>flower_arrangement)`
- `go_down`
- `.22(icl>firearm>thing,equ>twenty-two)`

4 UNLKB

La table suivante présente les quatre premiers niveaux de la taxonomie de l'UNLKB.

1st Level	**2nd Level**	**3rd Level**	**4th Level**
Universal Word uw	nominal concept	thing	abstract thing
			concrete thing
			functional thing
			place(icl>thing)
			pronominal thing
			time(icl>thing)
			volitional thing
	verbal concept	be	be(aoj>thing)
			be(aoj>thing,obj>thing)
		do	do(agt>thing)
			do(agt>thing,bas>thing,obj>thing)
			do(agt>thing,gol>thing)
			do(agt>thing,gol>thing,obj>thing)
			do(agt>thing,gol>thing,obj>thing,ptn>thing)
			do(agt>thing,gol>thing,obj>thing,ptn>thing,src>thing)
			do(agt>thing,gol>thing,obj>thing,src>thing)
			do(agt>thing,gol>thing,obj>uw)
			do(agt>thing,gol>thing,ptn>thing)
			do(agt>thing,gol>thing,ptn>thing,src>thing)
			do(agt>thing,gol>thing,src>thing)
			do(agt>thing,obj>thing)
			do(agt>thing,obj>thing,opl>thing)
			do(agt>thing,obj>thing,ptn>thing)
			do(agt>thing,obj>thing,ptn>thing,src>thing)
			do(agt>thing,obj>thing,src>thing)
			do(agt>thing,ptn>thing)
			do(agt>thing,ptn>thing,src>thing)
			do(agt>thing,src>thing)
		occur	occur(gol>thing,obj>thing)
			occur(gol>thing,obj>thing,src>thing)
			occur(obj>thing)
			occur(obj>thing,opl>thing)
			occur(obj>thing,src>thing)
	adjectival concept	uw(aoj>thing)	uw(aoj>abstract thing)
			uw(aoj>concrete thing)
			uw(aoj>functional thing)
			uw(aoj>place)
			uw(aoj>thing,bas>thing)
			uw(aoj>thing,obj>thing)
			uw(aoj>volitional thing)
		uw(mod<thing)	uw(mod<abstract thing)
			uw(mod<concrete thing)
			uw(mod<functional thing)
			uw(mod<place)
			uw(mod<volitional thing)
		uw(qua<thing)	
	adverbial concept	how	how(man<adjective concept)
			how(man<adjective concept,obj>thing)Â
			how(man<verbal concept)
			how(man<verbal concept,obj>thing)
			how(obj>thing)
			how(man<thing)

5 Un exemple d'hypergraphe UNL

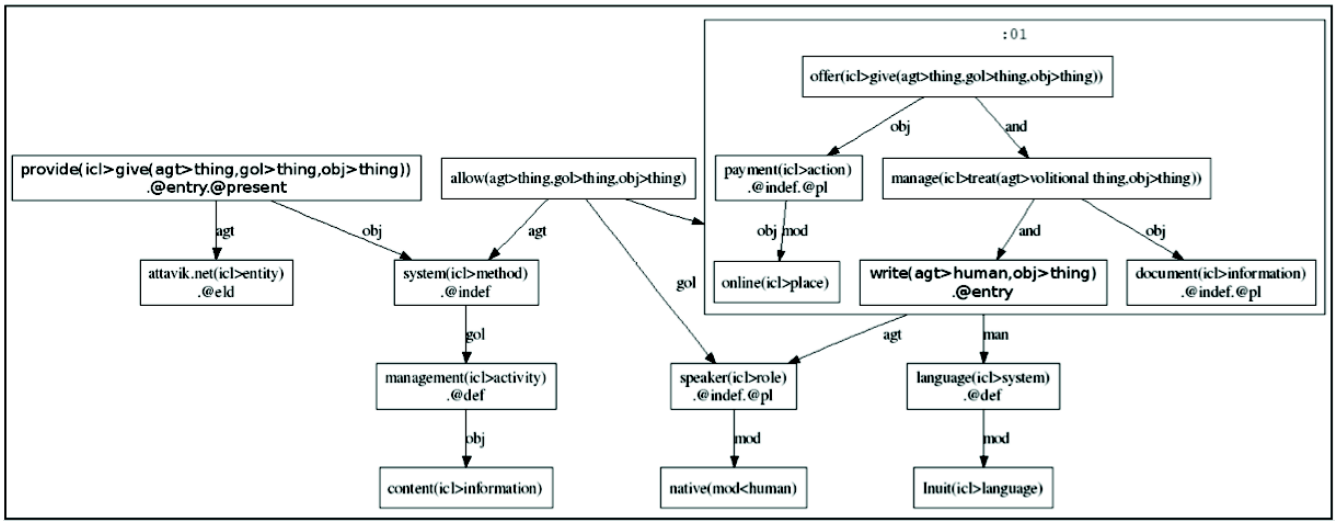


FIGURE 1 – Exemple d'hypergraphe UNL représentant la phrase "*Attawik.net provides a content management system that allows native speakers to write, manage documents and offer online payments in the Inuit language*".

```
agt(provide(icl>give(agt>thing,gol>thing,obj>thing)).@entry.@present, attavik.net(icl>entity).
    @eld)
obj(provide(icl>give(agt>thing, gol>thing,obj>thing)).@entry.@present, system(icl>method).
    @indef)
gol(system(icl>method).@indef,management(icl>activity).@def) obj(management(icl>activity).@def
    ,content(icl>information)) agt(allow(agt>thing,obj>thing,gol>thing),system(icl>method).
    @indef) gol(allow(agt>thing,obj>thing, gol>thing), speaker(icl>role).@indef.@pl) obj(allow
    (agt>thing,obj>thing, gol>thing), :01) obj:01(offer(icl>give(agt>thing,gol>thing,obj>thing
    )), payment(icl>action).@indef.@pl) and:01(offer(icl>give(agt>thing,gol>thing,obj>thing)),
     manage(icl>treat(agt>volitional
thing,obj>thing))) obj:01(manage(icl>treat(agt>volitional thing,obj>thing)), document(icl>
    information).@indef.@pl) and:01(manage(icl>treat(agt>volitional thing,obj>thing)), write(
    agt>human,obj>thing).@entry) agt(:01, speaker(icl>role).@indef.@pl) man(:01, language(icl>
    system).@def) mod(speaker(icl>role).@indef.@pl,native(mod<human)) mod:01(payment(icl>
    action).@indef.@pl,online(icl>place)) mod(language(icl>system).@def,Inuit(icl>language))
```

Annexe F

Implémentation XSLT transformant la sortie XML du logiciel NooJ en Graphe-Q.

```
<!--_____________________________________________________________
        Auteur : David Rouquet (David.Rouquet@imag.fr)
        XSLT 2, exécuter par exemple avec Saxon 9 HE

        Transformation de la sortie XML du logiciel NooJ en Graphe-Q.
_____________________________________________________________-->
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/201/XMLSchema" exclude-result-prefixes="xs" version="2.0">
    <xsl:output indent="no" method="text" encoding="UTF-8"/>
    <xsl:strip-space elements="*"/>
    <xsl:template match="text">
        <xsl:value-of select="."/>
        <xsl:text>.
</xsl:text>
    </xsl:template>
    <!--
Template principal : pour chaque unité lexicale (LU) d'une légende (caption) on lance le
    template NUM.
Dans le XML source, une LU représente une interprétation pour une segmentation possible de la
    légende ; un fils unique d'un LU est une autre interprétation de la même segmentation.
Si une LU a plusieurs fils (d'autres LU), ils représentent des interprétations pour une sous
    segmentation possible.
La segmentation du texte dans le XML source est donc présentée de façon récursive. La
    segmentation de niveau 1 est celle qui donne les segments les plus longs.
Ces segments sont ensuite éventuellement divisés en sous-segments et ainsi de suite.
NooJ, qui produit le XML source, ne gère pas l'export XML pour des segmentations qui se
    chevauchent (exemples : "white paper wall" ou "border line break").
-->
    <xsl:template match="caption">
        <xsl:text>**</xsl:text>
        <xsl:value-of select="@id"/>
        <xsl:text>.
</xsl:text>
        <!-- ne pas indenter la balise text ci dessus sinon la sortie sera indentée aussi-->
        <xsl:for-each select="child::LU">
            <xsl:call-template name="NUM">
                <!-- node1 est le précédent noeud de branchement rencontré dans la
                    construction du graphe-Q
    node2 est le noeud de branchement suivant.
    La valeur initiale de node2 correspond au dernier noeud du graphe-Q -->
```

```
                <xsl:with-param name="node1"/>
                <xsl:with-param name="node2">00<xsl:value-of select="count(preceding-sibling::
                    LU)+2"
                    /></xsl:with-param>
                <!-- mark sert siplement à repérer lorsque l'on traite le premier noeud (
                    exception dans le traitement récursif)-->
                <xsl:with-param name="mark">1</xsl:with-param>
            </xsl:call-template>
        </xsl:for-each>
    </xsl:template>

    <!--
Le template NUM est appellé seulement pour les LU qui doivent être numérotés.
-->
    <xsl:template name="NUM">
        <xsl:param name="node1"/>
        <xsl:param name="node2"/>
        <xsl:param name="mark"/>
        <!-- construct-node1 et 2 sont les numéros des noeuds à construire -->
        <xsl:variable name="construct-node1">
            <xsl:choose>
                <!-- quand un noeud LU n'a pas de frère précédent à ce niveau, il doit être
                    numéroté comme le précédent noeud courant (bifurcation dans le graphe-Q)
                    -->
                <xsl:when test="count(preceding-sibling::LU)=0">
                    <xsl:choose>
                        <xsl:when test="$mark=1">
                            <xsl:text>001</xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$node1"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$node1"/>00<xsl:value-of
                        select="count(preceding-sibling::LU)+1"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="construct-node2">
            <xsl:choose>
                <xsl:when test="count(following-sibling::LU)=0">
                    <xsl:value-of select="$node2"/>
                </xsl:when>
                <xsl:otherwise><xsl:value-of select="$node1"/>00<xsl:value-of
                        select="count(preceding-sibling::LU)+2"/></xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <!-- Le numéro du premier noeud est inséré ici, le suivant le sera dans le template
            ADDLEMMA, après avoir décoré l'arc les reliant -->
        <xsl:text>-</xsl:text>
        <xsl:value-of select="$construct-node1"/>
        <xsl:text>- </xsl:text>
        <!-- Insertion de la "forme" de l'occurrence décorant l'arc du graphe, telle que
            trouvée dans le texte source -->
        <xsl:text>$OCC($FORME(</xsl:text>
        <xsl:call-template name="ADDTEXT">
            <xsl:with-param name="text">
                <xsl:for-each select="descendant-or-self::LU">
                    <xsl:if test="not(*)"><xsl:value-of select="."/><xsl:text> </xsl:text> </
                        xsl:if>
                </xsl:for-each>
            </xsl:with-param>
        </xsl:call-template>
        <xsl:text>)</xsl:text>

        <xsl:call-template name="ADDLEMMA">
```

```
            <xsl:with-param name="node1">
                <xsl:value-of select="$construct-node1"/>
            </xsl:with-param>
            <xsl:with-param name="node2">
                <xsl:value-of select="$construct-node2"/>
            </xsl:with-param>
        </xsl:call-template>
    </xsl:template>

    <!--  ADDLEMMA ajoute les lemmes correspondant aux interprétations possible d'un segment,
        ou appelle NUM pour une ambiguïté de segmentation   -->
    <xsl:template match="LU" name="ADDLEMMA">
        <xsl:param name="node1"/>
        <xsl:param name="node2"/>
        <!-- les "choose" suivant servent :
    1) à enlever les redondances de la sortie NooJ (on ne s'intéresse que à LEMMA et CAT)
    2) à insérer la "forme" de l'occurrence dans LEMMA quand CAT=UNK (car dans ce cas le
        @LEMMA du XML source est vide)
-->
        <xsl:choose>
            <xsl:when test="@CAT = 'UNK'">
                <xsl:text>,$LU($LEMMA(</xsl:text>
                <xsl:call-template name="ADDTEXT">
                    <xsl:with-param name="text">
                        <xsl:value-of select="."/>
                    </xsl:with-param>
                </xsl:call-template>
                <xsl:text>), $CAT(</xsl:text>
                <xsl:value-of select="@CAT"/>
                <xsl:text>))</xsl:text>
            </xsl:when>
            <xsl:when test="not(@LEMMA = parent::*/@LEMMA)">
                <xsl:text>,$LU($LEMMA(</xsl:text>
                <xsl:call-template name="ADDTEXT">
                    <xsl:with-param name="text">
                        <xsl:value-of select="@LEMMA"/>
                    </xsl:with-param>
                </xsl:call-template>
                <xsl:text>), $CAT(</xsl:text>
                <xsl:value-of select="@CAT"/>
                <xsl:text>))</xsl:text>
            </xsl:when>
            <xsl:otherwise>
                <xsl:if test="not(@CAT = parent::*/@CAT)">
                    <xsl:text>,$LU($LEMMA(</xsl:text>
                    <xsl:call-template name="ADDTEXT">
                        <xsl:with-param name="text">
                            <xsl:value-of select="@LEMMA"/>
                        </xsl:with-param>
                    </xsl:call-template>
                    <xsl:text>), $CAT(</xsl:text>
                    <xsl:value-of select="@CAT"/>
                    <xsl:text>))</xsl:text>
                </xsl:if>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <!-- si un noeud LU à plus d'un fils, on est devant une ambiguité de segmentation,
                 on appelle donc le template NUM pour créer les noeuds en conséquence  -->
            <xsl:when test="count(child::LU) &gt; 1">
                <xsl:text>)-</xsl:text>
                <xsl:value-of select="$node2"/>
                <xsl:text>-
</xsl:text>
                <!-- ne pas indenter la balise text ci dessus sinon la sortie sera indentée
                    aussi-->
                <xsl:for-each select="child::LU">
                    <xsl:call-template name="NUM">
```

```
                        <xsl:with-param name="node1">
                            <xsl:value-of select="$node1"/>
                        </xsl:with-param>
                        <xsl:with-param name="node2">
                            <xsl:value-of select="$node2"/>
                        </xsl:with-param>
                        <xsl:with-param name="mark">2</xsl:with-param>
                    </xsl:call-template>
                </xsl:for-each>
            </xsl:when>
            <!-- si un noeud LU n'a pas d'enfant,on ajoute le noeud de fin de l'arrête -->
            <xsl:when test="count(child::LU) = 0">
                <xsl:text>)-</xsl:text>
                <xsl:value-of select="$node2"/>
                <xsl:text>-
</xsl:text>
            </xsl:when>
            <!-- ne pas indenter la balise text ci dessus sinon la sortie sera indentée aussi
                -->
            <!-- si le noeud LU n'a qu'un enfant, il s'agit d'une nouvelle interprétention
                pour le segment, on l'ajoute et on appelle de nouveau ADDLEMMA -->
            <xsl:otherwise>
                <xsl:apply-templates select="child::LU">
                    <xsl:with-param name="node1">
                        <xsl:value-of select="$node1"/>
                    </xsl:with-param>
                    <xsl:with-param name="node2">
                        <xsl:value-of select="$node2"/>
                    </xsl:with-param>
                </xsl:apply-templates>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>

    <!-- Le template suivant permet d'ajouter les textes (valeur d'une balise LU ou d'un
        attribut LEMMA) avec les caractères spéciaux sous une forme adéquate pour les systèmes
        -Q) -->
    <xsl:template name="ADDTEXT">
        <xsl:param name="text"/>
        <xsl:variable name="forme">
            <xsl:value-of select="translate($text, '
                ÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙÚÛÜÝàáâãäåçèéêëìíîïñòóôõöùúûüýÿ', '
                AAAAAACEEEEIIIINOOOOOUUUUYaaaaaaceeeeiiiinoooooouuuuyy')"/>
        </xsl:variable>
        <xsl:variable name="forme0">
            <xsl:value-of select="replace($forme,'[^A-Za-z0-9\-\+/\*\$=\. !\?:\[\]@#]',' ')"/>
        </xsl:variable>
        <xsl:variable name="forme1">
            <xsl:value-of select="replace($forme0,'(^\s+)|(\s+$)','')"/>
        </xsl:variable>
        <xsl:variable name="forme2">
            <xsl:value-of select="replace($forme1,'[\s]+',' ')"/>
        </xsl:variable>
        <xsl:variable name="forme3">
            <xsl:value-of
                select="replace($forme2,'^(.*)([\-\+/\*\$=\. !\?:\[\]@#])(.*)','MOTCP($1$2$3)
                ')"
            />
        </xsl:variable>
        <xsl:variable name="forme4">
            <xsl:value-of
                select="replace($forme3,'(\-|\+|/|\*|\$|=|\.| |!|\?|:|\[|\]|@|#)',',$1,')"/>
        </xsl:variable>
        <xsl:variable name="forme5">
            <xsl:value-of select="replace($forme4,' ','#sp')"/>
        </xsl:variable>
        <xsl:variable name="forme6">
            <xsl:value-of select="replace($forme5,'!','#exc')"/>
```

```
        </xsl:variable>
        <xsl:variable name="forme7">
            <xsl:value-of select="replace($forme6,'\?','#int')"/>
        </xsl:variable>
        <xsl:variable name="forme8">
            <xsl:value-of select="replace($forme7,'@','#at')"/>
        </xsl:variable>
        <xsl:variable name="forme9">
            <xsl:value-of select="replace($forme8,'=','#eq')"/>
        </xsl:variable>
        <xsl:variable name="forme9bis">
            <xsl:value-of select="replace($forme9,',,',',')"/>
        </xsl:variable>
        <xsl:variable name="forme9ter">
            <xsl:value-of select="replace($forme9bis,',\)',')')"/>
        </xsl:variable>
        <xsl:variable name="forme10">
            <xsl:value-of select="replace($forme9ter,'\(','(')"/>
        </xsl:variable>

        <xsl:variable name="forme11">
            <xsl:choose>
                <xsl:when test="$forme10=''">#sp</xsl:when>
                <xsl:when test="$forme10='MOTCP()'">#sp</xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="replace($forme10,'MOTCP','\$MOTCP')"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:value-of select="$forme11"/>
    </xsl:template>

</xsl:stylesheet>
```

Annexe G

Implémentation PERL d'un algorithme adapté de Lesk pour la désambiguïsation d'un alignement ontologie UW

```
################################################################################
# Auteur : David Rouquet (David.Rouquet@imag.fr)
# PERL 5.11
################################################################################
#
# Ce programme donne un score aux liens d'un alignement $ALIGN entre une ontologie $OWL et un
    lexique d'UW $UWS
# en comparant le voisinage d'un concept dans l'ontologie avec les restrictions, la définition
     et l'exemple associé
# à l'UW correspondante.
# Le résultat est un alignement contenu dans le fichier alignDesambIclDefExVois.rdf.
#
# This program scores an alignment $ALIGN between an ontology $OWL and an UW lexicon $UWS, by
    comparing
# the neighbourhood of a concept with restrictions, definitions, and exemples in corresponding
     UW.
# Resulting alignment is produced in the file alignDesambIclDefExVois.rdf
#
#
#
use warnings;

# Importe le module XML::Simple qui permet la création d'une table de références correspondant
     à l'arbre d'un fichier XML.
# Importe le module Data::Dumper qui permet une meilleure présentation des donnée en sortie.
use XML::Simple;
use Data::Dumper;

# Construction de la structure XML::Simple en mémoire vive
# (Ontologie OWL, lexique d'UW en SKOS, et alignement dans le format RDF d'EXMO)
my $OWL = XMLin('./100606_OMNIA_v6.owl', forcearray=>1);
my $UWS = XMLin('./Kaiko_UWpp_sentiWN_OWLapi.owl', forcearray=>1);
my $ALIGN = XMLin('./Kaiko_align_UWpp-OMNIAv6_StringEq.rdf', forcearray=>1);


#boucle sur chaque cellule (lien) de l'alignement
for my $Cell (@{$ALIGN->{'Alignment'}->[0]->{'map'}})
        {
```

```
#récupère concept et UW
my $cept = $Cell->{'Cell'}->[0]->{'entity2'}->[0]->{'rdf:resource'} ;
my $uwid = $Cell->{'Cell'}->[0]->{'entity1'}->[0]->{'rdf:resource'} ;
my %ceptVois = ('father'=>[],'brother'=>[],'son'=>[]) ;


#Cré le voisinage du concept (parents, frères et fils)
for my $ceptOWL (@{$OWL->{'owl:Class'}})
{
        #on commence par les pères
        if ($cept eq $ceptOWL->{'rdf:about'})
                {
                for my $ceptFather (@{$ceptOWL->{'rdfs:subClassOf'}})
                        {
                                my $ceptFatherWord = $ceptFather->{'rdf:resource'};
                                $ceptFatherWord =~ tr/\#//d ;
                                $ceptFatherWord =~ tr/\_/ / ;
                                $ceptFatherWord = uc($ceptFatherWord);
                                push(@{$ceptVois{father}}, $ceptFatherWord) ;

                                #puis les frères
                                for my $ceptTEMPbrother (@{$OWL->{'owl:Class'}})
                                {
                                for my $ceptTEMPfather (@{$ceptTEMPbrother->{'rdfs:
                                    subClassOf'}})
                                {
                                        if ($ceptFather->{'rdf:resource'} eq $
                                            ceptTEMPfather->{'rdf:resource'})
                                        {
                                                my $ceptBro = $ceptTEMPbrother->{'rdf:
                                                    about'};
                                                if ($cept ne $ceptBro)
                                                {
                                                $ceptBro =~ tr/\#//d ;
                                                $ceptBro =~ tr/\_/ / ;
                                                $ceptBro = uc($ceptBro);
                                                push(@{$ceptVois{brother}}, $ceptBro)
                                                    ;
                                                }
                                        }
                                }
                                }

                        }
                }
        #et enfin les fils
        for my $ceptOWLfather (@{$ceptOWL->{'rdfs:subClassOf'}})
        {
                if ($cept eq $ceptOWLfather->{'rdf:resource'})
                {
                        my $ceptSon = $ceptOWL->{'rdf:about'} ;
                        $ceptSon =~ tr/\#//d ;
                        $ceptSon =~ tr/\_/ / ;
                        $ceptSon = uc($ceptSon);
                        push(@{$ceptVois{son}}, $ceptSon) ;

                }
        }
}

# Crée la variable %uw qui correspond à l'UW dont l'identifiant est $uwid sous forme
    de hash.
# Par exemple, la chaîne manual(icl>exercise>thing, equ>manual_of_arms)
# devient %uw = (hw=>manual, icl=>[exercise,thing], eq=>[manual_of_arms])
# !!! les vecteurs sont stockés dans la table sous formes de références !!!
#
# Ici, on pourrrait optimiser en ayant au préalable indexé les UW par leur identifiant
    .
```

```
# Ce n'est pas le cas dans la structure crée par XML::Simple à partir du fichier SKOS,
# on est donc obligé de chercher l'identifiant $uwid parmi toutes les UW.

for my $uwTemp (@{$UWS->{'skos:Concept'}})
        {
        my $uwTempId = $uwTemp->{'rdf:about'};
                if ($uwTempId eq $uwid)
                {
                $uwLexeme = $uwTemp->{'skos:prefLabel'}->[0] ;
                ($hw, $restr) = split(/\(|\)/, $uwLexeme, 3);
                %uw=(hw => $hw);

                my @restr = split(/, /, $restr) ;
                for my $unitRestr (@restr)
                        {
                        @restrList = split(/\>/, $unitRestr) ;
                                $restrName = shift(@restrList) ;
                                $uw{$restrName} = [@restrList] ;
                        }

        # Cherche les éléments du voisinage du concept dans les restriction, la def et
             l'ex de l'uw.
        # Ajoute 0,5 au score de l'alignement si une égalité de chaîne est trouvée (
            les chaînes sont normalisées en majuscule).
        for my $voisType ('father','brother','son')
        {
                for my $voisCeptElt (@{$ceptVois{$voisType}})
                {
                        for my $voisUWtype ('skos:prefLabel','skos:example','skos:
                            definition')
                        {
                                #       print Dumper($uwTemp->{$voisUWtype}->[0]) ;
                                my $voisUWelt = $uwTemp->{$voisUWtype}->[0] ;
                                $voisUWelt =~ tr/\_/ / ;
                                if (uc($voisUWelt) =~ m/$voisCeptElt/)
                                {
                                        print '--->' ;
                                        print Dumper($uwLexeme) ;
                                        print Dumper($cept) ;
                                        print Dumper($voisUWelt) ;
                                        print Dumper($voisCeptElt) ;

                                        $Cell->{'Cell'}->[0]->{'measure'}->[0]->{'
                                            content'}=  $Cell->{'Cell'}->[0]->{'
                                            measure'}->[0]->{'content'} + 0.5 ;

                                }
                        }
                }
        }

                }
        }

        # Boucle sur les restrictions icl de l'%uw et
        # ajoute 1 au score de l'alignement si la restriction est une étiquette d'un
            père du concept
for my $iclWord (@{$uw{'icl'}})
        {
        $iclWord = uc($iclWord) ;
        for my $ceptOWL (@{$OWL->{'owl:Class'}})
                {

                if ($cept eq $ceptOWL->{'rdf:about'})
                        {
                        for my $ceptFather (@{$ceptOWL->{'rdfs:subClassOf'}})
                                {
```

```
                                        my $ceptFatherWord = $ceptFather->{'rdf:
                                            resource'};
                                        $ceptFatherWord =~ tr/\#//d ;
                                        $ceptFatherWord = uc($ceptFatherWord);

                                    if ($iclWord eq $ceptFatherWord)
                                        {
                                                        print '--->' ;
                                                        print Dumper($uwLexeme) ;
                                                        print Dumper($ceptFatherWord)
                                                            ;


                                        $Cell->{'Cell'}->[0]->{'measure'}->[0]->{'
                                            content'}=  $Cell->{'Cell'}->[0]->{'
                                            measure'}->[0]->{'content'} + 1 ;

                                        }
                                    }
                                }
                            }
                        }

                        open FILE, ">", "alignDesambIclDefExVois.rdf" ;
                        print FILE XMLout($ALIGN) ;
                        close FILE ;
}
```

Annexe H

Systèmes-Q

Septembre 1970—saisi par les étudiants de RICM, 2001-02, rév. Ch. Boitet 090528

LES SYSTEMES-Q
OU UN FORMALISME POUR ANALYSER ET SYNTHETISER DES PHRASES SUR ORDINATEUR

Alain COLMERAUER

Université de Montréal, Groupe TAUM, publication interne n° 43

Ce travail a été subventionné par le Conseil National de la Recherche du Canada : ceci dans le cadre d'un octroi à titre personnel et dans le cadre du projet de Traduction Automatique de l'Université de Montréal.

RESUME

Un système-q est un ensemble de règles permettant de faire certaines transformations sur des graphes orientés. Chaque flèche d'un tel graphe est surmontée d'une expression parenthésée. Les transformations peuvent correspondre à une analyse, à une synthèse de phrase, ou à une manipulation formelle de ce genre. Il est possible d'utiliser le même système-q pour décrire à la fois un processus et le processus inverse, comme par exemple l'analyse et la synthèse d'une même phrase.

ABSTRACT

A q-system is a set of rules that allows certain transformations to be carried out on oriented graphs. An expression with parentheses appears on each arrow of such a graph. The transformation may correspond to an analysis, a synthesis of a sentence, or to a similar formal manipulation. It is possible to use the same q-system to describe both a process and the inverse of this process, as for example the analysis and the synthesis of a sentence.

TABLE DES MATIERES

INTRODUCTION

Utiliser un ordinateur pour analyser une phrase est une entreprise difficile. Le problème principal est de nature combinatoire : pris isolément chaque groupe d'éléments de la phrase peut se combiner de différentes façons avec d'autres groupes et en former de nouveaux qui à leur tour peuvent se recombiner, et ainsi de suite. En général il n'existe qu'une seule façon de regrouper la totalité des éléments, mais, pour la découvrir, il faut essayer tous les groupements partiels possibles. Pour représenter d'une façon économique cette multitude de groupements, j'utilise un graphe orienté où chaque flèche est surmontée d'une expression parenthésée représentant un arbre. Un système-q n'est rien d'autre qu'un ensemble de règles permettant de transformer un tel graphe en un autre graphe. Cette transformation peut correspondre à une analyse, à une synthèse de phrase, ou à une autre manipulation formelle de ce genre.

Une des originalités des systèmes-q est leur aspect réversible. Ceci veut dire que pratiquement le même système-q peut être utilisé pour décrire une transformation et la transformation inverse, comme par exemple l'analyse et la synthèse de phrases. J'en donne un exemple à la fin de cet exposé.

Un autre aspect intéressant est la possibilité d'enchaîner plusieurs systèmes-q les uns à la suite des autres. Chacun prend comme données les résultats du précédent. Cette technique est largement utilisée dans notre projet de traduction automatique, puisqu'une phrase anglaise ne "subit" pas moins de quinze systèmes-q avant d'être traduite en français. Deux systèmes-q "s'occupent" de la morphologie anglaise, un autre de l'analyse de l'anglais, deux du passage d'une "structure" anglaise à une structure "française", un de la synthèse du français et neuf de la morphologie française.

Pour présenter le formalisme des systèmes-q, j'utiliserai une technique propre à la description des langages de programmation. Je commencerai par définir les éléments simples, puis à partir d'eux les plus complexes, en utilisant toujours la forme normale de Backus.

Bien entendu les systèmes-q ont été implantés sur l'ordinateur CDC 6400 de l'université de Montréal. Un premier programme écrit en Algol a été opérationnel dès octobre 1969. Actuellement nous utilisons une traduction plus efficace de ce programme en Fortran.

I. CARACTERES ET ETIQUETTES

Le caractère constitue la plus petite unité d'information. Il est classé en quatre catégories.

```
<caractère> ::= <lettre> | <chiffre> | <signe significatif>
                | <signe non significatif>
<lettre> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<chiffre> ::= 0|1|2|3|4|5|6|7|8|9
<signe significatif> ::= + | - | * | / | ( | ) | $ | = | ≠ | . | ,
<signe non significatif> ::= [ | ] | : | ↲ | ∨ | ∧ | ↑ | ↓ | ^ | < | > | ≤ | ≥ | ¬
```

Je n'ai pas prévu les lettres minuscules car elles n'existent pas sur notre ordinateur CDC 6400. De même, on peut concevoir une toute autre liste de signes non significatifs. Le choix que j'ai fait correspond au reste des caractères disponibles sur notre ordinateur.

Une étiquette est une suite de caractères : le premier peut être quelconque mais les autres caractère éventuels doivent être des chiffres ou des lettres.

```
<étiquette>  ::= <caractère> | <étiquette> <lettre>
                 | <étiquette> <chiffre>
```

Voici par exemple cinq étiquettes correctes :

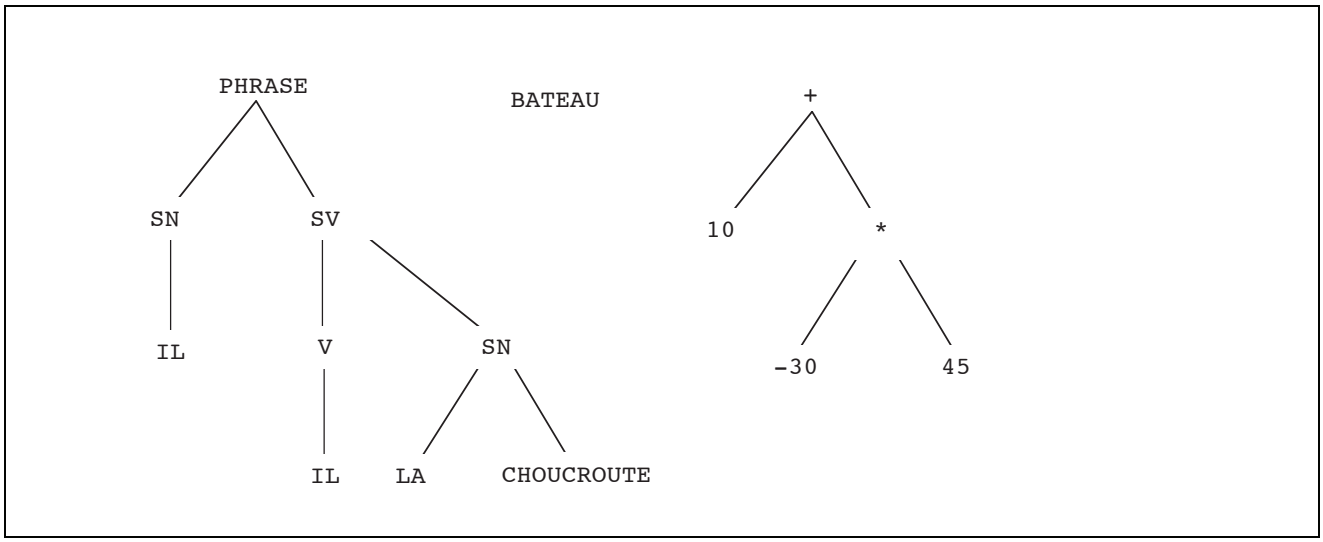
```
+   -PL3   (   HIBOU   ,A
```

et en voici cinq incorrectes

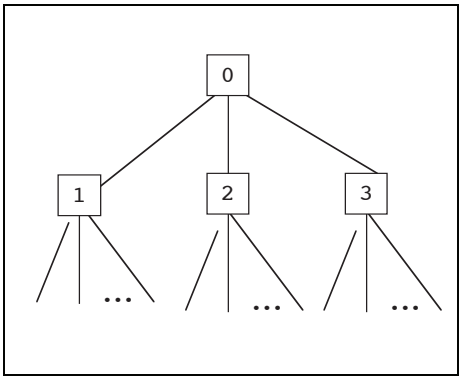
```
/OUI/   (A)   DIT-IL   3.14   %.
```

II. ARBRES ET LISTES

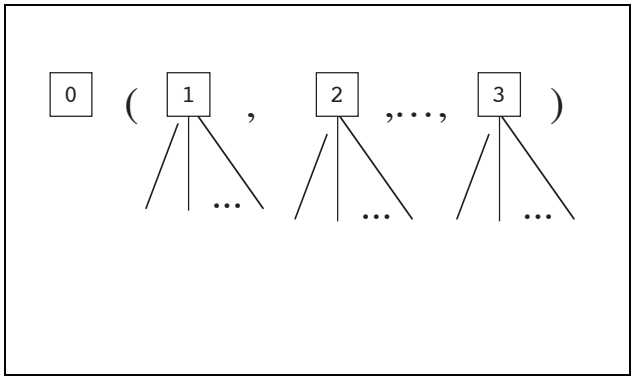
L'arbre est l'élément de base. Il correspond à ce que l'on a coutume de représenter par des schémas du genre :



On impose que les mots, les abréviations ou, d'une façon générale, les suites de caractères apparaissant dans de tels schémas soient des étiquettes. J'appelle arbre une expression formée d'étiquettes, de parenthèses et de virgules. Une telle expression se construit en prenant un schéma d'arbre et en remplaçant, autant de fois qu'il est nécessaire, la configuration



par la configuration



Les trois schémas précédents correspondant donc respectivement aux trois arbres :

```
PHRASE(SN(IL),SV(LA,CHOUCROUTE)))
    BATEAU
+(10,*(-30,45))
```

J'appelle liste toute suite d'arbres séparés par des virgules. Voici deux exemples de listes :

```
SN(IL),SV(V(MANGE))
A,B,C,D
```

Je note -NUL- la liste vide et je considère que, quelle que soit la liste *l* et quelle que l'étiquette *e*, les trois écritures

```
l,-NUL-    -NUL-,l       l
```

sont équivalentes, ainsi que les deux écritures

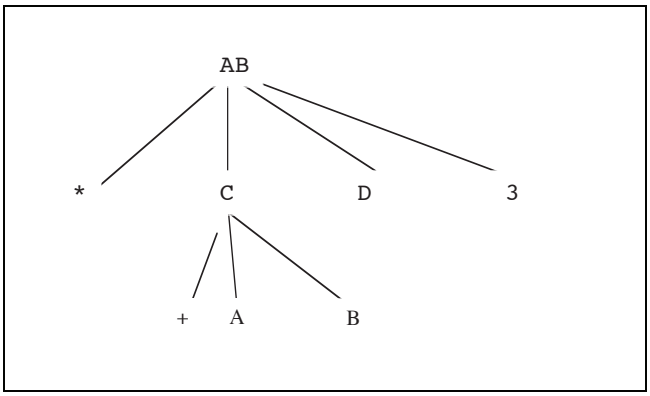
```
e(-NUL-)      e
```

J'introduis un opérateur noté $$. Il est appelé désintégrateur et, appliqué à une étiquette, il la transforme en une liste de caractères. Plus exactement, je considère que les deux écritures

$$\$\$c_1c_2\ldots c_n \qquad\qquad c_1, c_2, \ldots, c_n$$

sont équivalentes à condition que les c_i soient des caractères et que $c_1c_2\ldots c_n$ soit une étiquette.

En tenant compte de ce qui précède, le schéma d'arbre



peut être aussi bien représenté par

```
AB(*,C(+,A,B),D,3)
```

que par

```
AB(*(-NUL-),C($$+AB),D,3)
```

ou par

```
AB(*,-NUL-,-NUL-,C(+,A,B),D,3)
```

Voici maintenant les règles précises de formation d'un arbre et d'une liste.

```
<arbre> ::= <étiquette> | <étiquette> ( <liste> )
<liste> ::= <liste simple> | <liste simple> , <liste>
<liste simple> ::= -NUL- | <arbre> | $$ <étiquette>
```

III. ARBRES PARAMETRES ET LISTES PARAMETREES

Grosso modo, un arbre paramétré est un arbre dans lequel apparaissent éventuellement certains paramètres. Ces paramètres représentent des listes d'arbres ou des listes encore inconnues. La forme d'un paramètre est définie par :

```
<paramètre étiquette> ::=  <abcdef> * <indice>
<paramètre arbre> ::=  <ijklmn> * <indice>
<paramètre liste> ::=  <uvwxyz> * <indice>
<abcdef> ::=  A | B | C | D | E | F
<ijklmn> ::=  I | J | K | L | M | N
<uvwxyz> ::=  U | V | W | X | Y | Z
<indice> ::=  |  <chiffre>
```

Un paramètre est donc composé d'une lettre *l* suivie d'un astérisque suivi d'un chiffre *n* éventuellement absent.

Si *l* est compris entre A et F alors le paramètre représente une étiquette.

Si *l* est compris entre I et N alors le paramètre représente un arbre.

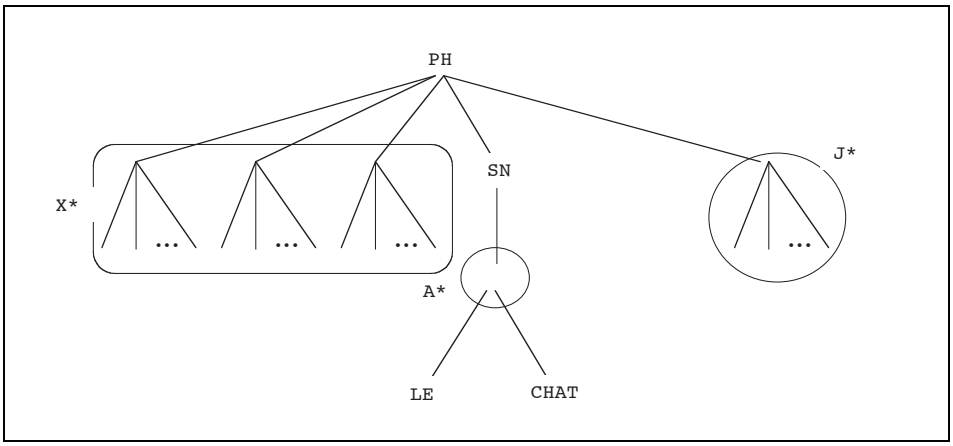
Si *l* est compris entre U et Z alors le paramètre représente une liste.

L'absence du chiffre *l* est équivalente à la présence du chiffre 0.

Voici par exemple un arbre paramétré

```
PH(X*,SN(A*(LE,CHAT)),I*)
```

On peut le représenter par le schéma



Si on substitue respectivement aux paramètres

```
X*          A*          I*
```

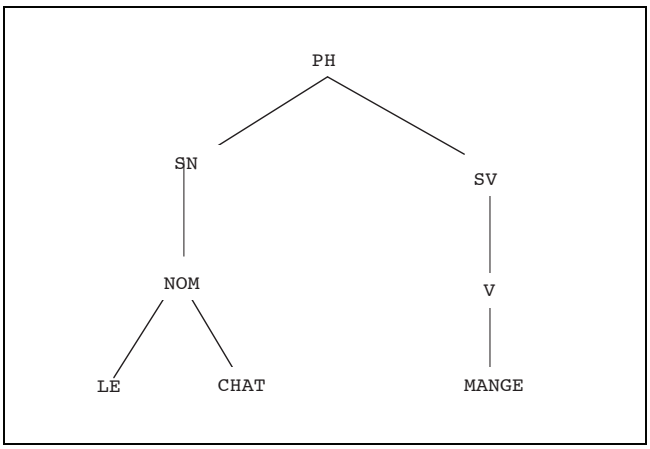
les expressions

```
-NUL-       NOM         SV(V(MANGE))
```

on obtient l'arbre

```
PH(SN(NOM(LE,CHAT)),SV(V(MANGE)))
```

qui correspond au schéma



Voici maintenant la définition précise d'un arbre et d'une liste paramétrés.

```
<arbre paramétré> ::= <paramètre arbre> | <étiquette paramétrée>
                    | <étiquette paramétrée> ( <liste paramétrée> )

<liste paramétrée> ::= <liste simple paramétrée> |
<liste simple paramétrée>, <liste paramétrée>
```

```
<liste simple paramétrée> ::= <paramètre liste> | -NUL-
                            | <arbre paramétré> | $$ <étiquette paramétrée>
<étiquette paramétrée> ::= <paramètre étiquette> | <étiquette>
```

Un arbre (ou une liste) paramétré représente l'ensemble de tous les arbres (ou listes) paramétrés que l'on peut obtenir, en substituant à chacun de ses "paramètres étiquettes" une étiquette quelconque, à chacun des ses "paramètres arbres" un arbre quelconque et à chacun des ses "paramètres listes" une liste quelconque. Par "substituer" j'entends que l'on remplace différentes occurrences d'un même paramètre par la même chose.

IV. RELATIONS ENTRE DES LISTES

Un arbre étant un cas particulier de liste, il est plus intéressant de définir des relations binaires entre des listes qu'entre des arbres. J'aurai besoin de quatre relations notées :

= ≠ -DANS- -HORS-

Les relations = et ≠ sont triviales, elles correspondent à l'égalité et à l'inégalité entre deux listes. Il faut cependant tenir compte des équivalences d'écriture introduites par la liste vide et l'opérateur désintégrateur $$ (voir section 2). Par exemple on a

```
0,1,2,3,4,5,6,7,8,9 = $$0123456789
A(-NUL-) = A
AB ≠ AB(X)
-NUL- = -NUL-,-NUL-
```

En convenant que e_1, e_2 représentent deux étiquettes quelconques, a_1, a_2 deux arbres quelconques et l_1, l_2 deux listes quelconques, la relation -DANS- se définit récursivement comme

-NUL- -DANS- l_2	≡	vrai
l_1 -DANS- -NUL-	≡	l_1 = -NUL-
a_1, l_1 -DANS- l_2	≡	a_1 -DANS- l_2 et l_1 -DANS- l_2
a_1 -DANS- a_2, l_2	≡	a_1–DANS- a_2 ou a_1 -DANS- l_2
$a_1(l_1)$ -DANS- $e_2(l_2)$	≡	$e_1 = e_2$ et l_1 -DANS- l_2

et la relation -HORS- qui n'est pas l'exacte négation de -DANS- se définit récursivement comme

-NUL- -HORS- l_1	≡	faux
l_1 -HORS- -NUL-	≡	faux
a_1, l_1 -HORS- l_2	≡	a_1 -HORS- l_2 et (l_1 -HORS- l_2 ou l_2 = -NUL-)
a_1 -HORS- a_2, l_2	≡	a_1 -HORS- a_2 et (a_1 -HORS- l_2 ou l_2 = -NUL-)
$e_1(l_1)$ -HORS- $e_2(l_2)$	≡	$e_1 = e_2$ ou l_1 -HORSl l_2

Il faut remarquer que -HORS- est une relation symétrique.

Bien entendu, dans ces deux dernières définitions, il faut tenir compte des équivalences d'écriture qu'induisent la liste vide et le désintégrateur.

Voici quelques exemples où les relations de -DANS- et -HORS- sont vérifiées.

1, 9, 7, 1	-HORS-	0, 2, 3
$$1971	-DANS-	$$0123456789
1, 9, 7, 1	-DANS-	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
PH	-DANS-	PH(SN(BORIS), SV(DORT))
SN	-HORS-	PH(SN(BORIS), SV(DORT))
A, B(A, B), A	-DANS-	B(B(A), A), A(C)
A, B(A, A), A	-DANS-	B(B(A), B), B(C)

Dans le cas où $\boldsymbol{l}_1$ et $\boldsymbol{l}_2$ sont de simples listes d'étiquettes, les opérateurs -HORS- et -DANS- se définissent très simplement. En effet, si $\boldsymbol{E}_1$ et $\boldsymbol{E}_2$ désignent respectivement l'ensemble des étiquettes apparaissant dans $\boldsymbol{l}_1$ et $\boldsymbol{l}_2$ on a :

$\boldsymbol{l}_1$ -DANS- $\boldsymbol{l}_2$	≡	$\boldsymbol{E}_1 \subset \boldsymbol{E}_2$
$\boldsymbol{l}_1$ -HORS- $\boldsymbol{l}_2$	≡	$\boldsymbol{E}_1 \cap \boldsymbol{E}_2 = \varnothing$ et $\boldsymbol{E}_1 \neq \varnothing$ et $\boldsymbol{E}_2 \neq \varnothing$

V. CHAINES

On a coutume d'appeler chaîne une suite finie de symboles écrits les uns à côté des autres, par exemple

abacb

Sur ces chaînes, on peut faire des transformations du genre : remplacer telle sous-chaîne x par telle autre sous-chaîne y ; on note

$x \rightarrow y$

Par exemple, la transformation *bac* → *ac* appliquée sur la chaîne précédente produirait *aacb*.

Je me propose de faire des transformations très analogues. Cependant, au lieu de considérer qu'un symbole ne peut être décomposé en éléments plus simples, je définis un symbole comme étant tout un arbre. Afin d'éviter certaines confusions, on est alors amené à utiliser un séparateur, en l'occurrence le signe plus, pour séparer les différents arbres qui composent une chaîne.

```
<chaîne>  ::=  <arbre>  |  <arbre>  +  <chaîne>
```

Voici quelques exemples de chaînes :

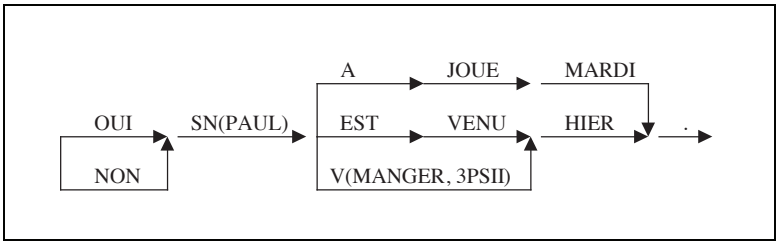
```
SN(PAUL) + VOIT + SN(PIERRE)
[ (SN) + PAUL + ] + VOIT + [ (SN) + PIERRE + ]
+ + + + +
```

Le dernier exemple n'est pas ambigu; le deuxième et le quatrième plus ("+") servent de séparateur alors que les autres plus sont des étiquettes.

VI. GRAPHES DE CHAINES

Plutôt que de faire des transformations sur une seule chaîne, je me propose d'en faire sur tout un ensemble. Pour représenter d'une façon économique un tel ensemble, j'utilise un graphe de chaînes.

Un graphe de chaînes est un ensemble de flèches, chacune surmontée d'un arbre, qui relient un ensemble de sommets entre eux. En voici un exemple.



J'appelle chemin tout suite non vide et finie de flèches

$$f_1, f_2, \ldots, f_n$$

qui est telle que n=1 ou que n>1 et que la flèche f_i "arrive" à un sommet d'où "part" la flèche f_{i+1} (i = 1, 2…, n-1).

Soit a_i l'arbre qui surmonte chaque flèche f_I. On dira que le chemin porte la chaîne

$$a_1 + a_2 + \ldots + a_n$$

Un graphe de chaînes doit satisfaire deux conditions :

(1) il ne contient pas de circuit, c'est-à-dire qu'il n'y a pas de chemin allant d'un sommet à ce même sommet,

(2) il y a deux sommets appelés respectivement entrée et sortie qui sont tels que toute flèche du graphe figure sur un chemin allant de l'entrée à la sortie. De la condition (1), il découle qu'il ne peut y avoir qu'une seule entrée et qu'une seule sortie.

Un graphe de chaînes représente l'ensemble des chaînes allant de l'entrée à la sortie. Dans l'exemple précédent, cet ensemble est :

```
OUI + SN(PAUL) + EST + VENU + HIER + .
OUI + SN(PAUL) + A + JOUE + MARDI + .
OUI + SN(PAUL) + V(MANGER, 3PSII) + HIER + .
NON + SN(PAUL) + EST + VENU + HIER + .
NON + SN(PAUL) + A + JOUE + MARDI + .
NON + SN(PAUL) + V(MANGER, 3PSII) + HIER + .
```

Pour représenter formellement un graphe de chaînes, je représente chaque sommet par un entier non négatif encadré de deux traits d'union et chaque flèche par un triplet constitué du sommet d'où part la flèche suivi de l'arbre qui la surmonte suivi du sommet où elle arrive. Il me suffit alors d'énumérer les flèches qui composent le graphe considéré. Le graphe de chaînes précédent peut donc être représenté par

```
-1- OUI –2-
-1- NON –2-
-2- SN(PAUL) –3-
-3- EST –4-
-4- VENU –5-
-3- V(MANGER, 3PSII) –5-
–5- HIER –6-
–3- A –7-
–7- JOUE –8-
–8- MARDI –6-
-6- . –9-
```

Afin d'abréger l'écriture, on permet de remplacer toute sous-suite de flèches de la forme :

$-i_0-\ a_1\ -i_1-$
$-i_1-\ a_2\ -i_2-$

$.....$
$-i_{n-1}-\ a_n\ -i_n-$

(où a_1, a_2, ..., a_n sont des arbres) qui est telle que les sommets $-i_1-$, $-i_2-$, ... , $-i_{n-1}-$ n'apparaissent pas ailleurs dans la représentation du graphe, par

$-i_0-\ a_1 + a_2 + ... + a_n\ -i_n-$

Une telle suite de flèches est appelée chemin sans carrefour. Le graphe précédent peut donc aussi être noté

```
-1- OUI -2-
-1- NON -2-
-2- SN(PAUL) -3-
-3- EST + VENU -4-
-3- V(MANGER, 3PSII) -4-
-4- HIER -5-
-3- A + JOUE + MARDI -5-
-5- . -6-
```

D'une façon formelle je suis donc amené à définir un graphe d'arbres comme

```
<graphe de chaînes> ::= <chemin sans carrefour>
                      | <chemin sans carrefour> <graphe de chaînes>
<chemin sans carrefour> :: = <sommet> <chaîne> <sommet>
<chaîne> ::= <arbre> | <arbre> + <chaîne>
<sommet> ::= -<entier>-
<entier> ::= <chiffre> | <chiffre> <entier>
```

Un graphe d'arbre est donc de la forme

$i_1\ c_1\ j_1$
$i_1\ c_2\ j_2$
$.......$
$i_n\ c_n\ j_n$

où i_1, i_2, ..., i_n sont des entiers non négatifs et c_1, c_2, ... , c_n des chaînes.

J'impose que, quels que soient les entiers p et q ($1 \le p \le n$ et $1 \le q \le n$),

$i_p = j_q$ entraîne $p>q$

Cette restriction est absolument équivalente à celle qui interdit l'existence d'un circuit.

On impose aussi de ne pas utiliser la liste vide -NUL- et le désintégrateur $$ pour représenter les arbres d'un graphe de chaînes. Ceci n'est pas une restriction puisqu'il existe toujours une écriture équivalente qui ne les utilise pas.

VII. SYSTEME-Q SIMPLIFIE

Un système-q simplifié est composé d'une ensemble de règles simplifiées de la forme

$a_1 + a_2 + ... + a_p == b_1 + b_2 + ... + b_q$.

Les a_i et les b_i sont des arbres quelconques. Les chaînes $a_1 + a_2 + ... + a_p$ et $b_1 + b_2 + ... + b_q$ sont respectivement le membre gauche et le membre droit de la règle. Le double signe égal se lit ''se réécrit en'' et le point marque la fin de la règle.

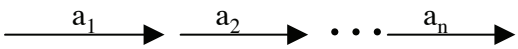
Cet ensemble de règles s'applique sur un graphe de chaînes donné et le transforme en un nouveau graphe de chaînes. La transformation se fait en deux phases distinctes : tout d'abord addition d'un certain nombre de flèches au graphe, ensuite suppression de certaines flèches jugées inutiles.

A. Première phase : addition de flèches

On considère toute suite de flèches de la forme

$-i_0- a_1 -i_1-$
$-i_1- a_2 -i_2-$
.....
$-i_{p-1}- a_p -i_p-$

qui est contenue dans le graphe considéré, autrement dit on considère tout chemin



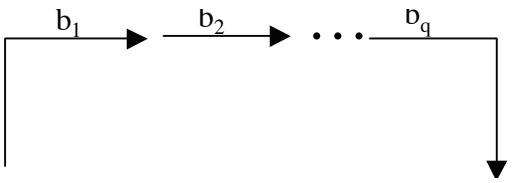
Si une règle de la forme

$a_1 + a_2 + ... + a_p = = b_1 + b_2 + ... + b_q$

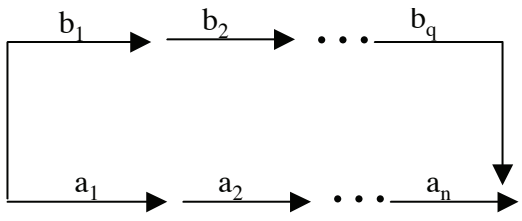
figure dans le système-q considéré, on l'applique sur ce chemin en ajoutant au graphe les flèches

$-j_0- b_1 -j_1-$
$-j_1- b_2 -j_2-$
.....
$-j_{q-1}- b_q -j_q-$

où $j_0 = i_0$ et $j_q = i_p$ et où les éventuels sommets $-j_1-$, $-j_2-$, ..., $-j_{q-1}-$ sont de nouveaux sommets. On ajoute donc au graphe le tout nouveau chemin



et on obtient



On prend soin de ne pas appliquer plusieurs fois la même règle sur le même chemin et on continue le processus jusqu'à ce qu'il n'y ait plus de règles à appliquer. Si le processus ne converge pas, le résultat est indéfini, sinon il est indépendant de l'ordre dans lequel on a appliqué les règles.

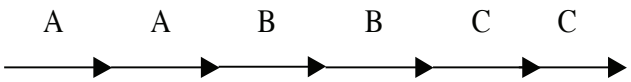
B. Deuxième phase : suppression de flèches

On supprime tout d'abord les flèches figurant dans des chemins sur lesquels s'est appliquée au moins une règle. On supprime ensuite les flèches qui ne figurent plus dans un chemin allant de l'entrée à la sortie du graphe.

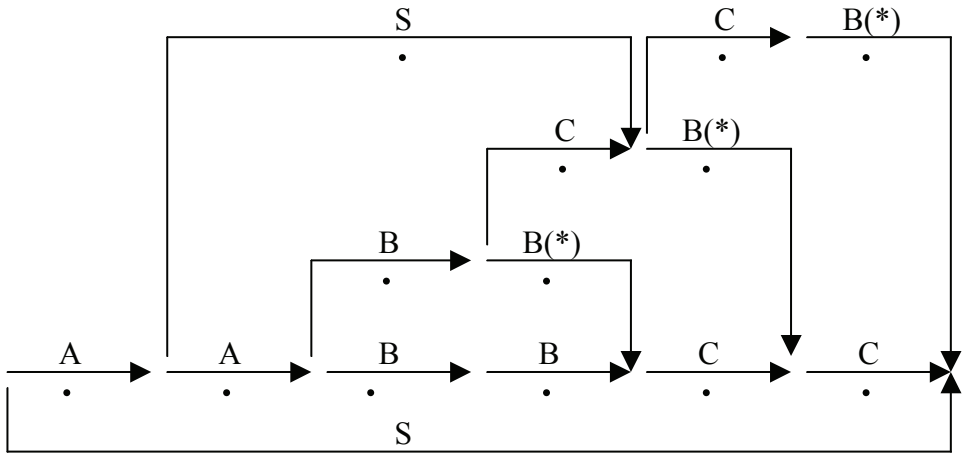
Soit par exemple à appliquer le système-q simplifié

```
A + B + C = = S.
A + S + B(*) + C = = S.
B(*) + C = = C + B(*).
B + B = = B + B(*).
```

sur le graphe de chaînes

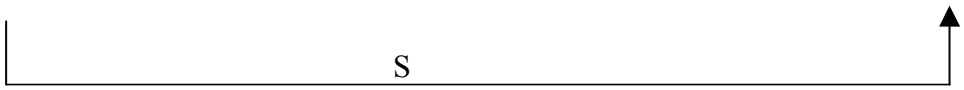


Après la première phase, on obtient



J'ai pris soin de marquer d'un point les flèches ayant servi à construire d'autres flèches.

Après la deuxième phase, il ne reste donc plus que



Autrement dit, ce système-q transforme

```
-0- A + A + B + B + C + C -1-
```

en

```
-0- S -1-
```

D'une façon plus générale, on peut montrer que l'on n'obtient

```
-0- S -1-
```

que si le graphe d'arbre initial est de la forme

```
-0- A + A + ... + A + B + B + ... + B + C + C + ... + C -1-
```

et que le nombre de A est le même que celui de B et que celui de C.

VIII. SYSTEME-Q

D'une façon générale, un système-q est une suite de règles et de commentaires. Bien entendu, l'absence ou la présence des commentaires ne change rien au système.

```
<système-q> ::= | < système-q> <règles> | <système-q> <commentaire>
<commentaire> ::= <texte> .
<texte> ::= ** | <texte> <caractère autre que point et égal>
<caractère autre que point et égal> ::= + | - | * | / | { | } | $ | ≠ | ,
                                      | <lettre> | <chiffre>
                                      | <signe non significatif>
```

Les règles peuvent contenir des paramètres et une condition. De plus, si une règle a une certaine partie identique à celle de la règle précédente, on peut omettre cette partie et la remplacer par un double trait d'union.

```
<règle> ::= <membre gauche> = = <membre droit> <condition>
<membre gauche> ::= <chaîne paramétrée> | --
<membre droit> ::= <chaîne paramétrée> | --
<chaîne paramétrée> ::= <arbre paramétré>
                      | <arbre paramétré> + <chaîne paramétrée>
<condition> ::= | / <expression booléenne>
<expression booléenne> ::= <secondaire booléen>
                         | <secondaire booléen> -OU- <expression booléenne>
<secondaire booléen> ::= <primaire booléen>
                       | <primaire booléen> -ET- <secondaire booléen>
<primaire booléen> ::= -- | -NON- <primaire booléen>
                     | (. <expression booléenne> .)
                     | <liste paramétrée> <relation> <liste paramétrée>
<relation> ::= = | ≠ | -DANS- | -HORS-
```

Un système-q est équivalent au système-q simplifié obtenu après avoir fait dans l'ordre les quatre transformations suivantes.

A. Élimination des commentaires

On supprime tous les commentaires

B. Élimination des doubles traits d'union

On parcourt les règles dans l'ordre dans lequel elles sont énumérées et on remplace chaque occurrence du double trait d'union

(a) par le membre gauche de la règle précédente s'il apparaît à la place d'un membre gauche

(b) par le membre droit de la règle précédente s'il apparaît à la place d'un membre droit

(c) par (. suivi de l'expression booléenne de la règle précédente suivie de .) s'il apparaît dans une condition.

De ceci il découle que la première règle ne peut contenir "--" et qu'une règle contenant "--" dans sa condition ne peut être précédée d'une règle ayant une condition vide.

C. Élimination des paramètres

On remplace chaque règle contenant des paramètres par l'infinité de règles obtenues en substituant à chaque paramètre, suivant sa nature, une étiquette, un arbre ou une liste quelconque. Dans cette infinité on considère que deux règles obtenues en substituant aux paramètres des choses différentes sont distinctes même si elles sont représentées par la même suite de caractères.

D. Élimination des conditions

On considère les règles ayant des conditions non vides. On supprime celles dont la condition n'est pas satisfaite et on supprime la condition des autres. Une condition est considérée comme satisfaite si et seulement si l'expression booléenne qui la représente a la valeur vraie. Pour évaluer cette expression booléenne, on considère que –OU-, -ET-, -NON- sont les opérations définies classiques et que =, ≠, -DANS-, -HORS- sont les relations définies à la section IV. Chaque fois que le parenthésage introduit par (. et .) ne précise pas l'ordre d'évaluation des différents composants de la condition, on évalue d'abord =, ≠, -HORS-, -DANS- ensuite -NON- ensuite -ET- et ensuite -OU-.

Pour des raisons d'efficacité, on impose que tout paramètre qui figure dans le membre droit ou dans la condition d'une règle doit aussi figurer dans son membre gauche.

Voici un exemple de système-q.

```
** EXEMPLE DE SYSTEME-Q.
A* + A*(U*) == A*(1,U*) / A* -DANS- A, B, C.
A(U*) + B(U*) + C(U*) == S(1, U*).
```

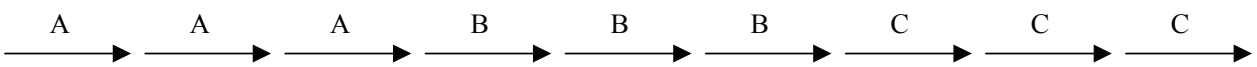
Il est équivalent au système-q simplifié qui contient notamment les règles

```
A + A == A(1).
A + A(1) == A(1).
A + A(1,1) == A(1,1).
.........................
B + B == B(1).
B + B(1) == B(1).
B + B(1,1) == B(1,1).
.....................
C + C == C(1).
C + C(1) == C(1).
C + C(1,1) == C(1,1).
.....................
A + B + C == S(1).
A(1) + B(1) + C(1) == S(1,1).
A(1,1) + B(1,1) + C(1,1) == S(1,1,1).
.....................................
```

Si on l'applique sur

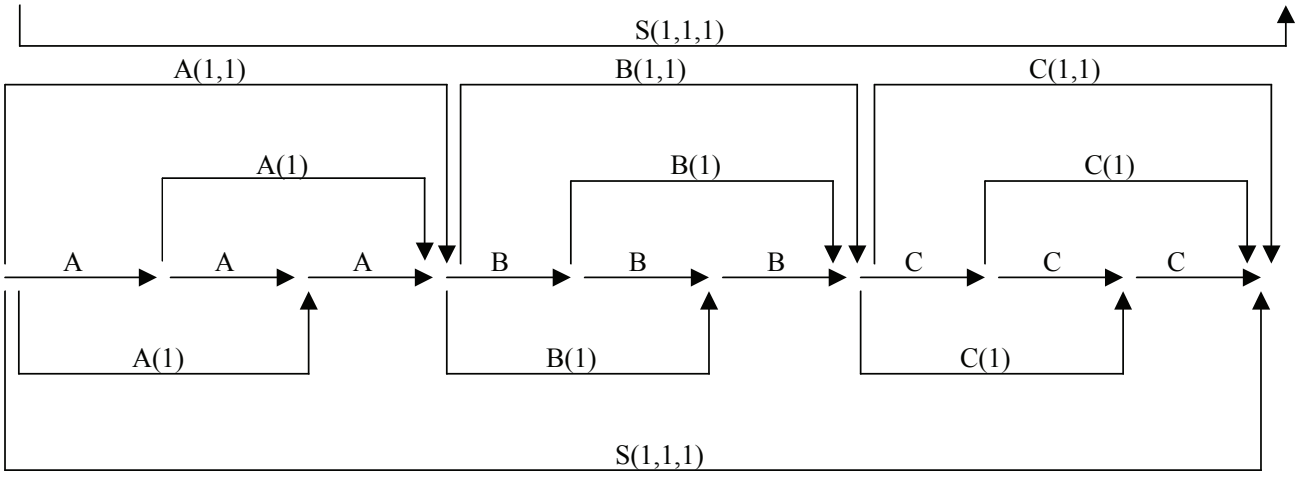
```
-0- A + A + A+ B + B + B + C + C + C –1-
```

c'est à dire



on obtient après la première phase

et après la deuxième



c'est-à-dire

```
-0- S(1,1,1) -1-
```

Voici maintenant quelques types de règles fort utiles.

La règle

```
A* == MOT($$A*).
```

permet de découper une étiquette en une liste de caractères et donc d'accéder aux éléments qui la composent. Appliquée à l'étiquette

```
VERCINGETORIX
```

elle produirait

```
MOT(V,E,R,C,I,N,G,E,T,O,R,I,X)
```

La règle

```
MOT($$A*) == A*.
```

au contraire est utilisée pour faire l'opération inverse, c'est-à-dire composer une étiquette à partir d'une liste de caractères.

Les deux règles

```
A*(U*) == [(A*) + ](U*) / U* = -NUL- -ET- A* -HQRS- [,].
](I*,U*) == I* + ](U*).
```

sont utilisées pour décomposer un arbre en une chaîne « structurée ».

Appliquées à

```
A(B,C(A,E(F)))
```

elles produiraient

```
[(A) + B +[(C) + A + [(E) + F + ] + ] + ]
```

Inversement, les deux règles

```
[(A*) + ](U*) == A*(U*).
I* + ](U*) == ](I*,U*) / [,] –HORS- I*.
```

permettent de retrouver l'arbre représenté par une chaîne ainsi structurée.

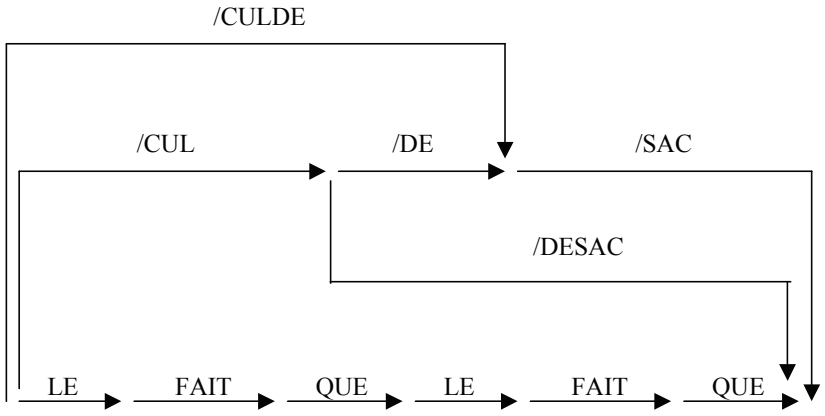
Si l'on veut supprimer d'un graphe de chaînes tout chemin portant une chaîne donnée, par exemple la chaîne

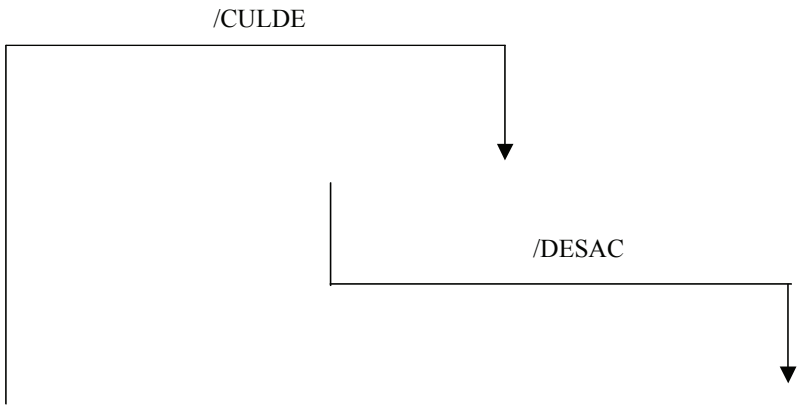
```
LE + FAIT + QUE + LE + FAIT + QUE
```

on peut utiliser les trois règles

```
LE + FAIT + QUE + LE + FAIT + QUE == /CUL + /DE + /SAC .
/CUL + /DE == /CULDE.            /DE + /SAC == /DESAC .
```

En effet, après la phase d'addition de flèches, on obtient





Après le début de la phrase d'élimination de flèches, il ne reste plus que :

et donc à la fin de cette dernière phase, il ne reste plus rien.

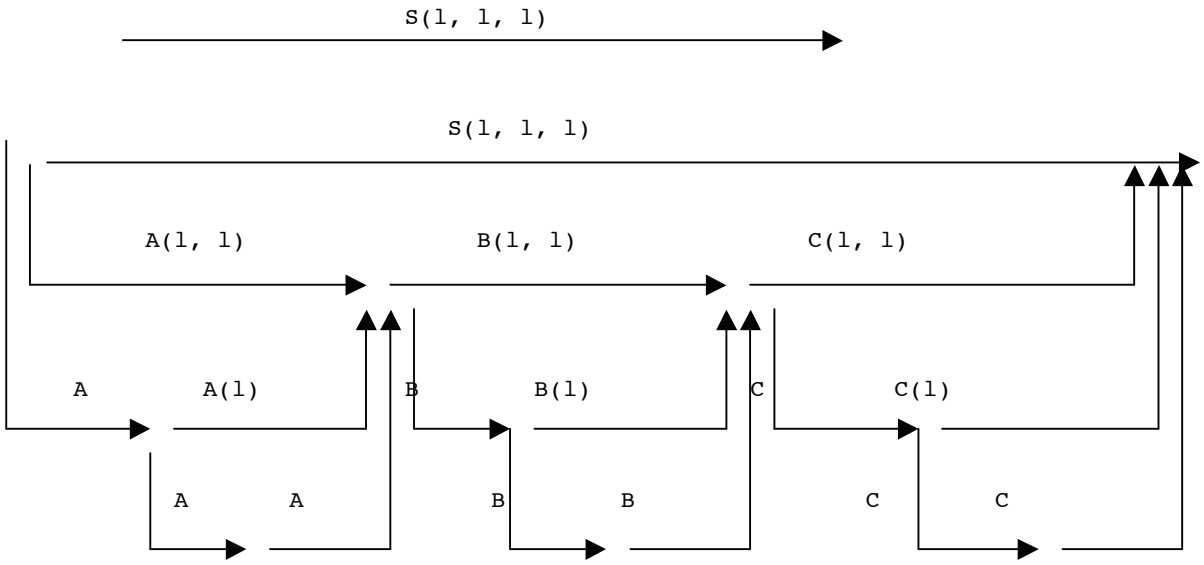
IX. REVERSIBILITE

Dans un système-q, les membres gauches et les membres droits de règles sont structurés exactement de la même façon. On peut donc se demander ce qui se produit si on les intervertit dans chaque règle. Le nouveau système-q ainsi obtenu permet en général de faire un traitement inverse à celui du système-q original : retrouver les données à partir des résultats.

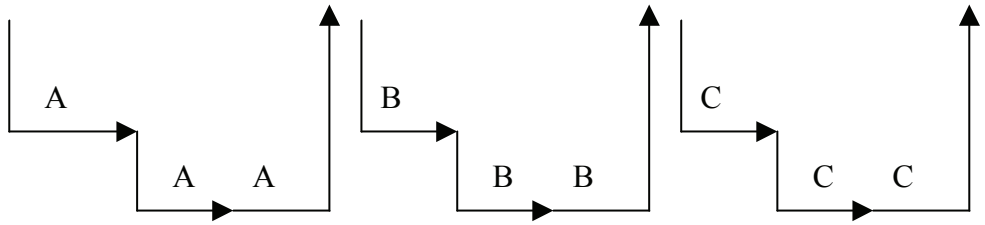
Par exemple, l'inverse du système-q donné au paragraphe précédent est

```
A*(l, U*) == A* + A*(U*) / A* -DANS- A,B,C.
S(l, U*)  == A(U*) + B(U*) + C(U*).
```

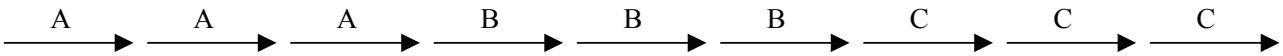
Appliqué au résultat



et après le deuxième



ce qui correspond bien au graphe



qui était la donnée du système-q originel.

En fait, il n'est pas nécessaire d'intervertir effectivement les membres gauches avec les membres droits. Il suffit de faire précéder le système-q considéré du mot -INV- pour qu'automatiquement il soit considéré comme inversé. Il faut noter qu'à ce moment la restriction sur les occurrences des paramètres se transforme en : tout paramètre figurant dans un membre gauche ou dans une condition doit aussi figurer dans le membre droit correspondant.

X. TRAITEMENT-Q

On est souvent amené à transformer un graphe de chaînes en plusieurs phases. Chaque phase consiste à appliquer un système-q éventuellement inversé. La première s'applique sur le graphe de départ et chaque autre phase s'applique sur le résultat de la phase précédente. J'appelle traitement-q une telle suite de phases.

Les phases d'un traitement-q sont écrites les unes à la suite des autres, dans l'ordre où l'on veut les appliquer. Le mot -REQ- sert à les séparer. Lors de l'exécution effective d'un traitement-q l'ordinateur imprime le ou les graphes qui lui sont fournis en données et les graphes obtenus après l'application de chaque phase. Si pour une phase donnée on veut les détails de l'application de chaque règle, il suffit de faire précéder cette phase du mot -DET-

```
<traitement-q> ::= <phase> | <phase> -REQ- <traitement-q>
<phase> ::= <détails> <inversion> <système-q>
<détails> ::= | -DET-
<inversion> ::= | -INV-
```

Si les données d'un système-q sont plusieurs graphes, il faut les séparer par le signe / .

```
<données d'un traitement-q> ::=
                  <graphe de chaînes>
                | <graphe de chaînes> / <données d'un traitement-q>
```

La section suivante est consacrée à un exemple de traitement-q en quatre phases.

XI. ANALYSE ET SYNTHESE DE PHRASES DU FRANÇAIS

Je présente ici un exemple de système-q permettant d'analyser et synthétiser un ensemble restreint mais cependant infini de phrases du français.

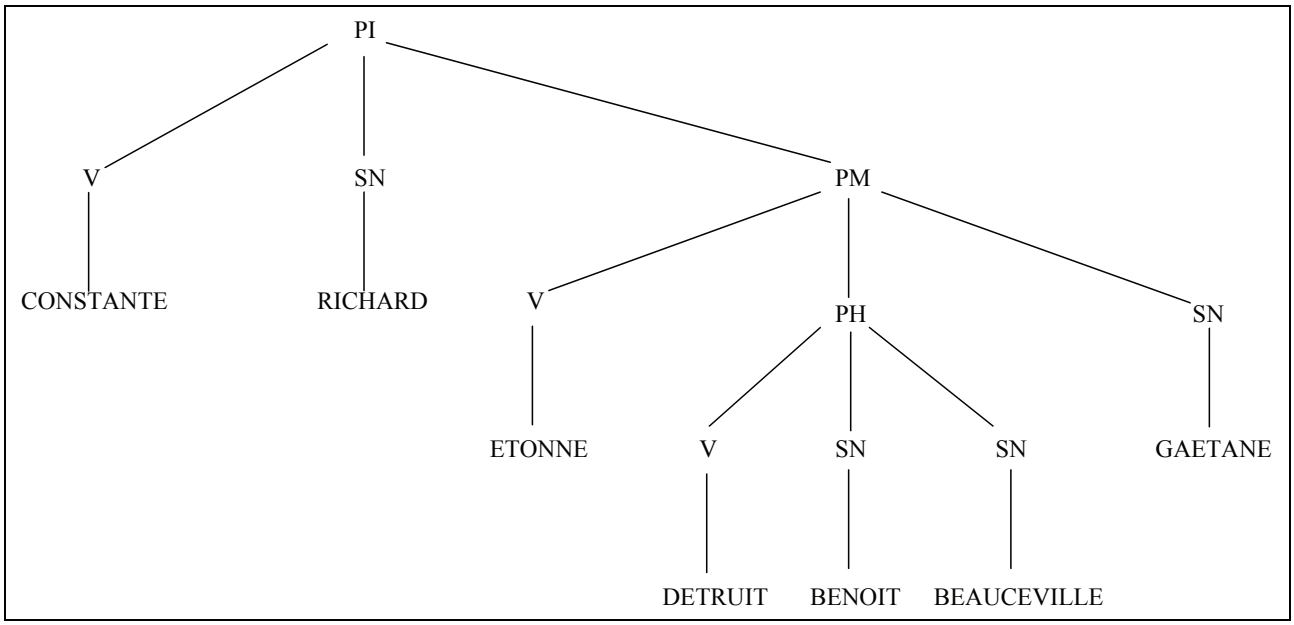
A chaque phrase correspond une phrase normalisée représentée par un arbre. A deux phrases ayant le même sens correspond la même phrase normalisée. Voici l'ensemble de toutes les phrases normalisées que je considère :

```
<phrase normalisée> ::= PI (<intérieur de phrase>)
<intérieur de phrase> ::= V (verbe), <argument>, <argument>
<argument> ::= <phrase> | SN (<nom propre>)
<phrase> ::= PH (<intérieur de phrase>)
<verbe> ::= DETRUIT | ETONNE | CONSTATE
<nom propre> ::= BEAUCEVILLE | BENOIT | GAETANE | RICHARD
```

Par exemple :

```
PI (V (CONSTATE), SN (RICHARD), PH (V (ETONNE), PH (V (DETRUIT), SN (BENOIT), SN
(BEAUCEVILLE)), SN (GAETANE)))
```

Ceci est une phrase normalisée. On peut la représenter par le schéma :



Le système-q qui suit permet de synthétiser un ensemble de paraphrases à partir d'une phrase normalisée de ce type. Ces paraphrases sont obtenues soit en choisissant la voix active, soit la voix passive, soit en nominalisant c'est à dire en transformant DETRUIRE en DESTRUCTION, ETONNE en ETONNEMENT et CONSTATE en CONSTATATION.

Le même système-q, mais inversé, permet d'analyser ces paraphrases et de retrouver la phrase normalisée qui leur correspond.

```
** FORME ACTIVE OU PASSIVE.
PI (V(A*), I*, J*)== SUJ + I* + V(A*) + OBJ + J*.
```

```
--   == SUJ + J* + EST + V(A*) + PPA + I*.

** NOMINALISATION D'UNE PHRASE.
PH(V(A*), I*, J*)== SN(V(A*))+ DDE + J* + PPA + I*.

** FORMES DEFINITIVES DES PREPOSITIONS.
SUJ + PH(U*)== LE + FAIT + QUE + PI(U* ).
OBJ + PH(U*)==  QUE + PI(U* ).
PPA + PH(U*) == PAR + LE + FAIT + QUE + PI(U* ).
DDE + PH(U*) == DE + CE + QUE + PI(U* ).

SUJ + SN(I*)== N(I*).
OBJ + SN(I*)== N(I*).
PPA + SN(I*)== PAR + N(I*).
DDE + SN(I*)== DE + N(I*).

** NOMS CONCRETS.
N(A*)== A* / A* - DANS - BEAUCEVILLE, BENOIT, GAETANE, RICHARD.

** NOMS ABSTRAITS.
N(V(DETRUIT))== LA + DESTRUCTION.
N(V(ETONNE))== L + ETONNEMENT.
N(V(CONSTATE))== LA + CONSTATATION.

** VERBES.
N( A* ) = = A* / A* - DANS – DETRUIT, ETONNE, CONSTATE.
```

Ce système-q est utilisé dans un traitement-q qui engendre toutes les paraphrases d'une phrase donnée. Ce traitement-q est composé de quatre phrases : une phrase d'analyse, une phrase de synthèse et deux phrases composées chacune d'une seule règle pour sortir les résultats en clair.

```
-INV- ** PHRASE D'ANALYSE.
            système-q précédent

-REQ- ** PHRASE DE SYNTHESE.
            système-q précédent

-REQ- A* + .(U*) = = .(A*, U*).
-REQ- .(A*, U*) = = A* + .(U*).
```

Par exemple, le graphe de chaînes qui représente une phrase

```
-0- RICHARD + CONSTATE + LA + DESTRUCTION + DE +
BEAUCEVILLE + PAR + BENOIT + ETONNE + GAETANE + . -1-
```

est transformé après la première phrase en

```
-O1- PI(V(CONSTATE)), SN(RICHARD), PH(V(ETONNE), PH(V(DETRUIT),
SN(BENOIT), SN(BEAUCEVILLE), SN(GAETANE))) + . -O2-
```

après la deuxième en

```
-O1- L + ETONNEMENT + DE + GAETANE -O2-
-O2- PAR + LA + DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT -O3-
-O2- PAR + LE + FAIT + QUE + -O4-
-O4- BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT -O3-
-O4- BENOIT + DETRUIT + BEAUCEVILLE -O3-
-O1- LE + FAIT + QUE -O5-
-05- GAETANE + EST + ETONNE -06-
-06- PAR + LA DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT -03-
-06- PAR + LE + FAIT + QUE -07-
-07- BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT -03-
-07- BENOIT + DETRUIT + BEAUCEVILLE -03-
-05- LA + DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT -08-
-05- LE + FAIT + QUE -09-
-09- BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT -08-
-09- BENOIT + DETRUIT + BEAUCEVILLE -08-
-08- ETONNE + GAETANE -03-
-03- EST + CONSTATE + PAR + RICHARD -10-
-01- RICHARD + CONSTATE -11-
-11- L + ETONNEMENT + DE + GAETANE -12-
-12- PAR + LA + DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT -10-
-12- PAR + LE + FAIT + QUE -13-
-13- BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT -10-
-13- BENOIT + DETRUIT + BEAUCEVILLE -10-
-11- QUE -14-
-14- GAETANE + EST + ETONNE -15-
-15- PAR + LA + DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT -10-
-15- PAR + LE + FAIT + QUE -16-
-16- BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT -10-
-16- BENOIT + DETRUIT + BEAUCEVILLE -10-
-14- LA + DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT -17-
-14- LE + FAIT + QUE -18-
-18- BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT -17-
-18- BENOIT + DETRUIT + BEAUCEVILLE -17-
-17- ETONNE + GAETANE -10-
-10- . -19-
```

et après les deux dernières phases en

```
-01- RICHARD + CONSTATE + QUE + LE + FAIT + QUE + BENOIT + DETRUIT + BEAUCEVILLE +
ETONNE + GAETANE + . -02-
-01- RICHARD + CONSTATE + QUE + LE + FAIT + QUE + BEAUCEVILLE + EST + DETRUIT + PAR
+ BENOIT + ETONNE + GAETANE + . -02-
-01- RICHARD + CONSTATE + QUE + LA + DESTRUCTION + DE + BEAUCEVILLE + PAR + BENOIT +
ETONNE + GAETANE + . -02-
-01- RICHARD + CONSTATE + QUE + GAETANE + EST + ETONNE + PAR + LE + FAIT + QUE +
BENOIT + DETRUIT + BEAUCEVILLE  + . -02-
-01- RICHARD + CONSTATE + QUE + GAETANE + EST + ETONNE + PAR + LE + FAIT + QUE +
BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT + . -02-
-01- RICHARD + CONSTATE + QUE + GAETANE + EST + ETONNE + PAR + LA + DESTRUCTION + DE
+ BEAUCEVILLE + PAR + BENOIT + . -02-
-01- RICHARD + CONSTATE + L + ETONNEMENT + DE + GAETANE + PAR + LE + FAIT + QUE +
BENOIT + DETRUIT + BEAUCEVILLE + . -02-
-01- RICHARD + CONSTATE + L + ETONNEMENT + DE + GAETANE + PAR + LE + FAIT + QUE +
BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT + . -02-
```

```
-01- RICHARD + CONSTATE + L + ETONNEMENT + DE + GAETANE + PAR + LA + DESTRUCTION +
DE + BEAUCEVILLE + PAR + BENOIT + . -02-
-01- LE + FAIT + QUE + LE + FAIT + QUE + BENOIT + DETRUIT + BEAUCEVILLE + ETONNE +
GAETANE + EST + CONSTATE + PAR + RICHARD + . -02-


-01- LE + FAIT + QUE  + LE + FAIT + QUE + BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT
+ ETONNE + GAETANE + EST + CONSTATE + PAR + RICHARD + . -02-
-01- LE + FAIT + QUE  + LA + DESTRUCTION  + DE + BEAUCEVILLE + PAR + BENOIT + ETONNE
+ GAETANE + EST + CONSTATE + PAR + RICHARD + . -02-
-01- LE + FAIT + QUE  + GAETANE + EST +  ETONNE +  PAR + LE + FAIT + QUE + BENOIT +
DETRUIT + BEAUCEVILLE +  EST + CONSTATE + PAR + RICHARD + . -02-
-01- LE + FAIT + QUE  + GAETANE + EST +  ETONNE +  PAR + LE + FAIT + QUE +
BEAUCEVILLE + EST + DETRUIT + PAR + BENOIT + EST + CONSTATE + PAR + RICHARD + . -02-
-01- LE + FAIT + QUE  + GAETANE + EST +  ETONNE +  PAR + LA + DESTRUCTION  + DE +
BEAUCEVILLE + PAR + BENOIT + EST + CONSTATE + PAR + RICHARD + . -02-
-01- L + ETONNEMENT + DE  + GAETANE + PAR + LE + FAIT + QUE + BENOIT + DETRUIT +
BEAUCEVILLE + EST + CONSTATE + PAR + RICHARD + . -02-
-01- L + ETONNEMENT + DE  + GAETANE + PAR + LE + FAIT + QUE + BEAUCEVILLE + EST +
DETRUIT + PAR + BENOIT + EST + CONSTATE + PAR + RICHARD + . -02-
-01- L + ETONNEMENT + DE  + GAETANE + PAR + LA + DESTRUCTION  + DE + BEAUCEVILLE +
PAR + BENOIT + EST + CONSTATE + PAR + RICHARD + . -02-
```

APPENDICE : ÉVOLUTION EN 1970-71

Depuis septembre 1970, deux changements mineurs ont été apportés à la définition des systèmes-q :

1) le caractère ":" n'existe plus ; il a été remplacé par le caractère "%".

2) lors de l'exécution effective d'un traitement-q, l'ordinateur n'imprime pas les graphes obtenus après l'application des phases qui sont précédées du mot –SUP-.

On pourrait redéfinir la syntaxe d'une phase comme suit :

```
Phase:  Impression Inversion SystemeQ ;
Impression:  | "-DET-" | "-SUP-" ;
Inversion:  | "-INV-" ;
```

Aussi, depuis septembre 1971, nous utilisons des programmes écrits en COMPASS pour exécuter efficacement les systèmes-q. Les remarques ci-dessus concernent la version COMPASS.

Gilles Stewart
Groupe de recherches pour la Traduction Automatique

REGLES DE SYNTAXE

A. Première forme : syntaxe en forme de Backus (BNF)

```
<données d'un traitement-q> ::= <graphe de chaîne> |
              <graphe de chaîne> / <données d'un traitement-q>

<graphe de chaîne> ::= <chemin sans carrefour> |
      <chemin sans carrefour> <graphe de chaîne>

<chemin sans carrefour> ::= <sommet> <chaîne> <sommet>

<sommet> ::= - <entier> -
<entier> ::= <chiffre> | <chiffre> <entier>

<chaîne> ::= <arbre> | <arbre> + <chaîne>

<arbre> ::= <étiquette> | <étiquette> ( <liste> )
<liste> ::= <liste simple> | <liste simple> , <liste>
<liste> ::= <arbre>

<traitement-q> ::= <phrase> |  <phrase> -REQ- <traitement-q>
<phrase> ::= <détails> <inversion> <système-q>
<détails> ::= | -DET-
<inversion> ::= | -INV-

<système-q> ::=   | <système-q> <règles> | <système-q> <commentaire>
<commentaire> ::= <texte>
<texte> ::= ** |  <texte> <caractère autre que point et égal>
<caractère autre que point et égal> ::= + | - | * | / | ( | ) | $ | = | , | <lettre>
| <chiffre> | <signe non significatif>
```

B. Seconde forme : syntaxe en format PCCTS

Cette forme est utilisable directement par PCCTS, un outil de type LEX+YACC acceptant des expressions régulières en parties droites et bien d'autres extensions. Il faut transformer un peu ce qui précède, et en particulier éliminer la récursivité à gauche.

```
DonneesQ:
    GrapheDeChaines
  | GrapheDeChaines "/" DonneesQ
  ;

GrapheDeChaines:
    CheminSimple
  | CheminSimple GrapheDeChaines
  ;

CheminSimple:
    Sommet Chaine Sommet
  ;

Sommet:
    "-" Entier "-"
  ;

Chaine:
    Arbre
  |  Arbre "+" Chaine
  ;

Arbre:
    Etiquette
  | Etiquette "(" Liste ")"
  ;

Liste:
    ListeSimple
  | ListeSimple Liste
  ;


TraitementsQ:
    Phase
  | Phase "-REQ-" TraitementsQ
  ;

Phase:
  Details Inversion SystemeQ
  ;

Details:
  | "-DET-"
  ;
```

```
Inversion:
  | "-INV-"
  ;

SystemeQ:
  | SystemeQ Regle
  | SystemeQ Commentaire
  ;

Commentaire:
    Texte "."
  ;

Texte:
    "**"
  | Texte
    CaractereNonPointOuEgal
  ;
   /* Pb pour le caractère
      "différent de" */

CaractereNonPointOuEgal:
  | "+" | "-" | "*" | "/" | "("
  | ")" | "$" | "#" | ","
  | Lettre
  | Chiffre
  | SigneNonSignificatif
  ;



Regle:
   MembreGauche "==" MembreDroit
          Condition "."
  ;

MembreDroit:
    ChaineParametree
  | "--"
  ;

 MembreGauche:
    ChaineParametree
  | "--"
  ;

ChaineParametree:
```

```
    ArbreParametre
  | ArbreParametre "+"
        ChaineParametree
  ;

Condition:
  | "/" ExpressionBooleenne
  ;

ExpressionBooleenne:
    SecondaireBooleen
  | SecondaireBooleen "-OU-"
        ExpressionBooleenne
  ;

SecondaireBooleen:
    PrimaireBooleen
  | PrimaireBooleen "-ET-"
        SecondaireBooleen
  ;

PrimaireBooleen:
    "--"
  | "-NON-" PrimaireBooleen
  | "(." ExpressionBooleenne ".)"
  | ListeParametree Relation
        ListeParametree
  ;

Relation:
    "=" | "#" | "-DANS-" | "-HORS-"
  ;

ArbreParametre:
    ParametreArbre
  | EtiquetteParametree
  | EtiquetteParametree "("
        ListeParametree ")"
  ;

ListeParametree:
    ListeSimpleParametree
  | ListeSimpleParametree ","
        ListeParametree
  ;

ListeSimpleParametree:
    ParametreListe
  | "-NUL-"
  | ArbreParametre
  | "$$" EtiquetteParametree
  ;

EtiquetteParametree:
    ParametreEtiquette
  | Etiquette
  ;

ParametreEtiquette:
    Abcdef "*" Indice
  ;

ParametreArbre:
    Ijklmn "*" Indice
  ;

ParametreListe:
    Uvwxyz "*" Indice
  ;

Indice:
  | Chiffre
  ;

Etiquette:
    Caractere
  | Etiquette Lettre
  | Etiquette Chiffre
  ;

Caractere:
    Lettre
  | Chiffre
  | SigneSignificatif
  | SigneNonSignificatif
  ;
```

```
Lettre:
    "A" | "B" | "C" | "D" | "E"
  | "F" | "G" | "H" | "I" | "J"
  | "K" | "L" | "M" | "N" | "O"
  | "P" | "Q" | "R" | "S" | "T"
  | "U" | "V" | "W" | "X" | "Y"
  | "Z"
  ;

Chiffre:
    "0" | "1" | "2" | "3" | "4"
  | "5" | "6" | "7" | "8" | "9"
  ;
```

```
SigneSignificatif:
    "+" | "-" | "*" | "/" | "("
  | ")" | "$" | "=" | "#" | "."
  | ","
  ;

SigneNonSignificatif:
    "~" | "[" | "]" | ":" | "<"
  | ">" | "^" | "%" | "£" | "@"
  | "•" | ";"
  ;
```

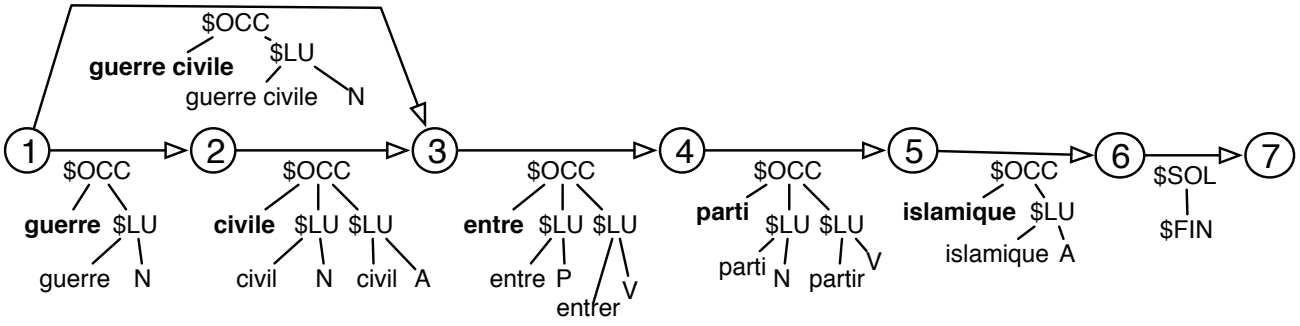
(Ch.Boitet & JC. Durand, 2002)

REGLES-Q UTILES POUR MANIPULER LE FORMAT D'ECHANGE DU PROJET OMNIA

Cette partie a été ajoutée au document initial par David Rouquet.

Nous donnons ici deux exemples montrant comment une simple règle-Q permet de faire des manipulations utiles sur la forme du format d'échange utilisé dans le système OMNIA.

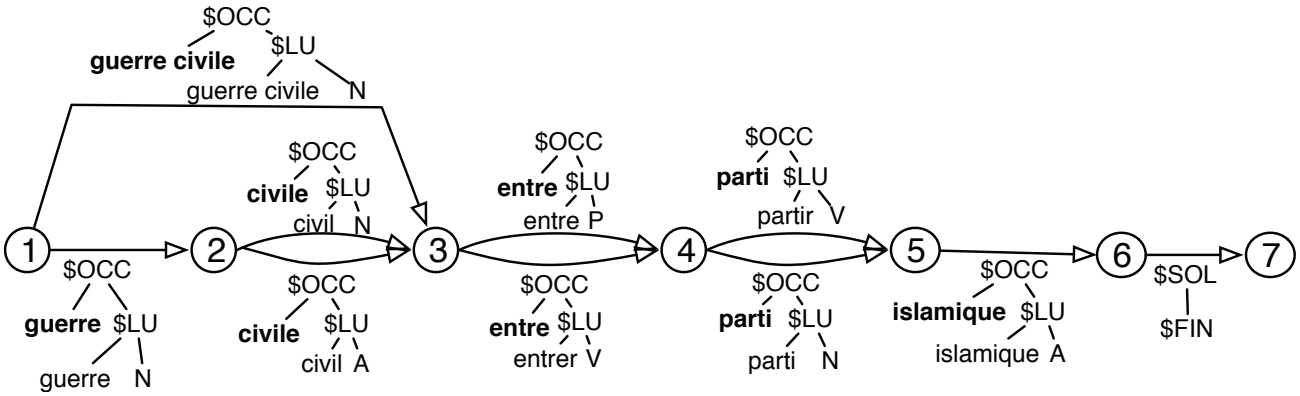
Considérons le graphe-Q suivant, qui permet de représenter le morceau de phrase « *[...]guerre civile entre parti islamique[...]* » dans le système OMNIA. Les arcs représentent les différentes segmentations possibles de la phrase. Pour chaque segment, les différentes unités lexicales ($LU) possibles pour une occurrence ($OCC) sont rassemblées dans l'arbre porté par l'arc. On ajoute à la fin un arc portant l'arbre $SOL($FIN) pour marquer la fin du fragment de phrase traité.



Si l'on souhaite que les arcs ne portent qu'une interprétation possible pour un segment (c'est à dire une seule unité lexicale), on peut appliquer un système-Q formé de la seule règle :

```
$OCC(A*,U*,$LU(V*),W*) == $OCC(A*,$LU(V*)) / $LU –DANS- U*,V*.
```

qui produira le graphe suivant :

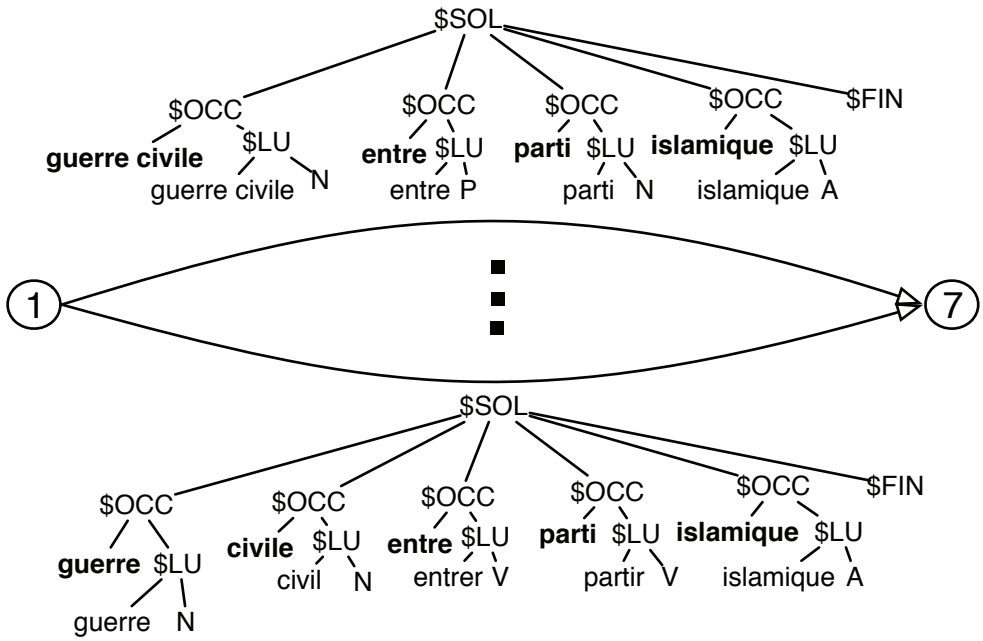


Il est maintenant possible de créer un arc pour chaque chemin interprétatif allant du nœud 1 au nœud 6.

Pour cela, nous appliquons maintenant la règle :

```
I* + $SOL(U*) == $SOL(I*,U*).
```

Nous obtenons le graphe suivant :

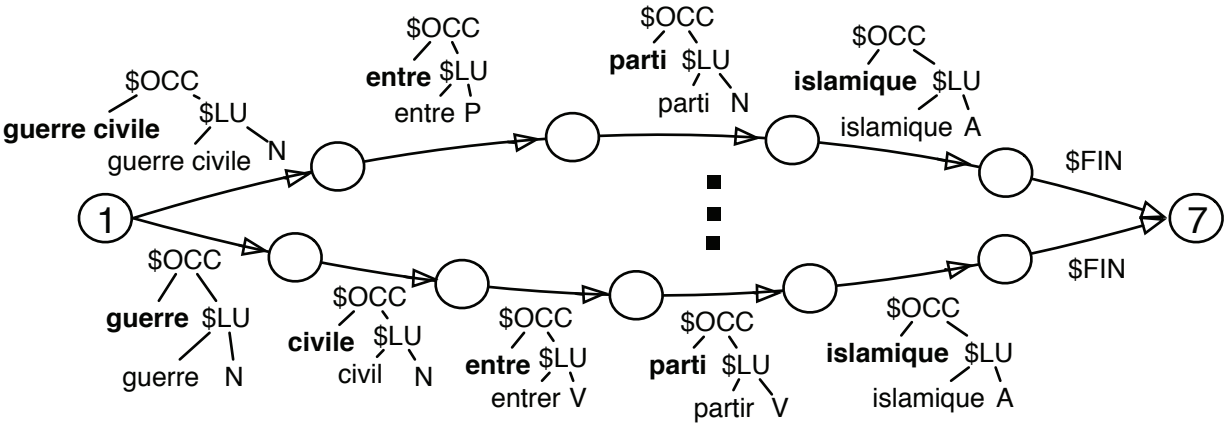


dont chaque arc représente un chemin interprétatif possible. Seulement deux de ces chemins figurent sur le graphique pour des raisons de place et de lisibilité.

Si nous désirons maintenant produire toutes les homophrases, avec un arc pour chaque unité lexicale, nous pouvons appliquer un nouveau traitement-Q composé de deux règles :

```
$SOL(I*,J*,U*) == I* + $SOL(J*,U*).
$SOL($FIN) == $FIN.
```

Nous obtenons alors le graphe suivant dans lequel les nœuds sans branchement sont restés anonymes :



Nous avons vu ici que l'utilisation du langage de règles spécialisé pour la manipulation des graphes-Q permet de réaliser très simplement des manipulations utiles sur le form at d'échange utilisé dans le système OMNIA.

En résumé, cette manipulation complexe est réalisée par le traitement-Q suivant :

```
-REQ-     $OCC(A*,U*,$LU(V*),W*) == $OCC(A*,$LU(V*)) / $LU –DANS- U*,V*.
-REQ-     I* + $SOL(U*) == $SOL(I*,U*).
-REQ-     $SOL(I*,J*,U*) == I* + $SOL(J*,U*).
          $SOL($FIN) == $FIN.
```