Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue

Mathieu Mangeot

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

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

Discipline : INFORMATIQUE

présentée et soutenue publiquement

par

M. Mathieu MANGEOT-LEREBOURS

le 27 septembre 2001

ENVIRONNEMENTS CENTRALISÉS ET DISTRIBUÉS POUR LEXICOGRAPHES ET LEXICOLOGUES EN CONTEXTE MULTILINGUE

JURY

Président	Laurent TRILLING
Rapporteurs	Jacques CHAUCHÉ
	Laurent ROMARY
Examinateurs	Frédéric ANDRÈS
	Jean-Pierre CHANOD
	Alain POLGUÈRE
Directeur de thèse	Christian BOITET
Co-directeur	Gilles SÉRASSET

Thèse préparée au sein des laboratoires GETA-CLIPS (IMAG, CNRS & UJF) et XRCE

Remerciements

En premier lieu, je remercie Christian Boitet, mon directeur de thèse qui m'a toujours soutenu, motivé et encouragé sans faillir et ce, depuis mon arrivée au GETA en septembre 1996. Les coups de gueule justifiés poussés avec tact et psychologie m'ont aidé à rebondir dans les moments difficiles. Christian m'a surtout donné goût à la recherche et au monde des langues.

Je remercie aussi Gilles Sérasset, mon codirecteur avec lequel j'ai beaucoup appris sur le plan technique et plus généralement sur une certaine philosophie de la recherche que je partage majoritairement. Kalimero a tous comptes faits bien apprécié de partager le bureau de Grincheux.

Jean-Pierre Chanod a su convaincre XEROX de l'intérêt de mon travail. Qu'il soit donc remercié tant pour la bourse CIFRE qui en a résulté que pour son accueil et son soutien constant.

Je remercie aussi Marie-Hélène Corréard pour m'avoir mis le pied à l'étrier dans le monde des dictionnaires. La collaboration entre un informaticien et une lexicographe fut très fructueuse. Dommage qu'elle ait dû quitter XEROX au milieu de mon gué.

Je suis très honoré que Laurent Romary, spécialiste reconnu de l'utilisation de XML pour le traitement des langues naturelles comme en témoignent sa participation très active aux projets SILFIDE, DHYDRO et SALT, ait volontiers accepté de rapporter sur ma thèse. Ses conseils et ses pointeurs ont d'ailleurs été très profitables pour mon travail.

Je tiens à remercier Jacques Chauché, spécialiste du TALN et très intéressé par le problème des ressources lexicales d'avoir volontiers accepté de rapporter sur ma thèse et de ses remarques pertinentes.

J'ai bien connu et apprécié Laurent Trilling en tant que professeur depuis mon année de licence et responsable des échanges avec l'étranger. Il a considérablement œuvré à l'IRISA pour le TALN et maintenant au LSR pour l'IA. Je suis donc très heureux qu'il ait accepté de présider mon jury de thèse.

Je suis également très heureux qu'Alain Polguère, linguiste renommé, ait accepté de participer à mon jury comme examinateur. Le projet Papillon, sujet de la dernière partie de ce manuscrit est en grande partie fondé sur ses travaux sur la lexicologie explicative et combinatoire.

Je voudrais aussi remercier Frédéric Andrès, spécialiste des bases de données multimédia, pour avoir accepté de participer à mon jury et m'avoir invité à Tokyo pour y passer les deux prochaines années dans son laboratoire et me permettre ainsi de continuer mes recherches dans le cadre du projet Papillon.

Je pense aussi à mes autres collègues du GETA et de l'équipe MLTT et plus généralement des laboratoires CLIPS et XRCE, pour tous les bons moments passés en leur compagnie.

Enfin, je suis reconnaissant envers ma famille et mes amis pour avoir subi et accepté les "mauvais côtés" de la vie du thésard, surtout le stress intense, et l'indisponibilité mentale ; et pour m'avoir patiemment réconforté.

みっちゃん：どうも　ありがとう　ございました。

Jobb egy mentő ötlet mint öt mentő egylet.

Table des matières

Table des figures

Introduction

Situation et motivations

Partout dans le monde, les centres de recherche publics et privés en traitement automatique des langues naturelles (TALN) accumulent de plus en plus de ressources lexicales de formats hétérogènes pour les besoins de leurs diverses applications. Ces ressources sont difficiles à maintenir et à manipuler. Il faut souvent reconstruire de zéro un dictionnaire ad hoc pour chaque nouvelle application.

L'utilisation d'Internet favorise la communication entre individus. Ceux-ci ont, de ce fait, besoin de communiquer à travers des langues différentes et donc d'utiliser et aussi éventuellement de créer pour leurs propres besoins des dictionnaires à usage humain. Les serveurs Web proposant la consultation en ligne de dictionnaires se développent sur Internet pour répondre à cette demande mais, là aussi, leur grand nombre et la quasi impossibilité de configurer le résultat des requêtes lancées sur ces serveurs ou de modifier les données de ces serveurs freinent les utilisateurs.

La consultation de dictionnaires non plus imprimés mais accessibles sur ordinateur peut être considérablement enrichie grâce à des outils disponibles sur l'ordinateur. Il n'est maintenant plus nécessaire de se limiter à la recherche d'un article correspondant à un mot précis. Il est possible de faire des recherches multicritères avec prétraitement de la requête et d'obtenir plusieurs articles ordonnés selon un certain critère.

La démocratisation d'Internet, la baisse des prix des ordinateurs familiaux et le succès des discussions en ligne rendent maintenant possible le travail collaboratif des internautes depuis leur domicile. De plus, il est aussi envisageable grâce à l'esprit communautaire et libre d'Internet de trouver des contributeurs travaillant bénévolement pour le développement de ressources libres de droits.

Intérêt de notre travail

Le regroupement des ressources lexicales aux formats hétérogènes stockées localement ou à distance est une première étape indispensable à leur manipulation et à leur réutilisation. Est-il possible d'utiliser à la fois des ressources locales et distantes ? Faut-il convertir toutes les ressources dans un format commun au risque de perdre de l'information ou est-il possible d'utiliser directement des ressources de formats hétérogènes ? Quelles techniques peut-on utiliser pour convertir des ressources et les manipuler ?

Les utilisations des ressources lexicales peuvent être très variées. Les outils de TALN ont besoin de dictionnaires très précis où l'information est codée de manière explicite. Les humains utilisent des dictionnaires de manières très différentes selon qu'ils sont apprenants d'une langue, traducteurs, linguistes ou simple curieux. Est-il possible d'utiliser une même ressource lexicale pour répondre à des besoins très différents ? Quelles sont alors les contraintes que les ressources doivent respecter ? Est-il possible de consulter plusieurs ressources en même temps et de paramétrer le résultat des requêtes de consultation ? Comment enrichir et élargir la consultation des ressources en combinant plusieurs outils ?

La création de nouvelles ressources lexicales par plusieurs contributeurs travaillant en collaboration à travers Internet est très intéressante puisque ceux-ci ont des niveaux de compétences très variés. Un spécialiste d'une langue s'occupera des informations relatives à cette langue; un traducteur mettra en relation les termes des langues qu'il connaît; un locuteur fournira des exemples ou des idiomes dans sa langue, etc. De plus, ces ressources peuvent rester en constante évolution, s'enrichir continuellement et suivre les changements des langues.

Il faut donc, pour cela, concevoir des outils pour gérer les différents intervenants et leur niveau de compétences variés. Nous devons aussi proposer des outils permettant de contribuer facilement et directement en ligne à la construction de nouvelles ressources. Pour garantir une portabilité et une compatibilité avec un maximum d'outils existants et à venir, nous baserons nos définitions sur le standard XML et ses dérivés (Namespace, XLINK, XPointer, XPath, XSLT, Schemas, etc.).

Organisation de la thèse

Dans la première partie de ce document intitulée "Contexte actuel de la dictionnairique", nous exposerons certaines notions du domaine de la lexicographie computationnelle puis nous examinerons en détail plusieurs dictionnaires variés. Nous étudierons ensuite les applications de consultation de dictionnaires, des outils de manipulation de ressources et des méthodes de construction de nouveaux dictionnaires. Nous continuerons cette partie par une explication des standards relatifs aux dictionnaires qui nous ont paru intéressants pour la suite de nos travaux. Enfin, nous terminerons par l'étude de projets sur les dictionnaires basés sur ces standards.

Dans la seconde partie, intitulée "Exploration de nouvelles directions, bilan et cahier des charges d'un environnement avancé", nous explorons plusieurs directions de recherche sur la consultation et la construction de dictionnaires. Nous exposons d'abord nos travaux sur la consultation en ligne de ressources lexicales hétérogènes locales ou distantes. Ensuite, nous détaillerons deux méthodes de construction de dictionnaires : l'une "démocratique" et l'autre en ligne pour des dictionnaires avec des structures simples. Nous relatons l'exploration de plusieurs outils d'aide à la consultation comme des correcteurs orthographiques, des annoteurs de documents, des conjugueurs, des plug-ins, etc. Enfin, nous établissons le cahier des charges d'un environnement plus "générique" en tirant le bilan de nos expériences.

Dans la troisième partie intitulée "Spécification d'un environnement de gestion et consultation de bases lexicales et dictionnaires", nous spécifions et de définissons un environnement complet de manipulation, création et consultation de dictionnaires. Nous dressons d'abord la liste des spécifications de notre environnement provenant de nos diverses expériences. Nous détaillons ensuite le système de bases lexicales SUBLIM qui répond en grande partie à nos spécifications du point de vue de l'architecture interne de notre environnement tout en soulignant ses manques pour nos objectifs. Nous décrivons ensuite l'architecture interne qui reprend SUBLIM avec une notation XML puis son architecture générale en montrant quels outils il est possible d'utiliser. Enfin, nous détaillons les interactions des différents utilisateurs avec notre environnement pour consulter et construire des dictionnaires.

Dans la dernière partie intitulée "Application de notre environnement à Papillon, projet de de base lexicale multilingue sur Internet", nous appliquons nos outils sur un cas concret de construction d'une base lexicale multilingue pour le projet Papillon. Nous présentons d'abord l'historique, les buts et l'architecture générale du projet Papillon. Nous définissons ensuite les principes lexicologiques, puis nous présentons les spécifications externes ainsi que l'analyse générale et l'implémentation du projet. Nous terminons par des évaluations préliminaires au projet.

Problèmes particuliers intéressants

Dans cette thèse, nous identifierons certains problèmes durs tels que la structuration et la manipulation de données hétérogènes, la visualisation d'une grande quantité de données et la construction en coopération par des personnes aux compétences diverses.

Nous résoudrons séparément ces problèmes grâce à des expérimentations variées sur la consultation de ressources hétérogènes, l'enrichissement et personnalisation du résultat ainsi que la construction de ressources en coopération.

Nous serons ensuite en mesure de concevoir un environnement répondant à tous ces problèmes se plaçant au dessus des bases de données utilisées pour le stockage et intégrant un serveur pour la construction coopérative. Son noyau inclura un formalisme générique de définition des structures.

Nous appliquerons enfin cet environnement au projet Papillon de développement par des bénévoles sur Internet d'une base lexicale multilingue dont l'architecture est constituée d'un dictionnaire monolingue pour chaque langue et d'un dictionnaire pivot d'acceptions interlingues reliant les articles monolingues.

A : Contexte actuel de la "dictionnairique"

Introduction

Avant de commencer, il est utile de décrire le contexte actuel de la "dictionnairique", branche relativement jeune du TALN. Ce terme est un peu plus général que "lexicographie computationnelle" qui fait référence aux outils et techniques de construction de dictionnaires.

Nous commençons bien sûr par fixer notre technologie de base; nous illustrons la variété des contenus et des structures des "dictionnaires" en étudiant en détail le format, la structure et la présentation de plusieurs dictionnaires. Nous décrivons des dictionnaires monolingues, multilingues, à usage humain ou machinal, des dictionnaires à structures internes simples et d'autres beaucoup plus complexes comme le Dictionnaire Explicatif et Contemporain (DEC) d'Igor Mel'tchuk [Mel'tchuk84,88,92].

Nous nous plaçons ensuite du côté des utilisateurs de dictionnaires en étudiant plusieurs outils de consultation. Certains dictionnaires comme le Collins on-line ou le Oxford Hachette [Corréard94] sont consultables par des applications installées localement sur des postes de travail. D'autres sont accessibles via des serveurs Web sur Internet comme le site dictionary.com. Ces outils ont des limitations. Nous verrons les améliorations éventuelles que l'on pourrait effectuer.

Nous continuons cette partie en nous plaçant du côté des lexicographes et lexicologues qui construisent les dictionnaires. Nous étudions les méthodes de manipulation de dictionnaires décrites dans la thèse de Haï Doan-Nguyen [Doan-Nguyen98a]. Elles permettent d'une part de récupérer des dictionnaires, c'est à dire de les transformer de leur format d'origine vers un format plus facile à manipuler et d'autre part d'effectuer des opérations ensemblistes sur plusieurs dictionnaires.

Nous étudions ensuite plusieurs techniques de construction de dictionnaires. Certaines fonctionnent directement via le Web (le projet SAIKAM de dictionnaire japonais-thaï [SAIKAM]). D'autres utilisent des éditeurs de texte classiques comme Word™ (technique du dictionnaire français-anglais-malais [Gut96]), des éditeurs de documents structurés SGML (technique du dictionnaire bilingue canadien [Roberts99]) ou encore des éditeurs spécialisés conçus de façon ad hoc (DECID pour le DEC).

Ensuite, nous expliquons les standards principaux utilisés dans la représentation des dictionnaires comme SGML (Standard Generalized Markup Language), puis les standards actuels comme Unicode et XML (eXtended Markup Language) qui nous ont paru intéressants pour la suite de nos travaux.

Enfin, nous terminerons par une étude de travaux récents en lexicographie et terminologie basés sur XML comme les projet DHYDRO [Descotte00a,00b] et SALT [SALT].

1. Notions du domaine

1.1. Définition des termes utilisés dans la thèse

1.1.1. Introduction

Un *dictionnaire* est composé d'un ensemble de volumes. Un *volume* est un ensemble d'*articles* triés selon un ordre spécifique. La liste ordonnées de ces articles constitue la *nomenclature* du dictionnaire. L'ordre utilisé est généralement l'ordre alphabétique de la langue des *mots-vedettes*. Un *article* est composé d'un mot-vedette et d'un corps.

Un texte est composé de *mots*. Ces mots sont les *formes de surface* des *lemmes*. Les formes de surface ne correspondent pas toujours de façon évidente aux mots-vedettes du dictionnaire. Par exemple, un verbe n'est jamais indexé selon ses formes conjuguées (*vais*, *allais*,*irai*) mais selon l'infinitif (*aller*); un nom commun (*dictionnaires*,*pauvresse*,*actrices*) est pratiquement toujours indexé selon son singulier (*dictionnaire*, *pauvre*, *acteur*). Généralement, les mots-vedettes d'une langue donnée sont les lemmes correspondant aux formes de surface que l'on trouve dans les textes.

Un dictionnaire *terminologique* rassemble généralement des termes d'un domaine précis de la langue. Un dictionnaire *général* rassemble des mots-vedettes sans se spécialiser dans un domaine particulier et contient des informations assez riches et variées. Un *lexique* est un dictionnaire monolingue terminologique dont la microstructure est très simple.

Le terme de *ressource lexicale* est un terme plus large que celui de *dictionnaire*. Les ressources peuvent être de plusieurs natures : des dictionnaires, des lexiques, des corpus, des thésaurus, etc. Les ressources peuvent être utilisées soit par des *humains* soit par des *machines*.

1.1.2. La macrostructure des dictionnaires

L'organisation des volumes forme la *macrostructure* du dictionnaire. La figure A.1 représente les principaux exemples de macrostructures.

La macrostructure la plus simple consiste en un seul volume. Les mots-vedettes des dictionnaires composés d'un seul volume appartiennent à la même langue. La nomenclature de ces dictionnaires ne dépend que d'une seule langue. Ce sont des dictionnaires *monodirectionnels*. On trouve des dictionnaires monolingues mais aussi des dictionnaires multilingues indexés selon une seule langue. Ces derniers sont appelés dictionnaires multicibles ou furcoïdes [Boitet86a,86b].

Une macrostructure fréquemment utilisée est celle du dictionnaire bilingue en deux volumes, un volume trié selon les mots-vedettes d'une langue et donnant les traductions de ces mots-vedettes dans une autre langue, et un autre volume symétrique. Ce sont des dictionnaires bilingues *bidirectionnels*. Ces dictionnaires bilingues sont rarement bijectifs (à chaque mot-vedette d'une langue ne correspond pas une traduction et une seule dans l'autre langue) sauf certains lexiques terminologiques simplifiés. Par exemple, un dictionnaire bilingue bidirectionnel anglais-français est composé de deux volumes : un volume anglais->français et un volume français->anglais.

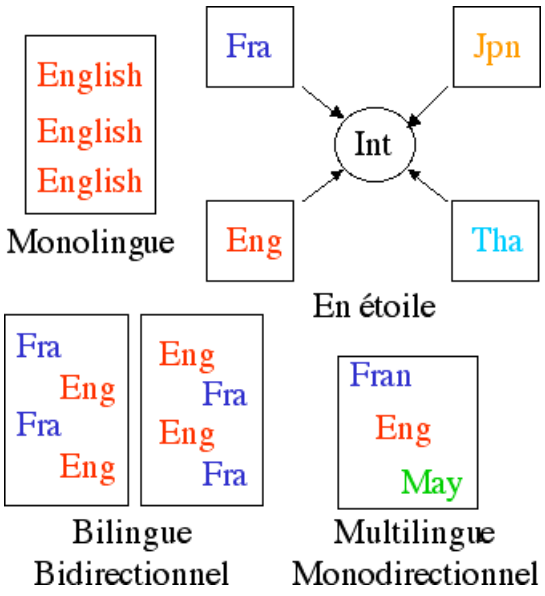


FIG. A.1 – *exemples de macrostructures*

Une macrostructure plus élaborée destinée aux *bases lexicales multilingues* consiste à organiser en étoile autour d'un dictionnaire central de concepts ou d'acceptions des dictionnaires monolingues contenant dans chaque langue de la base les traductions des concepts ou acceptions du dictionnaire central. Le dictionnaire central joue le rôle de *pivot* de la base. Ce sont des bases de *concepts* ou d'*acceptions* comme la base PARAX [Blanc95,97] développée au GETA ou le projet ULTRA dirigé par Yorick Wilks [Farwell92].

Dans une base de concepts, ceux-ci sont définis en premier. On cherche ensuite comment ils sont traduits dans chaque langue de la base. À l'inverse, dans une base d'acceptions, celles-ci sont au départ des liens de traduction entre deux ou plusieurs langues. Elles peuvent devenir des concepts lorsque la base a été suffisament complétée, équilibrée et raffinée.

1.1.3. La microstructure des dictionnaires

La structure logique de l'article forme la *microstructure* du dictionnaire. La microstructure varie beaucoup selon les dictionnaires. Elle peut être vue comme une structure composée d'*objets linguistiques*. Parmi ces objets, nous pouvons trouver : le mot-vedette, sa prononciation, les catégories grammaticales que peut avoir ce mot-vedette (nom, pronom, verbe, adjectif, adverbe, etc.), des définitions, des traductions, des exemples, des collocations, une étymologie, des sens, des gloses, des étiquettes (figuré, commerce, pharmacie, aéronautique, botanique, etc.), des régimes lexicaux, des fonctions lexicales, etc.

Un mot décrit dans un dictionnaire est appelé *vocable.* Les mots prennent très souvent plusieurs sens différents. Par exemple, le verbe `blanchir` a trois sens principaux : *blanchir des légumes*, *blanchir de l'argent sale* (blanchiment) et *blanchir un vêtement* (blanchissage). Un sens de mot est aussi appelé *lexie*. Au vocable `blanchir` correspond donc trois lexies. Le *contexte* d'un article est constitué des articles qui précèdent et suivent cet article selon la nomenclature du dictionnaire. Il est souvent très utile lors de la rédaction d'un article ou de sa lecture de pouvoir aussi accéder au contexte de cet article de façon à voir les articles suivants et précédents.

Une *acception* monolingue est une unité sémantique d'une langue. Une *base d'acceptions* fournit un lien entre les acceptions monolingues des différents dictionnaires. L'ensemble des *acceptions interlingues*

est l'union des ensembles des acceptions monolingues des différents dictionnaires de la base.

Un *concept* est une représentation abstraite mentale et générale. Une *ontologie* est un ensemble hiérarchisé de concepts, de faits et de règles qui représente une modélisation du monde. Une base de concepts possède donc un dictionnaire interlingue représentant une ontologie, contrairement à une base d'acceptions qui n'a pas de dictionnaire interlingue.

Ces objets linguistiques sont structurés de manière plus ou moins complexe. Les microstructures de certains dictionnaires se limitent à une suite de paires attribut-valeur. D'autres sont plus élaborées, et comprennent des arbres, des tableaux, des graphes, etc. Une microstructure fréquemment utilisée est celle de l'entrée sous forme d'arbre. L'avantage d'une telle structure est qu'elle est facilement représentable dans un format utilisant des balises. Il est aussi possible de représenter la plupart des microstructures à l'aide de structures de traits [Ide95a].

Lorsque la microstructure d'un article est représentée par des éléments spécifiques, c'est une structure *explicite*. Elle peut être interprétée par une machine et elle n'est pas ambiguë. Lorsqu'au contraire la microstructure n'est représentée que par des éléments de *présentation* de l'article, c'est une structure *implicite*. Elle ne peut être interprétée que par des humains; elle est souvent ambiguë.

1.1.4. Le format des données

Un dictionnaire peut revêtir plusieurs formes. Imprimé, c'est un *dictionnaire papier*. Représenté par des fichiers de caractères, c'est un *dictionnaire électronique*. Inclus dans un fichier binaire et utilisé par une application spécifique, c'est un *dictionnaire compilé*.

Historiquement, les premiers dictionnaires étaient exclusivement en format papier. Par la suite, pour simplifier la manipulation des données, celles-ci ont été stockées sous forme électronique. Dans un premier temps, seules les indications de typographie de la présentation étaient incluses dans ces données. Puis, petit à petit, une séparation s'est faite entre la structure logique du dictionnaire et sa présentation.

Les dictionnaires construits plus récemment sont stockés uniquement avec leur structure logique et les informations typographiques ont disparu. De plus, les dictionnaire ne sont plus élaborés seulement pour des humains. Des informations non pertinentes pour les humains mais cruciales pour des machines sont ajoutées dans la structure logique. Ces dictionnaires peuvent donc être utilisés directement par des machines.

Les dictionnaires compilés ne sont pas utilisables sans les informations nécessaires pour les décoder. Dans la suite, nous ne parlerons que de dictionnaires électroniques.

Chaque dictionnaire électronique est représenté dans les fichiers de caractères par un certain *format* particulier. Les formats sont différents selon les dictionnaires.

Le format le plus simple consiste à disposer chaque entrée sur une ligne. L'entrée est suivie éventuellement d'informations linguistiques séparées par un ou plusieurs caractères spéciaux.

Beaucoup de dictionnaires utilisant une microstructure sous forme d'arbre sont représentés dans un format utilisant des balises comme la norme SGML (Standard Generalized Markup Language) [ISO86] et plus récemment XML (eXtended Markup Language) [W3C98a] pour représenter les données. L'information textuelle est contenue entre une balise ouvrante et une balise fermante. L'ensemble est appelé un élément. Voici un exemple :

```
<headword number="1">abr&eacute;ger</headword>
```

L'élément se nomme <headword>. Des attributs peuvent être associés à l'élément. Ici, nous utilisons l'attribut number (numéro). Pour assurer la portabilité et la compatibilité d'un document encodé en SGML ou en XML, on utilise des entités pour représenter les caractères spéciaux. Dans l'exemple, le `"é"` est représenté par `"&eacute;"`.

Un élément peut englober d'autres éléments de façon à construire une structure d'arbre. Ici, l'élément <entry> englobe les éléments <headword> et <pos>.

```
<entry>
  <headword>abr&eacute;ger</headword>
  <pos>v.t.</pos>
</entry>
```

Le projet de la TEI (Text Encoding Initiative) [Ide95b,Johnson95] terminé en mai 94 avait pour but d'unifier la sémantique des balises SGML pour encoder les textes. Le groupe de travail sur les dictionnaires a notamment publié une Définition de Type de Document (DTD) générale pour encoder les dictionnaires.

Certains dictionnaires enfin utilisent directement un format de présentation pour le stockage. Ces formats ne reflètent pas directement la structure logique de ces dictionnaires. Il faut alors leur appliquer un traitement pour obtenir une structure logique plus directement utilisable. C'est le cas des formats RTF (Rich Text Format) et HTML (HyperText Markup Language) [HTML 4.0]. De plus, les traitement ne peuvent pas être totalement automatiques et sont très coûteux [Doan-Nguyen98a,98b].

1.1.5. La présentation des informations

La structure de présentation de l'article (polices, couleurs, tailles) est appelée *présentation* du dictionnaire. Cette structure est indépendante de la structure logique même si, en général, elle en reflète certaines parties. Un dictionnaire peut avoir plusieurs présentations différentes mais il n'aura toujours qu'une seule microstructure. De plus en plus de dictionnaires sont disponibles non seulement dans des version papier mais aussi sous forme d'applications sur ordinateur (par exemple le Collins ou le Hachette-Oxford) ou sur la Toile (par exemple, le FeM [FeM] ou le Websters [dictionary.com]) . À chaque version est associée une présentation différente, mais toutes les versions sont élaborées à partir de la même structure logique. C'est le cas du FeM (voir 1.2.1).

1.2. Exemples de dictionnaires à usage humain

1.2.1. Un dictionnaire monodirectionnel trilingue : le FeM

Introduction

Le dictionnaire français-malais "Kamus Perancis-Melayu Dewan" [Gut96], a été construit en coopération entre le service Culturel de l'Ambassade de France à Kuala Lumpur, le Dewan Bahasa dan Pustaka, l'Unit Terjemahan Melalui Komputer (Universiti Sains Malaysia, Penang) et le Groupe d'Étude pour la Traduction Automatique, GETA (Université Joseph Fourier, Grenoble &CNRS) sous la coordination de l'association Champollion.

C'est un dictionnaire trilingue monodirectionnel. Sa macrostructure est donc constituée d'un seul volume. C'est un dictionnaire à usage humain. Il comporte environ 20 000 articles et 50 000 sens de mots ou lexies. Dans sa version papier définitive, l'anglais a été supprimé, alors qu'il a été conservé dans les variantes électroniques.

La technique de construction de ce dictionnaire français-malais en s'aidant de l'anglais comme langue pivot a été reprise dans deux projets appelés Fe* : français-anglais-thaï et français-anglais- vietnamien. Ces dictionnaires sont en cours de construction.

Nous avons beaucoup utilisé le FeM dans nos expériences. De plus, sa microstructure est relativement simple. Au départ, l'anglais étant une langue pivot mais il y avait des doublons dans les traductions. En effet, un vocable français ayant deux lexies pouvait avoir une traduction anglaise distincte pour chaque lexie, puis ces deux traductions anglaises se traduire de la même façon en malais. Depuis 1995, le découpage des sens se fait selon la langue source, le français. La microstructure est donc en fourche, les langues cibles étant les branches de la fourche. Sa disponibilité et son originalité nous ont incité à le présenter ici.

Format interne du dictionnaire

La microstructure du dictionnaire est composée d'une suite de paires attributs/valeur. Les articles du dictionnaires étant principalement manipulés par des applications programmées en LISP [Steele90], le format interne du dictionnaire est une forme LISP très facile à analyser. Le dictionnaire est stocké dans un ou deux fichiers par lettre. La taille des fichiers varie de 25 à 500 kilooctets. La taille totale est de 6,8 mégaoctets. La figure A.2 montre un article au format original.

```
(:fem-entry
  (:ENTRY "abréger")
  (:FRENCH_PRON "abre-je-")
  (:FRENCH_CAT "v.tr.")
  (:FRENCH_GLOSS "un texte")
  (:ENGLISH_EQU "to shorten")
  (:ENGLISH_EQU "to abridge")
  (:MALAY_EQU "memendekkan")
  (:MALAY_EQU "meringkaskan")
)
```

FIG. A.2 – *l'article* `abréger` *du FeM au format original (LISP)*

Ce dictionnaire a été converti au format rtf pour être édité à l'aide du logiciel Word™. La figure A.3 montre le même article en format rtf.

```
/** en tête du fichier rtf **/
{\rtf1\mac \deff8\deflang1033{\fonttbl{\f0\froman\fcharset77\fprq2 Tms
Rmn;}
/** définition des polices **/
{\f1\fnil\fcharset2\fprq2 Symbol;}
..f2..f3.......f52..f53..
{\f54\fnil\fcharset77\fprq2 \'96\'7b\'96\'be\'92\'a9\'81|\'82l;}}
{\stylesheet{\widctlpar \f8\lang1036 \snext0 Normal;}
{\*\cs10 \additive Default Paragraph Font;}
/** définition des styles **/
{\s16\widctlpar \b\f8\fs28\ul\lang1036 \sbasedon15\snext15 french_entry;}
..s17..s18.....s57..s58..  {\s59\li3960\sb60\widctlpar \f8\cf2\lang1036
\sbasedon52\snext59 malay_pron;}}
/** fin de l'en-tête et début du fichier **/
\par \pard\plain \s16\widctlpar \b\f8\fs28\lang1036 abr\'8eger
\par \pard\plain \s28\widctlpar \caps\f3\lang1036 /abre-je-/
\par \pard\plain \s18\widctlpar \f8\lang1036 v.tr.  \par \pard\plain
\s21\widctlpar \i\f9\lang1036 (un texte)
\par \pard\plain \s34\widctlpar \b\f8\cf6\lang1036 to shorten
\par to abridge
\par \pard\plain \s20\widctlpar \b\f8\cf4\lang1036 memendekkan
\par meringkaskan
```

FIG. A.3 – *l'article* `abréger` *du FeM en format rtf source*

Ces deux articles sont équivalents. Ils contiennent exactement les mêmes informations. L'application

de conversion d'un format à l'autre est bijective et sans perte d'information. Ces exemples montrent que les formats internes peuvent être très différents et pourtant contenir les mêmes informations. Il est donc préférable de choisir comme format interne un format lisible par l'humain.

Présentation du dictionnaire

La figure A.4 montre un exemple de présentation du même article tel que l'utilisateur le voit.

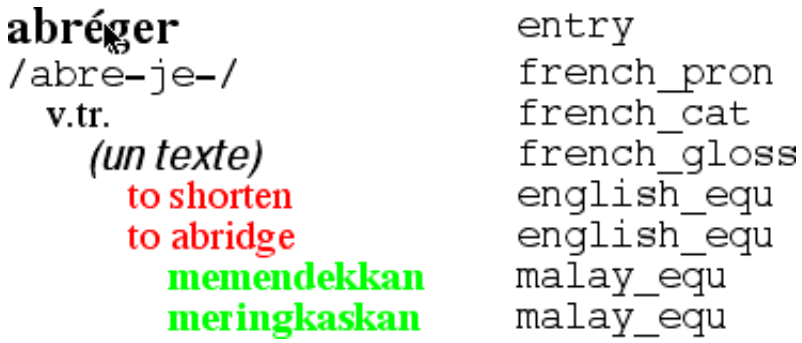


FIG. A.4 – *l'article* `abréger` *du FeM (avec indication des styles)*

Cette présentation est utilisée par les lexicographes lorsqu'ils rédigent les articles. La présentation utilisée dans le dictionnaire imprimé diffère. Cet exemple montre qu'il est possible d'associer différentes présentations au même format interne.

1.2.2. Un dictionnaire d'usage monolingue : le NODE

Introduction

Le New Oxford Dictionary of English [Pearsall98] est un dictionnaire monolingue anglais. Il a été publié par les presses universitaires d'Oxford (OUP) en 1998. C'est un dictionnaire à usage humain. Sa macrostructure consiste donc en un seul volume. Sa microstructure contient la plupart des éléments d'un dictionnaire monolingue classique parmi lesquels l'étymologie des entrées, des exemples et des informations encyclopédiques. Il contient environ 93 000 articles. Comme ce dictionnaire était disponible au XRCE à des fins de recherche, nous avons pu l'utiliser pour nos expériences.

Format interne du dictionnaire

Comme ce dictionnaire a été publié par un éditeur spécialisé, son format interne original est SGML, le format le plus répandu chez les éditeurs. Sa microstructure est représentée sous forme d'un arbre pour chaque article. Le dictionnaire est stocké dans un seul fichier d'une taille de 38 mégaoctets. Pour la prononciation, l'alphabet utilisé est l'Alphabet Phonétique International (API). Dans les exemples suivants, aucune police API n'était disponible. La pornonciation est représentée par une transcription.

Le tableau A.1 décrit les éléments SGML de l'exemple de la figure A.5.

Présentation du dictionnaire

La figure A.6 montre le même article dans une présentation lisible par l'humain.

élément	explication	traduction
<d>	date	date
<df>	definition	définition
<etym>	etymology	étymologie
<ex>	example	exemple
<f>	form	forme ou patron
<ff>	foreign form	forme étrangère
<fg>	form group	groupe de formes
<gg>	grammar group	groupe grammatical
<hg>	headword block	bloc du mot-vedette
<hw>	headword	mot-vedette
<la>	language	langue
<pr>	pronunciation	prononciation
<ph>	phonetic IPA	phonétique API
<ps>	part-of-speech	catégorie grammaticale
<s1>	sense n°1	sens n°1
<se>	standard entry	entrée standard
<sy>	syntax indicator	indicateur de syntaxe
<tr>	translation	traduction

TAB. A.1 – *éléments SGML du NODE*

```
<se><hg><hw>abbreviate</hw> <pr><ph>@"bri:vIeIt</ph></pr></hg>
<s1><ps>verb</ps> <gg>with <sy>obj.</sy></gg>
<fg>usu. <f>be abbreviated</f></fg><df>shorten (a word, phrase, or
text):</df>
<ex>the business of artists and repertoire, commonly abbreviated to A &amp;amp. R</ex> &ex.
<eg><gg>as <sy>adj.</sy></gg><fg><f>abbreviated</f></fg><ex>this book is
an abbreviated version of the earlier work.</ex></eg></s1>
<etym><d><la>late Middle English</la></d>: from <la>late
Latin</la> <ff>abbreviat-</ff> <tr>shortened</tr>, from the
verb <ff>abbreviare</ff>, from <la>Latin</la> <ff>brevis</ff>
<tr>short</tr>.</etym>
</se>
```

FIG. A.5 – *l'article* `abbreviate` *du NODE en format original (SGML)*

1.2.3. Un dictionnaire d'usage bilingue : le DHO

Introduction

Le dictionnaire Hachette-Oxford [Corréard94] est un dictionnaire bilingue anglais-français à usage humain. Il a été publié conjointement par Hachette et les presses universitaires d'Oxford (OUP) en 1994. C'est un dictionnaire bilingue bidirectionnel. Sa macrostructure consiste en deux volumes. Le volume français->anglais comporte environ 39 000 articles et le volume anglais->français environ 48 000 articles.

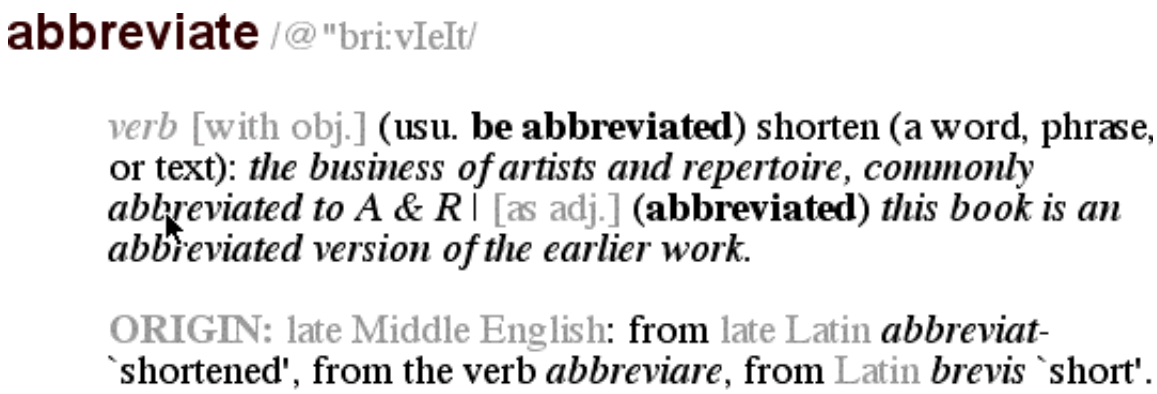


FIG. A.6 – *présentation de l'article* `abbreviate` *du NODE*

Format du dictionnaire

Ce dictionnaire est aussi publié par OUP. Le format interne est donc SGML. Le dictionnaire est stocké dans deux fichiers, un pour le volume anglais-français et un pour le volume français-anglais. Leur taille est d'environ 15 mégaoctets chacun. La prononciation est notée avec l'API.

Le tableau A.2 décrit les éléments SGML de l'exemple de la figure A.7.

élément	explication	traduction
<co>	collocate	collocation
<hg>	headword block	bloc du mot-vedette
<hw>	headword	mot-vedette
<ic>	indicator	indicateur
<pr>	pronunciation	prononciation
<ph>	phonetic IPA	phonétique API
<ps>	part-of-speech	catégorie grammaticale
<s2>	sense n°2	sens n°2
<sl>	source language	langue source
<se>	standard entry	entrée standard

TAB. A.2 – *éléments SGML du DHO*

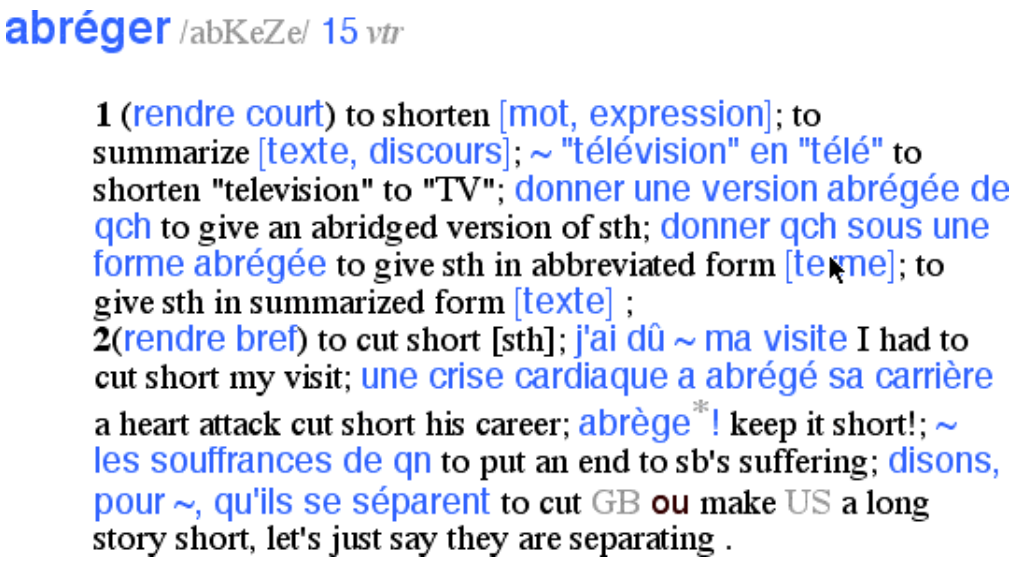
```
<se><hw>abr&ea.ger</hw><pr><ph>abKeZe</ph></pr>
<hg><ps> vtr</ps></hg><s2 num=1>(<ic>rendre court</ic>) to shorten
[<co>mot,expression</co>]; to summarize [<co> texte, discours</co>];
<sl>&hw.  &oq.t&ea.l&ea.vision&cq.  en &oq.t&ea.l&ea.&cq.</sl> to
shorten &oq.television&cq.  to &oq.TV&cq; (...)  </se>
```

FIG. A.7 – *l'article* `abréger` *du DHO en format original (SGML).*

Le mot-vedette `abréger` est suivi de sa prononciation, de sa catégorie grammaticale (vtr) et de ses traductions anglaises. Celles-ci sont différenciées par le contexte (collocations). On observe que les traductions ne sont pas marquées. Pour l'humain, cela ne pose pas de problèmes. Par contre, la machine ne peut utiliser ce dictionnaire pour rechercher des traductions. Il faut dans ce cas récupérer le dictionnaire pour marquer les traductions.

Présentation du dictionnaire

La figure A.8 est une présentation du même article lisible par l'humain.


abréger /abKeZe/ 15 vtr

1 (rendre court) to shorten [mot, expression]; to summarize [texte, discours]; ~ "télévision" en "télé" to shorten "television" to "TV"; donner une version abrégée de qch to give an abridged version of sth; donner qch sous une forme abrégée to give sth in abbreviated form [terme]; to give sth in summarized form [texte] ;
2(rendre bref) to cut short [sth]; j'ai dû ~ ma visite I had to cut short my visit; une crise cardiaque a abrégé sa carrière a heart attack cut short his career; abrège* ! keep it short!; ~ les souffrances de qn to put an end to sb's suffering; disons, pour ~, qu'ils se séparent to cut GB ou make US a long story short, let's just say they are separating .

FIG. A.8 – *présentation de l'article* `abréger` *du DHO*

1.2.4. Un dictionnaire très complexe : le DEC

Introduction

Le Dictionnaire Explicatif et Combinatoire du français contemporain [Mel'tchuk92] est un dictionnaire monolingue français. Le même type de DEC a été d'abord développé pour le russe par Igor Mel'tchuk, en collaboration avec de nombreuses personnes (en particulier avec le laboratoire du professeur Rosenweig). Comme il s'agit d'un travail de recherche en lexicographie, le DEC comporte peu de vocables, mais chacun est très détaillé. Il y a actuellement quatre volumes regroupant 558 vocables en tout.

La microstructure du DEC est définie par la lexicologie explicative et combinatoire [Mel'tchuk95], elle-même issue de la théorie sens-texte. Cette théorie fournit les informations nécessaires pour passer d'une idée (le sens) à sa réalisation dans une langue donnée (le texte).

Cette microstructure est très complexe : on trouve des tableaux de régime, des fonctions lexicales, etc. Pour représenter ces structures variées, nous avons donc besoin d'un langage de représentation de structures générique et riche.

Format et présentation

Chaque article est stocké dans un fichier à part dans des formats divers (ici en format HTML). La taille du fichier de l'article `Averse` de la figure A.9 est d'environ 7,5 ko. À l'origine, les articles étaient écrits en format Word™. Le même format est alors utilisé à la fois pour la structure interne, le stockage et la présentation. Étant donné que des éléments de style comme le gras et l'italique sont utilisés pour représenter la structure interne, celle-ci n'est pas explicitement compréhensible et elle peut être ambiguë.

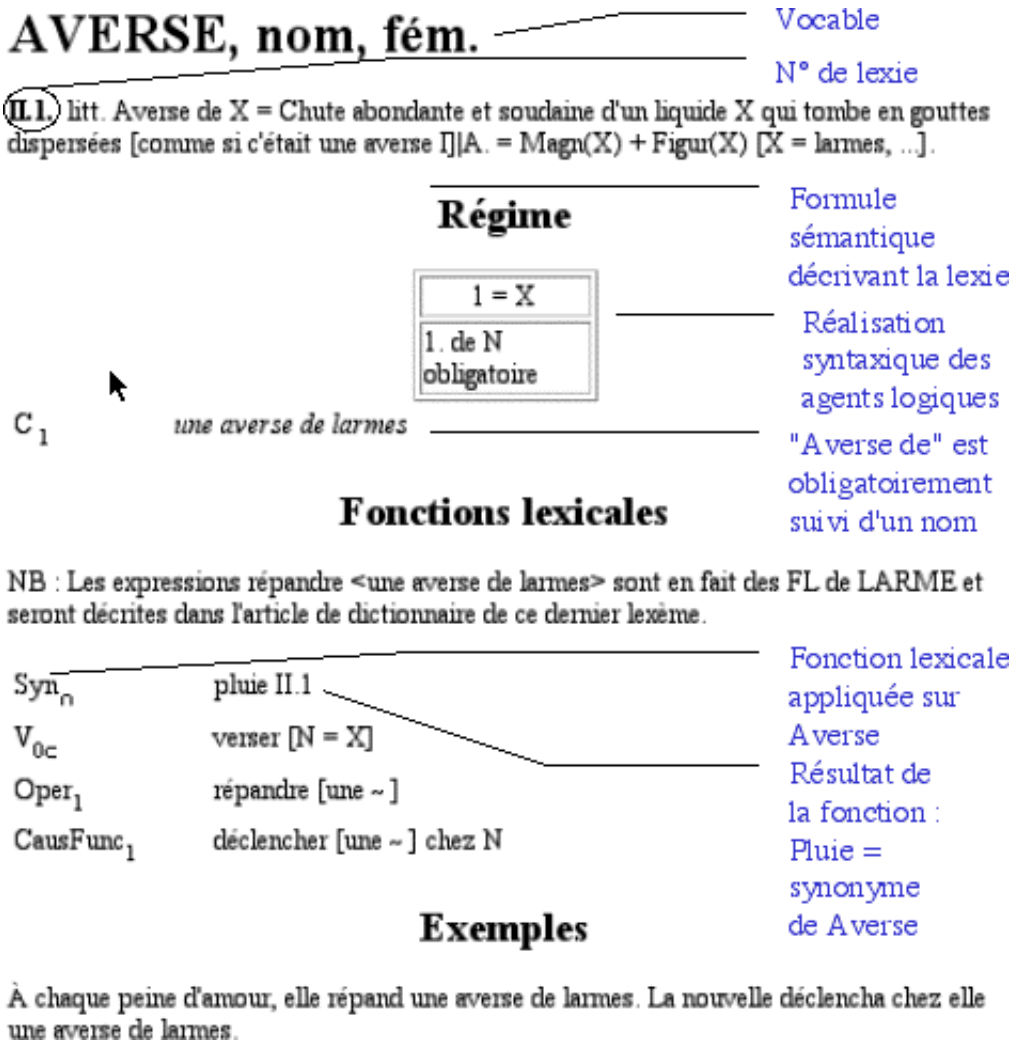


FIG. A.9 – *extraits du vocable* `averse` *du DEC en HTML*

1.2.5. Une simplification du DEC : la base DiCo

Introduction

Les récents travaux d'Alain Polguère [Polguère00] sur la construction de bases lexicales pour des linguiciels et la rédaction de dictionnaires publics généraux sont une application directe de la lexicologie explicative et combinatoire [Mel'tchuk95]. Le projet DiCo vise à construire une base lexicale du français de grande taille et à générer à partir de cette base un dictionnaire d'usage public : le Lexique Actif du Français (LAF). Cette recherche est menée conjointement par Igor Mel'tchuk et Alain Polguère avec l'aide des étudiants de l'Observatoire de Linguistique Sens-Texte de l'université de Montréal au Canada. La base DiCo est en cours de rédaction. Il est prévu à terme d'obtenir environ 3 000 vocables ayant chacun plusieurs lexies (en moyenne trois lexies).

Format du dictionnaire

Cette base lexicale est gérée par l'outil FileMaker®. Chaque entrée de la base correspond à une lexie. Un vocable peut avoir une ou plusieurs lexies selon qu'il est monosémique ou polysémique. Les lexies d'un même vocable auront généralement le même nom et les mêmes propriétés grammaticales. chaque lexie est composée de huit champs différents. Nous donnons dans l'exemple de la figure A.10 représentant l'unique lexie du vocable monosémique `MEURTRE` le nom de tous les champs suivis de leur valeur.

1. `Nom de l'unité lexicale :` `MEURTRE`
2. `Propriétés grammaticales :` `nom, masc`
3. `Formule sémantique :` `action de tuer:`
   `_ PAR L'individu X DE L'individu Y`
4. `Régime :` `X = I = de N, A-poss`
   `Y = II = de N, A-poss`
5. `(Quasi-)synonymes:` `{QSyn} assassinat, homicide#1; crime`
6. `Fonctions lexicales`
   - `{V0} tuer`
   - `{A0} meurtrier-adj`
   - `{S1} auteur [de ART _ ]`
     `//meurtrier-n /* Nom pour X*/`
7. `Exemples:` `La mésentente pourrait être le mobile du meurtre.`
8. `Idiomes:`
   - `_appel au meurtre_`
   - `_crier au meurtre_`

FIG. A.10 – *extraits de la lexie MEURTRE de la base DiCo*

Nous pouvons voir dans cette figure que le découpage structurel pourrait être plus fin que ces huit champs. En effet, le texte de certains champs est en fait un contenu structuré. Par exemple, dans la formule sémantique ou le régime, X et Y réfèrent à des actants; dans la fonction lexicale `{S1}`, la double barre `//` a une sémantique particulière : elle distingue les résultats de la fonction n'englobant pas le nom de l'unité lexicale (`auteur`) de ceux qui l'englobent (`meurtrier`). Cependant, une telle structure serait très complexe à représenter dans une base de données. C'est pourquoi on limite le découpage d'une entrée en huit champs seulement. Les utilisateurs savent ensuite interpréter la structure interne restant dans le texte des différents champs.

Présentation du dictionnaire

Le LAF est directement généré à partir de la base DiCo. Il tente de rapprocher la lexicographie "théorique" et la lexicographie "commerciale" en utilisant la lexicologie explicative et combinatoire. La figure A.11

représente l'article du LAF généré à partir de l'entrée de DiCo décrite plus haut.

MEURTRE, nom, masc

ACTION DE TUER: Meurtre par l'individu X [*de* N, A__] de l'individu Y [*de* N, A__] ≈ assassinat, homicide1; crime VERBE tuer ADJECTIF meurtrier$_{Adj}$ NOM POUR X auteur [de ART –] //meurtrier$_{N}$ NOM POUR Y victime [de ART –] TRÈS CHOQUANT atroce, affreux, brutal, horrible, inqualifiable, odieux QUI A ÉTÉ PRÉPARÉ avec préméditation, prémédité postpos //assassinat TEL QU'IL Y A DEUX/TROIS/QUATRE Y double/triple/quadruple antépos [*Les victimes de ce double meurtre sont un père de famille et son fils de 15 ans.*] FAIRE UN M. accomplir, commettre, perpétrer [ART –]; tremper [dans ART –] [*Il a refusé de tremper dans ce meurtre odieux.*] CAUSER QUE X FASSE UN M. pousser [N_ au –] RAISON D'UN M. mobile [de ART –] S'OCCUPER D'UN M. enquêter [sur ART –]; élucider [ART –], trouver l'auteur [de ART –]; punir, châtier [ART –]; venger [ART –] SERVICE DE POLICE QUI S'OCCUPE DES M. brigade criminelle PRÉPARER UN M. préméditer, préparer [ART –] //comploter ESSAYER DE FAIRE CROIRE QU'UN M. EST UN N maquiller [ART – en N] [*Il a maquillé ce meurtre en accident/suicide.*] FAIT DE TENTER UN M. tentative [de –] CRI LANCÉ PAR QQN. QUI ASSISTE À _RISQUE D'ÊTRE VICTIME D'_UN M. « AU ~ ! ». *C'est ici que le double meurtre a été commis. Soupçonné du meurtre de son épouse, il a été arrêté par les gendarmes mercredi. Il devrait comparaître aux assises dans trois semaines comme auteur présumé du meurtre d'un quinquagénaire.* ◊ _appel au meurtre_, _crier au meurtre_.

FIG. A.11 – *l'article MEURTRE du LAF*

Nous voyons que cet article est destiné à un lecteur humain. L'utilisateur peut dans un premier temps lire l'article rapidement de manière naturelle. Il peut ensuite, s'il est attentif aux détails de mise en page, retrouver toutes les informations présentes dans la base DiCo, notamment la formule sémantique ou le régime.

Ce formalisme est très intéressant puisque, à partir des mêmes données, il permet de produire des ressources aussi bien pour des systèmes lexicaux que pour des dictionnaires d'usage grand public. De plus, il permet de populariser la lexicologie explicative et combinatoire provenant de la théorie sens-texte. Nous avons ici un net progrès par rapport aux autres dictionnaires vus précédemment qui n'étaient destinés qu'à un usage uniquement humain et ne pouvaient donc pas être facilement utilisables par une machine sans transformation.

L'utilisation d'une base de donnée limite cependant la structuration des entrées en champs. Il serait intéressant d'utiliser un système de structuration plus élaboré du type XML afin de noter explicitement toute la structure des articles.

1.2.6. Conclusion

De cette revue de quelques dictionnaires à usage humain, il ressort que certaines informations ont une structure implicite. Elles sont codées par une présentation spéciale (styles, polices ou couleurs différentes). Cela ne pose pas de problèmes à l'humain, qui peut tout de suite déduire la structure de la présentation. Par contre, un programme ne peut déduire automatiquement cette structure. Si l'on veut utiliser ces dictionnaires avec des machines, il faut donc trouver un moyen de récupérer ces informations.

Une seconde observation est que ces dictionnaires ont des structures très variées. Si l'on veut représenter un maximum de dictionnaires avec le même langage, il faut donc que ce langage soit générique et qu'il puisse représenter un grand nombre de structures linguistiques comme des arbres, des tableaux de régimes, des fonctions lexicales, etc.

Un troisième point est que tous ces dictionnaires ont des formats différents. Si l'on veut les utiliser en même temps, il faut donc trouver un moyen d'unifier les formats soit avec un format commun, soit avec des

outils de transformation dynamiques.

Enfin, la sémantique des marques ou balises contenues dans les dictionnaires n'est pas non plus unifiée. Pour représenter le mot-vedette, on utilise parfois l'élément <hw>, parfois l'élément (entry), parfois l'élément <h1>, etc. Il faut donc là aussi trouver un moyen de s'accorder pour pouvoir manipuler différents dictionnaires à l'aide des opérations de fusion, d'union, d'intersection, etc.

1.3. Exemples de dictionnaires à usage machinal

1.3.1. Un dictionnaire provenant de la traduction automatique : le RUSFRA

Le système RUS-FRA [Boitet82a,82b] est un système de traduction de seconde génération développé au sein du laboratoire GETA à l'aide de l'environnement ARIANE-78 puis converti en ARIANE-G5. Pour les étapes d'analyse et de génération morphologiques ainsi que le transfert lexical, ARIANE-G5 utilise des dictionnaires. Examinons leur structure. Le langage ATEF utilise, pour l'analyse morphologique trois sortes de dictionnaires monolingues : de bases, d'affixes et de tournures. Chaque dictionnaire est une liste d'articles dont voici la syntaxe simplifiée en figure A.12.

```
<article de D. de bases> ::= <morphe> == <format M> (<format S ou G>,
<UL>).
<article de D. d'affixes> ::= <morphe> == <format M> (<format S ou
G>).
<article de D. de tournures> ::= <tournure> == <format M> (<format S
ou G>, <UL>).
<morphe> ::= <suite de symboles non blancs de 34 caractères>.
<tournure> ::= <suite de symboles sans sous-suite de 2 blancs de 34
caractères au plus>.
<format i> ::= <identificateur>.
```

FIG. A.12 – *syntaxe du langage ATEF*

L'exemple de la figure A.13 est tiré du dictionnaire de bases. Il contient quatre articles.

```
ACETATE
==N1      (SUBST   ,ACETATE                ). ACETIC
==A       (VOID    ,ACETIC                 ).
DUMP
==V1Z     (PN1     ,DUMP                   ).
DUMP
==N1Z     (LOC     ,DUMP                   ).
```

FIG. A.13 – *articles provenant d'un dictionnaire au format ATEF*

La syntaxe est positionnelle, le signe "==" étant en colonnes 35 et 36, la "("en colonne 45, la "," ou ")" en colonne 54, et ")." en colonnes 63-64 (affixes) ou 79-80 (base ou tournures). Suit éventuellement un commentaire. Ainsi, aucun délimiteur n'est nécessaire. On accède aux dictionnaires par les morphes ou les tournures.

Le langage TRANSF permet d'écrire des dictionnaires bilingues pour la traduction. Un article est caractérisé par un nom d'unité lexicale source noté entre deux apostrophes, un séparateur "==" , et une liste de

triplets contenant chacun :

- une condition (expression de condition propre ou appel à une procédure de conditions),
- une arborescence image du nœud en cours,
- une partie affectation comprenant, pour chaque sommet de l'arborescence image le nom du sommet, le symbole ":" , le nom d'unité lexicale cible affecté à ce sommet, suivi éventuellement d'une liste d'affectations de valeurs de variables cibles, qui peut comporter un nom de format préfixé ou des expressions d'affectation.

La figure A.14 donne un exemple tiré du dictionnaire de traduction russe-français.

```
'OBRATITQSYA' == PG -E- 5K / /ADRESSER , +VMB2/
                 / /TRAITER , +UMB1,$ACC,$NRF/
                 ...
                 / /TRANSFORMER , +VBF1.
```

FIG. A.14 – *article du dictionnaire de traduction russe->français*

Ce dictionnaire de traduction automatique a été simplifié semi-automatiquement pour obtenir le dictionnaire RUSFRA [Boitet82c, Nedobejkine94] du GETA. Les accents toniques du russe y ont été ajoutés afin d'obtenir un dictionnaire plus utile. Il comporte environ 10 000 unités lexicales pour chaque langue correspondant à environ 26 000 lemmes pour le français et 30 000 lemmes pour le russe.

Les articles de la figure A.15 sont des articles de ce dictionnaire "naturel". Ils sont divisés en 2 parties. La première est composée de l'unité lexicale russe suivie du lemme correspondant et d'une variable utilisée par le système. La deuxième est composée de l'unité lexicale française correspondant à la traduction de l'entrée russe, suivie d'un numéro unique, d'un code morphosyntaxique et enfin du lemme français correspondant.

```
((obratitq obrathaμtq $vi)
   (2(adresser )$v adresser ))
((obratitq obrathaμtqsya $vi.r )
   (8(traiter ) $v traiter ))
((obratitq obrathaμtq $vi)
   (4(transformer) $v transformer))
```

FIG. A.15 – *trois articles du dictionnaire RUSFRA*

1.3.2. Une base de données lexicales pour la phonologie : BDLex

BDLex [Pérennou92] est un projet développé dans le cadre du GDRPRC CHM par le groupe IHMPT de l'IRIT, Université Paul Sabatier de Toulouse. Son objectif était de rendre disponibles différents matériaux lexicaux utilisés dans les interfaces en langage naturel écrit ou oral et dans les systèmes d'aide linguistique. Un lexique de formes fléchies représentées aux plans morpho-syntaxique, phonologique et orthographique a été produit. La version la plus complète contient plus de 430 000 formes fléchies générées à partir de 50 000 entrées canoniques. Cette version est distribuée par l'association ELRA (European Language Resource Association) [ELRA].

Les entrées lexicales de BDLex sont des lemmes. À chaque entrée lexicale sont associés plusieurs champs :

– une représentation phonologique sous-jacente dans les champs PHON_SYLL et FPH. BDLEX fournit de plus les homophonies, la représentation en classes phonétiques et le nombre de syllabes (voir figure A.18),

– une représentation en phonogrammes (champ PHONOGRAMMES). Ceux-ci jouent un rôle important dans le cadre de la correction lexicale ou encore de la transcription graphèmes-phonèmes,

– des statistiques lexicales représentées par un ensemble d'indices de fréquences d'origine diverses (fréquence de Catach, fréquence élémentaire).

La figure A.16 représente un extrait de cette base. Tous les champs ne sont pas représentés.

```
GRAPH_ACC PHON_SYLL FPH CS PHONOGRAMMES
aigre-doux E/gr/du s" J (ai,E) (g,g) (r,r) (e, ) (-,) (d,d) (ou,u)
(x,s") amygdale A/mi/dAl N (a,A) (m,m) (y,i) (g,) (d,d) (a,A) (l,l)
(e,)
axe Aks N (a,A) (x,ks) (e,)
bahut /bA/y N (b,b) (a,A) (h,) (u,y) (t, )
chat-huant /_A/y/ã N (ch,_) (a,A) (t, ) (-, ) (h,) (u,y) (an,ã) (t,
) dix-huit /di/zÁi t' J (d,d) (i,i) (x,z) (-, ) (h, ) (ui,Ái) (t,t')
exact eg/zA kt" J (e,e) (x,gz) (a,A) (ct,kt")
iceberg Ajs/bErg N (i,Aj) (c,s) (e, ) (b,b) (e,E) (r,r) (g,g) hautbois
/*O/bwA N (h,*) (au,O) (t, ) (b,b) (oi,wA) (s, )
homme Øm N (h, ) (o,Ø) (mm,m) (e, )
onze /*õz J ( ,*) (on,õ) (z,z) (e, )
skate /skEjt N (s,s) (k,k) (a,Ej) (t,t) (e, )
tocsin /tOk/sÉ N (t,t) (o,O) (c,k) (s,s) (in,É)
```

FIG. A.16 – *extrait de la base BDLex*

La figure A.17 représente un autre exemple avec les indices associés.

```
GRAPH_ACC HG CS FREQ F_Catach FREQ_Elementaire
alors 11 A C1 B0 111 22
avoir 21 V C0 B0 TR 11 2
chaussure 11 N B0 701
être 21 V C0 B0 TR 4 1
de 11 p C0 B0 2 3
la 11 d C0 B0 1 7
rayonner 11 V B0
```

FIG. A.17 – *extrait de BDLex avec les indices associés*

Chaque entrée lexicale est munie de marques de frontière spécifiant la nature du terme placé immédiatement après. Lorsqu'une partie du mot est une autre entrée lexicale, celle-ci n'est pas décomposée. Actuellement, 68 préfixes et 107 suffixes ont été introduits dans BDLEX. Ceux-ci peuvent être utilisés pour procéder à une

analyse morphologique dérivationnelle. Les matériaux lexicaux de BDLEX sont disponibles sous l'environnement ORACLE sur station de travail SUN. L'accès aux informations peut s'effectuer au grâce aux outils dont dispose ORACLE.

Ce dictionnaire est typiquement à usage informatique. De plus, les informations de ce dictionnaire sont codées et difficilement utilisables par un humain.

1.3.3. Une base de concepts multilingue : la base Mémodata

Cette base multilingue [Dutoit92] est basée sur le Dicologique. Sa macrostructure est constituée d'un dictionnaire pivot où sont décrits les concepts, et d'un dictionnaire pour chaque langue (allemand, anglais, espagnol, italien et français) dans laquelle est traduit chaque concept. La base Mémodata compte environ 47 000 concepts tous traduits dans chaque langue de la base. Cette base est également distribuée par l'association loi 1901 ELRA [ELRA].

Sa microstructure est très simple : pour chaque concept, on trouve une ligne avec le numéro de concept, ";", une lettre pour indiquer la langue (A pour les concepts, D pour l'allemand, E pour l'espagnol, S pour l'espagnol, F pour le français et I pour l'italien), la traduction du concept, ";" et sa catégorie grammaticale. Le format utilisé est un format texte simple. Les traductions des concepts sont stockées par langue. Chaque fichier a une taille d'environ 1,2 mégaoctets. La figure A.18 représente les concepts 91 et 92 et leurs traductions. Chaque paragraphe ou groupe de lignes provient d'un fichier différent :

```
91;A;bientôt Av
92;A;abréviation TL|rendre plus simple, plus élémentaire V
91;D;demnächst;adv
91;D;in Kürze;adv
91;D;nächstens;adv
92;D;abkürzen;v_trans
91;E;soon;adv
92;E;abbreviate;v_trans
92;E;shorten;v_trans
91;S;dentro de poco;adv
92;S;abreviar;v_trans
91;F;avant peu;adv
91;F;dans peu;adv
91;F;sous peu;adv
92;F;abréger;v_trans
91;I;fra poco;adv
91;I;fra poco tempo;adv
91;I;tra poco;adv
92;I;abbreviare;v_trans
```

FIG. A.18 – *concepts 91 et 92 et leurs traductions dans la base Mémodata*

Un concept peut être traduit dans une langue donnée par une ou plusieurs traductions. Le concept n°91 est ici traduit par 3 traductions dans toutes les langues sauf en anglais et en espagnol où il n'y a qu'une traduction.

1.3.4. Des bases lexicales utilisables en traduction automatique : les bases UNL

Fondé à l'IAS (Institute of Advanced Studies) de l'UNU (Université des Nations Unies) à Tokyo en avril 1996, le projet UNL [UNL96,97] rassemble maintenant des partenaires du monde entier, avec plus de 14 langues couvertes. Le but de ce projet est la définition d'un format d'échange (le langage UNL) codant la sémantique d'un document de manière suffisamment précise pour permettre sa "déconversion" dans la langue maternelle du lecteur. Le modèle développé dans le projet UNL est fondé sur une représentation interlingue (sous forme de graphes sémantiques) à partir de laquelle on peut générer des textes dans la langue de notre choix.

Depuis le 1^{er} janvier 2000, les spécifications du "langage UNL" sont ouvertes à tous sur le serveur de l'IAS. Les enjeux scientifiques de cette recherche sont d'ordre conceptuel, linguistique et informatique. Il s'agit d'articuler :

– un modèle de représentation sémantique ;

– des outils permettant de faire l'aller-retour entre un document en langue naturelle et ce modèle;

– un système d'information où les donnée échangées sont indépendantes des langues.

Le modèle développé dans le projet UNL est fondé sur une représentation interlingue (sous forme de graphes sémantiques décrivant la structure abstraite des énoncés) à partir de laquelle on peut générer des textes dans toute langue disposant d'un déconvertisseur. Il existe actuellement des générateurs expérimentaux couvrant 14 langues et développés dans 15 pays différents. L'axe principal de recherche a porté dans un premier temps sur la définition du langage d'échange UNL et sur sa validation, par le développement de "déconvertisseurs" (outils réalisant la déconversion ou traduction automatique d'un document UNL en un document en langue naturelle).

La macrostructure de la base est constituée d'une part pour chaque langue du projet d'un dictionnaire bilingue associant les mots de cette langue avec les unités lexicales de l'UNL que l'on appelle UW (Universal Words) et d'autre part d'une base regroupant toutes les UW et d'une base de relations reliant des couples d'UW. Une telle relation porte un poids qui indique sa fréquence d'apparition dans les corpus.

La base de relations est appelée Knowledge Base. Cette base ainsi que la base regroupant toutes les UW sont maintenues par l'équipe responsable du projet.

Les dictionnaires bilingues sont maintenus par les équipes partenaires. Chaque partenaire est responsable du dictionnaire associant sa langue et l'UNL. Le GETA s'occupe donc du dictionnaire français-UNL.

La microstructure du dictionnaire décrit pour chaque article une correspondance entre un mot français suivi de variables utilisées dans le système de traduction automatique ARIANE puis d'une UW. L'UW est composée d'un mot-vedette en anglais suivi de restrictions sémantiques.

Dans l'exemple de la figure A.19, le mot français `abréger` est suivi des variables ARIANE suivantes : `{AUX (AVOIR), CAT (CATV), VAL1 (GN)}` et de l'UW `"shorten(obj>word)"`. Cette UW est une dénotation d'ensembles d'acceptions interlingues. Elle est composée du mot-vedette anglais `shorten` suivi de la restriction sémantique `obj>word`. Ici, cette restriction de sens signifie que `shorten` s'applique seulement aux mots `(obj>word)`.

```
[abréger]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"shorten(obj>word)";
```

FIG. A.19 – *l'article* `abréger` *du dictionnaire français-UNL au format original*

Ce dictionnaire est en constant développement afin d'ajouter de nouveaux mots ou de distinguer plusieurs sens différents. Il contient actuellement environ 40 000 articles.

1.3.5. Conclusion

Les dictionnaires que nous venons de présenter sont à usage "machinal", ce qui a pour conséquence que leur structure est parfaitement définie et leur contenu jamais ambigu.

Il apparaît aussi que, souvent, le contenu de ces dictionnaires peut aussi être intéressant pour des usages humains. La base Mémodata en est le meilleur exemple. Comment faire ? Le format externe, comme nous l'avons vu, est lisible par des développeurs mais inutilisable par des lecteurs humains. Quant au format compilé, destiné à permettre les accès rapides, il est toujours illisible par l'humain.

On voit donc encore une fois l'intérêt de définir, pour tout dictionnaire et pour toute ressource lexicale, une structure interne "pivot", et de considérer les autres formes comme des "présentations", y compris la forme "source", qui doit être strictement équivalente. Les autres formes (compilées pour des applications, ou filtrées pour des présentations) peuvent par contre ne contenir qu'une partie de l'information.

2. Outils de consultation de dictionnaires

Un aspect important des dictionnaires à usage humain est évidemment leur "consultabilité", déterminée par la puissance des outils de recherche, par la clarté et la souplesse de la présentation du contenu, et enfin par la qualité de l'interface utilisateur en général.

2.1. Applications de consultation sur ordinateur

Les applications de consultation de dictionnaires sur ordinateurs sont pratiques car les réponses aux requêtes sont quasi-instantanées. De plus, du point de vue des éditeurs, c'est un moyen de vendre leurs dictionnaires sur cédéroms, ce qu'ils ne peuvent pas faire avec des serveurs de consultation sur le Web. On trouve donc principalement des versions électroniques des grands dictionnaires imprimés du commerce. Étant donné que le service est payant, les dictionnaires sont le plus souvent de meilleure qualité que ceux que l'on trouve sur le Web. La qualité est garantie grâce à l'équivalence de ces dictionnaires avec leur version papier.

2.1.1. Une application basique : le *Collins on-line*

Introduction

Le Collins On-Line est un dictionnaire électronique bilingue français-anglais de Harper Collins publisher développé par AND software. Son utilisation est relativement simple et convient très bien pour une recherche rapide d'une traduction d'un mot. La rapidité permet à l'utilisateur de ne pas perdre le contexte de la phrase contenant le mot cherché.

Interface

Dans une première fenêtre, il y a un cadre de saisie du mot recherché dans la langue source avec, dans le deuxième cadre, au fur et à mesure que l'on rentre les lettres du mot, les mots se rapprochant le plus des lettres saisies par ordre alphabétique. Il faut sélectionner un mot du cadre du bas pour faire une recherche. Ensuite, il y a deux boutons au bas de la fenêtre : Retourner et Rechercher.

Si l'on clique sur le bouton Rechercher, une deuxième fenêtre apparaît, comportant la traduction du mot préalablement sélectionné. Elle contient pour chaque catégorie grammaticale (n, vt, vi etc.) la traduction du mot avec une précision de sens dans la langue source entre parenthèses s'il y a plusieurs traductions possibles, et la traduction de locutions contenant le mot, avec une précision de sens si nécessaire. À la fin de la liste sont indiquées les sous-entrées du dictionnaire. Les deux fenêtres sont représentées dans la figure A.20.

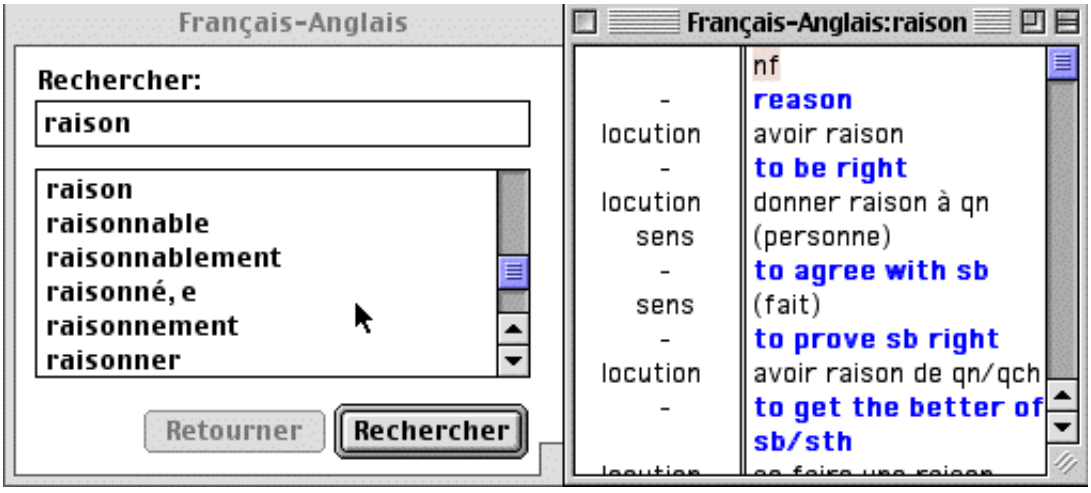


FIG. A.20 – *l'article* `raison` *du Collins on-line*

Discussion

Le Collins est bâti sur des structures de donnée relativement simples et figées. Il est conçu pour une utilisation commerciale. Une fois que la présentation des articles est définie, il n'y a pas de possibilités d'évolution.

Cette interface a cependant deux propriétés intéressantes. Le contexte du mot recherché est systématiquement affiché, ce qui permet de consulter des articles voisins de celui que l'on cherche. Dans la fenêtre de résultat, les traductions anglaises sont en bleu. Elles n'ont pas la même couleur que le texte français et se distinguent donc plus facilement.

2.1.2. Une application plus riche : *Oxford Superlex*

Introduction

L'application Oxford Superlex permet de consulter les dictionnaires publiés par les presses universitaires d'Oxford (OUP). Les dictionnaires disponibles sont des versions électroniques des dictionnaires imprimés. Le contenu n'est pas modifié. Aucune information n'est rajoutée. L'application apporte simplement une facilité de recherche des mots.

L'application est indépendante des dictionnaires. L'utilisateur peut ajouter un dictionnaire dans l'application à tout moment. Dans notre exemple, nous avons trois dictionnaires disponibles : le Oxford-Hachette anglais-français, le Oxford-Duden anglais-allemand et un anglais-espagnol. Par contre, les dictionnaires ne sont pas consultables en même temps. Il faut sélectionner un dictionnaire avant de faire des recherches.

Cette application est disponible sur PC et sur Macintosh. Nous pouvons voir d'ailleurs qu'aucun travail n'a été fait pour la version Macintosh. En effet, les caractères ne sont pas codés de la même manière sur les deux plates-formes. Il faut donc recoder les accents des dictionnaires. Dans l'exemple, les accents n'ont pas été recodés, ce qui génère des problèmes à l'affichage. Au lieu de lire "é", on lit "È", etc.

Interface

L'utilisateur choisit d'abord le dictionnaire qu'il veut consulter dans la liste en haut. Ici nous avons sélectionné le Oxford-Hachette anglais-français. Il choisit ensuite le volume anglais->français ou français-

>anglais. Il peut enfin consulter le dictionnaire en tapant le début du mot qu'il cherche dans la case de recherche.

FIG. A.21 – *l'article* `abréger` *du Oxford Superlex*

La figure A.21 montre l'article `abréger` et l'interface du Oxford Superlex. Le contexte de l'article trouvé s'affiche dans la partie gauche de la fenêtre. L'utilisateur clique ensuite sur l'article qu'il veut consulter et celui-ci s'affiche dans la partie droite.

Discussion

Cette application est une simple adaptation électronique des dictionnaires imprimés. Les seuls avantages qu'elle rapporte sont la rapidité de recherche et quelques facilités comme l'historique des recherches. Sur Macintosh, les caractères ne sont pas encodés correctement.

La qualité des données consultées nous fait regretter les limitations de l'application. En effet, on ne peut pas faire de recherche multidictionnaires, rechercher un mot autre que le mot-vedette présent dans le texte de l'article, ni rechercher un mot à partir de sa prononciation.

2.1.3. Une application évoluée : *MoBiDic*

Introduction

MoBiDic (MorphoLogic Bilingual Dictionary) [Proszéky97] est une application développée par la société hongroise Morphologic. Cette application permet des recherches multidictionnaire. Il est possible de consulter entre autres 21 dictionnaires anglais-hongrois dont le English-Hungarian School Dictionary de 35 000 articles, 13 dictionnaires allemand-hongrois dont le German-Hungarian Concise Dictionary de 65 000 articles, deux dictionnaires français-hongrois, deux dictionnaires italien-hongrois et d'autres dictionnaires spécialisés multilingues.

Le système permet de consulter tous ces dictionnaires avec la même interface. Il utilise des lemmatiseurs pour aider la consultation. Les utilisateurs ont aussi la possibilité de créer leurs propres dictionnaires et de les ajouter au système.

Interface

L'utilisateur peut d'abord paramétrer de nombreuses options : le nombre de dictionnaires qu'il consulte, la langue d'interface, la taille des fenêtres, etc. L'interface est composée d'une fenêtre de consultation et de réglages d'options puis de 4 fenêtres de résultat. La figure A.22 représente l'interface et toutes les fenêtres de résultat.

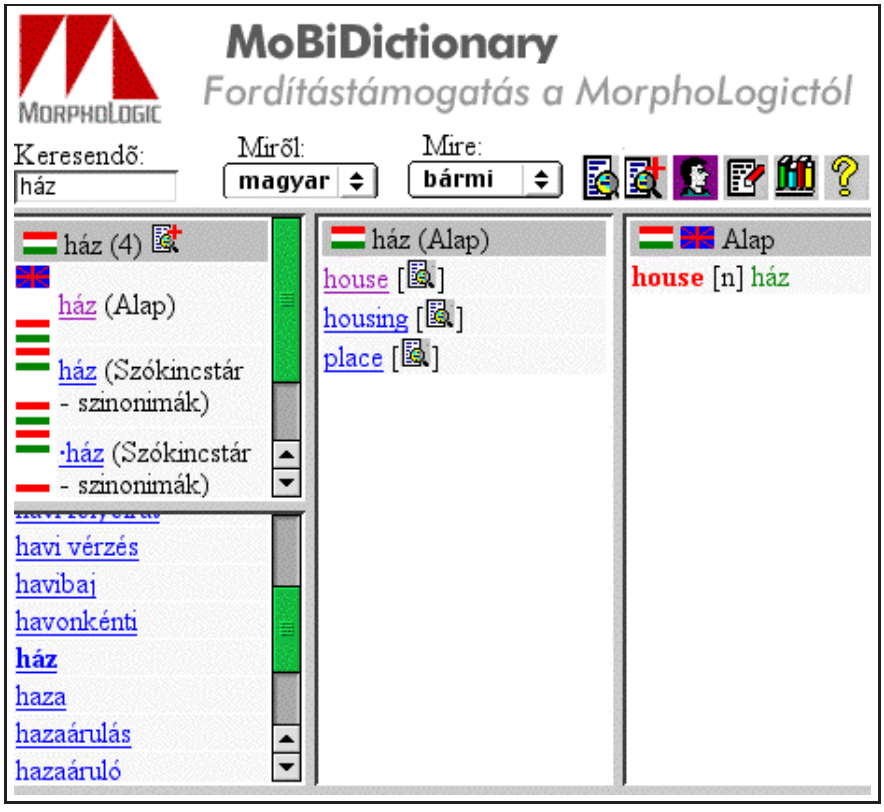


FIG. A.22 – *résultats d'une requête sur MoBiDictionary*

Lorsqu'on interroge le système, la première fenêtre affiche pour chaque dictionnaire dans lequel le système a trouvé une réponse une ligne avec le mot et le nom du dictionnaire. Dans la deuxième fenêtre se

trouve le contexte de l'entrée que l'on sélectionne, c'est à dire les 5 mots précédents et les 5 mots suivants selon la nomenclature du dictionnaire. La troisième fenêtre affiche tous les mots du dictionnaire sélectionné correspondant à la requête. Enfin, la dernière fenêtre affiche l'article complet.

Discussion

Cette application apporte de nombreuses améliorations dans la consultation de dictionnaires : une recherche multidictionnaire, une aide à la consultation grâce à un lemmatiseur, une vue du contexte de l'article sélectionné, la possibilité de configurer beaucoup d'options et de rajouter ses propres dictionnaires.

Elle pourrait être enrichie en proposant la personnalisation d'un article complet au niveau des informations proposées et de leur style. De plus, la dernière fenêtre n'affiche qu'un article à la fois. Il est donc impossible de comparer plusieurs articles.

2.2. Consultation de dictionnaires sur Internet

Les serveurs proposant une consultation de dictionnaires en ligne sur le Web sont de plus en plus nombreux étant donné la croissance de l'Internet et le besoin en ressources linguistiques qu'il provoque avec le multilinguisme. Malheureusement, les ressources proposées sont généralement de qualité moyenne car le service est gratuit. On comprend facilement que les éditeurs hésitent à mettre leurs ressources disponibles sur le Web car dans ce cas, ils ne pourraient plus les vendre.

2.2.1. Consultation simple du dictionnaire universel francophone

Introduction

Le Dictionnaire Universel Francophone est issu des efforts combinés de l'AUPELF-UREF (Agence francophone pour l'enseignement supérieur et la recherche) et des Éditions Hachette. Le dictionnaire est consultable en ligne depuis novembre 1999 à cette adresse [DUF].

Interface

L'interface Web de ce dictionnaire est très sommaire. L'utilisateur entre un mot dans un formulaire HTML dans la partie gauche de son navigateur et clique sur le bouton "chercher". La requête est alors envoyée au serveur. Une liste des articles correspondant à ce mot est renvoyée et affichée toujours sur la gauche du navigateur. L'utilisateur clique alors sur l'article qu'il souhaite lire et celui-ci apparaît dans la partie droite de son navigateur. Tous les mots de l'article sont en fait des liens hypertexte qui renvoient aux articles définissant ces mots. Cela permet une navigation au hasard dans le dictionnaire. La figure A.23 représente l'interface HTML du dictionnaire.

L'utilisateur n'a qu'une seule option de consultation : il peut chercher des mots commençant par une suite de lettres qu'il tape dans le formulaire. De plus, la consultation est ralentie car elle se passe en deux temps. L'utilisateur doit cliquer sur l'article précis qu'il désire lire avant qu'il ne s'affiche. Il serait par exemple intéressant de consulter le dictionnaire avec des expressions régulières (par exemple, tous les mots contenant la racine "coup" comme coup, coupole, couple, couper, découper, découpage, etc.). Mais cependant, lorsqu'un dictionnaire n'est pas public et c'est le cas, les concepteurs réduisent volontairement les options d'accès pour éviter le pillage.

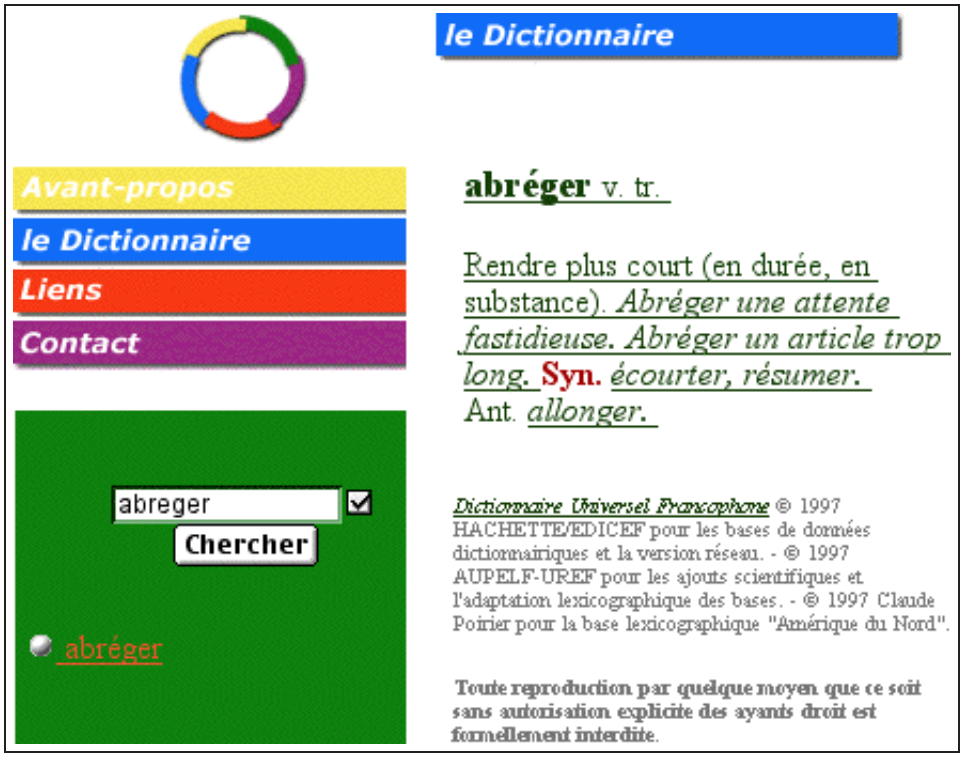


FIG. A.23 – *interface et résultats de la consultation du DUF*

2.2.2. Consultation plus évoluée d'un dictionnaire : *EDict*

Introduction

Le projet EDICT a démarré en 1991. Il est dirigé par Jim Breen, professeur au campus de Melbourne de l'université Monash en Australie. Il consiste en deux parties :

– la création et le maintien de trois documents : un lexique japonais-anglais (EDICT) qui contient actuellement plus de 70 000 articles, un dictionnaire de noms propres (ENAMDICT) qui contient plus de 200 000 noms et une base d'informations sur les kanji (KANJIDIC) qui détaille les 6 353 kanji inclus dans le standard JIS X 0208,

– le développement de logiciels et de serveurs pour consulter ces ressources.

Depuis, le projet JMDict (Japanese Multilingual Dictionary) a pris la suite en 1999. Les buts sont de convertir les ressources au format XML [XML 1.0] et encodé en Unicode [ISO93] UTF-8 et aussi d'ajouter d'autres langues cibles. Il y a actuellement environ 10 000 articles japonais-français et 18 000 articles japonais-allemand.

Toutes les ressources sont consultables en ligne [EDICT]. De plus, les sources sont téléchargeables gratuitement par FTP.

Interface

Avec la même interface, il est possible de chercher un mot anglais, un mot japonais en choisissant son dictionnaire dans une liste, d'examiner un kanji dans un mot composé de plusieurs kanji ou de consulter la

base des kanji. La figure A.24 montre l'interface et les résultats de la recherche dans le dictionnaire EDICT.

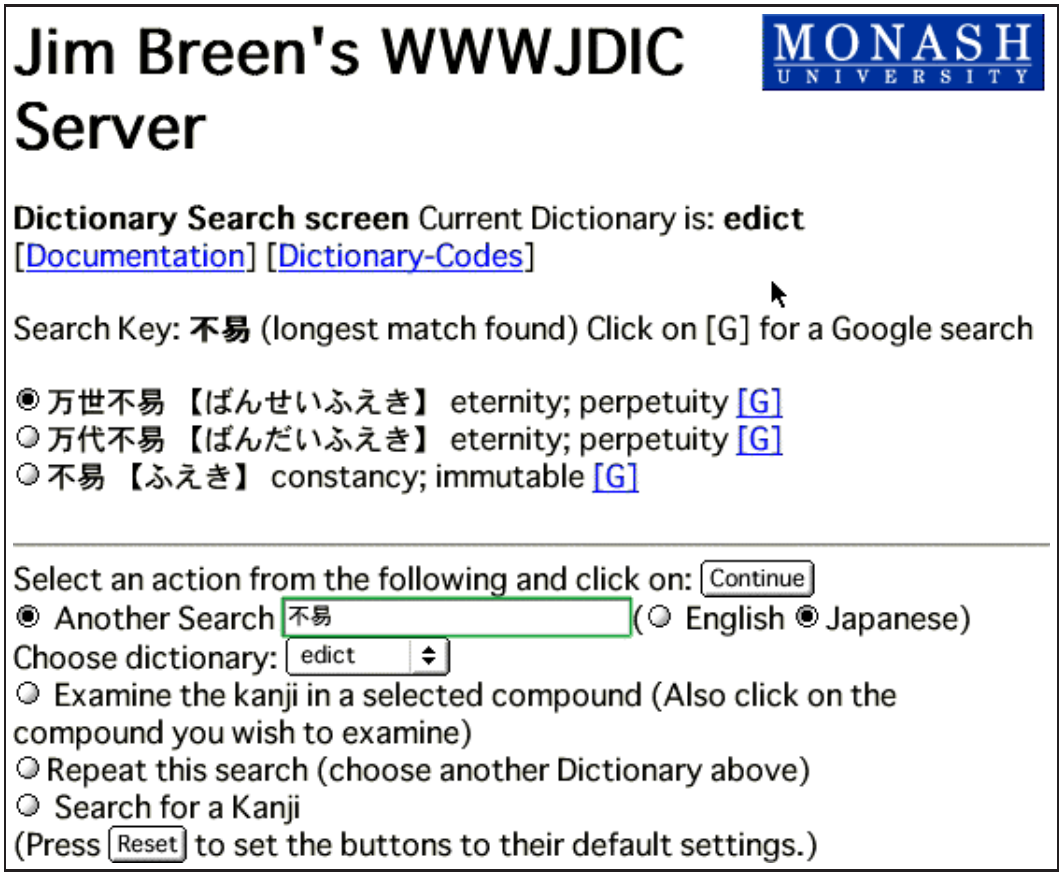


FIG. A.24 – *interface et résultats de WWWJDict*

La figure A.25 montre l'article obtenu de la figure A.24 au format original XML.

```
<entry>
  <ent_seq>1491240</ent_seq>
  <k_ele><keb> 不易 </keb></k_ele>
  <r_ele><reb> ふえき </reb></r_ele>
  <sense><gloss>constancy</gloss>
  <gloss>immutable</gloss></sense>
</entry>
```

FIG. A.25 – *article de EDICT au format XML*

2.2.3. Consultation de plusieurs dictionnaires : le site *dictionary.com*

Introduction

Le site Web dictionary.com [Dictionary.com] est maintenu par la société Lexico basée en Californie. Il permet de consulter des ressources en ligne depuis 1998. Ce serveur consulte plusieurs dictionnaires en

même temps et affiche tous les articles dans une même fenêtre.

Tous les dictionnaires disponibles sont monolingues anglais. On trouve le American Heritage® Dictionary of the English Language, le Webster's Revised Unabridged Dictionary, le WordNet® version 1.6, des informations provenant du CIA World Factbook, le Free On-line Dictionary of Computing de l'U.S. Gazetteer contenant plus de 13 000 termes, le Acronym Finder contenant 50 000 entrée, le On-line Medical Dictionary incluant 60 000 termes et 5 000 définitions du glossaire financier InvestorWords. Ils totalisent un ensemble de 600 000 entrées.

Interface

L'interface représentée par la figure A.26 est du type de celle du dictionnaire universel francophone. L'utilisateur rentre un mot dans un formulaire HTML et clique sur un bouton pour lancer la recherche. Il n'a aucune option de consultation comme la recherche avec les premières lettres ou des expressions régulières, l'utilisation d'un lemmatiseur, etc.

FIG. A.26 – *interface du serveur dictionary.com*

Résultat

La consultation de plusieurs ressources permet d'obtenir des articles ayant le même mot-vedette mais appartenant à des dictionnaires différents. Par exemple, pour le mot-vedette "do", on obtient 19 articles provenant de six dictionnaires différents. La figure A.27 montre le résultat d'une requête sur le mot anglais `abbreviation`.

2.2.4. Consultation d'une base terminologique multilingue : *EURODICAUTOM*

Introduction

Eurodicautom [Eurodicautom] est la base de données terminologique multilingue du service de traduction de la Commission Européenne. Développée initialement pour assister les traducteurs internes, elle est aujourd'hui consultée par un nombre croissant de fonctionnaires de la Communauté Européenne ainsi que par des professionnels de la langue grâce au projet MLIS (Multilingual Information Society) piloté par la DGXIII. Les données contenues sont disponibles en 12 langues et constamment mises à jour. Ces langues sont le danois, le hollandais, l'anglais, le français, l'allemand, le grec, l'italien, le portugais et l'espagnol.

La base terminologique EURODICAUTOM comprend 700 000 entrées (couvrant en moyenne 5 ou 6 langues) et un fichier d'abréviations et d'acronymes comprenant 150 000 entrées mis à jour chaque mois avec environ 2 000 items. La base couvre un large spectre de la connaissance humaine même si le cœur est relatif aux thèmes de la Commission Européenne.

Interface

L'interface Web d'EURODICAUTOM est représentée par la figure A.28. L'utilisateur sélectionne les langues source et cibles puis éventuellement le ou les domaines terminologiques.

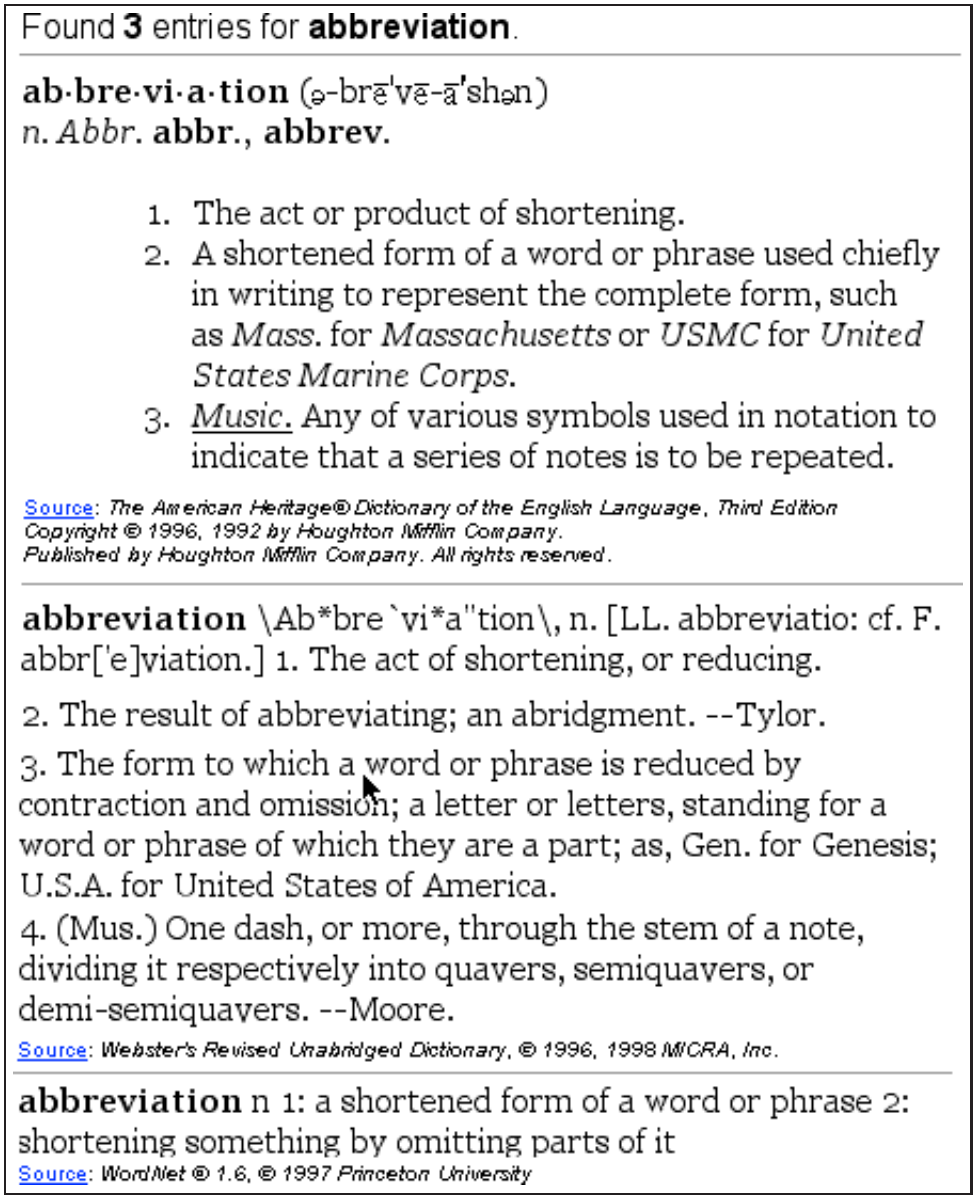


FIG. A.27 – *réponses d'une requête sur dictionary.com*

Résultat

Les outils de bases de données conviennent très bien pour stocker et utiliser des bases terminologiques du type Eurodicautom. La structure des entrées de ces bases est constituée en général au premier niveau d'un mot ou d'une structure attribut-valeur.

Dans l'exemple de la figure A.29, chaque valeur est stockée dans un champ de la base de données. Le champ porte le nom de l'attribut (*Collection, ID Number, Date, Reliability, Subject, Term, Reference*). Par contre, pour une structure d'article plus complexe comme le DEC, il n'est pas possible de décomposer tous les différents éléments de l'article en champs séparés.

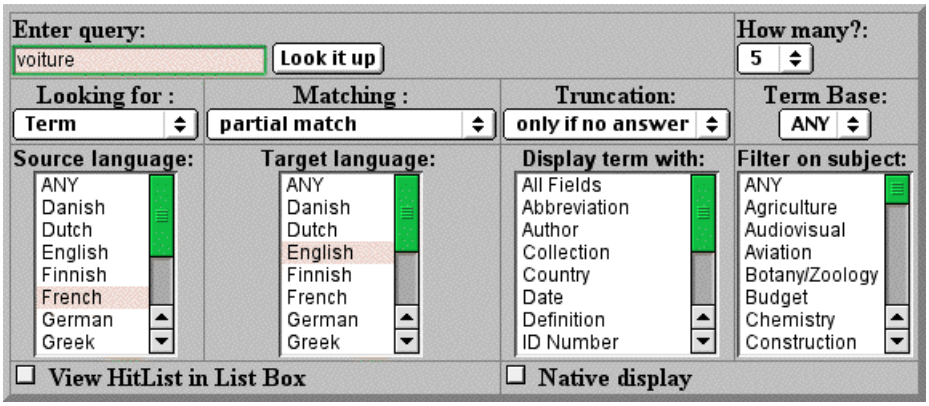


FIG. A.28 – *interface Web de la base terminologique EuroDicAutom*

Document 2 Prev Next HitList Query Options FeedBack

	Terminology Office	Terminology Office, European Commission, Brussels (=BTB)
	Collection	Common Customs Tariff (=TDC77)
	ID Number	0086204
	Date	911022
	Reliability	4
	Subject	conventions (=CEN) rolling stock (=TR4) Unknown CM (=)
French	*Term*	*voiture*
	Reference	TARIF DOUANIER; NOTEX CE 86.09 A
English	*Term*	coach
	Reference	CUSTOM TARIFF; NOTEX CE 86.09 A

FIG. A.29 – *terme* `voiture` *de la base Eurodicautom*

2.2.5. Conclusion

En général, les applications comme les serveurs Web de consultation de dictionnaires ne proposent pas de recherches multilingues. De plus, s'ils font des recherches dans plusieurs dictionnaires, ce sont presque toujours des dictionnaires anglais-autre langue. Il est donc très difficile de trouver de bons dictionnaires bilingues sans utiliser l'anglais.

Les outils d'aide à la recherche comme les lemmatiseurs, les correcteurs orthographiques sont très rare-

ment utilisés, pourtant on les trouve dans d'autres applications. Les options de recherche évoluées comme les expressions régulières ne sont jamais disponibles. Le résultat des requêtes n'est pas personnalisable par l'utilisateur.

Tous ces inconvénients sont compréhensibles dans le cas de serveurs Web, car les propriétaires des dictionnaires ne veulent pas que l'on puisse pirater entièrement leurs ressources.

3. Outils de manipulation de dictionnaires

Nous nous intéressons maintenant aux outils qui permettent de manipuler des dictionnaires déjà existants. Le plus souvent, on désire réutiliser des dictionnaires existants en les transformant et en les combinant suivant plusieurs opérations comme la fusion ou l'intersection.

Il existe des progrès récents dans la récupération et la fusion comme RÉCUPDIC et PRODUCDIC [Doan-Nguyen98a,98b]. Ces méthodologies ont été décrites par Haï Doan-Nguyen dans le cadre de sa thèse.

3.1. Une méthode de récupération de dictionnaires : RÉCUPDIC

3.1.1. Présentation

Cette méthodologie permet de "récupérer" un dictionnaire dans son format d'origine et de le transformer en une structure plus profonde où toute l'information est explicite. Elle inclut deux étapes principales :

– la transduction utilise des outils du type des macros Word^{TM} à base de rechercher/remplacer pour nettoyer le dictionnaire, marquer le plus d'information possible et produire un fichier en format texte "pur" (ASCII). Ainsi, ? pourra être remplacé par $Symbol_S$ ou toute autre notation non ambiguë.

– la traduction par analyse utilise un formalisme nommé H-grammar. L'utilisateur décrit la grammaire du dictionnaire à récupérer en H-grammar. Il ajoute ensuite les actions de construction d'objets et de détection d'erreurs. La détection d'erreurs permet de corriger automatiquement les erreurs les plus fréquentes. Si un détail est faux dans un article, il n'est pas rejeté en bloc. Un compilateur utilise ensuite la description pour construire l'ensemble d'objets constituant une représentation structurée du dictionnaire.

3.1.2. Exemple d'article avant récupération

La figure A.30 représente un article du dictionnaire BABEL au format d'origine avant la récupération. Il s'agit d'un dictionnaire d'abréviations.

```
.COM    Command (file name extension) +
        Commercial Business (Domain Name) [Internet]
```

FIG. A.30 – *article de BABEL avant récupération*

Il arrive fréquemment qu'un article ne vérifie pas la syntaxe indiquée par ses auteurs. Dans BABEL, par exemple, on peut trouver des parenthèses en trop, on a des "+" oubliés, etc. Il faut alors normaliser. L'article de la figure A.29 donné en exemple est correct.

Cet article a une structure implicite : c'est sa présentation qui reflète sa structure. Les différentes informations sont distinguées par leur mise en forme et des caractères spéciaux (les parenthèses "()", le "+", les

crochets "[]", etc.). Il faut donc modifier cet article si nous voulons le réutiliser. Pour récupérer l'article avec l'outil H-grammar, il faut écrire une grammaire de récupération dans ce formalisme. Voyons maintenant comment écrire une grammaire H-grammar.

3.1.3. Grammaire de récupération

Une grammaire de récupération H-grammar se compose de six mots-clefs avec leurs instructions :

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

La figure A.31 montre le squelette d'une règle d'analyse syntaxique.

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

FIG. A.31 – *squelette de règle d'analyse syntaxique de H-grammar*

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

A est un non-terminal. ai1, ai2, ... et ao1, ao2, ... sont respectivement les variables d'entrée et de sortie de la règle.

Dans la partie droite,

- B, C, ... peuvent être un non-terminal, un terminal, le symbole nul §, une action, ou une condition;
- bi1, bi2, ..., ci1, ci2, ..., etc. sont respectivement les expressions d'entrée de B, C, etc;
- bo1, bo2, ..., co1, co2, ..., etc. sont respectivement les variables de sortie de B, C, etc.

Les variables de sortie de la règle (ao1, ao2, ...) doivent se trouver parmi les variables de sortie de la partie droite (bo1, bo2, ..., co1, co2, ..., etc.). Les expressions d'entrée suivent la syntaxe d'expressions de LISP, et peuvent contenir des variables d'entrée (ai1, ai2, ...) et des variables de sortie des unités précédentes de la partie droite (par exemple, ci1 peut contenir bo1, bo2, ...). Le nombre de variables et expressions pour chaque unité peut varier à partir de zéro.

La figure A.32 montre une grammaire H-grammar permettant la récupération des articles de BABEL. Dans les règles syntaxiques, le caractère "§" correspond au symbole nul, le caractère ">" devant un nom de symbole comme ">hwd" indique que ce symbole est terminal. Dans les règles d'analyse lexicale, la notation "_¨10" signifie que le symbole est composé de 10 caractères, le symbole `to-cparen` correspond

```
#grammar babel-glossary /* Acquisition du glossaire BABEL de I. Kind
*/
#syntax-rules :1:  babel-entry(;entry) ->>hwd(;hwd) body(;body) --
(#!babel((trim-whites hwd) body); entry).  :2:  body(;body) ->
sense(;S1) sense*(;S*) -- cons(S1 S*; body).  :3:  sense(;S)->
>exps(;exps) expl?(;expl) subj?(;subj) -- (#!sense((trim-whites exps)
(if expl (trim-whites expl)) (if subj (trim-whites subj))); S).
:4:  expl?(;expl) -> "(" >to-cparen(;expl) ")".
:5:  expl?(;expl) -> §(nil;expl).
:6:  subj?(;subj) -> "[" >to-cbrak(;subj) "]".
:7:  subj?(;subj) -> §(nil; subj).
:8:  sense*(;S*) -> "+" sense(;S1) sense*(;S*1) -- cons(S1 S*1; S*).
:9:  sense*(;S*) -> §(nil; S*).
#start-symbol babel-entry /*symbole de départ de la grammaire*/
#lexical-rules hwd -> _¨10.  /* Headword prend 10 caractères */
exps -> ![+([]*.  to-cparen -> >[)].  to-cbrak -> >[\]].
#lexical-order ("+" "(" ")" "[" "]" hwd exps expl subj)
#working-code (sia-defclass babel () (hwd body))
/* classe définitions */
(sia-defclass sense () (exps expl subj))
(defun trim-whites (string) (string-trim '(#\Space #\Tab #\Newline)
string))
```

FIG. A.32 – *grammaire H-grammar de récupération de BABEL*

à une chaîne de caractères se terminant par une parenthèse fermante, le symbole `to-cbrak` correspond à une chaîne de caractères se terminant par un crochet fermant.

Expliquons maintenant les règles d'analyse syntaxique **#syntax-rules** :

`sense*` est un simple non-terminal. Il ne s'agit pas de `sense` suivi de l'opérateur de Kleene. De même `expl?` est un non terminal normal. `sense*` donne finalement une liste de `sense` et `expl?` donne 0 ou 1 `expl` mais il s'agit de simples conventions de l'auteur de la grammaire.

La règle n°1 produit un article babel `babel-entry` à partir du mot-vedette `hwd` et d'un corps `body`.

La règle n°2 produit un corps `body` à partir d'une liste de sens `sense*`.

La règle n°3 produit un sens à partir d'une définition `exps`, d'une explication `expl` et d'un domaine `subj`.

Les règles n°4 et 5 produisent une explication `expl` à partir d'un texte entre parenthèses "()".

Les règles n°6 et 7 produisent un domaine `subj` à partir d'un texte entre crochets "[]".

Les règles n°8 et 9 produisent une liste de sens `sense*` à partir de 2 sens `sense` séparés par un "+".

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

3.1.4. Exemple d'article après récupération

La figure A.33 montre le résultat de la récupération de l'article BABEL original après compilation avec H-grammar :

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

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

FIG. A.33 – *article de BABEL après récupération (objet LISP)*

lexicaux. Il est aussi très facile de transformer cet article en un autre format exprimant une structure explicite comme XML par exemple.

3.2. Un outil de manipulation de dictionnaires : PRODUCDIC

3.2.1. Présentation

Lorsque les dictionnaires sont "récupérés" de leur format d'origine vers une structure plus profonde, il faut maintenant un outil qui permet de composer de nouveaux dictionnaires à partir du résultat de la récupération. PRODUCDIC permet d'accomplir cette étape. Cet outil ensembliste du type langage de programmation est construit à partir de fonctions LISP qui permettent des manipulations sur les structures profondes récupérées. Il permet 7 types d'opérations sur les dictionnaires :

1. Sélection

La sélection d'un sous-ensemble `B` des éléments de l'ensemble `A` qui satisfont un prédicat `P`, peut être implémentée comme suit :

```
B := NIL;
for-all a in A do
   if P(a) then add(a,B);
```

2. Extraction

L'opération générale de création d'un objet à partir d'une donnée `a` quelconque s'écrit :

create-obj `class` **from** `a` **assign-list** `((slot1 . f1)(slot2 . f2) ...)`

Cette opération crée un objet x de la classe class et affecte les valeurs f1(a), f2(a), etc. aux slots slot1, slot2, etc.

3. Regroupement

regroup-by-partition `set` **partition-by** `func` **unite-into** `class` **assign-list** `list`

Cette opération partitionne `set` avec `func`, et transforme chaque sous-ensemble en un objet de classe `class` en utilisant `list`.

4. Inversion

L'inversion se compose de deux étapes : le regroupement et la division. Pour diviser une donnée `a`, on peut utiliser l'opération suivante :

split `a` **by** `F` **assign-list** `((slot1 . f1)(slot2 . f2)...);`

5. Enchaînement

Les fonctions d'enchaînement sont utilisées dans l'exemple suivant pour produire un dictionnaire français-vietnamien avec un français-anglais et un anglais-vietnamien.

6. Combinaison parallèle

Pour combiner en parallèle deux dictionnaires Dict1 et Dict2 pour obtenir Dict3, on passe par deux étapes :

– création des articles de Dict3 à partir de Dict1,

– intégration des articles de Dict2 à Dict3.

7. Combinaison en étoile

Comme c'est une généralisation de l'enchaînement et de la combinaison parallèle, la combinaison en étoile peut être implémentée avec les opérations présentées précédemment.

3.2.2. Exemple

Dans la production en ligne, un ou plusieurs articles sont produits à chaque demande. Par exemple, étant donné un mot français, on peut créer dynamiquement un article français-vietnamien par enchaînement, en cherchant un article français-anglais et quelques articles anglais-vietnamien, dans les dictionnaires correspondants. Voici les structures des articles de départ :

```
(fe-entry
  (fre string)
  (eng* (list-of (eng string)))
(ev-entry
  (eng string)
  (vie* (list-of (vie string)))
```

Voici la structure de l'article à créer :

```
(fv-entry
  (fre string)
  (vie* (list-of (vie string)))
```

Voici l'algorithme de création dynamique de ces articles :

```
create-FV-from-French(French, FE-dict, EV-dict)=def
FE-entry := find-entry(French, FE-dict);
  V-set := NIL;
  for-all eng in eng*(FE-entry) do
        EV-entry := find-entry(eng, EV-dict);
        V-set := union(V-set, vie*(EV-entry));
  FV-entry := create-FV-entry(fre(FE-entry), V-set);
```

return `FV-entry;`

La production en ligne ne permet pas de vérifier la qualité linguistique des articles produits. Elle sert plutôt à produire des squelettes ou brouillons de dictionnaires dont la qualité linguistique sera révisée par des spécialistes.

Donnée : FE-dict	**Donnée : EV-dict**	**Résultat : FV-dict**
aimer : love, like	**love :** yêu, thuöng **like :** thích	**aimer :** yêu, thuöng, thích

TAB. A.3 – *résultat de l'application de l'algorithme au mot-vedette* `aimer`

3.3. Conclusion

Ces techniques ont été éprouvées puisque RÉCUPDIC a permis de récupérer plus de 1 650 000 articles et que 543 000 articles ont été produits avec PRODUCDIC. Cet environnement est très puissant mais il n'est pas adapté pour un linguiste (lexicographe, lexicologue). En effet, il faut savoir programmer en Macintosh Common Lisp (MCL) pour pouvoir l'utiliser. Ces outils sont spécialisés.

Il faudrait ajouter une interface utilisable par un linguiste mais l'outil reste assez abstrait. Il se pose donc un problème de mise en œuvre pratique. Il faut donc continuer et améliorer cette technique et carrosser l'outil pour le rendre utilisable par un non informaticien.

4. Méthodes de construction de dictionnaires

Il y a plusieurs tâches à réaliser lors de la construction d'un dictionnaire, correspondant à différents "profils" :

- le lexicologue définit les informations qui seront contenues dans le dictionnaire, spécifie leur forme et donne les critères permettant de définir les unités lexicales;
- l'informaticien crée les outils spécifiques au dictionnaire ainsi défini et met au point la méthodologie qui sera utilisée lors de la construction du dictionnaire. Il construit de plus les interfaces nécessaires au lexicographe (poste de travail);
- le lexicographe construit le dictionnaire selon les spécifications ainsi faites, en créant des nouvelles unités ou en complétant des unités déjà existantes.

Dans la pratique, il ne peut y avoir de lexicologue qui ne soit en même temps lexicographe. Dans la suite, nous utiliserons cependant le terme de lexicologue pour désigner la personne qui définit les informations contenues dans le dictionnaire mais aussi qui contrôle le travail des lexicographes (rédacteur en chef de la publication d'un dictionnaire).

La création d'outils pour les dictionnaires pose des problèmes informatiques venant surtout de la masse et de la variété des informations à construire. La construction d'un dictionnaire est un travail mené en collaboration par différents lexicographes qui doivent respecter une cohérence, non seulement en ce qui concerne la forme spécifiée par le lexicologue (abréviations, balises...), mais aussi sur le fond (même critère de sélection des sens, mêmes critères de décomposition en entrées et sous-entrées dans le cas d'homographes...).

Enfin, les choix faits par certains lexicographes peuvent influer sur les décisions que devront prendre d'autres lexicographes (liens syntaxiques ou sémantiques entre entrées). Les outils informatiques construits doivent donc tenir compte de l'aspect distribué du travail de lexicographie.

Lors du travail de lexicographie, il peut arriver que le lexicologue souhaite modifier la structure du dictionnaire afin de mieux prendre en compte certains phénomènes qui ont été mal évalués ou sous-estimés. Cela peut se traduire par un changement des interfaces d'édition et par une modification des éventuels outils de vérification automatique de cohérence. Un outil pour lexicographes doit donc être suffisamment paramétrable et évolutif pour autoriser de tels changements.

Il existe différentes méthodologies pour créer et maintenir des dictionnaires complexes. Nous parlerons de construction :

- *directe* , utilisant l'interface d'une base de données,
- *démocratique* , utilisant un éditeur du commerce comme "pseudo-éditeur syntaxique",

– *classique* , utilisant des éditeurs syntaxiques de type SGML,

– *spécialisée* , utilisant un vrai éditeur syntaxique ad hoc créé pour le dictionnaire en question,

– *en ligne* , par des contributeurs travaillant directement sur le Web.

Nous allons maintenant analyser ces différentes méthodes pour déterminer leurs avantages et leurs inconvénients.

4.1. Constructions "directe" et "démocratique" : exemple du FeM

4.1.1. Introduction

La construction du dictionnaire FeM présenté dans la section 1.2.1 a été en partie réalisée par le GETA. Du fait de la difficulté de trouver suffisamment de lexicographes compétents en français et malais, le travail a débuté sur la base d'un dictionnaire français-anglais. Les entrées (français-anglais)-malais étaient ensuite révisées par un lexicographe expérimenté.

Au départ, la méthode de construction était "directe". Les interfaces de rédaction, récupération et de manipulation du dictionnaire ont été programmées avec 4D, un système commercial de bases de données, mais des problèmes sont rapidement apparus. Les informations stockées dans la base de données sont verrouillées. Il n'est pas possible d'avoir d'inconsistance dans la base. Or, il est parfois nécessaire de passer par un état intermédiaire incohérent comme par exemple deux fois le même article puis de vérifier a posteriori ces erreurs. Cela n'était pas possible avec une base de données. De plus, lors de la rédaction des articles, le fait de ne pas voir le contexte de l'article en cours de rédaction (i.e. les articles "voisins", dans leur totalité ou non) est handicapant. Il n'était pas possible d'avoir une vue globale du dictionnaire.

4.1.2. Méthode de construction "démocratique" des articles

C'est alors que l'idée est venue d'utiliser finalement un outil du commerce comme un éditeur pseudo-syntaxique pour le travail d'indexage. Cette idée a été proposée et implémentée pour la première fois en 1992 par Christian Boitet puis reprise par la suite [Gaschler94a,94b]. Le choix s'est porté sur le logiciel de traitement de texte Word car il présentait de nombreux avantages :

– il fonctionne sur Mac et sur PC,

– les lexicographes savaient déjà l'utiliser,

– il était déjà disponible sur les machines des partenaires,

– contrairement à de nombreux outils d'indexage, il permet de voir tout un ensemble d'entrées de manière compacte et d'utiliser le copier/coller.

Un article du dictionnaire de travail se présente alors en Word sous forme d'une suite de paragraphes. Chaque paragraphe contient un élément d'information. Le style du paragraphe permet de savoir de quel élément il s'agit. Les lexicographes travaillent sur des fichiers RTF (Rich Text Format) qui sont ensuite analysés et intégrés dans la base. Les outils construits pour cela (analyse du RTF) ont été très simples à créer, du fait de l'utilisation d'un paragraphe par élément d'information.

Il a même été possible, dans un premier temps, d'utiliser les outils de Recherche/Remplacement intégrés à Word qui ont permis de créer, sans aucun effort, des fichiers texte balisés utilisables directement par la base centrale. La méthodologie employée est schématisée dans la figure A.34.

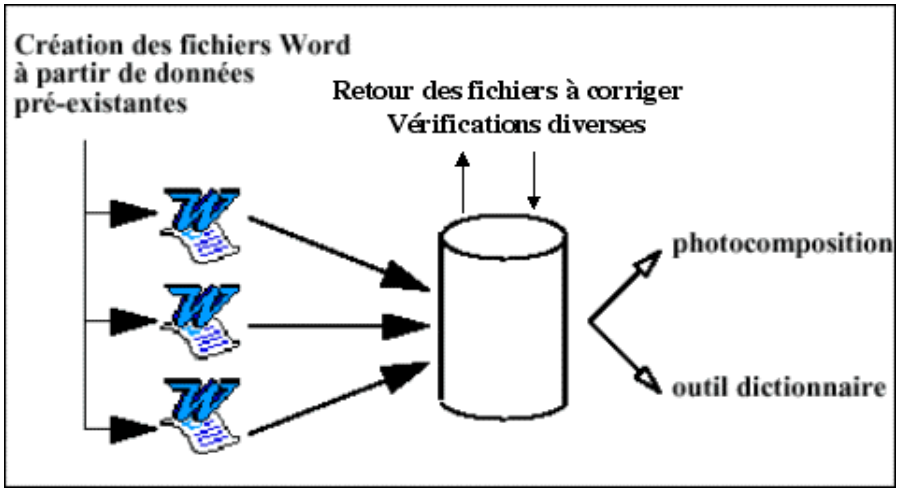


FIG. A.34 – *méthodologie de création du FeM*

4.1.3. Bilan de la méthode

L'utilisation d'un traitement de texte simple a permis la construction de ce dictionnaire de 20 000 entrées dont 50 000 acceptions, 10 000 exemples et 8 000 tournures à partir d'un "brouillon" initial par composition d'un dictionnaire français-anglais et d'un anglais-malais. Le principal avantage de cette méthode est sa simplicité. Les principaux développements informatiques ont porté sur l'exploitation de la base et non sur sa création. Le seul développement nécessaire pour la création de la base a été l'analyse des fichiers Word/RTF.

La distribution du travail entre les différents lexicographes est, elle aussi, très simple mais ce mode de distribution (basé sur l'échange des fichiers Word) est assez rigide. Enfin, la création d'un dictionnaire est un processus assez long. Aussi, il est bon de compter sur un logiciel qui évolue durant tout ce temps. La contrepartie de cet avantage est que l'on est dépendant d'un format propriétaire. Néanmoins le sous-ensemble du format RTF utilisé a toujours été compatible avec les versions de Word utilisées.

L'inconvénient majeur de cette méthode est qu il n'existait aucun outil permettant au lexicographe de vérifier le travail en cours. On ne peut, en effet, constater la malformation d'une entrée que lorsqu'on l'intègre à la base. Aussi, ce processus d'intégration ne peut se faire que sous le contrôle d'un administrateur lexicologue chargé de corriger les erreurs des lexicographes (mauvais choix de style, abréviation inconnue, etc.). Il est aussi possible de faire une partie du contrôle à la source en fournissant les outils nécessaires aux lexicographes sous forme de macros Wrod par exemple.

4.2. Création "classique" avec un éditeur structuré SGML

4.2.1. Introduction

Cette méthode est très répandue car elle est utilisée par tous les éditeurs de dictionnaires imprimés qui ont pris l'habitude d'encoder leurs dictionnaires dans le format SGML. C'est aussi la première méthode informatisée qui a permis de construire des dictionnaires à usage humain. Cette méthode est utilisée pour construire le Dictionnaire Canadien Bilingue (DCB) [DCB].

Les objectifs du projet du DCB sont les suivants :

– la réalisation d'un dictionnaire bilingue canadien (anglais-français, français-anglais) à l'intention

d'utilisateurs ayant de bonnes connaissances linguistiques dans les deux langues (date de publication : 2004);

– la constitution d'une base de données de textes canadiens généraux et spécialisés en anglais et en français;

– la constitution d'une base de données dictionnairique à usages multiples;

– le développement de la recherche en lexicographie bilingue au Canada.

Trois universités canadiennes sont impliquées dans ce projet : l'université d'Ottawa, l'université de Montréal et l'université Laval. Les ateliers de rédaction du dictionnaire sont situés à l'université de Montréal et à l'université d'Ottawa. Les rédacteurs sont pour la plupart des étudiants en linguistique et traduction de ces deux universités. Ce projet a donné lieu à de nombreuses publications. Nous en avons utilisé principalement deux comme sources d'information : [Langlois97] et [Roberts99].

4.2.2. Préparation des articles

Les données sélectionnées par les lexicographes pour un mot-vedette sont compilées dans un article de format prédéterminé. Ce format, qui correspond une DTD SGML, est assez complexe car il est conçu pour tenir compte de tous les renseignements qui peuvent figurer dans n'importe quel article. Néanmoins, il est souple puisque seuls certains éléments d'information sont obligatoires quel que soit l'article : le mot-vedette, la catégorie grammaticale, une division sémantique, par exemple. En outre, le format hiérarchique permet d'une part d'identifier les parties les plus importantes de l'article (par ex. zone d'introduction, division(s) sémantique(s), section des composés, section des expressions figées), et, d'autre part, de subdiviser chacune de ces parties en sous-parties, qui sont elles-mêmes subdivisées plus loin, ce qui permet d'ajouter beaucoup de détails sur chaque partie importante.

Une fois le format de l'article type établi par un comité spécial de l'équipe du DCB travaillant avec un consultant, une DTD a été créée. Cette DTD permet aux lexicographes de rédiger des entrées en SGML. Pour ce faire, ils utilisent divers logiciels, dont WordPerfect SGML.

Les logiciels analysent la DTD pour produire un squelette d'article et proposent des facilités de rédaction qui s'adaptent à la DTD : la liste des balises disponible à un endroit précis de la structure, l'insertion ou la suppression d'un nouvelle balise, etc. De ce fait, la structure de l'article en cours de rédaction est toujours conforme à la DTD. La rédaction est guidée par la structure. La figure A.35 représente l'article `fier-à-bras` tel qu'il apparaît à l'écran lorsque le lexicographe le rédige en SGML.

Le lexicographe a deux options d'affichage : il peut voir les balises comme dans l'exemple ou ne voir que le texte. Cette deuxième option correspond à la vue de l'article que les lecteurs du dictionnaire auront par la suite.

Les articles préparés en format SGML sont ensuite stockés dans une base de données lexicographiques. Il est aussi possible d'imprimer chaque article de façon à ce qu'il ressemble à un vrai article de dictionnaire.

4.2.3. Révision des entrées

C'est sur l'article imprimé que se penchent les réviseurs; en effet, ces derniers préfèrent travailler sur l'article complet, ce qui n'est pas toujours possible lorsqu'ils révisent l'article à l'écran. La possibilité d'étaler côte à côte l'article et les documents consultés et imprimés par les lexicographes lors de la rédaction facilite la révision.

À cette étape, l'informatique joue plutôt un rôle d'arrière-plan. Il arrive, par exemple, que les réviseurs aient besoin de consulter eux-mêmes les corpus pour clarifier certains points ou pour trouver d'autres

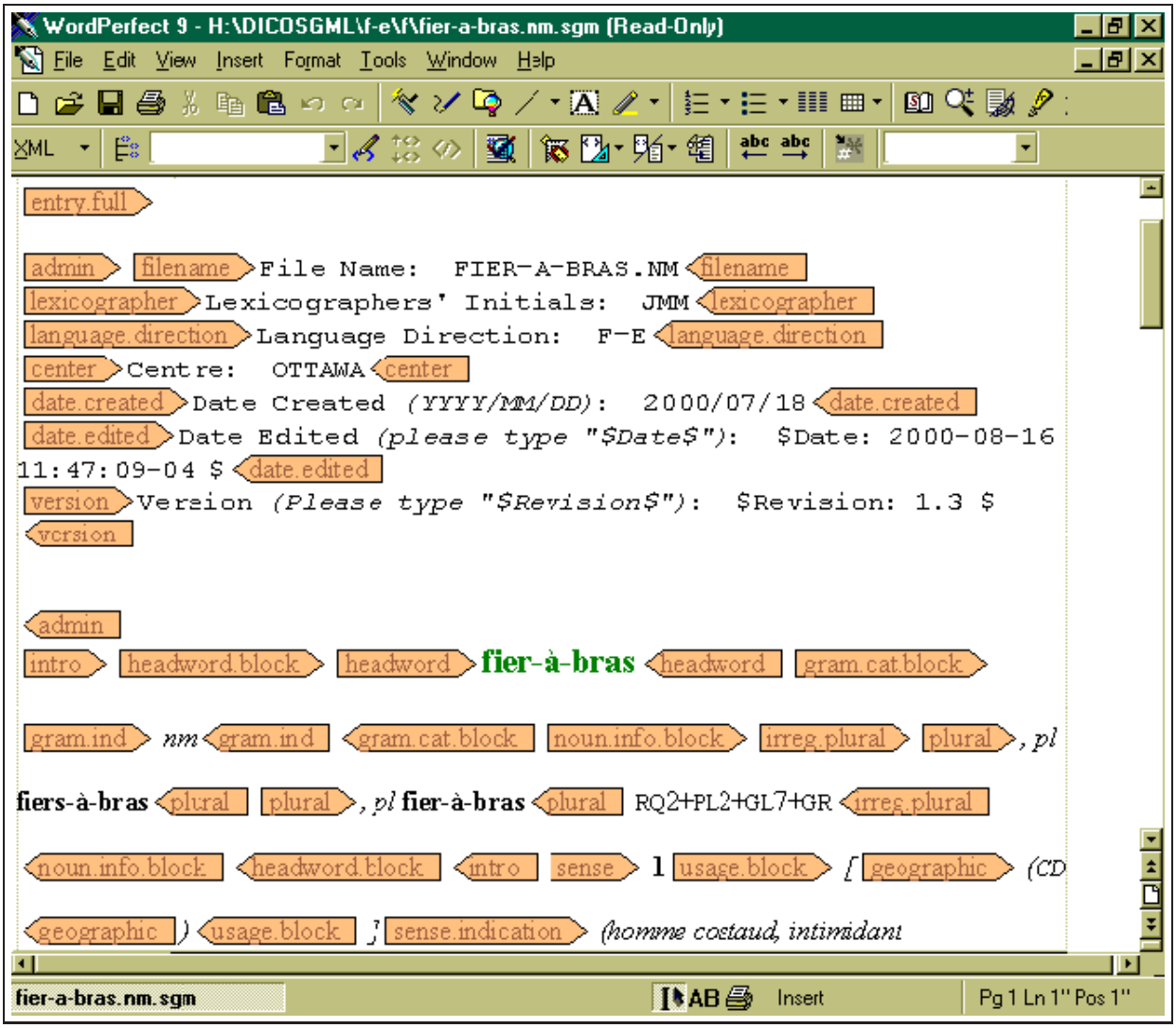


FIG. A.35 – *rédaction d'un article du DCB avec WordPerfect*

exemples d'usage. De plus, après chaque révision, le lexicographe responsable de l'article modifie, en fonction des changements proposés, la version informatisée de l'article. Toutes les versions d'un article sont sauvegardées dans une base de données lexicographiques, ce qui permet à un réviseur d'examiner les changements déjà apportés par les autres réviseurs.

4.3. Construction "spécialisée" pour des dictionnaires de traduction automatique

4.3.1. Introduction

Le logiciel ATLAS [Bachut84a,84b] conçu par Daniel Bachut permet d'introduire des mots nouveaux et les codes associés dans un dictionnaire de TA. Il gère des manuels d'indexage pour linguistes. Son code a été écrit en Pascal et il a été compilé sur un système VM/ESA d'IBM. Les dictionnaires remplis à l'aide

d'ATLAS sont utilisés par le système de traduction ARIANE-G5.

4.3.2. Les manuels d'indexage

Les manuels d'indexage représentent les arbres de décision utilisés par les lexicographes lors de l'indexation d'une entrée. Ils expliquent comment affecter les différents codes utilisés lors de la traduction. Le linguiste édite son manuel avec un éditeur de textes quelconque. Il le compile ensuite avec ATLAS. Lorsqu'ATLAS détecte une erreur, il signale sa position et permet au linguiste de la corriger. La figure A.36 montre un exemple de manuel d'indexage. La figure A.37 montre la forme arborescente pour le manuel papier correspondant.

```
ROOT(q) :  'type of word to be indexed ?'  ;
1 :  'noun'--> NOUN ;
2 :  'verb'--> VERB ;
3 :  'adjective'--> ADJ ;
4 :  'invariant'--> INVAR .
ADJ(Q) : 'what is the adjective type?'  ;
** this includes adj with no comp or sup.
1 :  'comp with MORE'--> AD1 ;
2 :  'comp with ER'--> AD2 ;
3 :  'irregular'--> AD3 .
AD1(Q) : 'ambiguous adjective ?'  ;
1 :  'yes' --> AZ(V): 'obscure';
2 :  'no' --> A(V): 'expensive'.
AD2(Q) : 'what is the type of the adjective ?'  ;
-- type 1 == comp with ER, sup with EST
-- type 2 == comp with ER, sup with ST
-- AMBIGUOUS ie.  'normal ambiguous eg.  'fast'
-- 'normal + comp ambiguous eg.  'light'.
-- 'comp ambiguous eg.
1 :  'type 1, ambiguous'--> A1Z(V): 'light';
2 :  'type 1, non ambig'--> A1(V): 'soft';
3 :  'type 2, ambig'--> A2Z(V): 'saf(e)';
4 :  'type 2, non ambig'--> A2(V): 'unsaf(e)'.
AD3(Q) : 'there are 3 bases to be indexed' ;
1 :  'normal'--> AD4 ;
2 :  'comparative'--> AD5 ;
3 :  'superlative'--> A(V): 'best,driest'.
NOUN(Q) : 'is the noun both regular and variable?'  ; -- example of
irregular noun :  mouse
1 :  'yes'--> NREG ;
2 :  'no'--> NIRG .
```

FIG. A.36 – *exemple de manuel d'indexage source pour l'outil ATLAS*

Dès que la compilation est terminée, le linguiste peut utiliser des fonctions pour rajouter des cartes dans son manuel ou en supprimer interactivement. Il peut aussi visualiser et imprimer tout ou partie de son manuel. Le manuel peut être aussi visualisé sous forme arborescente. L'interpréteur de menus permet

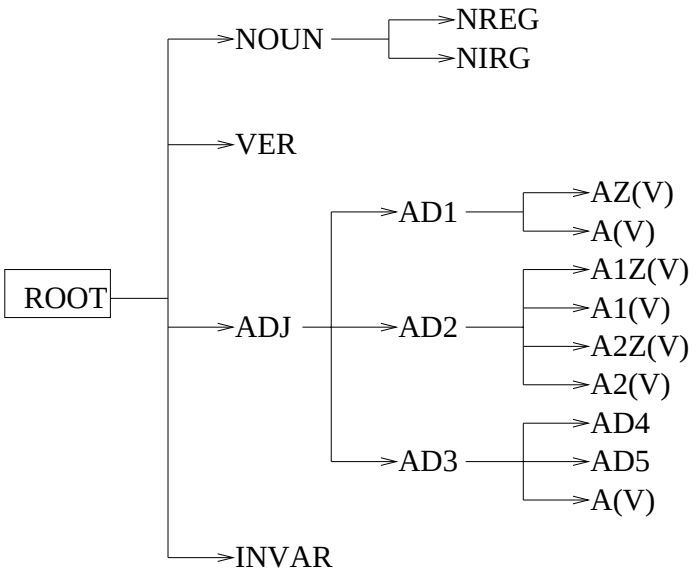


FIG. A.37 – *forme arborescente pour le manuel papier correspondant*

d'indexer interactivement un mot dans un dictionnaire en suivant le format du manuel d'indexage compilé au préalable.

4.3.3. Discussion

ATLAS propose un menu relativement complet de fonctions de manipulation de manuels d'indexage. À l'époque, l'indexation directe dans un dictionnaire (sous XEDIT interagissant avec ATLAS) était cependant très lente, et donc peu utilisée. Il serait très utile tel quel. Seule l'intégration est à revoir.

4.4. Construction "spécialisée" pour des dictionnaires d'usage : l'outil DECID

4.4.1. Introduction

Le projet de recherche NADIA-DEC [Sérasset 97a,97b] (1994-1996), réalisé en collaboration entre le GETA et le GRESLET (département de linguistique et traduction de l'université de Montréal) et soutenu par le réseau LTT de l'AUPELF-UREF avait pour but l'informatisation du Dictionnaire Explicatif et Combinatoire du Français Contemporain (DEC).

Au départ, le dictionnaire était disponible sous forme de fichier Word correspondant à la version imprimée [Mel'tchuk92]. Ce projet a permis de récupérer le DEC vers un format reflétant sa structure interne de manière plus explicite, ce qui a ensuite permis de reconvertir ce dictionnaire vers différents formats cibles. Un de ces formats est associé à un éditeur spécialisé, DECID, lui aussi développé par Gilles Sérasset [Sérasset96, 97c] dans le cadre du même projet.

À cet égard, le projet NADIA-DEC se distingue des autres projets d'informatisation du DEC qui se basent a priori sur une structure informatique simplifiée et qui n'informatisent que le sous-ensemble de données commun entre le DEC et cette structure. Le DEC est un dictionnaire très complexe. L'utilisation de

Word comme interface pour lexicographe n'est donc pas possible (même s'il a été possible par ailleurs de récupérer et de regénérer les fichiers originaux).

4.4.2. L'éditeur spécialisé DECID

Dans la méthodologie adoptée, l'édition se fait directement au niveau de la base lexicale. Lors de la construction de DECID, l'accent a été mis sur le confort du lexicographe. L'interface a été directement inspirée de la version papier du DEC.

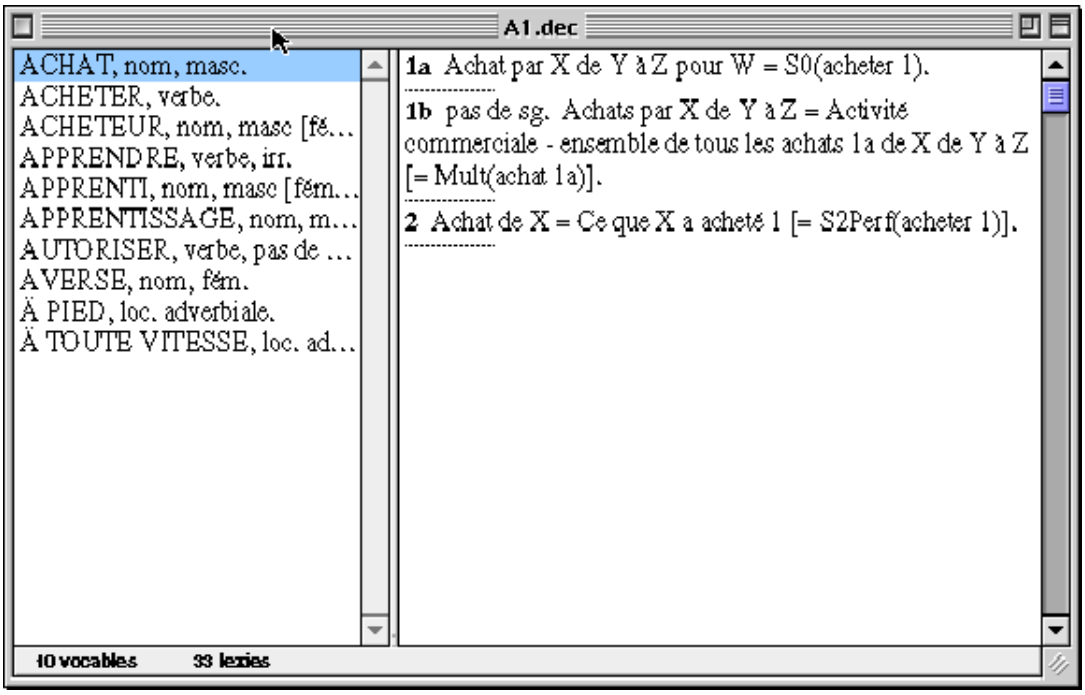


FIG. A.38 – *fenêtre principale de DECID*

Dès que l'on crée un dictionnaire, la fenêtre principale du dictionnaire apparaît (voir figure A.38). Dans la première partie, il y a la liste des vocables. Si l'on clique sur l'un d'eux, les lexies correspondantes apparaissent dans la liste de droite. On passe en mode édition en appuyant sur la touche Entrée du pavé numérique. En double-cliquant sur un résumé, on ouvre la fenêtre de la lexie correspondante.

La seconde fenêtre (voir figure A.39) présente une lexie et permet de l'éditer. L'édition des fonctions lexicales est une tâche difficile lorsque les lexicographes travaillent sur un traitement de texte. Il faut faire attention à bien mettre les majuscules au bon endroit, passer en exposant ou en indice les parties qui doivent l'être, etc. Bref, au lieu de travailler sur la signification d'une fonction lexicale, le lexicographe travaille sur sa forme. Avec DECID, le lexicographe peut éditer la fonction $\mathtt{Perm}_{\ 1}\mathtt{IncepReal}_3\ ^{+}\mathtt{usual}$ simplement en tapant la séquence : perm1incepreal3+usual. La mise en forme est totalement prise en charge par le logiciel.

4.4.3. Discussion

Cet éditeur déjà utilisable, utilisé et utile n'est qu'une première étape vers un outil plus ambitieux. Il faudrait le doter d'un système de vérification de contraintes de cohérence et l'intégrer à des outils plus

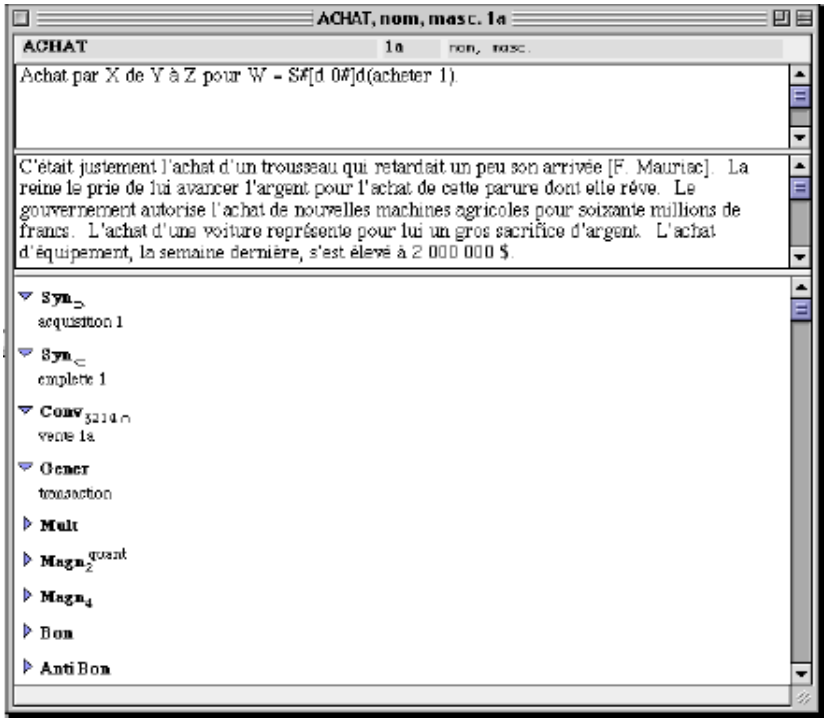


FIG. A.39 – *fenêtre de lexie de DECID*

généraux utilisés par le lexicographe. Comme le DEC est un travail de lexicologie, la structure des entrées est en permanente évolution. Cela rend très difficile la maintenance d'un outil ad hoc. Or DECID a été construit en fonction d'un état précis du DEC à des fins d'expérimentation d'interfaces pour lexicographes; il faudrait donc le transformer en un éditeur plus générique ou au moins paramétrable par les lexicologues.

4.5. Construction "en ligne" par des contributeurs : le projet SAIKAM

4.5.1. Introduction

SAIKAM [Ampornaramveth98,00] est un projet lancé en 1998 par l'ATPIJ (association des professionnels thaïs au Japon) avec l'aide du NECTEC (National Electronics and Computer Technology Center) en Thaïlande et du NACSIS/NII (National Institute on Informatics) au Japon. Son but principal est le développement d'un environnement intégré en ligne pour la construction collaborative d'un lexique japonais-thaï sur Internet.

La base lexicale est implémentée en PostgreSQL. Cette base de données est consultée par des CGIs installés sur un serveur HTTPd Apache. Il existe principalement deux interfaces, une pour les lexicographes et une pour les utilisateurs (consultation et contribution).

4.5.2. Interface de rédaction en ligne

Les lexicographes du monde entier peuvent se connecter à la base lexicale centralisée et mettre à jour son contenu en utilisant des navigateurs Web standard. Les utilisateurs doivent s'identifier au préalable. Ils peuvent ensuite se connecter au système en entrant leur login et leur mot de passe puis éditer le contenu des

dictionnaires en suivant ces trois étapes. À chaque lexicographe est affecté un "panier" de mots.

Le lexicographe doit donc premièrement remplir son panier avec les mots qu'il souhaite réviser. Il les révise ensuite en ligne à l'aide d'un formulaire HTML. Une fois que ses entrées sont complétées, il les retire de son panier. Elles pourront ensuite être révisées par d'autres lexicographes. La figure A.40 montre le formulaire HTML pour l'édition d'un article japonais.

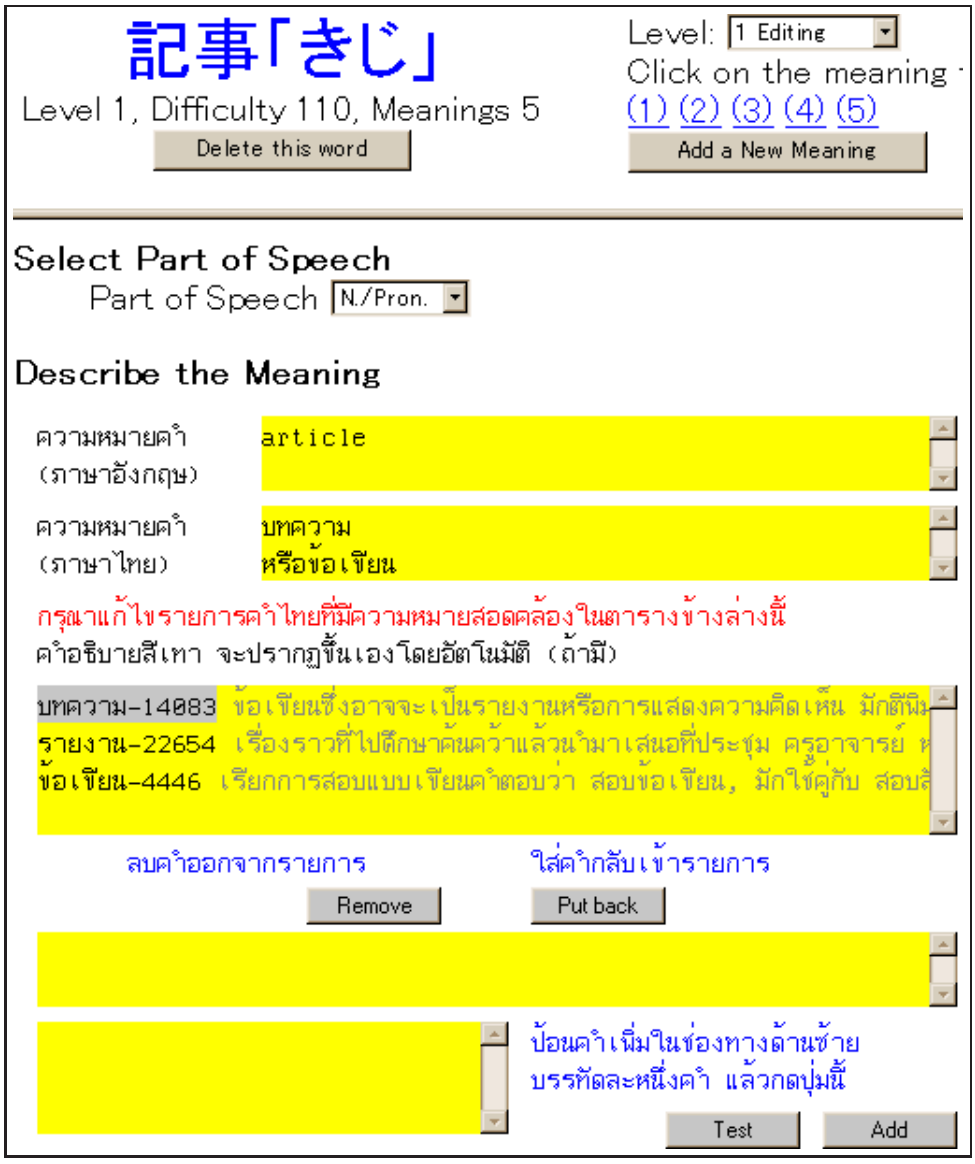


FIG. A.40 – *interface d'édition de SAIKAM*

Le thaï utilise un alphabet spécial dont la méthode de saisie est peu répandue. Elle ne se trouve donc que sur peu de machines. Pour améliorer l'accessibilité de l'interface, un clavier virtuel thaï a été programmé en java. Il permet aux contributeurs travaillant sur une machine qui n'est pas équipée de la méthode de saisie du thaï de pouvoir quand même rédiger les articles en ligne. Lors de la consultation, un choix est proposé à l'utilisateur pour le thaï. S'il a des polices thaï installées sur son ordinateur, le serveur enverra le texte encodé en TIS-160 (encodage spécifique au thaï). Si, par contre, son ordinateur n'en est pas équipé, le

serveur enverra alors des images au format gif représentant le texte thaï à la place.

La base lexicale a été initialisée avec des listes de mots japonais-anglais et thaï-anglais. Les liens initiaux ont été générés en calculant un score mesurant la similarité entre leurs définitions anglaises. La plupart du temps, les lexicographes suppriment des liens existants incorrects. C'est un travail plus facile que d'entrer de nouveaux liens. Pour l'instant, 88 contributeurs se sont enregistrés sur le serveur. La couverture actuelle est d'environ 1 700 termes révisés en ligne et de 2 000 termes révisés hors connexion qui seront intégrés à la base.

4.5.3. Interface de consultation

La figure A.41 montre un article de SAIKAM vu à travers l'interface de consultation.

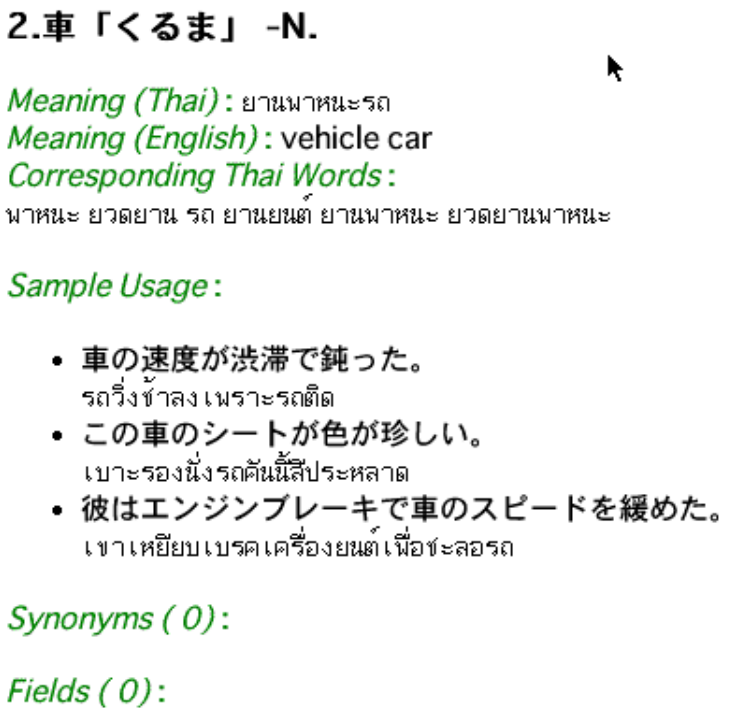


FIG. A.41 – *article* `kuruma`*(voiture) du dictionnaire japonais- thaï*

En plus de la recherche de mots, SAIKAM fournit des outils de recherche de corpus pour le japonais. L'utilisateur peut chercher un mot selon sa fréquence, sa prononciation ou son niveau de difficulté.

4.5.4. Discussion

Ce projet est très prometteur dans le domaine de la coopération sur Internet pour la construction de bases lexicales. Nous pouvons cependant regretter l'absence de lexicologues contrôlant les données. De ce fait, le degré de qualité d'une telle base est difficile à évaluer.

4.6. Conclusion

La méthode démocratique utilise des outils très répandus. Elle est donc facile à mettre en œuvre. Les changements de structure des dictionnaires en cours de construction, pourvu qu'ils restent petits, sont tout à fait possibles. Cette méthode ne bénéficie pas par contre pour l'instant d'outils d'aide à la rédaction et de vérification. Elle n'est pas valable non plus pour des dictionnaires fortement structurés. Enfin, elle n'est pas non plus réactive. Il faut du temps pour distribuer le travail et le réintégrer.

La méthode classique permet de réagir très facilement à un changement de structure des dictionnaires. Il est aussi plus facile d'élaborer des outils de vérification des articles. Par contre, elle ne permet pas non plus de travailler directement en ligne. De plus, il faut équiper tous les rédacteurs d'un éditeur adéquat.

La méthode spécialisée est très pratique pour la rédaction des articles. On peut proposer des outils d'aide à la rédaction et de vérification. De plus, il est possible d'éditer des dictionnaires très complexes comme le DEC. Par contre, ces éditeurs ne s'adaptent pas bien aux changements de structure. Il faut alors les reprogrammer. Enfin, ils ne permettent pas non plus de travailler en ligne en "collecticiel".

La méthode en ligne est très intéressante car elle permet aux contributeurs de travailler simultanément en "collecticiel". Ils n'ont besoin que d'un simple navigateur Web. Elle ne semble par contre valable que pour des dictionnaires à structure simple ou pour des modifications d'articles très localisées. Il faut aussi enrichir cette solution grâce à des outils d'aide à la rédaction. De plus, la gestion des contributions doit être rigoureuse et supervisée par un petit groupe de spécialistes pour éviter la pollution volontaire ou non du dictionnaire par des contributions incorrectes.

Ces méthodes présentent toutes des avantages et des inconvénients bien distincts. Il semble que l'on n'aie pas encore trouvé la solution idéale. Peut-être faudrait-il envisager de pouvoir utiliser toutes ces méthodes en parallèle selon les besoins : travailler en ligne pour de petites contributions spécialisées, et avec un éditeur spécialisé pour la rédaction et la vérification d'articles entiers.

5. Standards liés à la représentation de dictionnaires

Dans cette partie, nous présenterons les standards que nous avons estimés les plus importants pour la représentation des dictionnaires. En effet, pour garantir le plus de portabilité, de compatibilité et de réutilisabilité possible à nos dictionnaires, il faut utiliser au maximum les standards existants à tous les stades de l'élaboration des dictionnaires et surtout lors de la définition de leur structure.

5.1. Pour les caractères : Unicode et ses transcriptions

Le standard ISO UNICODE [ISO93] a été créé en 1993. Les versions du standard sont complètement compatibles et synchronisées avec les versions correspondantes du standard international ISO/IEC 10646. Il résout les problèmes d'encodage des caractères dans différentes langues en spécifiant un numéro unique pour chaque caractère, quelle que soit la plate-forme, quel que soit le logiciel, quelle que soit la langue.

Avant l'invention d'Unicode, des centaines de systèmes de codage de caractères ont été créés. Pas un seul d'entre eux n'était satisfaisant : par exemple, l'Union Européenne a besoin de plusieurs systèmes de codage pour couvrir toutes ses langues d'usage (ISO 8859-1 à 16) [ISO99a]. Même pour une seule langue comme le français, aucun système de codage ne couvrait tous les caractères (il manque entre autres le œ collé dans l'ISO 8859-1 ou ISOLATIN 1 qui est un caractère mais pas une lettre), les signes de ponctuation et les symboles techniques en usage courant.

UNICODE n'est pas un encodage. C'est une table mettant en correspondance un caractère avec un numéro unique. Il est possible de représenter une suite de caractères de la table UNICODE avec plusieurs encodages différents. Comme la table UNICODE possède un nombre de caractères largement supérieur à 256, il n'est pas possible de les représenter sur un octet. La majorité des machines actuelles utilise cependant un codage des caractères sur un octet.

On utilise alors soit le numéro du caractère par exemple U-00FC en hexadécimal pour représenter "ü", soit un système d'encodage variable sur plusieurs octets. UTF-8 (Unicode Transformation Format) représente les caractères Unicode sur un nombre variable d'octets. Les caractères de la table ASCII (American Standard Code for Information Interchange), et, plus précisément, les caractères de la table Unicode compris entre U-0000 et U-007F seront représentés à l'identique sur un octet (le premier bit de l'octet est à 0). Les caractères compris entre U-0080 et U-07FF seront encodés avec deux octets (le premier bit du premier octet est à 1, cela indique qu'il faut lire le deuxième octet pour reconstituer le caractère), etc.

Il devient alors possible de n'utiliser qu'une seule table de codage pour représenter un dictionnaire multilingue comprenant par exemple, du français, du japonais et de l'arabe. L'utilisation d'Unicode se répand de plus en plus, bien que la majorité des plates-formes ne l'utilisent pas encore en natif (c'est à dire qu'il faut toujours effectuer une transformation pour obtenir le caractère).

5.2. Pour la structure des documents : le balisage

5.2.1. Le standard des éditeurs : SGML

SGML [ISO86] est un standard international pour la définition de méthodes de représentation de documents sous forme électronique. C'est un langage de balisage de l'information à l'aide d'étiquettes devenu une norme ISO en 1986.

Ce standard a été principalement utilisé dans le monde de l'édition. C'est pourquoi on trouve principalement des dictionnaires d'usage encodés en SGML comme le NODE ou le OHD décrits plus haut.

C'est un métalangage, c'est à dire un moyen de définir formellement un langage permettant la représentation d'un document électronique. Il permet donc de définir des ensemble d'étiquettes autorisées et requises et leur signification. Un tel ensemble constitue la DTD (Document Type Definition) qui est une sorte de grammaire hors-contexte.

SGML permet de définir des hiérarchies multiples et permet aussi de ne pas fermer ou ouvrir des balises. Pour l'analyse, il faut alors impérativement se servir de la DTD. Dans l'exemple suivant, toutes les balises ne sont pas fermées.

```
<semaine>
  <jours-feries>
    <jour num=6>samedi
    <jour num=7>dimanche
</semaine>
```

5.2.2. Un standard plus récent : XML et ses dérivés

XML [Connolly97] est apparu en 1997. C'est un sous-ensemble simplifié de SGML. Les recherches sur XML ont donné naissance à une recommandation du W3C [XML 1.0] respectant la norme UNICODE [ISO93]. XML rend possible la représentation d'une grande variété d'information. Toutes ces caractéristiques garantissent la lisibilité par de humains, ainsi que la pérennité et la compatibilité avec un nombre croissant d'outils. De plus, comme XML est un sous-ensemble de SGML, la conversion de dictionnaires bien formés au niveau XML n'est pas nécessaire. Comme XML est un format textuel, il sera toujours possible de lire les fichiers originaux encodés en XML.

Un document XML a une structure en forme d'arbre comme SGML mais avec un seul élément racine. Tous les éléments ont une balise ouvrante et une balise fermante. Les balises facultatives de SGML ne sont plus autorisées, ce qui rend beaucoup plus facile ("regard en avant" plus petit) l'analyse des documents XML.

L'exemple précédent en SGML est encodé en XML de la façon suivante :

```
<semaine>
  <jours-feries>
    <jour num="6">samedi</jour>
    <jour num="7">dimanche</jour>
  </jours-feries>
</semaine>
```

Un nombre croissant de normes ont déjà été établies autour de XML :

- définition d'espaces de nom avec Namespace [XML Namespaces];
- description de structures de documents avec les DTD (Définition de Type de Document) et XML Schema [XML Schemas];

– désignation de parties de documents avec XLink [Xlink], XPath [XPath] et Xpointer [XPointer];

– transformation de documents avec XSLT (eXtensible Stylesheet Language Transformation) [XSLT 1.0];

– ensembles d'appels de fonctions standard de manipulation de documents XML avec DOM (Document Object Model) [DOM] et SAX (Simple API for XML) [SAX 2.0];

– métadonnées sur les documents XML avec RDF (Resource Description Framework) [RDF];

– présentation de documents XML avec les feuilles de style XSL-FO (Formatting Objects) [XSL], CSS (Cascading Stylesheet Language) [CSS 2] ou encore DSSSL (Document Style Semantics and Specification Language) [DSSSL, ISO96].

XML est à l'heure actuelle le format idéal pour représenter le contenu des dictionnaires. Les nombreuses normes qui sont définies autour d'XML ainsi que le nombre de plus en plus important d'outils utilisant XML nous ont convaincu de l'utiliser pour nos travaux.

5.3. Pour la représentation du contenu

5.3.1. Proposition d'une structure très riche : le modèle GENELEX

Présentation

Le projet EUREKA GENELEX [GENELEX93] (GENEric LEXicon) s'est étendu principalement sur 3 ans (1990-1993). Le but principal était la construction d'un dictionnaire générique pour différentes langues européennes (le français, l'italien et l'espagnol). L'effort humain fut d'environ 250 hommes-années. GENELEX a produit un dictionnaire public d'environ 3 000 termes mais aussi des dictionnaires privés avec des parties provenant de dictionnaires "propriétaires" reformatés par chacun des membres du projet (ERLI, IBM et Larousse pour le français). Trois compétences distinctes ont été requises : celle du linguiste, celle du formalisateur et celle du lexicographe.

Le projet fonctionna de la façon suivante. D'abord le modèle GENELEX a été décrit à l'aide d'une DTD SGML, implémentant les contraintes imposées par le modèle, en particulier des listes fermées de catégories grammaticales et de traits morphologiques. Ce modèle propose une structuration de données lexicales en 3 couches : morphologie, syntaxe et sémantique. L'unité lexicale est le sens d'un mot.

Un logiciel GENELEX fut ensuite réalisé par chaque partenaire pour exploiter les données. C'est une surcouche logicielle au-dessus de différents systèmes de stockage d'objets persistants. Cela permit à chaque membre du projet de stocker ses données comme il le souhaitait avec ses propres outils. Les dictionnaires d'application étaient ensuite générés par extraction des données nécessaires dans une forme adaptée aux besoins.

Entre les deux, les fichiers SGML étaient chargés et transformés en objets pour être utilisés avec le logiciel.

Exemples

La version 1.5 du dictionnaire AlethDic de GSI-ERLI (LexiQuest depuis 1998) est encodée selon la structure GENELEX. AlethDic se compose de 128 066 unités morphologiques, 85 446 unités syntaxiques et 57 951 unités sémantiques. Il est stocké dans un fichier au format SGML de 42 mégaoctets. Les exemples des figures A .42, A.43 et A.44 sont tirés de ce fichier.

```
<Um_S id="UM54070"
catgram="NOM"
sscatgram="COMMUN"
autonomie="OUI"
usyn_l="US80176">
  <Umg mf="PG101">
  <Lib>semestriel</Lib>
</Umg>
</Um_S>
```

FIG. A.42 – *unité morphologique* `semestriel` *de AlethDic*

```
<Usyn
id="US80176"
description="D_SN">
  <Corresp_Usyn_Usem usem_cible="NO_semestriel_SE1_PG101">
</Usyn>
```

FIG. A.43 – *unité syntaxique* `semestriel` *de AlethDic*

```
<Usem id="NO_semestriel_SE1_PG101"
appellation="semestriel"
trait_sem_valpond_l="TSVP_OBJET_TS_classificateur_de_nom _C
TSVP_PLUS_TS_SEMIOTIQUE_T">
</Usem>
```

FIG. A.44 – *unité sémantique* `semestriel` *de AlethDic*

Discussion

LexiQuest a amélioré le modèle et complété les dictionnaires. Ce modèle est utilisé dans sa base lexicale universelle multilingue appelée LexiDict qui contient plus de 100 000 entrées dans deux ou trois langues avec 150 000 lexies par langue reliées à 100 000 concepts (interlingues).

Le modèle GENELEX a servi de référence dans de nombreux projets par la suite comme EAGLES et PAROLE. Il est très complet et bien détaillé grâce à la conception en objets. Cependant, il faut des spécialistes pour travailler sur un tel dictionnaire, car les informations sont "éclatées" dans de multiples fichiers, et car les interfaces suivent cette organisation en ouvrant autant de fenêtres que de niveaux d'information (morphologique, syntaxique, sémantique, conceptuel).

5.3.2. Essai de standardisation du contenu : la TEI/DEI

1. Présentation

La TEI (Text Encoding Initiative) [Ide95b, Johnson95] est un projet international (1994-2000) qui a eu pour but de développer des directives pour la préparation et l'échange de textes électroniques. Cette action était soutenue par de nombreuses associations ainsi que par le gouvernement américain et la commission européenne.

La TEI P3 propose des DTD pour un grand nombre de textes (proses, vers, drames, dialogues, etc) ainsi que les dictionnaires. Cependant, les experts de la TEI travaillant sur la partie dictionnaire ont conclu qu'il n'était pas possible de proposer une DTD simple pour coder tous les dictionnaires. Les problèmes rencontrés

sont :

– la contradiction entre la généralité de la description, qui doit être applicable à un grand nombre de dictionnaires, et le pouvoir descriptif, c'est à dire la possibilité de décrire avec précision la structure de n'importe quel dictionnaire;

– le besoin de permettre différents usages et vues du dictionnaire encodé, comme par exemple une version imprimée et une version base de données;

– la dualité dans les dictionnaires entre la structure de surface liée à la présentation et la structure profonde liée à l'organisation logique et linguistique, dite aussi microstructure.

Malgré ces problèmes, le chapitre 12 de la TEI P3 propose un certain nombre d'éléments d'information qu'il est intéressant de noter, comme :

– les informations sur la forme du mot (orthographe, prononciation, accentuation, etc.)

– les informations grammaticales (catégories, sous-catégories, etc.)

– les définitions ou traductions

– l'étymologie

– les renvois

– les entrées apparentées

– les informations d'usage

– les exemples.

2. Un exemple

La figure A.45 montre un exemple d'article de dictionnaire.

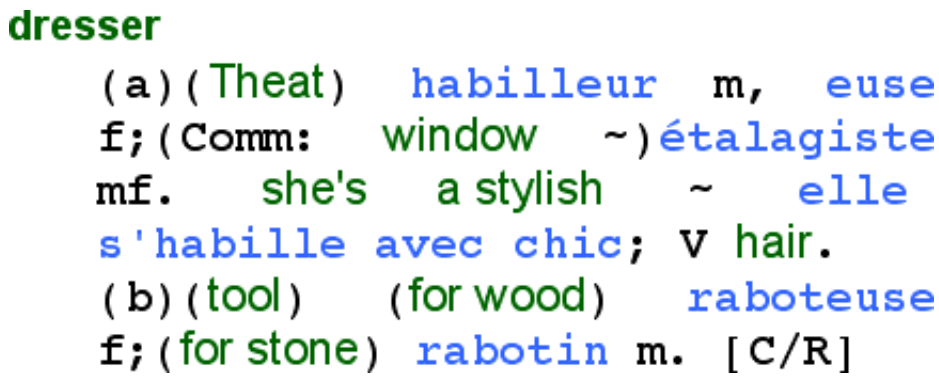


FIG. A.45 – *exemple d'article de dictionnaire anglais-français*

La figure A.46 montre le même exemple encodé avec les balises de la TEI.

```
<entry n='1'><form><orth>dresser</orth></form>
<gramGrp><pos>n</pos></gramGrp>
<sense n='a'>
<sense><usg type=dom>Theat</usg>
<trans><tr>habilleur</tr><gen>m</gen></trans>
<trans><tr>-euse</tr> <gen>f</gen></trans>
</sense>
<sense><usg type=dom>Comm</usg>
<form type=compound><orth>window<oRef></orth></form>
<trans><tr>étalagiste</tr><gen>mf</gen></trans>
</sense>
<eg><q>she's a stylish<oRef></q><trans><tr>elle s'habille avec
chic</tr>
</trans></eg>
<xr type=see>V.<ref target=hair>hair</ref></xr></sense>
<sense n='b'><usg type=category>tool</usg>
<sense><usg type=hint>for wood</usg>
<trans><tr>raboteuse</tr><gen>f</gen></trans>
</sense>
<sense><usg type=hint>for stone</usg>
<trans><tr>rabotin</tr><gen>m</gen></trans>
</sense>
</sense>
</entry>
```

FIG. A.46 – *exemple d'article encodé avec les balises de la TEI*

3. Discussion

La TEI a rencontré des difficultés dans le codage des dictionnaires car il semble très difficile d'imposer une norme. Chaque dictionnaire a une structure propre, et il n'est pas possible de représenter tous les dictionnaires avec la même structure, aussi complexe soit-elle.

Cependant le travail de la TEI sur les dictionnaires est intéressant à double titre. D'une part, la TEI a été élaborée avec de nombreux éditeurs. Cela a au moins permis d'unifier la dénomination des éléments SGML. D'autre part, même si les structures des dictionnaires sont différentes, il est possible de s'entendre sur leur contenu au niveau sémantique. Par exemple, on peut définir précisément ce qu'est un mot-vedette, une catégorie grammaticale, une traduction, un exemple, une étymologie, etc. L'ensemble des balises définies par la TEI sert alors de référence pour les définitions sémantiques des objets qu'ils représentent.

6. Exemples de projets récents basés sur XML

Pour illustrer ce qui précède, nous présentons ici deux projets en lexicographie et terminologie qui utilisent XML pour représenter les données. Il s'agit de projets très récents. Lorsque nous avons commencé notre travail de thèse, leurs conclusions n'étaient pas disponibles. Ce qui fait que nous n'avons pas pu les utiliser pour guider notre travail. Cependant, nous arrivons à des conclusions convergentes.

6.1. Plate-forme de gestion d'une base sur l'hydrographie : DHYDRO

6.1.1. Présentation

Le projet DHYDRO [DHYDRO, Descotte00a,00b] (Dictionnaire Hydrographique Multilingue Normalisé) est un projet MLIS de la communauté européenne. Ce projet a duré 18 mois entre 1998 et 2000. Le consortium a été organisé autour de l'organisme international (l'OHI). Les autres partenaires sont :

- l'équipe langue et dialogue du LORIA en Lorraine;
- le bureau hydrographique international (BHI) à Monaco;
- le service hydrographique et océanographique de la marine française (SHOM) à Paris;
- l'IDS (Institut für Deutsche Sprache) à Mannheim;
- le centre de recherche TERMISTI à Bruxelles.

Le projet DHYDRO avait pour objet de créer sur Internet un espace terminologique multilingue spécialisé dans le domaine de l'hydrographie. Cet espace est conçu autour du Dictionnaire Hydrographique International (DHI), publié par le BHI sous la forme de trois volumes indépendants (en anglais, en français et en espagnol) décrivant environ 7 000 concepts communs. La plate-forme DHYDRO intégrée au site Internet de l'OHI rassemble sur un site web interactif des services utiles aux terminologues, aux traducteurs et aux spécialistes de l'hydrographie, notamment les outils suivants :

- un outil d'édition collégiale et à distance d'une base terminologique multilingue (création ou recherche puis import de fiches multilingues, gestion locale des informations conceptuelles, lexicales et sémantiques selon les droits du rédacteur sur les langues concernées, export puis validation de la fiche modifiée).
- un outil d'exploration de publications multilingues de référence, alignées par paires de langues.
- un espace interactif de communication permettant aux rédacteurs d'interagir efficacement.

– des solutions pour dériver de cette base divers produits tels que des glossaires bilingues ou trilingues, et des dictionnaires monolingues.

– un outil de rétroconversion des 3 dictionnaires monolingues initiaux en une unique base terminologique (approche conceptuelle) multilingue.

– de larges possibilités d'extension du système (ajout de nouvelles langues et de responsables). DHYDRO comporte également une interface web offrant ainsi au plus grand nombre un accès direct aux données hydrographiques (recherche sur le contenu dans les diverses langues et présentation des résultats selon divers formats).

6.1.2. Généricité et flexibilité de Dhydro

Les outils sont adaptés mais totalement indépendants d'une part de tout domaine terminologique et d'autre part de toute plate-forme matérielle. Les efforts ont principalement porté sur :

– l'utilisation intensive de normes et de standards existant dans les champs des technologies de l'information, des modèles de données terminologiques (ISO 12620- Catégories de données, ISO 12200-MARTIF [Melby94, ISO99b]), le format d'encodage XML [XML 1.0] et des techniques de transformation de documents structurés par des feuilles de style XSL [XSLT 1.0].

– la paramétrisation sous la forme de documents XML de la totalité des données manipulées.

– la mise en œuvre d'un scénario éditorial aussi cohérent et robuste que possible et la répartition claire des responsabilités de chaque type d'acteur (consultant, rédacteur, administrateur).

– l'implication continue et soutenue des utilisateurs finals dans les phases de tests de ces outils grâce à la mise en place d'un espace interactif de discussion entre utilisateurs, experts et créateurs de terminologie hydrographique.

6.2. Intégration de lexiques et de bases terminologiques : SALT

6.2.1. Présentation

SALT (Standards-based Access to multilingual Lexicons and Terminologies) [SALT] est un projet commun de la NSF et du cinquième PCRD de la communauté européenne. Commencé en 1999, il se terminera en 2001. Les membres principaux du projet côté européen sont l'Institut für Übersetzer- und Dolmetscherausbildung de l'université de Vienne, l'Institut für Informationsmanagement, Fachhochschule de Köln, l'Accademia Europea di Bolzano, l'University of Surrey, le LORIA, l'Institut der Gesellschaft zur Förderung der Angewandten Informationsforschung e.V. de l'Universität des Saarlandes et, du côté américain, le Translation Research Group de la Brigham Young University et le Kent State University Institute for Applied Linguistics.

Le but de ce projet est d'intégrer des lexiques utilisés en traduction automatique d'une part et des bases de données terminologiques conceptuelles utilisées dans les outils d'aide à la traduction humaine d'autre part. Cette intégration se fera grâce à un format d'échange appelé XLT (eXchange format for Lex/Termdata). Le projet SALT comprend plusieurs tâches :

– tester et raffiner un format d'échange de données lexicales et terminologiques basé sur XML et appelé XLT;

– développer un site web pour tester XLT;

– développer des outils XLT pour concevoir des applications utilisant des données lexicographiques et terminologiques.

6.2.2. Exemple de document au format XLT

Le format XLT est basé tout d'abord sur XML. Il reprend ensuite le standard MARTIF (ISO 12200 [ISO99b], lui-même basé sur ISO 12620) [ISO99c] pour les bases terminologiques à usage humain et le format OLIF [OLIF] pour les dictionnaires de traduction automatique. Les caractères seront représentés en Unicode. Les outils XLT pourront convertir les données entre XLT, OLIF, [GENETER] et d'autres formats. L'exemple de la figure A.47 est un document XLT simple :

```
<martif type="DXLT" lang="en">
  <martifHeader>
    <fileDesc>
      <sourceDesc>
        <p>from an Oracle corporation termBase</p>
      </sourceDesc>
    </fileDesc>
    <encodingDesc>
      <p type="DCSName">DXLTdV04</p>
    </encodingDesc>
  </martifHeader>
  <text>
    <body>
      <termEntry id="ID67">
        <descrip type="subjectField">manufacturing</descrip>
        <descrip type="definition">A value between 0 and 1</descrip>
        <langSet lang="en">
          <tig>
            <term>alpha smoothing factor</term>
            <termNote type="termType">fullForm</termNote>
          </tig>
        </langSet>
        <langSet lang="hu">
          <tig>
            <term>Alfa simítísi tényezõ</term>
          </tig>
        </langSet>
      </termEntry>
    </body>
  </text>
</martif>
```

FIG. A.47 – *document XLT*

Ce document est conforme à la spécification noyau de la structure XSLT appelée XLTcdV04. L'élément <martifHeader> représente la méta-information sur l'entrée. Par exemple, cette entrée provient d'une

base terminologique d'Oracle Corporation.

L'élément <termEntry> représente une entrée de base terminologique. L'attribut ID sert d'identificateur unique dans tout le document. <descrip type='subjectField'> permet d'indiquer le domaine du terme et <descrip type='definition'> une définition. L'élément <langSet> contient une section de langue. Ici, il y a une section pour l'anglais et une section pour le hongrois. L'élément <tig> (term information group) représente une section terminologique composée d'un terme et de l'information associée. L'élément <termNote type='termType'> est la catégorie ISO 12620 du terme.

Conclusion

Cette partie nous a permis d'identifier les limites des outils actuels et d'imaginer des voies de recherche possibles pour nos futures expérimentations.

Pour la consultation de dictionnaires, il serait très intéressant de pouvoir accéder à la méta-information sur les ressources afin de distinguer leur qualité et leur couverture. Les utilisateurs aimeraient aussi pouvoir consulter plusieurs dictionnaires avec la même interface même si ces dictionnaires ont des formats hétérogènes. Ils pourraient de ce fait comparer plus facilement les articles des différents dictionnaires. Il nous semble aussi nécessaire de proposer des outils d'aide en amont ou en aval de la consultation, comme des correcteurs orthographiques et des lemmatiseurs (pour la recherche) ou des conjugueurs (pour l'utilisation). Enfin, il est indispensable que l'utilisateur puisse personnaliser le résultat de ses requêtes au niveau de la structure (informations à cacher, etc.) et de la présentation (style, couleurs, polices, etc.) afin de sélectionner uniquement les informations dont il a besoin dans une grande quantité d'information.

Pour la manipulation de dictionnaires, nous avons besoin de récupérer des ressources existantes, de produire à partir d'elles de nouvelles ressources et de faire des conversion entre formats. Les outils Récupdic et Producdic sont satisfaisants en terme de fonctionnalités. Il reste toutefois un problème d'interface et de portabilité.

Pour la construction de dictionnaires, il est possible de distinguer deux types d'apport : la rédaction d'articles entiers et les contributions localisées sur des parties d'articles. Pour la rédaction, il faut proposer des outils d'aide à la rédaction et aussi un mécanisme d'aller/retour entre les rédacteurs et la base pour pouvoir réviser le travail accompli. Pour les contributions, il faut des outils simples fonctionnant directement en ligne et permettant de partager les contributions entre plusieurs utilisateurs. L'utilisation d'un "tampon" est nécessaire pour donner un statut "provisoire" aux nouvelles données. Ensuite, il faut mettre au point une procédure de validation /correction /intégration des données.

Pour la structure interne des dictionnaires, nous souhaitons nous appuyer sur des standards existants pour garantir la portabilité et la compatibilité avec un maximum d'outils existants et à venir. C'est pourquoi nous pensons utiliser les standards UNICODE et XML ainsi que le résultat des recherches menées par la TEI. Nous devons enfin trouver un langage de structuration des informations lexicales générique pour représenter la grande variété des théories linguistiques et des structures existantes.

B : Exploration de nouvelles directions, bilan et cahier des charges d’un environnement avancé

Introduction

Nous avons voulu expérimenter nos idées au moyen de prototypes rapidement écrits pour tester différentes idées concernant la représentation et le traitement distribué des dictionnaires. Les projets de recherche menés au GETA et la grande variété des outils et des ressources linguistiques au centre européen de recherche de Xerox nous ont considérablement aidé dans nos expériences.

Essayer de construire directement un environnement qui permette à la fois la manipulation, la consultation et la construction de dictionnaires nous a semblé prématuré. C'est pourquoi nous avons préféré explorer plusieurs pistes séparément avant de spécifier les bases de notre environnement. Ces expérimentations nous ont permis de déterminer des solutions partielles possibles des problèmes posés par la conception d'un environnement de bases lexicales.

Nous avons commencé par le problème de la *consultation en ligne* de ressources lexicales. Notre but est de permettre de consulter la méta-information sur les ressources disponibles, de consulter plusieurs ressources hétérogènes à la fois à partir d'un navigateur, et aussi de regrouper des ressources locales et distantes avec un résultat transparent pour l'utilisateur. Nous avons réalisé pour cela DictList et DicoWeb. Le second est en usage expérimental continu depuis trois ans sur le site interne de XRCE. Une version publique est cependant disponible à l'adresse suivante [DicoWeb].

Nous nous sommes attaqué au problème de la *construction de dictionnaires* sous deux angles. D'une part, les besoins du projet UNL nous ont amené à améliorer la méthode "démocratique" de la construction du dictionnaire français-anglais-malais FeM (Word™ utilisé en "pseudo éditeur syntaxique") exposée en première partie. D'autre part, nous avons expérimenté une autre méthode de construction en ligne de dictionnaires à structures simples pour apprenants de langues étrangères (prototypes DicoSzótár [DicoSzótár] pour le hongrois et Nihongo [Nihongo] pour le japonais).

Nous avons réalisé et expérimenté plusieurs *outils améliorant la consultation*, grâce à l'ajout de modules d'aide en amont et en aval de la consultation, comme des lemmatiseurs, des correcteurs orthographiques, des conjugueurs, de svisualisateurs d'objets complexes comme des arbres, etc.

Nous avons ensuite cherché à améliorer la coopération entre utilisateurs et contributeurs en étudiant des outils permettant l'*annotation de documents* . Enfin, nous avons testé la consultation de ressources lexicales par une autre application.

Le bilan de ces expériences nous a enfin permis d'élaborer un cahier des charges d'un environnement de manipulation, de construction et consultation de dictionnaires plus avancé et tenant compte de nos observations.

1. Expériences sur la consultation en ligne

Les laboratoires GETA et XRCE disposent de nombreuses ressources lexicales pour leurs expériences. De plus, ils sont équipés de serveurs web accessibles par tous leurs membres. Ces ressources sont riches et très variées. Elles sont stockées dans des formats hétérogènes. Malheureusement, ces formats de stockage sont illisibles directement par des humains. C'est pourquoi les conditions étaient réunies pour expérimenter des serveurs Web de dictionnaires à usage humain.

1.1. Consultation de méta-informations sur les ressources

1.1.1. Présentation de l'outil

Pour y voir plus clair dans les ressources lexicales, nous avons d'abord organisé et standardisé leur rangement dans des répertoires, puis nous avons mis en place DictList, un outil de consultation de la méta-information dont nous disposions sur ces ressources. Cet outil devait permettre de répondre aux questions du type : avons-nous un dictionnaire français-russe ? Quelle est sa qualité et sa couverture ? Où se trouve-t-il ?

Plutôt que de stocker les méta-données dans un lieu qui leur est propre, nous avons choisi de les mettre à côté des données qu'elles représentent. ainsi, dès que l'on ajoute un nouvel ensemble de données, les méta-données sont ajoutées en même temps. Cela permet aussi de consulter ces méta-données directement lorsqu'on accède au système de fichier où sont stockées les données.

1.1.2. Protocole de nommage des fichiers

Pour permettre à notre outil d'accéder directement aux fichiers de méta-données, et pour clarifier l'organisation des fichiers, nous avons instauré un protocole de nommage des répertoires et des fichiers de données. Cela permet par exemple de prendre en compte automatiquement un nouvel ensemble de données sans modifications de l'outil.

Les ressources sont dans la mesure du possible rangées dans le même répertoire. Chaque ressource est placée dans un répertoire dont le nom est composé du nom de la ressource suivi des langues présentes dans la ressource dans l'ordre alphabétique. Chaque nom de fichier de ressource contient le nom du dictionnaire suivi de la langue source puis des langues cibles présentes dans le fichier. Le nom de version ainsi que l'encodage sont éventuellement ajoutés. L'extension représentant le format (txt, rtf, xml, html, sgml, etc.) termine le nom du fichier.

Par exemple le fichier représentant le volume français-anglais du dictionnaire Oxford-Hachette encodé en sgml aura comme nom : `ohd_fr_en-v2-ISO-8859-1.sgml`. Il sera rangé dans le répertoire de nom `OHD_en-fr` qui contiendra aussi le fichier `ohd_fr_en-v1-ISO-8859-1.sgml` .

1.1.3. Structures internes utilisées

Un fichier encodé en XML est placé dans le répertoire de chaque ressource. Il contient un certain nombre d'informations. Ce sont des méta-données sur ces ressources. Nous trouvons :

– le nom de la ressource,

– la catégorie (monolingue, bilingue, multilingue),

– les langues sources et cibles,

– le domaine (général, médecine, etc.),

– les dates de création de la ressource,

– les auteurs,

– le responsable,

– des informations complémentaires.

Pour chaque fichier, nous trouvons :

– le nom du fichier,

– la date d'installation,

– la version,

– l'encodage (ISO-8859-1, UTF-8, etc.),

– le nombre d'articles,

– le nombre de traductions pour des dictionnaires multilingues,

– des informations complémentaires.

Par exemple, le fichier XML de la figure B.1 décrit le dictionnaire EuroWordNet.

Une feuille de style écrite en XSLT [XSLT 1.0] permet de produire un fichier README au format texte à partir des fichiers XML (voir figure B.2). Ces fichiers sont placés dans les répertoires des ressources. Cela permet aux personnes parcourant directement l'arborescence de pouvoir lire ces fichiers expliquant le contenu des répertoires.

1.1.4. Architecture et interface de DictList

Un petit script CGI programmé en Perl permet alors de consulter à travers le web les informations disponibles sur les ressources installées localement. La figure B.3 représente l'architecture de cet outil.

Pour se servir de l'outil, l'utilisateur a besoin d'un navigateur web. L'interface est un formulaire HTML inséré dans la partie gauche d'une page web. L'utilisateur peut effectuer une recherche multicritères en choisissant parmi la catégorie, les langues sources et cibles, le domaine et le format. Lorsqu'il appuie sur le bouton "search", la requête HTTP est envoyée au CGI écrit en Perl. Le CGI consulte et sélectionne les fichiers README encodés en XML en fonction des critères, transforme le XML en HTML et renvoie le résultat dans la partie droite de la fenêtre. La figure B.4 représente l'interface et le résultat d'une requête sur DictList.

```
<dictionary-readme>
  <readme-info>
    <creation-date>6 May 1999</creation-date>
    <author>Laurent Griot</author>
  </readme-info>
  <general-info>
    <name>EuroWordNet</name>
    <category>multilingual dictionary</category>
    <domain>general</domain>
    <source-language>English</source-language>    <creation-date>see
version</creation-date>
    <responsible>Laurent Griot</responsible>
    <info>A directory containing the Eurowordnet data.</info>
  </general-info>
  <files-list>
    <file>
      <name>EuroWordNet1.6.txt</name>
      <creation-date>6 May 1999</creation-date>
    </file>
  </files-list>
</dictionary-readme>
```

FIG. B.1 – *description du dictionnaire EuroWordNet*

```
DICTIONARY README
=====================================
README INFO
creation date:  6 May 1999
author:  Laurent Griot
=====================================
GENERAL INFO
name:  EuroWordNet
category:  multilingual dictionary
domain:  general
source language:  English
creation date:  see version
responsible:  Laurent Griot
info:  A directory containing the Eurowordnet data.
=====================================
FILES LIST
file name:  EuroWordNet1.6.txt
creation date:  6 May 1999
```

FIG. B.2 – *description du dictionnaire EuroWordNet en format texte*

1.1.5. Discussion

Nous avons amélioré l'organisation des ressources lexicales grâce à notre méthodologie et à l'outil DictList. La méta-information permet de mieux connaître les ressources. Elle est indispensable pour faire

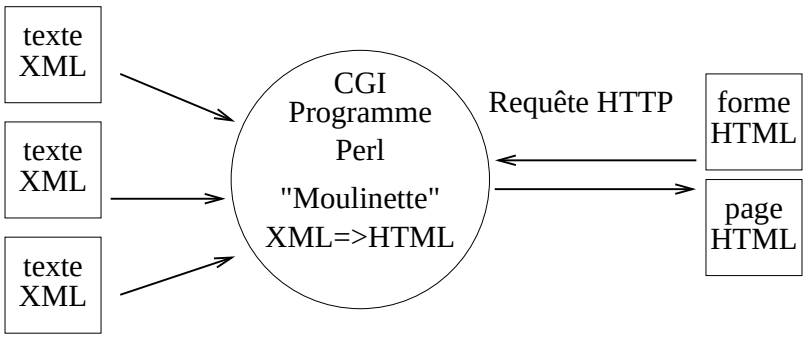


FIG. B.3 – *architecture de DictList*

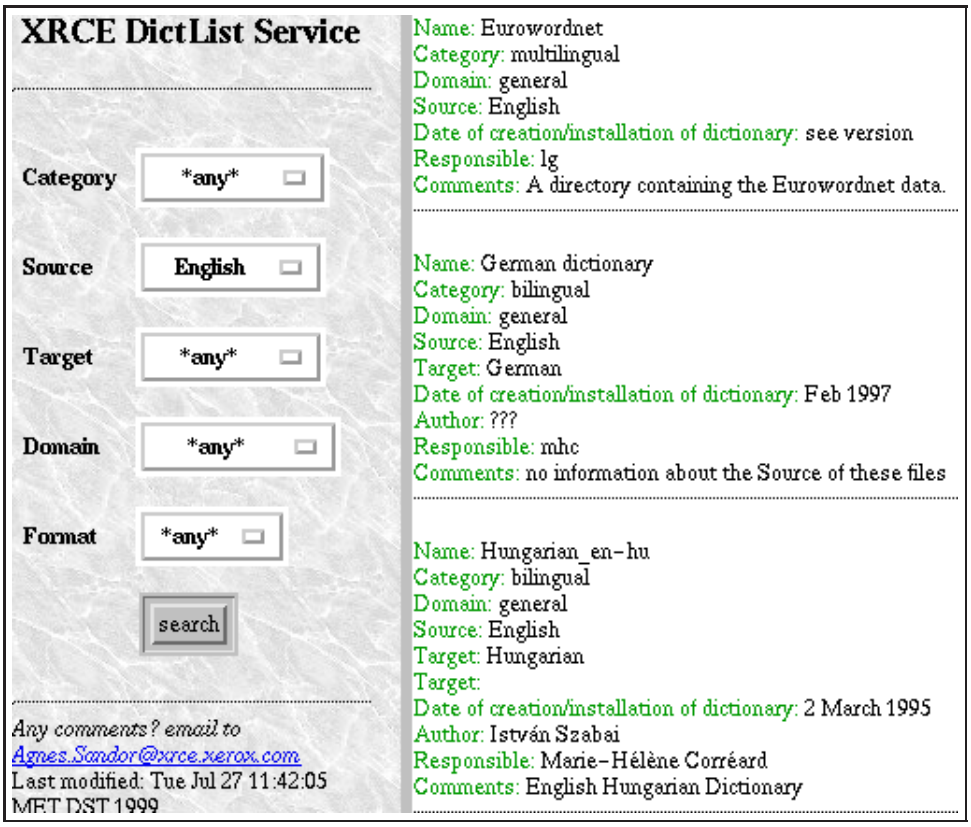


FIG. B.4 – *copie d'écran du serveur DictList*

des choix. L'utilisation de fichiers XML pour stocker les données ainsi que des feuilles de style XSLT a aussi été concluante.

Cependant, la majorité des utilisateurs regrettent de n'avoir que de la méta-information sur les ressources. Ils souhaiteraient accéder aussi directement à un extrait de ces ressources. Nous avons donc conçu un autre outil répondant à cette demande.

1.2. Consultation de plusieurs ressources hétérogènes : DicoWeb

1.2.1. Présentation

Les ressources lexicales disponibles sont stockées dans des fichiers texte, chacune selon son propre format. Après les avoir ordonnées selon la méthode définie pour le serveur DictList, nous avons construit une maquette qui permet de consulter tous ces fichiers texte en même temps à la volée et d'afficher les différents articles dans une seule fenêtre.

Nous avons appelé cette maquette "DicoWeb" pour "Dictionnaires sur le Web". DicoWeb est un serveur de dictionnaires conçu pour un usage humain. Il sert pour des expérimentations à XRCE. Pour des raisons légales, il n'est pas accessible au public. Nous présenterons son interface, son architecture et quelques points importants.

1.2.2. Architecture de DicoWeb

La méthode utilisée pour l'outil DictList a été reprise et améliorée. Un script cgi écrit en Perl fait la liaison entre l'utilisateur, les analyseurs morphologiques et les dictionnaires. Lorsque l'utilisateur a choisi ses langues source et cibles puis tapé son entrée, le résultat est envoyé au script. Si l'analyse morphologique est sélectionnée, ledit script envoie l'entrée à l'analyseur morphologique correspondant à la langue source. La réponse est ensuite décodée. La figure B.5 montre l'architecture générale de DicoWeb.

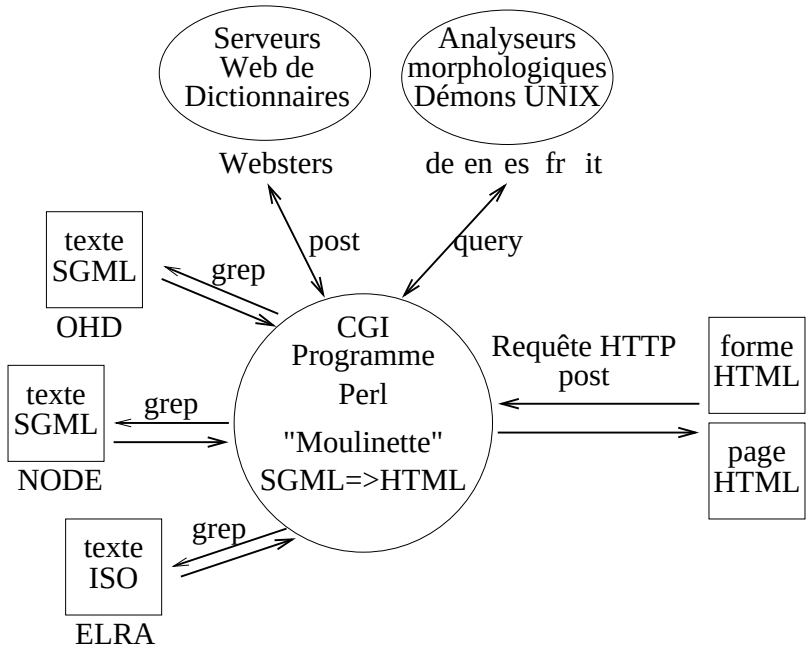


FIG. B.5 – *architecture générale de DicoWeb*

Les dictionnaires sont alors sélectionnés en fonction des langues cibles et les fichiers texte originaux sont parcourus par le script qui cherche l'entrée décrite par une expression régulière Perl. Les lignes vérifiant l'expression régulière sont alors sélectionnées puis passées à travers une "moulinette" qui transforme le texte source en HTML. Le tout est renvoyé sous forme de page HTML à l'utilisateur.

1.2.3. Interface de DicoWeb

La figure B.6 montre l'interface Web de DicoWeb. L'utilisateur sélectionne la langue source, dans laquelle il va taper l'entrée. Il peut sélectionner ensuite des langues cibles et/ou des ressources. Par défaut, toutes les langues cibles et toutes les ressources locales sont sélectionnées.

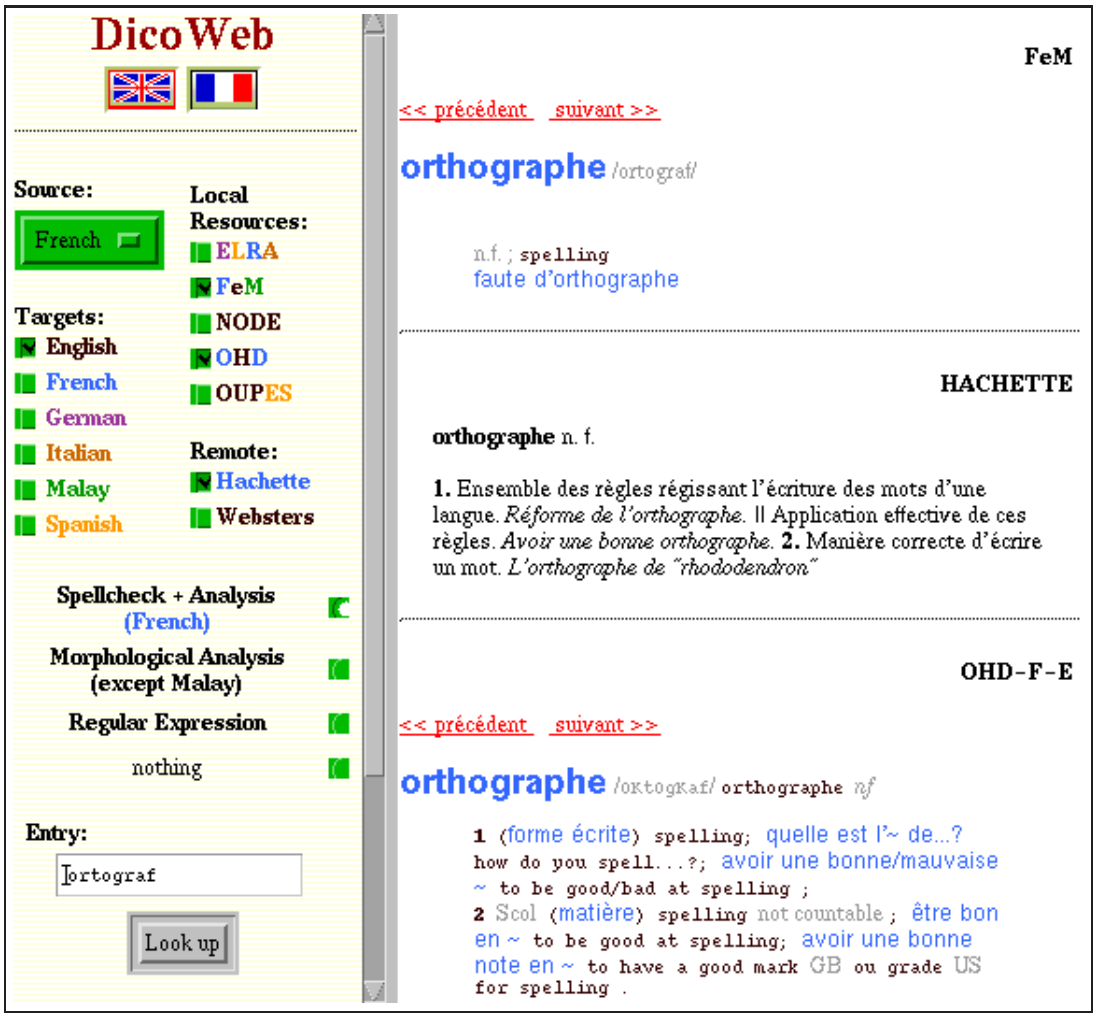


FIG. B.6 – *Interface Web de DicoWeb*

Il peut, avant de consulter les dictionnaires, envoyer le mot qu'il vient de taper à un analyseur morphologique en cochant la case correspondante. S'il clique sur les boutons `previous` ou `next` des parties OHD ou NODE, il pourra consulter les entrées précédentes et suivantes correspondant, dans l'ordre alphabétique, à celles affichées.

Dans un souci de clarté, nous fixons au départ une seule couleur ainsi qu'une police différente pour chaque langue, qu'elle soit source ou cible, et cela pour tous les dictionnaires. L'utilisateur s'habituera ainsi à ce mode de représentation. Pour construire nos interfaces, nous nous efforçons de suivre les principes cognitifs définis par Joëlle Coutaz et son équipe [Coutaz88].

1.2.4. Fonctionnalités originales

Recherche de l'entrée

Selon les langues sélectionnées, le script consulte les dictionnaires correspondants. Par exemple, si l'utilisateur ne sélectionne que l'anglais comme langue source et cible, le script consultera le dictionnaire NODE monolingue anglais et la base ELRA. S'il choisit le français comme langue source et l'anglais comme langue cible, le script consultera le dictionnaire FeM, le OHD français-anglais et la base ELRA. S'il choisit l'espagnol comme langue source, le script ne consultera que la base ELRA.

La base ELRA et le dictionnaire FeM sont multilingues. Si l'utilisateur n'a sélectionné que certaines langues cibles, par exemple l'anglais, le malais du FeM et les autres langues de la base ELRA ne sont pas affichées. Cela permet de personnaliser l'affichage en fonction des demandes de l'utilisateur. Les dictionnaires ne subissent aucune modification, ils sont consultés directement dans leur format d'origine.

Perl dispose d'un puissant langage d'expressions régulières. À chaque dictionnaire correspond une expression régulière. Pour chercher une entrée du OHD, par exemple, on utilisera le patron : `/<[hc]w>$entry</` où `$entry` représente l'entrée demandée.

Le dictionnaire FeM est unidirectionnel, du français vers l'anglais et le malais. Cependant, grâce aux expressions régulières, nous pouvons chercher la traduction d'un mot malais en français ou plus exactement, chercher dans quelles entrées françaises apparaît ce mot malais. L'utilisateur pourra alors se faire une idée de la traduction française de ce mot.

Pour la recherche dans la base ELRA, le script cherche d'abord les numéros de concept dans le dictionnaire correspondant à la langue source, puis cherche dans les dictionnaires correspondant aux langues cibles les traductions correspondant aux numéros de concept.

Prétraitement de l'entrée

Le prétraitement de l'entrée consiste à utiliser tous les modules permettant soit de corriger les éventuelles erreurs de l'utilisateur, soit de trouver, à partir d'une forme de surface, le ou les mots-vedettes correspondants.

Nous proposons donc, selon la disponibilité des modules dans chaque langue, d'utiliser d'abord un correcteur orthographique et de fautes de frappe. Pour l'outil DicoWeb fonctionnant à XRCE, nous avons utilisé le correcteur orthographique basé sur des règles de phonétique utilisé dans la nouvelle version du cédérom du dictionnaire Hachette-Multimédia. Si l'utilisateur tape `ortograf`, nous obtiendrons après correction `orthographe` (voir figure B.6).

Une fois l'entrée corrigée, pour obtenir une liste des lemmes correspondant à une forme de surface, nous utilisons un analyseur morphologique. L'entrée est d'abord envoyée par le script à l'analyseur morphologique correspondant à la langue source. Le résultat est ensuite décodé de façon à fournir une liste d'entrées plausibles. Ainsi, si l'utilisateur tape l'entrée `cochons`, la liste des nouvelles entrées sera `cocher` et `cochon`. Les analyseurs morphologiques sont des démons UNIX qui tournent en permanence. Ils répondent à des requêtes de différentes applications et étaient déjà utilisés avant que nous ne programmions cette interface.

Le but ici n'est pas de fournir une véritable recherche aidée par le contexte, mais de proposer une petite aide supplémentaire. En effet, il existe des outils spécialisés dans la recherche à l'aide du contexte. Ces outils évitent par exemple que, lorsque l'utilisateur tape `cochons`, il obtienne l'entrée `cocher , nom commun` qui n'a rien à voir avec sa première demande. Notre système n'est pas conçu pour résoudre ce genre de problème. Cependant, l'analyse morphologique de l'entrée peut s'avérer utile lorsqu'on ne maîtrise pas la langue source. La liste des nouvelles entrées est ensuite utilisée par le script pour consulter les dictionnaires.

L'utilisateur peut profiter directement du langage d'expressions régulières. En effet, s'il tape une entrée

sous forme d'expression régulière, celle-ci sera interprétée telle quelle par le script. Par exemple, si l'utilisateur tape `b.ll` (ici, le point correspond à n'importe quel caractère) et sélectionne l'anglais comme langue source, il obtiendra les entrées `ball, bell, bill` et `bull`. Des petits exemples lui sont donnés en ligne, ainsi que quelques explications.

Entrée précédente et suivante

Pour les dictionnaires classés par ordre alphabétique (ici tous sauf la base de concepts ELRA), il est possible de consulter les entrées précédant et suivant celles affichées. Pour cela, lorsque le script consulte un dictionnaire à la recherche d'une entrée, il compte les lignes. Lorsque l'utilisateur demande l'entrée précédente ou suivante, le script utilise ce numéro de ligne pour faire sa recherche. Elle s'effectue donc plus rapidement que lorsque le script effectue une recherche à l'aide d'une expression régulière. L'utilisateur se retrouve partiellement dans le contexte de la lecture d'un dictionnaire papier où le contexte de l'entrée est directement sous ses yeux.

Pages fabriquées à la volée

Pour éviter de convertir à chaque fois le texte source en HTML, nous aurions pu convertir en une seule fois tous les dictionnaires source. Cependant, même si cette solution réduit le temps d'attente lors de la recherche d'une entrée, elle présente deux inconvénients importants. D'abord, la fabrication à la volée des pages HTML permet d'une part de respecter le copyright en interdisant aux utilisateurs de récupérer entièrement le dictionnaire en une seule fois, ensuite, on peut retoucher le rendu final directement en modifiant le script Perl.

Ajout d'une nouvelle ressource

Les critères que doivent satisfaire les nouvelles ressources pour être ajoutées au système sont simples : l'entrée doit soit être disposée sur une seule ligne, soit pouvoir être extraite à l'aide d'un outil simple comme sggrep (grep pour SGML). Il suffit alors de formuler l'expression régulière adéquate pour trouver l'entrée du dictionnaire, puis d'associer une feuille de style au texte pour le rendu final.

La programmation d'une première version fonctionnelle de DicoWeb a pris moins d'un mois. Nous avons par la suite ajouté de nouvelles ressources très facilement, avec très peu de développement.

DicoWeb accède aux fichiers texte des ressources via des index également stockés dans des fichiers texte. Aucune information n'est stockée en mémoire. Malgré ce handicap, le temps d'accès moyen pour un article est de moins d'une seconde, ce qui est parfaitement acceptable pour un utilisateur humain. Les ressources sont accédées directement.

1.2.5. Discussion

DicoWeb n'est pour l'instant accessible qu'en interne par environ 110 personnes, avec actuellement plus de 100 accès par jour, ce qui démontre son utilisabilité.

Dans cette application, nous utilisons directement le format et la structure d'origine des ressources lexicales. Pour l'affichage, nous transformons à la volée le format en HTML pour pouvoir utiliser un navigateur. Si plusieurs articles correspondant au même mot-vedette sont trouvés, ils sont affichés à la suite.

Avantages

L'avantage majeur de cette technique très simple est qu'elle permet de visualiser le contenu de plusieurs ressources à la fois. De plus, du fait de l'utilisation directe du format d'origine, nous n'avons pas besoin de

récupérer les ressources pour les transformer dans un autre format. La fidélité par rapport à l'original est aussi garantie. Enfin, il n'y a pas de perte d'information.

La transformation à la volée des pages permet de faire rapidement des modifications dans la présentation du résultat. Nous avons la possibilité d'accéder aux articles précédant et suivant l'article visualisé selon la nomenclature du dictionnaire.

Inconvénients

Cette technique ne permet pas d'utiliser n'importe quelle structure. Il faut qu'elle soit simple et lisible par l'humain pour que l'on puisse la transformer aisément. Ces structures doivent être du genre SGML-HTML, ou des structures textuelles aussi simples.

Il est impossible de manipuler les ressources car elles ont des formats différents. Par exemple, la fusion d'articles correspondant au même mot-vedette mais provenant de ressources différentes est impossible. Nous devons utiliser les ressources telles quelles.

Nous demandons un article à la fois et nous le transformons. S'il fallait transformer toute la ressource avant de l'utiliser, nous ne pourrions pas utiliser ce type de serveur, car il ne permet jamais d'accéder à toute la ressource.

Il n'est pas encore possible d'afficher le contexte d'un article, c'est à dire d'afficher par exemple les 5 articles précédents et suivants selon la nomenclature. Nous ne pouvons obtenir qu'un article à la fois.

Dans les maquettes présentées jusqu'ici, le résultat n'est pas encore paramétrable par l'utilisateur.

1.3. Regroupement de ressources locales et distantes : DicoFeJ

1.3.1. Présentation

DicoFeJ est un serveur de dictionnaires français-anglais-japonais conçu selon l'architecture de Dico-Web. L'utilisateur consulte ce serveur à partir d'un terme français, anglais ou japonais. Nous utilisons deux ressources : un dictionnaire français-anglais provenant du FeM et le dictionnaire japonais-anglais : Edict [EDICT] de Jim Breen. Pour représenter du français et du japonais dans la même page, nous devons utiliser Unicode. Nous transformons donc à la volée l'encodage des résultats de l'ISO-LATIN-1 pour le français et de l'EUC-JP pour le japonais vers l'Unicode encodé en UTF-8.

Notre serveur réutilise le serveur distant du dictionnaire japonais-anglais développé par Jim Breen de l'université Monash à Melbourne en Australie [EDict]. Nous avons programmé un module interfacique (wrapper) qui consulte ce serveur, ramène le résultat et le convertit en Unicode. Ce résultat converti est ensuite affiché avec les autres résultats obtenus localement. Dans notre exemple, figure B.7, l'entrée du FeM a été obtenue localement tandis que l'article du EDICT provient du serveur distant de Jim Breen.

Lorsque l'utilisateur entre un mot français, DicoFeJ récupère les traductions anglaises correspondant aux mots-vedettes trouvés. Il consulte ensuite le dictionnaire japonais-anglais avec ces traductions. Nous affichons ensuite les entrées françaises du FeM et les entrées japonaises de EDICT à la suite. Nous construisons ainsi à la volée un brouillon de dictionnaire français-japonais.

D'autres ressources distantes sont disponibles. Ainsi, l'utilisateur peut compléter les informations sur un terme en consultant le dictionnaire universel francophone [DUF] s'il s'agit d'un terme français, ou le dictionnaire Websters si le terme est anglais. Il lui suffit de cocher les boutons correspondants (dictionnaires optionnels).

FIG. B.7 – *l'article* `neige` *du serveur dicofej*

1.3.2. Discussion

La transformation à la volée des pages permet de contourner l'impossibilité légale de stocker toutes les ressources localement. On accède aux ressources présentes sur ces serveurs distants (Hachette, Websters et Edict) grâce à des interfaces de connexion (wrappers) que nous avons programmées en Perl.

Pour nous permettre de visualiser du français et du japonais en même temps, nous utilisons la norme Unicode et son codage UTF-8.

Dans nos maquettes, nous utilisons aussi des modules complémentaires en amont et/ou en aval de la consultation des ressources. Les analyseurs morphologiques et les correcteurs orthographiques servent en amont pour obtenir un lemme à partir d'une forme de surface. L'interrogation du dictionnaire se fait ensuite avec le lemme.

Avantages

L'utilisateur dispose de plusieurs ressources dans la même interface. Il peut comparer les articles équivalents.

Les modules complémentaires permettent de proposer de nouvelles fonctionnalités aux utilisateurs, ce qui enrichit le concept de dictionnaire.

Inconvénients

Les interfaces de connexion aux ressources distantes doivent être programmées ad hoc.

Les informations provenant de ces ressources ont leur propre format qui est toujours un format de présentation et jamais un format logique. Elles ne sont donc pas analysables automatiquement et ne sont par conséquent utilisables que par des humains.

Les modules ne peuvent pas être clients et fournisseurs en même temps. De plus, l'adaptation des interfaces (wrappers) de ces modules doit aussi être faite à la main.

1.4. Personnalisation du résultat des requêtes : le FeM

1.4.1. Présentation

Le serveur du FeM a été construit en reprenant la technique de DicoWeb avec des scripts CGI écrits en Perl. Il dispose donc d'une recherche d'un article par expressions régulières Perl et d'un accès aux entrées précédentes et suivantes. Il est disponible sur le serveur public initialement financé par l'action SILFIDE (AUF-CNRS) [FeM].

Nous avons modifié la technique de DicoWeb pour la rendre plus réactive. Au lieu de séparer le formulaire HTML dans un cadre (frame) et le résultat des requêtes dans un autre cadre, le formulaire HTML est inclus dans la page du résultat. L'utilisateur du FeM est donc toujours devant la même page, dont le contenu est modifié dynamiquement à chaque nouvelle requête.

Un progrès, par rapport à DicoWeb, est la possibilité offerte à l'utilisateur de personnaliser le résultat de ses requêtes. Avant de consulter le serveur, l'utilisateur sélectionne dans la partie grisée à l'aide des boutons les éléments d'information qu'il souhaite voir apparaître dans le résultat de sa requête. Ainsi, si l'utilisateur est bilingue malais-français et s'il n'est pas intéressé par les traductions anglaises, il lui suffit de décocher les cases correspondant aux traductions anglaises comme dans l'exemple de la figure B.8.

1.4.2. Discussion

Avantages

Le résultat est paramétrable dynamiquement par l'utilisateur.

Inconvénients

L'utilisateur ne peut paramétrer qu'une partie du résultat en sélectionnant les catégories d'information qu'il souhaite afficher ou non. Il n'est pas encore possible de changer la présentation (couleurs, styles), ni la structure du résultat.

L'utilisateur paramètre son résultat à chaque session. Il ne peut pas encore garder ses préférences pour une nouvelle session. Pour cela, il semble nécessaire qu les utilisateurs s'inscrivent sur le serveur, et que ce dernier puisse les identifier à chaque nouvelle session.

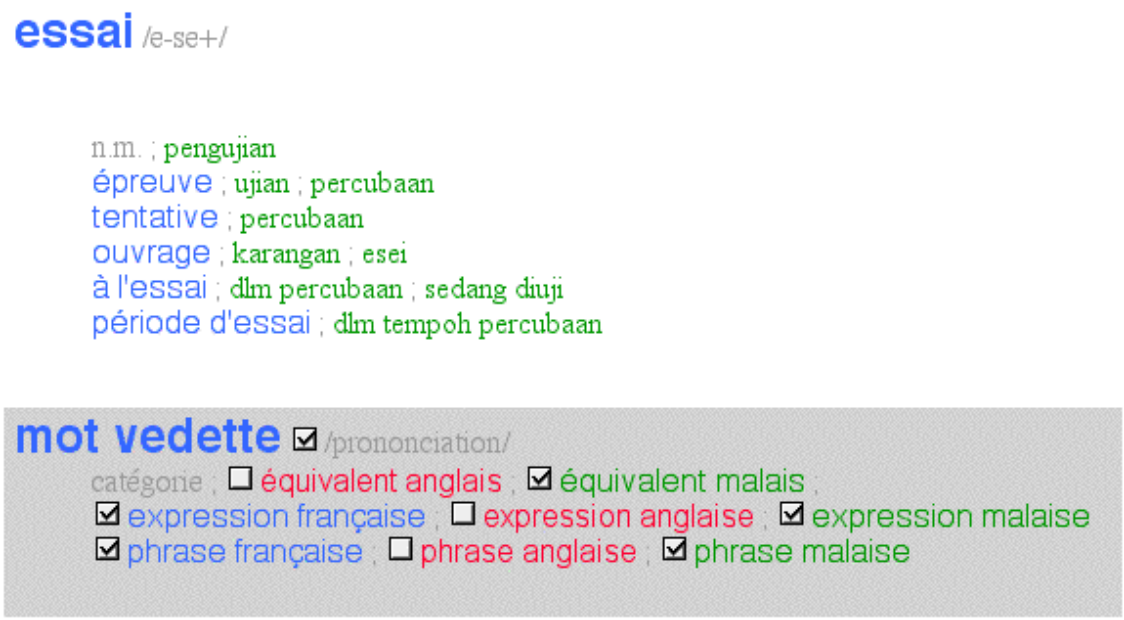


FIG. B.8 – *interface du serveur du FeM paramétrable*

2. Amélioration des méthodes de construction

2.1. Amélioration de la méthode démocratique du FeM pour UNL

2.1.1. Problématique

Le projet UNL a été présenté dans la section 1.3.4. de la partie A. Le langage UNL sert de représentation sémantico-linguistique pivot pour diverses applications (traduction automatique, RI multilingue). Il ne peut bien sûr représenter exactement toute l'information exprimée dans n'importe quelle langue, il s'agira toujours d'une approximation.

Les expressions UNL ne doivent pas seulement être définies rigoureusement, mais être aussi générales que possible pour être comprises par toutes les personnes chargées du développement des "enconvertisseurs" et des "déconvertisseurs".

Le vocabulaire UNL est formé de :

– UW (Universal Word ou en français Unité de Vocabulaire Virtuel) qui représentent des acceptions ou ensembles d'acceptions interlingues. Par convention, on a utilisé des mots anglais pour établir le vocabulaire UNL, car cette langue est compréhensible par la majorité des acteurs du projet.

– étiquettes de relations sémantiques;

– étiquettes d'attributs qui expriment, à un niveau interlingue, l'actualisation (détermination, nombre, genre), l'aspect, la modalité, l'emphase, etc. Un attribut particulier, `@entry`, indique l'élément central de l'énoncé représenté, pour la partie (scope) considérée.

Les UW dérivées du mot anglais « book » sont décrites comme suit :

```
book                          :  garde tout sens possible
book(icl>publications)        :  sens limité par une UW hyperordonnée
                              = livre en tant que publication
book(=accounts)               :  sens limité par une autre UW
                              = livre de comptes
book(obj>room)             :  restriction par une relation distinctive
                              = réserver (une chambre)
```

La figure B.9 montre un exemple d'expression UNL.

Il fallait à court terme indexer près de 200 000 entrées avec un coût maximal de 5 F par entrée. Pour satisfaire cette demande, il a donc fallu travailler avec plusieurs indexeurs en même temps, travaillant chez eux et non reliés au réseau. Il fallut ensuite regrouper les données en construisant une base lexicale pour différents outils et différents partenaires. La base lexicale est maintenant utilisée par le serveur de déconversion des graphes UNL vers des énoncés français.

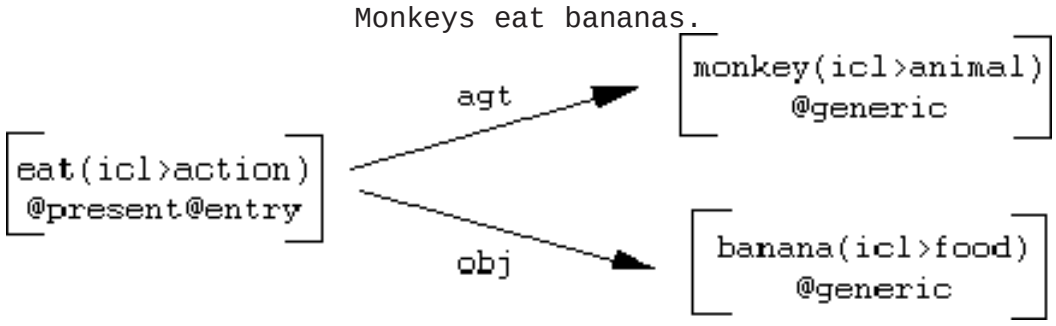


FIG. B.9 – *exemple de graphe UNL*

La solution mise en œuvre est représentée par la figure B.10

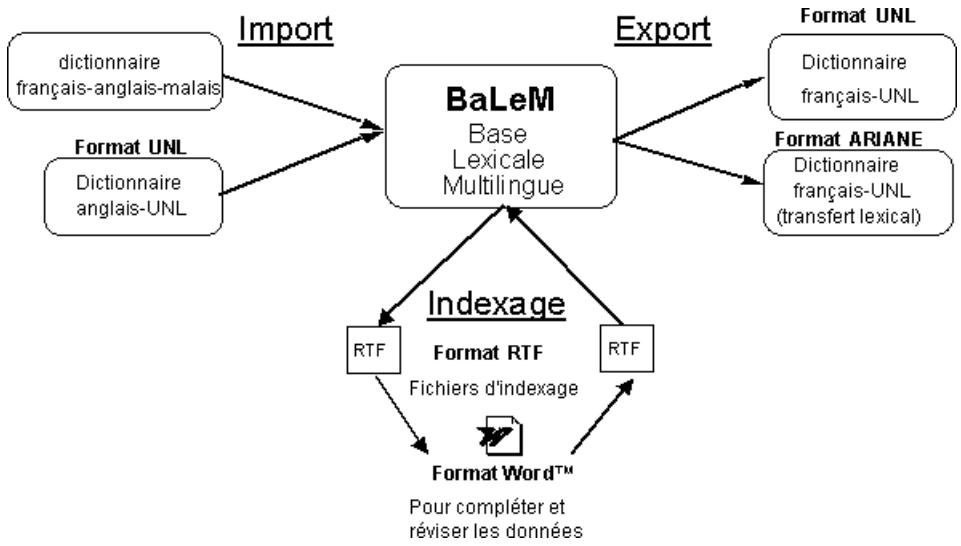


FIG. B.10 – *solution mise en œuvre*

2.1.2. Structure interne de la base

La structure de la base n'est pas figée, ce qui nous permet de la faire évoluer facilement en fonction des besoins. Nous pouvons à tout moment intégrer de nouveaux dictionnaires ou générer automatiquement des dictionnaires pour différents systèmes de déconversion/traduction comme ARIANE [Boitet82] ou le système DeCo utilisé à l'Université des Nations Unies par le centre UNL [UNL97] pour le japonais et l'anglais.

La base lexicale a été programmée en Macintosh Common LISP Object System (MCL) [Keene89, Steele90]. Ce langage à objets nous a permis de définir la structure interne en suivant un modèle à objets.

La figure B.11 montre trois classes d'objets liés entre eux :

– La classe appelée `Vocable` est similaire à une entrée de dictionnaire papier.

– La classe appelée `Acception française` représente une acception française selon le point de vue du GETA.

– Chaque UW UNL est représentée par une instance de la classe Acception UNL.

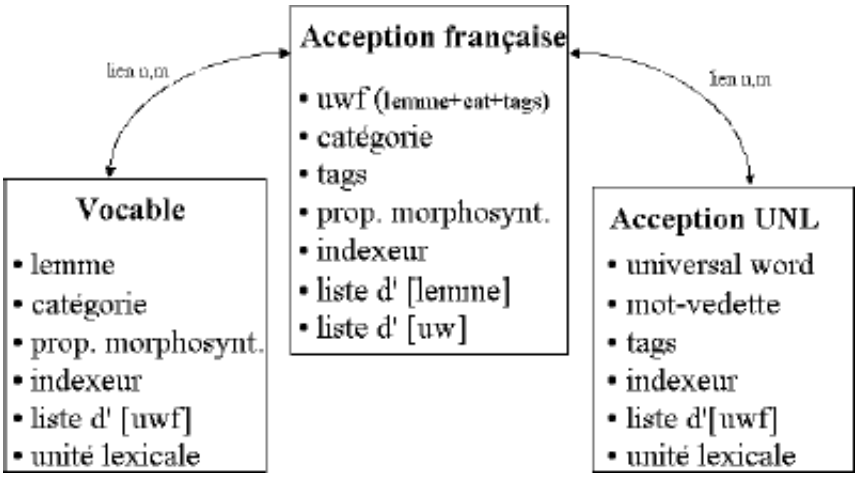


FIG. B.11 – *vision interne de la base lexicale*

Chaque instance de la classe `Acception UNL` est liée à une ou plusieurs instances de la classe `Acception française`. Chaque Vocable est lui aussi lié à une ou plusieurs instances de cette classe. Une instance de la classe `Acception française` peut être liée à un nombre quelconque de `Vocables` et/ou d'`Acceptions UNL`.

2.1.3. Rédaction des articles

Nous avons repris la solution employée pour la construction du dictionnaire FeM. Cette technique a permis de corriger, compléter ou créer 20 000 entrées contenant 50 000 acceptions en 9 mois. Nous avons ensuite amélioré la technique en séparant le travail en deux parties :

La base de données centrale est gérée par un lexicologue. Il récupère d'abord plusieurs dictionnaires qu'il fusionne. Il crée une description des entrées de la base sous forme de grammaire. Grâce à cette description, il prépare le résultat de cette fusion sous forme de fichiers Word™ qui sont envoyés aux indexeurs. Dans un deuxième temps, il récupère les fichiers révisés et complétés pour les intégrer dans la base après filtrage. Il peut renvoyer plusieurs fois les fichiers aux indexeurs, si le résultat n'est pas satisfaisant.

Les indexeurs travaillent à domicile sur leur ordinateur personnel. Ils n'ont besoin que du logiciel Word™ sur Mac ou PC. Pour faciliter le travail des lexicographes, nous avons ajouté des outils d'aide à l'indexation sous forme de macros Word™.

Les postes des lexicographes

Le lexicographe dispose d'une vue globale de l'extrait de dictionnaire avec lequel il travaille. Il peut corriger très rapidement les erreurs qu'il détecte et peut s'inspirer des articles précédents ou suivants, qu'il voit en totalité sans avoir à ouvrir de fenêtres supplémentaires.

Chaque unité d'information est donnée sous forme de paragraphe dans un style particulier [Gaschler94a, 94b]. À l'aide des macros, le lexicographe peut sélectionner la catégorie dans une liste (ce qui évite les erreurs dans les abréviations), vérifier la validité d'une entrée ou calculer l'ensemble des styles pouvant suivre le style courant, afin d'insérer un nouvel élément d'information [Mangeot97]. La figure B.12 montre un exemple de fichier d'édition d'un dictionnaire.

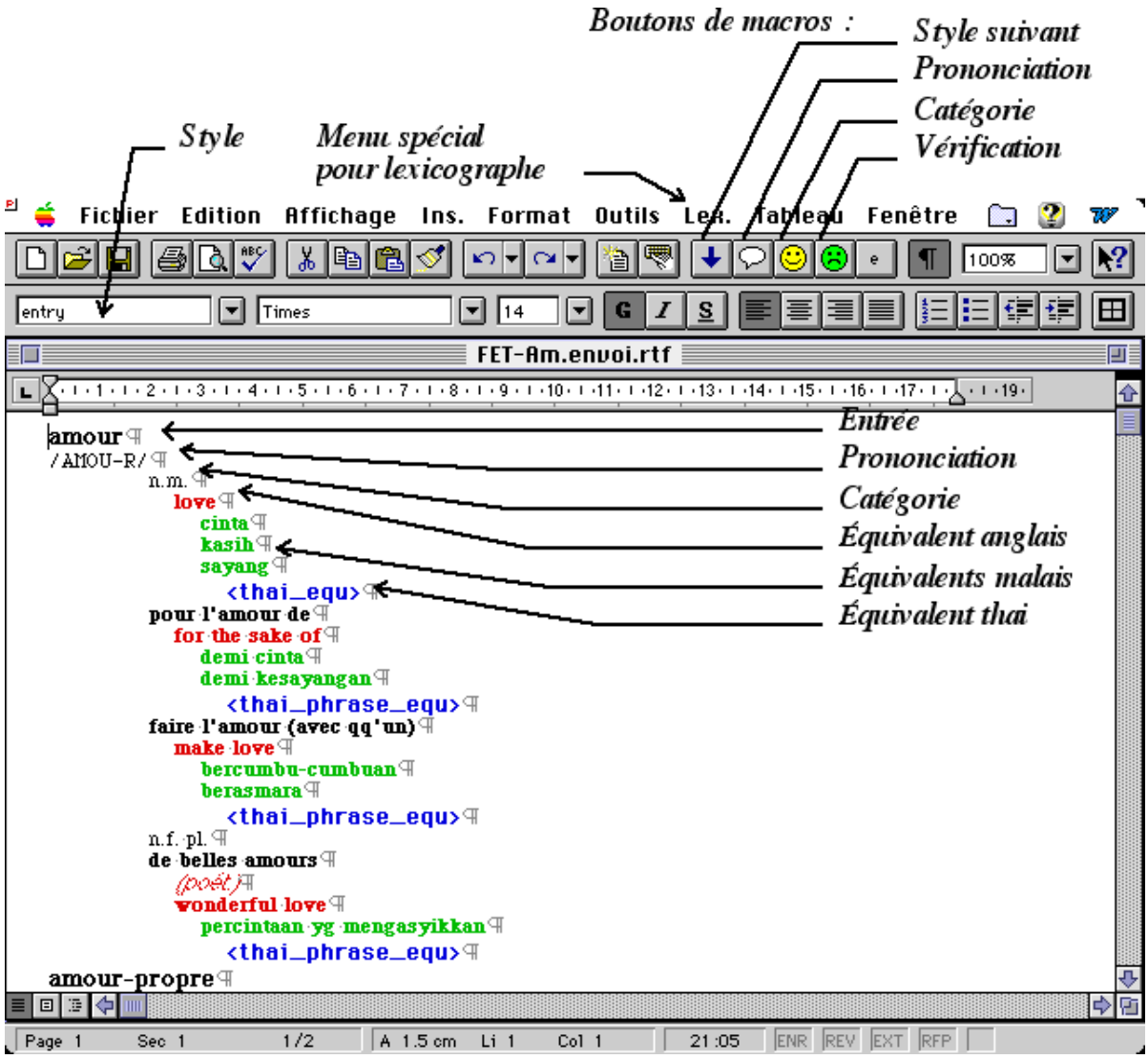


FIG. B.12 – *fichier d'édition du dictionnaire français-anglais-thaï*

Les outils d'aide à l'indexage

Pendant l'indexage, le lexicographe peut consulter ses propres données : dictionnaires papier, autres dictionnaires électroniques, et fichiers d'édition déjà complétés. Il est libre d'utiliser d'autres outils d'analyse de corpus ou d'étiquetage de sens qui peuvent l'aider à indexer ses termes.

Lorsque le lexicographe a fini de remplir un champ, il appelle la macro `style suivant` (voir figure B.13) soit par un bouton dans la barre d'outils, soit par un menu, soit encore par un équivalent clavier. Il sélectionne dans la liste des styles suivants autorisés celui dont il a besoin et la macro change le style automatiquement.

Grâce à la macro `liste valeurs` (voir figure B.14), le lexicographe obtient pour chaque champ la liste des valeurs possibles. Dans notre exemple, le champ de la catégorie grammaticale ne peut comporter que certaines valeurs. Si le lexicographe appelle la macro `liste valeurs`, elle affiche automatiquement la liste des catégories autorisées et insère la catégorie sélectionnée.

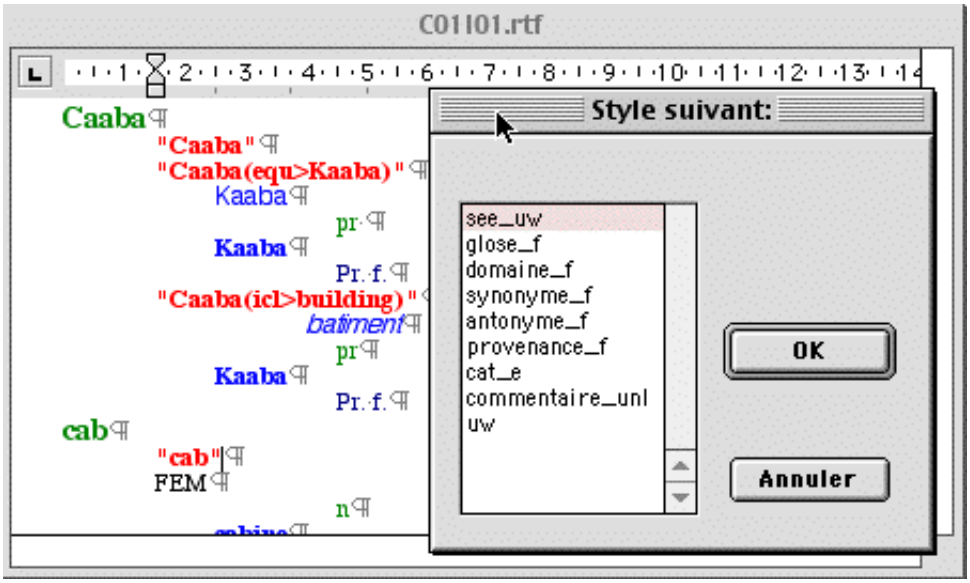


FIG. B.13 – *fenêtre de la macro* `style suivant`

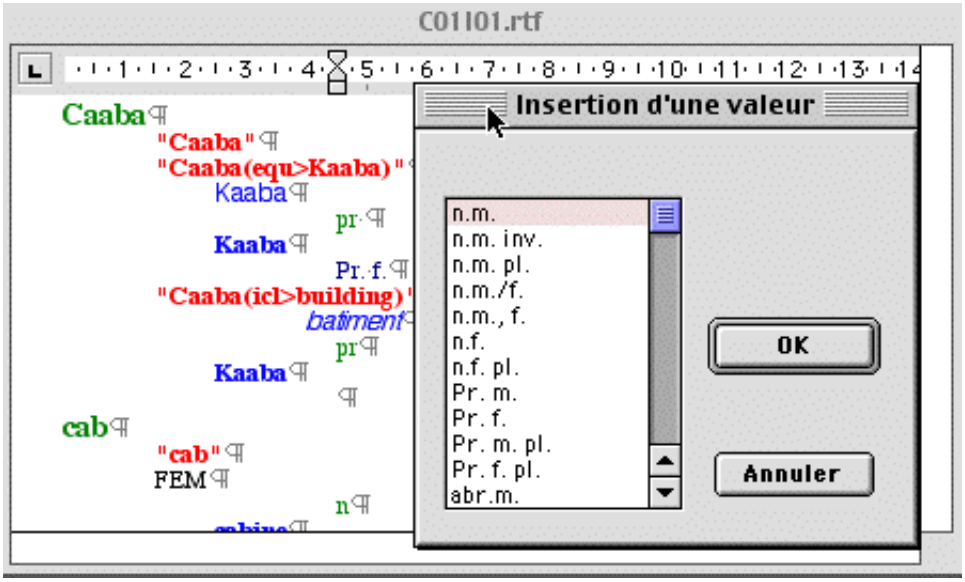


FIG. B.14 – *fenêtre de la macro* `liste valeurs`

Grâce à la macro `vérification`, le lexicographe peut vérifier si une valeur est bien permise pour le champ sélectionné. Dans notre exemple, la macro appliquée au champ de catégorie vérifie si la valeur sélectionnée appartient bien à la liste des catégories grammaticales définie par le lexicologue. Elle envoie un message d'erreur (voir figure B.15) si le champ n'est pas correctement rempli.

La macro `vérification générale` permet au lexicographe de vérifier la cohérence d'une entrée entière. Pour chaque style, elle vérifie si le style suivant est un style autorisé par la grammaire, et, si le style contient une liste fermée de valeurs, elle applique la macro `vérification`.

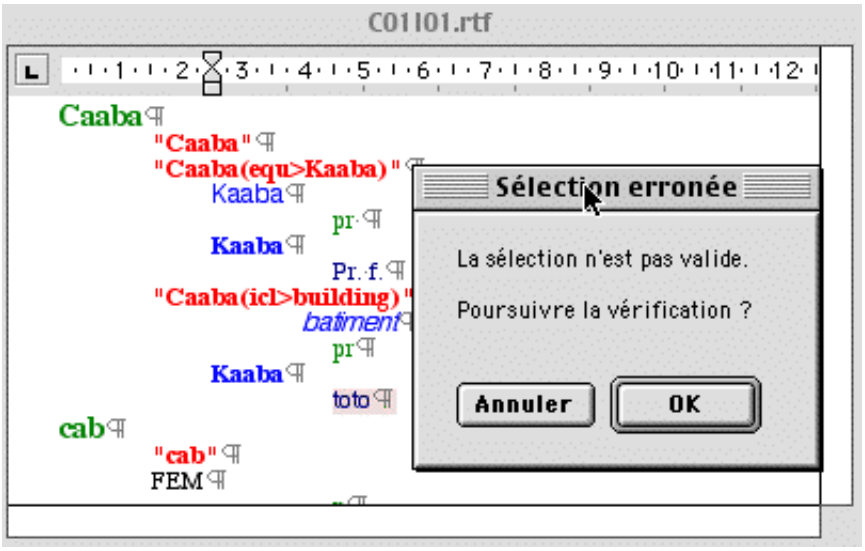


FIG. B.15 – *message d'erreur suite à la vérification d'une catégorie*

Après avoir vérifié ses entrées, le lexicographe enregistre le fichier en format RTF et le renvoie au lexicologue par disquette ou par réseau.

2.1.4. Discussion

La technique de construction du dictionnaire français-malais a été améliorée :

– nous avons indexé 20 000 UW en 7 mois (6 indexeurs à temps partiel travaillant chez eux),

– nous pouvons générer les fichiers RTF à partir de la base existante,

– nous proposons des outils d'aide à l'indexage qui permettent au lexicographe de vérifier la structure d'un article.

Avantages

Nos outils ont été conçus pour répondre à la demande du projet UNL. Nous les avons aussi testés avec les données du projet FeT. L'objectif de ce projet (éditer un dictionnaire trilingue français, anglais, thai) est différent de celui du projet UNL. Les outils ont pu être utilisés sans aucune modification. Cela confirme l'aspect générique de notre poste de travail.

L'utilisation de Word permet une démocratisation de la méthode. Les lexicographes peuvent travailler aussi bien sur Macintosh que sur PC. Ils n'ont pas besoin non plus d'utiliser des machines très puissantes. Des postes d'entrée de gamme suffisent.

Au premier abord, les outils d'aide à l'indexage semblent prometteurs. Ils permettent manifestement de gagner du temps et d'éviter des erreurs. De plus, si l'ensemble du dictionnaire n'est pas exactement conforme à une structure définie ou si certaines parties sont délicates, les lexicographes peuvent tout de même travailler sur un sous-ensemble de l'information disponible.

Inconvénients

Il reste tout de même des problèmes inhérents à la méthode. Ainsi, même si nous fournissons au lexicographe des outils permettant de vérifier la structure des entrées, des problèmes subsistent lors de la récupération. En effet, ces outils ne fonctionnent que lorsque le lexicographe les appelle. L'expérience montre qu'il ne le fait que rarement. D'autre part, on a toujours besoin d'un administrateur de la base pour vérifier le travail des indexeurs.

De plus, cette technique est restreinte à des dictionnaires à structures relativement simples (descriptibles par des grammaires LL(1)).

2.2. Construction en ligne de dictionnaires à structures simples : DicoSzótár et Nihongo

2.2.1. Présentation

Intéressé par les langues, nous suivons régulièrement des cours de langues étrangères. Nous avons donc profité de ce terrain d'expérimentation pour d'une part aider notre apprentissage avec des outils et d'autre part pour tester des directions de recherche pour la conception d'un environnement.

Nous avons ainsi conçu des serveurs utilisés à la fois pour la consultation et pour la rédaction de nouveaux articles. DicoSzótár est un serveur de dictionnaire pour apprenants du hongrois et Nihongo pour apprenants du japonais. Les buts principaux de ces expériences sont la consultation et la construction en ligne des dictionnaires. Les dictionnaires sont construits au fur et à mesure de l'apprentissage des mots par les participants aux leçons. Ils sont aussi utilisés pour apprendre le vocabulaire et réviser les leçons précédentes.

Les parties consultation des serveurs DicoSzótár et Nihongo sont conçues selon l'architecture DicoWeb. Les parties rédaction sont conçues de manière analogue. Les utilisateurs entrent les données en ligne à l'aide d'un formulaire HTML. Le serveur récupère les données et les inclut dans les fichiers où sont stockés les dictionnaires.

Pour DicoSzótár, deux dictionnaires bilingues sont en cours de construction : un dictionnaire français-hongrois et un dictionnaire hongrois-français. Chaque dictionnaire est actuellement composé d'environ 600 articles.

Pour Nihongo, nous avons noté les traductions avec des liens interlingues. Deux dictionnaires monolingues sont en cours de construction : un dictionnaire français de 210 articles et un dictionnaire japonais de 350 articles. Les articles de ces 2 dictionnaires sont reliés entre eux par des liens de traduction. Lors de la consultation, le serveur consulte d'abord le dictionnaire de la langue source, stocke temporairement les articles qui répondent à la requête puis consulte le dictionnaire de la langue cible et sélectionne les articles cibles reliés aux articles sources précédemment stockés. Il recompose ensuite à la volée des articles de dictionnaires bilingues d'usage avec les informations contenues dans tous les articles.

2.2.2. Structure des articles

Lors de la rédaction, l'utilisateur entre des informations sur un mot français et sur sa traduction dans l'autre langue. À partir de ces informations, le serveur construit un article pour le mot français et un article pour le mot de l'autre langue. Chaque article sera ensuite inséré dans le dictionnaire correspondant à sa langue.

Les articles sont encodés au format XML. Lors de la création des articles ou de la modification d'une partie de ces articles, des informations d'administration sont ajoutées pour permettre la révision des informations. Nous stockons aussi des informations relatives aux leçons pour permettre de réviser le vocabulaire d'une leçon en particulier.

La structure des articles de Nihongo est une simplification de celle utilisée pour les lexies de la base DiCo. Chaque article est en fait une lexie. Nous pouvons donc avoir plusieurs articles ayant le même mot-vedette si les sens qu'ils représentent sont différents. Pour la traduction du mot vedette dans l'autre langue, nous n'indiquons pas la traduction directement mais l'identificateur de la lexie correspondante.

La figure B.16 représente un exemple d'article du dictionnaire Nihongo français.

```
<lexie id="maison#1"
basic="yes"
indexer="Mathieu Mangeot"
date="Fri Oct 20 19:3 1:46 MET DST 2000"
status="à réviser">
  <headword nb="1">maison</headword>
  <pos>nom commun</pos>
  <jpn><refjpn href="家#1"/></jpn>
  <lesson-number>1</lesson-number>
  <lesson-date>20/10/00</lesson-date>
</lexie>
```

FIG. B.16 – *article du dictionnaire Nihongo français*

Chaque lexie a un identificateur unique `id` utilisé pour les liens. Nous notons le mot-vedette avec l'élément <headword>, la catégorie grammaticale avec l'élément <pos>, les traductions japonaises avec l'élément <jpn>. Chaque traduction japonaise est en fait un lien vers une lexie japonaise notée avec l'élément <refjpn>.

Nous notons ensuite des informations liées à l'apprentissage de la langue avec la date et le numéro de la leçon dans laquelle ce mot a été vu pour la première fois avec les éléments <lesson-date> et <lesson-number>.

Enfin, nous notons des informations d'administration pour nous permettre de contrôler la qualité des données. Nous notons le nom du lexicographe avec l'attribut `indexer`, la date avec l'attribut date et le statut de la lexie avec l'attribut `status`.

2.2.3. Interface de rédaction

À chaque nouvelle leçon, les nouveaux mots sont ajoutés au dictionnaire par un des participants à la leçon. Chaque contributeur possède son interface web personnalisée. L'interface de la figure B.17 est personnalisée pour Mathieu. Cela permet de noter le nom et le niveau du contributeur. À l'aide d'un formulaire HTML (voir figure B.17), il entre les données en ligne. Ces données sont ensuite insérées dans les dictionnaires et stockées au format XML sur le serveur. Elles sont ensuite consultables grâce à un outil similaire à DicoWeb.

Lorsque l'utilisateur entre les données en ligne, le serveur attribue automatiquement un numéro unique à chaque lexie. Dans le cas de la figure B.17, une lexie française pour `maison` et une lexie japonaise pour `uchi`. Si un autre utilisateur entre ensuite les mêmes mots, le serveur l'avertira. L'utilisateur devra alors cocher le bouton `forcer l'insertion de l'entrée` s'il estime que le nouveau mot entré est en fait une nouvelle lexie.

Par exemple, un premier utilisateur entre le mot français `aimer` et la traduction japonaise `suki` (dans le sens de like). Lorsqu'un deuxième utilisateur entrera le même verbe français `aimer` avec la traduction japonaise `aisuru` (dans le sens de love), il forcera l'insertion de l'entrée et un nouveau numéro unique sera affecté à ces lexies. Le dictionnaire sera alors composé de deux lexies pour le verbe `aimer`.

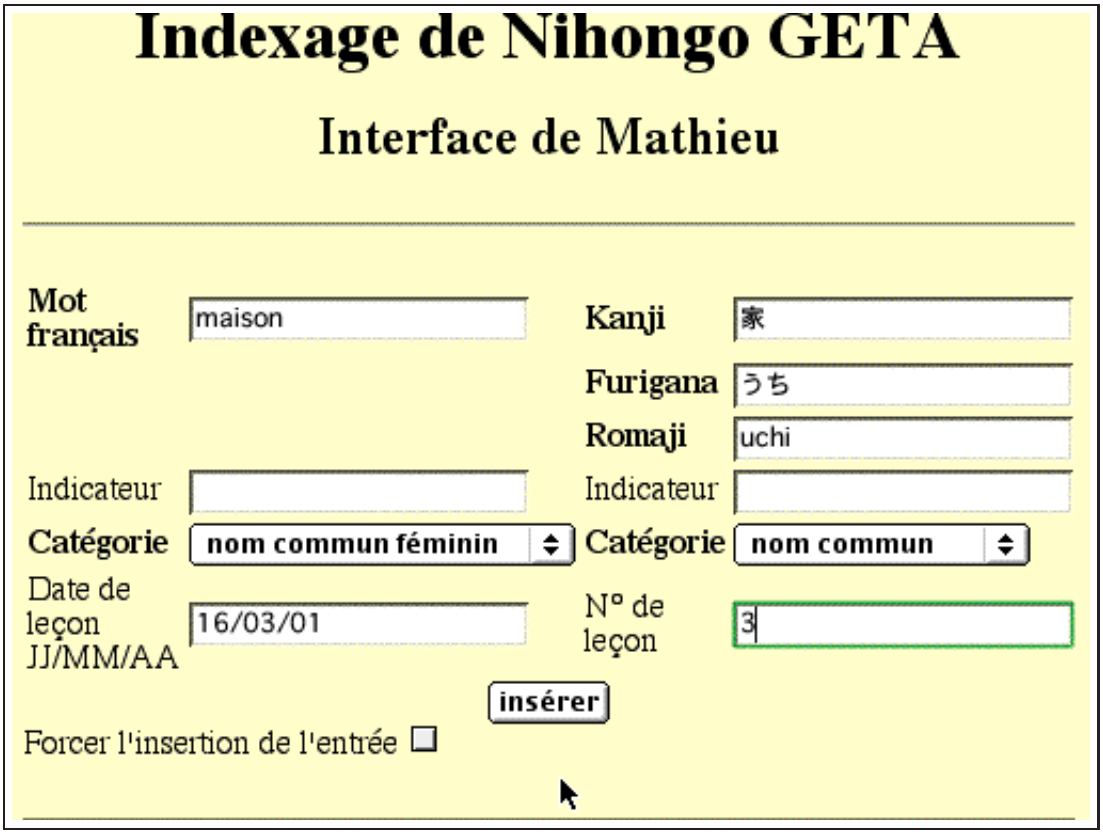


FIG. B.17 – *interface d'indexage en ligne du dictionnaire Nihongo*

2.2.4. Discussion

Avantages

Cette méthode est simple et efficace. Les utilisateurs n'ont besoin que d'un navigateur Web pour l'utiliser. Le formulaire HTML permet de contrôler la structure des articles. Les informations de gestion permettent de contrôler la qualité du dictionnaire.

Inconvénients

Nous n'avons pas pour l'instant développé d'interface de révision, de modification ou de suppression d'un article. Ces manipulations se font "à la main", directement avec un éditeur de texte sur les fichiers. Cette méthode demande donc plus de développements pour être vraiment opérationnelle.

L'utilisation d'un formulaire HTML limite aussi cette technique. En effet, la structure du dictionnaire à construire doit rester très simple. De plus, elle demande aux utilisateurs d'être connectés en permanence lors de la rédaction de l'article. En particulier, on peut intégrer de l'information multimédia, et aussi enrichir considérablement les foncitonnalités de consultation.

3. Nouvelles directions pour la consultation

L'utilisation de machines pour stocker et consulter les dictionnaires permet d'enrichir le concept de dictionnaire qui était basé jusqu'à récemment sur les dictionnaires imprimés à usage humain.

3.1. Elargissement du concept de dictionnaire : DicoSzótár

DicoSzótár est un dictionnaire pour apprenants du hongrois. Il est composé d'une partie français->hongrois et d'une partie hongrois->français. Ce dictionnaire est en cours de construction. Il nous a permis de tester plusieurs nouveaux concepts.

3.1.1. Utilisation de données multimédia

Nous avons d'abord testé l'utilisation de données multimédia en ajoutant dans certains articles une image pour illustrer le sens porté par le mot-vedette. Cette technique permet de construire un dictionnaire monolingue (ici en hongrois) mais consultable par tous les utilisateurs qui peuvent visualiser l'image. Ceux-ci comprendront la signification du mot-vedette dans leur langue grâce à l'image. L'exemple de la figure B.18 montre un article contenant une image.

Cette technique déjà utilisée depuis longtemps dans les encyclopédies atteint cependant rapidement ses limites. On ne peut pas tout illustrer par des images. De plus, certaines différences sont subtiles et très difficiles à représenter. Par exemple, il n'est pas évident de montrer la différence entre une vallée et une montagne ou un lac et un étang.

Nous avons aussi rajouté dans certains articles un fichier son de la prononciation du mot-vedette par un locuteur natif. Nous pourrions aussi utiliser un synthétiseur par l'intermédiaire d'un module externe.

Quant à la vidéo, l'Encyclopedia Universalis l'a utilisé depuis 1997 dans les cédéroms de l'encyclopédie..

3.1.2. Interface personnalisée pour apprenants : le quizz

Pour faciliter l'apprentissage du hongrois, nous avons construit une petite application qui utilise DicoSzótár. L'utilisateur sélectionne la langue source, le nombre de mots, la leçon et les catégories grammaticales qu'il veut réviser. L'application se connecte alors à DicoSzótár pour choisir au hasard dans le dictionnaire les mots correspondant à la leçon sélectionnée ainsi que leurs traductions. Ces mots sont ensuite affichés. L'utilisateur doit donner au système une traduction pour chaque mot.

L'application vérifie les traductions et affiche en rouge les corrections des traductions erronées (voir figure B.19). Un score final est donné à l'utilisateur en fonction du nombre d'erreurs qu'il a faites.

Cette interface est très pratique pour l'apprentissage du vocabulaire. Cependant, sa mise en place n'est possible que si les informations sont présentes dans tous les articles du dictionnaire, et elle dépend de la

FIG. B.18 – *article* `fa` *du serveur DicoSzótár*

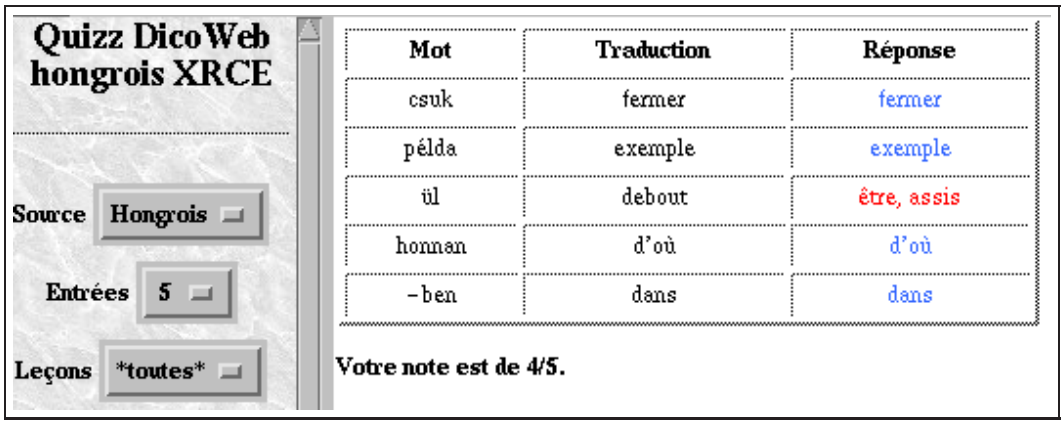


FIG. B.19 – *utilisation de DicoSzótár par un quizz*

structure du dictionnaire. L'application est relativement sommaire. Elle a principalement pour but d'illustrer l'utilisation d'un dictionnaire par une autre application. Pour des renseignements plus complets sur les environnements d'apprentissage, il est possible de se référer à la thèse de Thierry Selva [Selva00].

3.2. Visualisation au moyen d'arbres hyperboliques

3.2.1. Introduction

Le projet UNL utilise à l'heure actuelle environ 16 langues. Les bases lexicales UNL de chaque langue sont composées d'un dictionnaire bilingue reliant les unités lexicales UNL aux lemmes de la langue. Une unité lexicale (UW) est composée d'un mot-vedette (headword) anglais suivi éventuellement d'une liste de restrictions illustrant un sens précis du mot-vedette. Un seul mot-vedette regroupe donc plusieurs sens avec des restrictions différentes. Chaque sens peut avoir des traductions dans chaque langue du projet.

Pour visualiser un mot-vedette UNL, ses différentes acceptions et leurs différentes traductions, nous avons utilisé un visualisateur d'arbres hyperboliques développé par la société InXight. Ce visualisateur permet de naviguer dans l'arbre en cliquant sur les nœuds, et de relier des nœuds à des pages html grâce à des liens hypertexte. C'est une applet programmée en java qui lit un fichier texte représentant un arbre en entrée et l'affiche à l'écran comme un arbre hyperbolique. Il est possible de spécifier les couleurs de chaque nœud et de chaque arc.

3.2.2. Exemple d'arbre hyperbolique

L'utilisateur demande un mot-vedette. Notre maquette consulte alors les dictionnaires UNL disponibles pour chaque langue et extrait toutes les acceptions correspondant au mot-vedette demandé ainsi que leurs traductions. Un fichier texte représentant l'arbre est ensuite construit à la volée, puis affiché à l'aide de l'applet d'arbre hyperbolique sur l'écran de l'utilisateur. On peut ensuite naviguer dans l'arbre avec la souris.

Dans l'exemple de la figure B.20, le mot-vedette demandé est le mot anglais `desert`. Il est placé au centre de la figure et colorié en violet. Les acceptions reliées à ce mot-vedette sont coloriées en gris. Nous trouvons par exemple `desert(src>human), desert(src>group), desert(agt>human), desert(agt>animate,icl>event)`.

Nous avons adopté un principe ergonomique qui est d'associer à chaque langue une couleur. Nous gardons autant que possible les mêmes codes de couleur pour des applications différentes. Les traductions françaises des acceptions sont coloriées en bleu. Nous trouvons entre autres `désert2, faire_défection, délaisser, abandonner`. Les traductions italiennes sont coloriées en vert : `desertico` & `disertare`. Les traductions japonaises sont coloriées en rouge.

La version de l'applet utilisée ne permet pas d'utiliser Unicode. Les traductions japonaises sont donc mal codées. La nouvelle version a corrigé ce défaut mais nous n'avions pas encore pu nous la procurer au moment de rédiger cette section.

3.2.3. Discussion

L'expérience des codes de couleur est concluante. Cela permet à l'utilisateur de se repérer dans une base multilingue. Nous avons aussi montré avec cette maquette qu'il était possible de réutiliser des produits du commerce et de les adapter pour permettre de visualiser plus facilement une grande quantité de données lexicales.

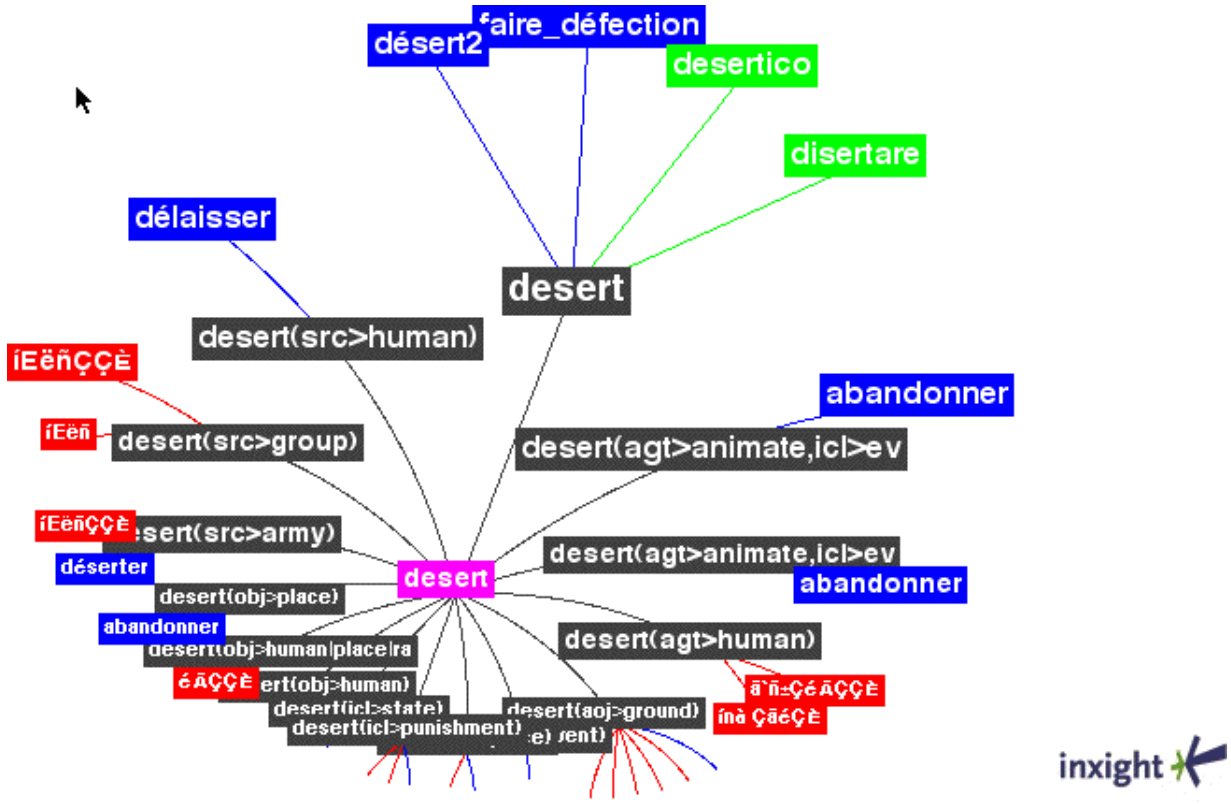


FIG. B.20 – *article* `desert` *de la base lexicale UNL*

3.3. Annotation d'un article de dictionnaire

Lors de la consultation d'un dictionnaire, les utilisateurs souhaitent souvent ajouter des remarques sur des articles déjà écrits. Lors de la rédaction, les rédacteurs souhaitent faire des commentaires sur des informations qu'ils veulent ajouter mais ne savent pas comment ajouter. Ces annotations doivent pouvoir être partagées entre groupes d'utilisateurs et de rédacteurs.

3.3.1. Notre outil

Partant de ce constat, nous avons implémenté une maquette permettant d'annoter les dictionnaires consultables sur nos serveurs comme DicoWeb, FeM, etc. Les annotations sont stockées sur un serveur. Chaque utilisateur gère son propre dictionnaire d'annotations. Il peut partager ses annotations avec plusieurs groupes d'utilisateurs.

Lorsqu'un utilisateur recherche une entrée, notre outil consulte le serveur d'annotations et affiche les annotations appartenant aux groupes dont fait partie l'utilisateur. Ensuite, l'utilisateur peut modifier ses annotations ou en créer de nouvelles à partager éventuellement entre plusieurs groupes d'utilisateurs.

Par la suite, nous avons découvert des outils équivalents implémentant les annotations que nous avons testé : l'outil *ThirdVoice* et l'annoteur de l'éditeur HTML *Amaya*.

3.3.2. L'outil *ThirdVoice*

Présentation

ThirdVoice^{TM} [ThirdVoice] est un outil disponible sur plate-forme Windows. Il permet d'annoter des documents installés sur le web et de partager les annotations en constituant des groupes d'utilisateurs. La figure B.21 montre un exemple d'annotation réalisée avec *ThirdVoice* sur la page web de *paperbag.com*.

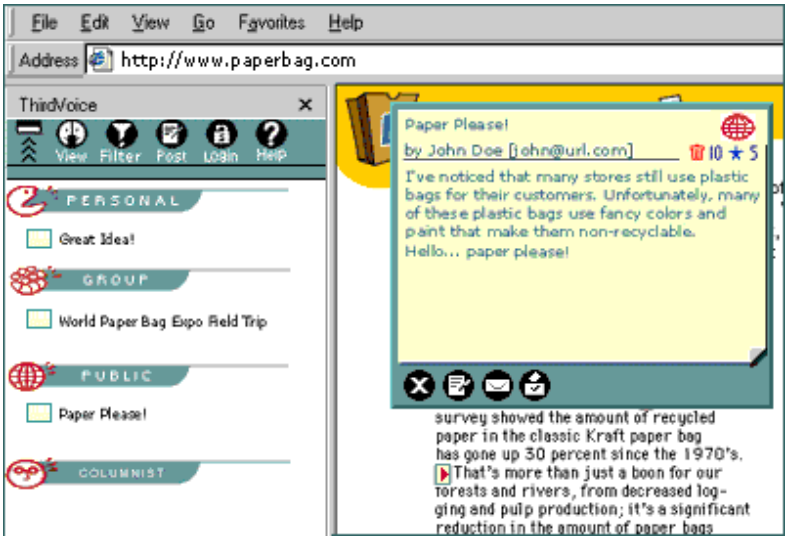


FIG. B.21 – *l'outil ThirdVoice d'annotation de pages Web*

Discussion

Cet outil est très satisfaisant en termes de fonctionnalités puisqu'il permet l'annotation de pages et le partage des annotations par des groupes d'utilisateurs. Cependant, les annotations sont stockées sur le serveur de ThirdVoice. Il n'est pas possible de les gérer soi-même, ce qui peut poser des problèmes de confidentialité. Il n'est pas non plus possible d'annoter des documents stockés localement. Ils doivent absolument être installés sur un serveur Web public. De plus, cet outil n'est disponible que sur Windows, ce qui limite aussi son utilisation.

3.3.3. L'annoteur d'*Amaya*

Présentation

Amaya [Amaya], le navigateur/éditeur du W3C (World Wide Web Consortium) implémente depuis peu un système d'annotation. L'utilisateur peut spécifier le serveur d'annotations ou encore stocker ses annotations en local. Il est donc possible de travailler en local sans se connecter au web. Les annotations sont stockées sous forme de fichiers xhtml [XHTML 1.0] et décrites par des fichiers XML utilisant les RDF et les XPointer.

RDF (Resource Description Framework) [RDF] est une structure pour les métadonnées. RDF permet une interopérabilité entre les applications qui échangent des informations sur le Web. RDF facilite le traitement automatique des ressources Web.

XPointer (XML Pointer Language) [XPointer] est une recommandation XML. C'est un langage utilisé comme base pour référencer une portion de document XML. XPointer est basé sur XPath [XPath]. Il permet l'examen d'une structure hiérarchique de document et le choix de ses parties internes basé sur diverses propriétés comme le type des éléments, les valeurs d'attributs, les caractères et leur position relative.

Exemple d'annotation

Tout d'abord, l'utilisateur configure son logiciel pour utiliser les annotations. Il doit indiquer son nom d'utilisateur et le serveur sur lequel seront stockées les annotations. Pour notre exemple, nous avons stocké les annotations en local pour pouvoir les analyser.

Les annotations sont indexées par document annoté. Lorsque l'utilisateur annote un document, un fichier d'index est créé ou modifié. Ce fichier d'index contient une liste d'associations entre un document annoté et un index d'annotations. Pour l'exemple, nous avons annoté le titre de cette section : l'annoteur Amaya. Le fichier d'index est composé des informations suivantes :

`file:/home/mmangeot/MM-These/partieB.html`
`file:/home/mmangeot/.amaya/annotations/index01`

Il indique donc que le fichier `partieB.html` a été annoté et que les annotations sont indexées dans le fichier `index01`. Ce fichier est au format XML. Il utilise la norme RDF pour noter les annotations sur un document. La figure B.22 représente un extrait de ce fichier.

```
<!-- déclaration des espaces de noms -->
<!-- r, a, http et d sont des raccourcis -->
<r:RDF xmlns:r="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:a="http://www.w3.org/2000/10/annotation-ns#"
xmlns:http="http://www.w3.org/1999/xx/http#"
xmlns:d="http://purl.org/dc/elements/1.0/">
<r:Description>
<r:type resource="http://www.w3.org/2000/10/annotation-ns#Annotation"/>
<r:type resource="http://www.w3.org/2000/10/annotationType#Comment"/>
<a:annotates r:resource="file:///home/mangeot/MM-These/partieB.html"/>

<a:context>
#xpointer(start-point(string-range(/html[1]/body[1]/p[85],"",58,1)))
</a:context>
<d:creator>mangeot</d:creator>
<a:created>2001-01-31T17:57:41</a:created>
<d:date>2001-01-31T17:59:26</d:date>
<a:body r:resource="file:///home/mangeot/.amaya/annotations/annot01.html"/>
</r:Description>
</r:RDF>
```

FIG. B.22 – *description d'annotations Amaya dans le format XML*

L'élément <type> note le type d'annotation, l'élément <annotates> note le document annoté (ici partieB.html), l'élément <context> note à l'aide d'un pointeur Xpointer l'endroit exact où se trouve l'annotation dans le document.

Ensuite, on trouve des informations d'administration comme le créateur de l'annotation noté par <creator> et la date de création de l'annotation notée par <created>.

Enfin, on trouve un pointeur vers l'annotation elle-même notée par l'attribut resource de l'élément <body> (ici le fichier `annot01.html`).

La figure B.23 montre finalement le fichier xhtml `annot01.html` qui représente le corps de l'annotation.

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <!-- titre généré automatiquement -->
    <title>Annotation of Partie B</title>
  </head>
  <body>
  <p>Voici une <strong>annotation</strong> portée par le titre de
cette section :  3.3.3.  L'annoteur d'Amaya</p>
  </body>
</html>
```

FIG. B.23 – *document XML représentant une annotation*

La figure B.24 montre l'annotation telle que l'utilisateur la voit sur son document.

Discussion

Le système d'annotations d'Amaya répond à pratiquement tous nos besoins. On peut regretter cependant l'absence de gestion de groupes d'utilisateurs donc l'impossibilité de partager des annotations.

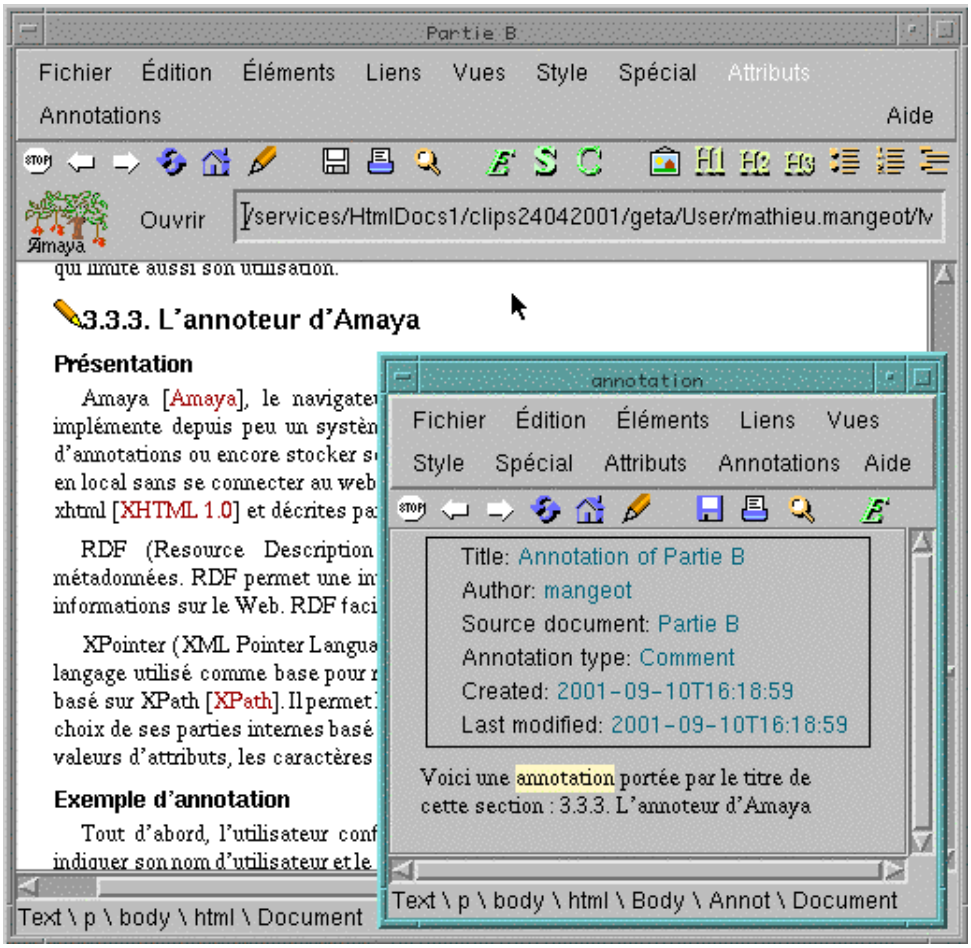


FIG. B.24 – *exemple de document annoté avec Amaya*

4. Coopération entre applications

Dans cette section, nous expérimentons la coopération entre applications. Nous avons testé la coopération dans les deux sens, la base lexicale pouvant jouer le rôle du client ou du serveur. D'un coté, nous utilisons dans nos applications des outils d'aide à la consultation ou des ressources installés sur des serveurs distants. De l'autre, nos applications peuvent être utilisées automatiquement par d'autres applications comme serveurs de dictionnaires.

4.1. Aide à la consultation grâce à des modules externes

4.1.1. Présentation

DicoSzótár, notre serveur de dictionnaire pour apprenants du hongrois, a été enrichi par l'utilisation de modules installés sur des serveurs distants. Nous avons associé des actions aux mots-vedettes de DicoSzótár pour faciliter l'apprentissage du hongrois. L'accusatif et le pluriel des noms hongrois n'étant pas toujours réguliers, nous avons associé un générateur d'accusatif et de pluriel aux entrées correspondantes. Nous avons aussi associé un conjugueur générique aux verbes.

DicoSzótár est installé sur le serveur du CLIPS [DicoSzótár]. Le conjugueur et le générateur d'accusatif résident sur le serveur public de XRCE [Démos]. Ils proviennent des outils d'analyse du hongrois. Il y a donc un wrapper qui fait le lien entre notre serveur et les outils installés sur le serveur de XRCE.

4.1.2. Utilisation d'un conjugueur

La figure B.25 montre un exemple d'utilisation d'un conjugueur. L'utilisateur recherche d'abord un article en indiquant un mot-vedette dans l'interface de départ. Le ou les articles correspondant au mot vedette s'affichent alors dans la partie droite du navigateur. Au bas des articles, une nouvelle interface est affichée pour les noms et les verbes. Un bouton permet de demander l'accusatif ou le pluriel des noms. Des listes sont affichées pour sélectionner le temps et le mode de conjugaison des verbes.

Lorsque l'utilisateur veut conjuguer un verbe, il indique le temps et le mode désirés. Le module de conjugaison installé sur un serveur distant est alors consulté automatiquement par l'application. Les résultats s'affichent ensuite dans la partie droite de la fenêtre (voir figure B.26).

4.2. Consultation par une application de traduction automatique

4.2.1. Présentation

Le serveur du dictionnaire UNL est implémenté en Common Lisp et tourne sur un Macintosh. Il est essentiellement utilisé par des machines. En effet, il sert principalement au serveur de déconversion qui transforme les graphes UNL en textes français. Pour que les applications clientes du serveur du dictionnaire UNL-français puissent communiquer avec le serveur, nous avons mis en place un protocole d'interrogation

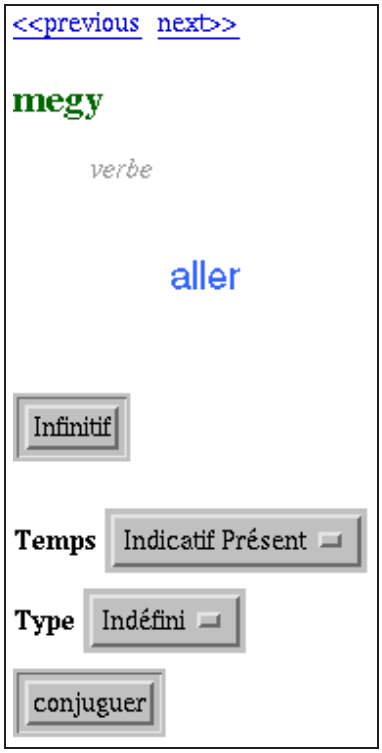


FIG. B.25 – *utilisation d'un conjugueur dans DicoSzótár*

Réponse du conjugueur

Singulier 1ère personne	megyek
Singulier 2ème personne	mégy
Singulier 2ème personne	mész
Singulier 3ème personne	megy
Pluriel 1ère personne	megyünk
Pluriel 2ème personne	mentek
Pluriel 3ème personne	mennek

FIG. B.26 – *résultat du conjugueur*

de dictionnaires fonctionnant par Telnet. Le Dictionary Server Protocol (DICT) est une transaction TCP basée sur un protocole de requête/réponse qui permet à un client d'accéder à des entrées de dictionnaires. Ce protocole est décrit dans la RFC 2229 [Dict].

4.2.2. Commandes disponibles

Le serveur écoute sur le port 2628 qui est réservé au protocole Telnet/DICT. Lorsqu'il reçoit une demande de communication de la part d'un client, il crée un processus fils qui s'occupera de traiter la requête. Chaque nom de commande est composé de quatre lettres. Pour l'instant, seules les commandes suivantes sont implémentées :

- `AUTH username password` /* authentification */
- `CLUW uw` /* UW la plus proche présente dans le dictionnaire */
- `HEAD headword` /* UWs ayant le même mot-tête */
- `HELP` /* affichage de ce texte */
- `LOAD [DIC] dictionnaire unl-français [/DIC]` /* charge en mémoire un dictionnaire */
- `QUIT` /* ferme la connexion */
- `TRAN {parametres} [UNL] graphe unl [/UNL]` /* traduction d'un graphe UNL */
- `UWLS uw` /* recherche les traductions en français d'une UW */
- `UWTR uw` /* recherche une traduction en français d'une UW */

Le processus fils traite la requête et renvoie la réponse au client.

4.2.3. Exemples de sessions

Lorsque le serveur de déconversion des graphes UNL en énoncés français reçoit un graphe UNL, il analyse le graphe et en extrait les UW qu'il contient. Il se connecte ensuite au serveur du dictionnaire UNL-français en ouvrant une session Telnet/Dict avec la commande `telnet silfide.imag.fr 2628`. Il s'identifie avec la commande `AUTH`, envoie la liste des UW à traduire en français avec la commande `UWLS`. Lorsqu'il reçoit le résultat, il ferme la session avec la commande `QUIT`.

Lorsque l'administrateur du serveur de dictionnaires souhaite mettre à jour le dictionnaire, il peut aussi le faire à distance. Pour cela, il se connecte au serveur du dictionnaire UNL-français en ouvrant aussi une session Telnet/Dict. Il s'identifie avec la commande `AUTH`, il envoie le nouveau dictionnaire avec la commande `LOAD` et ferme la session avec la commande QUIT.

4.3. Consultation par un outil de recherche : Sherlock

4.3.1. Présentation

L'application Sherlock sur Macintosh consulte automatiquement de nombreux CGIs distants. Elle permet d'effectuer des recherches sur plusieurs moteurs de recherche en parallèle, ce qui est un gain de temps considérable. Sherlock utilise des plug-ins pour se connecter aux différents CGIs implémentant des moteurs de recherche dans des domaines très variés.

Notre application DicoWeb peut être considérée comme un moteur de recherche dans un dictionnaire. Comme cette application est disponible sur le Web et fonctionne via un CGI, nous avons pu développer un plug-in pour Sherlock qui peut consulter notre serveur DicoWeb automatiquement. Cela permet de tester l'utilisabilité de DicoWeb par une autre application cliente.

4.3.2. Le plug-in Sherlock

Un plug-in est un fichier texte balisé en pseudo-XML et rangé dans un dossier spécial utilisé par l'application Sherlock. Dans ce fichier, nous devons indiquer les paramètres que Sherlock doit utiliser pour consulter l'application DicoWeb. Notre plug-in est présenté dans la figure B.27.

```
<-- pseudo XML d'APPLE pour Sherlock -->
<search
name="DicoWeb FeM"
action="http://clips.imag.fr/cgi-bin/geta/dicoweb/dicoweb.pl"
method="get">
<input name="SOURCE" value="fr">
<input name="RESEARCH" value="dict">
<input name="FORMNAME" value="sherlock-internal">
<input name="FEM" value="on">
<input name="ENTRY" user>
<interpret
bannerStart="<TITLE>"
bannerEnd="</TITLE>"
resultListStart="<!-- DICTIONARY START -->"
resultListEnd="<!-- DICTIONARY END -->"
resultItemStart="<!-- ENTRY START -->"
resultItemEnd="<!-- ENTRY END -->"
>
</search>
```

FIG. B.27 – *fichier de plug-in pour l'application Sherlock*

Le mot-clé `action` indique l'URL du CGI DicoWeb.

Le mot-clé `input` introduit un paramètre d'entrée du CGI.

Les mots-clés `resultListStart` et `resultListEnd` indiquent quelles sont les chaînes de caractères du résultat qui marquent le début et la fin de la liste des articles répondant à la requête.

Les mots-clés `resultItemStart` et `resultItemEnd` indiquent quelles sont les chaînes de caractères du résultat qui indiquent le début et la fin de la liste des articles répondant à la requête.

4.3.3. Interface de l'outil Sherlock

L'utilisateur entre le terme qu'il recherche dans l'interface de Sherlock. Sherlock se connecte alors à notre serveur grâce aux informations fournies par le plug-in et attend le résultat, qu'il affiche ensuite là où il affiche tous les résultats de recherche (voir figure B.28).

4.3.4. Discussion

Sherlock permet à l'utilisateur de faire une recherche multisite et d'afficher les résultats selon un ordre de pertinence établi par les serveurs consultés. Si tous les serveurs de dictionnaires développaient leur plug-in, nous pourrions faire une recherche multidictionnaire sans aucune programmation. Cependant, nous ne pourrions pas fusionner les informations renvoyées par les différents outils.

FIG. B.28 – *article* `essai` *du FeM dans l'application Sherlock*

5. Conclusion : cahier des charges d'un environnement unifié

Après avoir exploré plusieurs aspects du traitement des dictionnaires comme la consultation en ligne, la construction coopérative, et de nouvelles directions pour la consultation et la coopération entre applications, nous sommes maintenant en mesure d'élaborer un cahier des charges pour un environnement complet de création, manipulation et consultation de dictionnaires multilingues hétérogènes.

Le cahier des charges est différent de celui d'une entreprise sur deux aspects principaux : le temps et les coûts. Les recherches pour un nouvel outil ne sont pas limitées par le temps car il n y a pas le même souci de rentabilité. Sans contrainte temporelle, on peut donc attacher plus d'importance à la conception d'un tel environnement.

L'environnement devra pouvoir être capable de résister au temps, c'est à dire de s'adapter aux évolutions du domaine. Sa conception doit être pensée pour une utilisation à long terme. L'absence de contraintes de rendement nous permet de rester à un niveau générique et de ne pas nous restreindre à une tâche particulière.

Nous ferons d'abord un bilan des expériences précédentes, puis exposerons les problèmes encore non résolus et enfin terminerons par les contraintes d'implémentation.

5.1. Bilan des expériences précédentes

5.1.1. Sur la consultation en ligne

Pour la consultation de dictionnaires, nous sommes arrivé à :

– présenter la méta-information sur les ressources, importante pour pouvoir les sélectionner et les évaluer et le résultat est totalement satisfaisant;

– présenter des dictionnaires hétérogènes de façon unifiée, avec un résultat très satisfaisant, bien que l'on bute sur une impossibilité de principe : on ne peut pas traiter les informations renvoyées par les différents serveurs, et on ne peut donc pas les fusionner ou les filtrer à un niveau fin;

– donner des moyens à l'utilisateur de personnaliser ses requêtes. Le résultat est satisfaisant mais on voudrait faire bien plus.

5.1.2. Sur la construction de dictionnaires

Pour la construction de dictionnaires, nous avons fait progresser deux méthodes déjà employées avec succès pour des réalisations importantes (FeM, SAIKAM). La méthode "démocratique" permet de faire des allers-retours entre le lexicologue et les lexicographes, ces derniers travaillant sur Word^{TM}, un logiciel de traitement de texte du commerce très répandu. Nous proposons en plus des outils d'aide à l'indexation sous forme de macros.

L'autre méthode permet la construction en ligne de dictionnaires ayant des structures simples. Ces deux méthodes sont complémentaires. Il faut les améliorer et même les unifier pour pouvoir construire des dictionnaires plus complexes en ligne et localement.

5.1.3. Sur l'utilisation d'outils annexes

Nos outils utilisent des modules annexes pour l'annotation de documents et pour l'aide à la consultation comme les analyseurs morphologiques et les conjugueurs. Le résultat est très satisfaisant, en terme de fonctionnalités comme en temps de réponse : grâce à l'augmentation des débits sur le réseau, le fait qu'un module soit distant ne provoque pas d'attente supplémentaire perceptible, Par contre, ce que nous avons fait a été ad hoc et il faut absolument standardiser les interfaces de ces modules et prévoir des API (Application Programming Interface) pour pouvoir les changer et les adapter facilement (voir l'outil ODILE d'Isabelle Tomasino [Tomasino90]).

5.2. Problèmes restants non résolus

5.2.1. Construction en communauté à travers le Web

Lors de nos expériences précédentes, nous avons conçu une technique permettant de rédiger des articles de dictionnaires en ligne. Mais, même si nous prenons en compte des informations permettant la révision des données, la technique n'est valable que pour un tout petit groupe d'utilisateurs et elle ne permet pas encore la révision des données.

Il reste donc à mettre en place un vrai "collecticiel" qui permette la construction collaborative de dictionnaires via le Web comme dans le projet AllianceWeb [AllianceWeb]. Les membres de la communauté virtuelle concernés par cette construction ont des rôles différents. Les informations envoyées par les contributeurs doivent être révisées par un petit groupe d'experts. Cette organisation pose des problèmes de droits d'accès différents et de gestion des contributions, qui ne peuvent être intégrées qu'après révision.

Problème de droits d'accès

Si deux utilisateurs écrivent ou modifient la même donnée en même temps, des problèmes de conflits surgissent. De plus, certaines données doivent être protégées du grand public. Pour résoudre ces problèmes, il faut organiser un serveur gérant différents utilisateurs et groupes.

Une solution satisfaisante est que chaque utilisateur ait avoir un compte virtuel sur le serveur avec des droits d'accès particuliers. Les utilisateurs pourront librement constituer des groupes partageant les mêmes droits d'accès. Il sera aussi utile, voire nécessaire, d'associer à chaque utilisateur un profil d'intérêt et un profil de compétences.

Problème de gestion des contributions

Tous les utilisateurs qui possèdent un compte virtuel sur le serveur peuvent envoyer leurs contributions au serveur. Si ces contributions sont intégrées sans contrôle, la base lexicale risque de se trouver polluée par des contributions ou des corrections erronées. Il faut donc mettre en place un mécanisme de gestion des contributions par un groupe restreint d'experts qui permette de n'intégrer les contributions que si elles ont été validées.

Les contributions ne seront donc pas intégrées directement dans la base, mais elles seront stockées dans l'espace virtuel du contributeur jusqu'à ce qu'elles soient révisées, validées et intégrées.

5.2.2. Gestion d'une base multilingue

Lors de nos expériences précédentes, nous avons utilisé plusieurs dictionnaires bilingues, mais nous n'avons pas encore manipulé de véritable base lexicale multilingue. D'autre part, nous avons réutilisé les ressources lexicales dans leur format d'origine, et nous n'avons donc pas non plus résolu le problème de la structuration des informations lexicales.

Évolution des dictionnaires bilingues vers une base lexicale multilingue

Nous maîtrisons la fabrication de dictionnaires bilingues en utilisant diverses méthodes. Nous souhaitons maintenant monter d'un degré en construisant une base lexicale multilingue avec une architecture pivot et en extrayant de cette base des dictionnaires bilingues personnalisés. La construction d'une base multilingue à structure pivot limitera les efforts de rédaction, car les parties monolingues ne seront rédigées qu'une seule fois.

Des efforts on déjà été faits dans ce domaine avec l'élaboration de la maquette PARAX par Etienne Blanc [Blanc96,99] ou le projet ULTRA dirigé par Yorick Wilks [Farwell92]. Il faut maintenant passer à une réalisation à grande échelle pour permettre à toute une communauté de construire ensemble une base lexicale multilingue.

Problème de structuration des informations lexicales

Les bases de données usuelles ne permettent pas de représenter les informations lexicales avec une granularité très fine. Par exemple, la base lexicale DiCo est stockée dans une base de données du type FileMaker^{TM}. Chaque lexie est stockée dans 8 champs différents. Les champs ne sont en fait pas analysés jusqu'au bout. Ce sont des objets textuels structurés par des marques typographiques, et pas des balises sémantiques (caractérisant le contenu).

Par contre, les bases de données sont idéales pour le stockage, la préparation et le tri des informations. Elles seront donc utilisées au premier niveau comme support, mais ne le seront pas pour l'interaction directe avec les utilisateurs. Les articles des dictionnaires seront stockés tels quels, avec une granularité relativement grossière. Ils seront ensuite analysés pour trouver les informations nécessaires.

5.3. Contraintes d'implémentation

Nous voulons élaborer un environnement qui soit le plus portable possible. Sa conception doit s'appuyer sur un maximum de standards. De plus, nous souhaitons manipuler des structures hétérogènes avec les mêmes outils. Nous avons donc besoin d'un système générique de structuration des données lexicales.

5.3.1. Utiliser la technologie XML pour manipuler les données

Le format standard à l'heure actuelle pour la structuration des données est XML [XML 1.0]. Son importance croissante dans le domaine de l'informatique nous incite fortement à l'utiliser pour manipuler les données. Il nous semble intéressant pour plusieurs raisons :

– *compatibilité et portabilité* . XML est une recommandation du consortium W3C. Les outils compatibles avec cette recommandation peuvent donc lire tous les documents XML valides.

– *utilisation d'UNICODE* . Nous pouvons utiliser le standard UNICODE (avec le codage UTF-8 par exemple) directement dans les fichiers XML. Nous pourrons donc facilement représenter des dictionnaires multilingues.

– *multiplicité des normes et recommandations autour de XML* . XML a donné naissance à d'autres normes et recommandations qui l'utilisent directement comme XML-Namespace, XSL, XPATH ou XPOINTER. Ces normes étendent les fonctionnalités et la portabilité des documents XML.

– *multiplicité des outils* . Des API (interfaces de programmation) pour le protocole DOM (Document Object Model) [DOM] et SAX (Simple Api for XML) [SAX] sont disponibles pour pratiquement tous les langages de programmation.

– *héritage de SGML* . Ce format est dérivé de SGML. Un document SGML bien formé selon la recommandationC XML est un document XML. Cela facilite la récupération et l'utilisation de tous les dictionnaires déjà codés en SGML.

– *lisibilité par l'humain* . XML est lisible directement, ce qui est très utile pour les développeurs.

Cependant XML n'est qu'un format. Il nous faut ensuite spécifier et définir un système de manipulation des données basé sur les outils manipulant XML et les normes associées.

5.3.2. Utiliser un système générique de structuration de données lexicales

Nous souhaitons manipuler dans notre base des informations lexicales ayant des structures hétérogènes correspondant à diverses théories linguistiques et à divers types de ressources lexicales. Nous voulons aussi récupérer et utiliser des ressources existantes et produire des ressources dans des formats et structures spécifiques. Il faut donc trouver un moyen de représenter des structures riches et hétérogènes pour toutes les informations lexicales que l'on manipule.

Prévoir un formalisme souple et générique

Ce système doit implémenter un formalisme très souple permettant de représenter de nombreuses ressources hétérogènes. Les bases lexicales pourront contenir non seulement plusieurs dictionnaires pour plusieurs langues, mais aussi des objets qui ne seront pas des dictionnaires. Le formalisme devra ainsi permettre d'associer des arbres, des images, du son ou de la vidéo à un mot.

Les ressources traitables par le système envisagé peuvent être :

– des dictionnaires généraux monolingues (NODE, DEC);

– des dictionnaires généraux bilingues (OHD, OUPES);

– des dictionnaires généraux multilingues (FeM, Fe*);

– des bases multilingues de concepts ou d'acceptions (ELRA, UNL);

– des dictionnaires de systèmes de traduction (ARIANE RUS-FRA);

– des bases de données terminologiques (EURODICAUTOM);

– des mémoires de traduction (les réponses aux requêtes sont les segments alignés contenant le mot);

– des banques d'arbres (les réponses sont les arbres représentant des analyses de phrases incluant le mot);

– des corpus annotés ou non (les réponses sont les KWIC ou la fréquence d'apparition du mot);

– des nomenclatures;

– des listes de noms propres, des annuaires, etc.

Permettre la vérification des données

Certaines données seront récupérées à partir de ressources existantes, d'autres seront produites par des contributeurs. La base lexicale ne sera jamais dans un état figé. Il faut donc prévoir des procédures de validation des données pour surveiller en permanence l'état des informations contenues dans la base. Cela nécessite des outils permettant de mettre en place des contraintes sur les ressources et des vérificateurs de cohérence.

Grâce à ces outils, on pourra guider les contributeurs en leur proposant des informations à compléter, et aider les réviseurs en repérant certaines erreurs ou inconsistances.

Ces outils permettront de vérifier, d'enrichir des articles et de faire de multiples recherches. Il devront être associés à un langage de requêtes puissant.

Lorsqu'un article de dictionnaire sera mal formé, il faudra pouvoir le trouver. Les vérificateurs pourront être lancés par exemple en tâche de fond sur la base lexicale.

Par exemple, le lexicologue responsable d'une ressource peut avoir besoin d'extraire de la base en construction tous les articles qui n'ont pas de traduction. Il les enverra ensuite aux lexicographes qui les complèteront.

C : Spécification d’un environnement de gestion et consultation de bases lexicales et dictionnaires

Introduction

À partir du cahier des charges précédent, nous pouvons maintenant passer à la spécification d'un environnement complet de manipulation, création et consultation coopératives et distribuées de dictionnaires, du point de vue de l'architecture interne, de l'architecture lexicale et des interactions avec les différents utilisateurs.

Dans une première section, nous donnons les spécifications externes du noyau, du serveur de construction collaborative, et des fonctionnalités, en nous basant sur les expériences précédentes.

Dans la deuxième partie, nous définissons plus précisément le noyau en utilisant le formalisme SUBLIM de Gilles Sérasset que nous étendons et traduisons en XML.

Nous donnons ensuite les spécifications internes du serveur de construction collaborative et d'utilisation mutualisée d'une base lexicale multilingue riche.

Enfin, nous précisons les fonctionnalités de manipulation et d'échange avec d'autres programmes, puis de consultation et de contribution humaines.

1. Spécifications externes de l'environnement

Dans ce chapitre, nous décrivons les spécifications externes d'un environnement permettant de récupérer des ressources existantes, d'en construire de façon collaborative (aspect "collecticiel") grâce aux contributions et d'en produire de nouvelles pour l'export à partir des données existantes.

Nous spécifions d'abord le noyau de notre environnement, puis le principe de développement en communauté de ressources libres, et enfin l'intégration des expériences précédentes.

1.1. Spécification du noyau

Pour les raisons exposées précédemment, nous avons été conduits à prendre de la distance par rapport aux bases de données classiques qui ne serviront qu'au stockage, et ne seront pas directement utilisées pour la manipulation du contenu, réalisé par traitement des chaînes XML contenues dans les divers champs.

Il nous faut donc passer au niveau supérieur et redécrire un environnement complet en partant du noyau de cet environnement. Nous devons choisir un formalisme de représentation des données puis des outils pour les manipuler au niveau interne.

1.1.1. Choix du formalisme de représentation

Le formalisme de représentation des données sera basé sur SUBLIM, un Système Universel de gestion de Bases Lexicales Informatisées Multilingues décrit dans la thèse de Gilles Sérasset [Sérasset94e]. Ce système permet au lexicologue de spécifier la structure interne d'une base lexicale en utilisant deux langages de haut niveau : le langage LEXARD pour la macrostructure de la base et des dictionnaires et le langage LINGARD pour la microstructure des dictionnaires.

Avec ce système, il est donc possible de représenter de nombreuses structures des ressources hétérogènes provenant de la récupération. Le système possède aussi un formalisme permettant de décrire les vérifications que l'on souhaite appliquer aux données.

1.1.2. Manipulation des ressources

Les ressources existantes ont toutes des formats physiques et des structures logiques différents. Si l'on veut les intégrer à la base lexicale, il faut les convertir dans notre formalisme. Cette structure logique et linguistique unifiée unique facilitera la comparaison des ressources. Nous avons donc besoin d'outils pour récupérer les ressources existantes et les transformer dans notre formalisme et pour manipuler les ressources une fois converties de façon à produire de nouvelles ressources.

Différents types de ressources

Le serveur de la base permet d'accéder à différentes ressources, à des stades différents d'intégration dans la base :

– les dictionnaires externes sont vus dans leur format, éventuellement ensembles dans une même fenêtre (exemple de DicoWeb),

– les dictionnaires récupérés avec leur structure logique plus ou moins complète (exemple du FeM en XML),

– les dictionnaires récupérés et en cours de fusion et révision par des contributeurs (soupe lexicale),

– les dictionnaires générés à partir du contenu de la base.

Récupération de ressources

Pour la récupération de ressources, nous avons choisi le système RÉCUPDIC [Doan-Nguyen96a]. Ce système décrit en partie A est spécialisé pour la récupération dictionnairique. Il se compose de méthodes et d'outils puissants et faciles à utiliser. Il permet de ne jamais éliminer d'information, et de garder dans la mesure du possible la structure logique d'origine, si elle existe.

Pour faciliter la gestion de la base lexicale, il faut d'autre part garder la méta-information disponible et donner un label de qualité ainsi que des degrés de certification pour chaque ressource que l'on intègre à la base.

Production à partir de ressources existantes

Le lexicologue qui veut produire une nouvelle ressource à partir de ressources existantes a besoin d'outils pour concevoir la macrostructure et la microstructure de son dictionnaire. Ces outils doivent lui permettre de décrire en partie les interfaces de rédaction de ce nouveau dictionnaire.

Nous avons choisi le système PRODUCDIC [Doan-Nguyen96a]. Ce système, décrit en partie A, permet d'effectuer des opérations ensemblistes (union, intersection, soustraction) sur des ensembles de dictionnaires pour spécifier et réaliser des processus de production de façon générique et efficace.

Ainsi, PRODUCDIC permet, grâce aux opérations de fusion, de créer un squelette de dictionnaire de façon à ne pas partir de zéro lors de la construction d'une nouvelle ressource. Le squelette est ensuite révisé article par article.

Il permet aussi d'extraire de la base des nouvelles ressources avec des formats spécifiques. Ces ressources sont alors exportées selon les besoins des utilisateurs.

1.1.3. Construction de nouvelles ressources

Les lexicologues qui supervisent la construction de nouvelles ressources et l'intégration des contributions ont besoin d'outils pour contrôler le flux des données et pour appliquer des mécanismes de vérification des données. En effet, les ressources sont en constante évolution.

1.2. Développement partagé de ressources libres

Le développement actuel d'Internet et son esprit communautaire nous permettent d'envisager le développement partagé de ressources libres de droits. Nous souhaitons mettre en place un environnement qui permette à

n'importe quel utilisateur de notre serveur de contribuer à la construction des ressources directement, c'est-à-dire en ligne, à travers le Web.

Les contributions sont ensuite gérées par un petit groupe de spécialistes lexicologues. Ceux-ci vérifient les contributions et décident de les intégrer ou non aux ressources existantes. Comme on risque d'arriver à de nombreuses contributions, il faut permettre à des contributeurs de "préparer le travail" des lexicographes gérant la base en annotant les contributions d'autres contributeurs.

1.2.1. Principe général socio-économique du partage

Nous souhaitons que les utilisateurs partagent les ressources personnelles qu'ils ont construites. Ils sont invités à envoyer à la base leurs contributions qui seront ensuite partagées avec tout ou partie des utilisateurs de la base.

Utilisateurs et groupes

Les utilisateurs de la base ont chacun un compte où sont stockés leurs profils, leurs préférences, leurs contributions et leurs annnotations. Ils peuvent ensuite constituer librement des groupes en fonction d'intérêts ou d'activités communs.

Au départ, la base contient trois groupes prédéfinis. L'univers regroupe tous les utilisateurs de la base. Le groupe des administrateurs administre le serveur et le base lexicale. Le groupe des lexicologues contrôle la distribution du travail à faire et révise les contributions reçues avant de les intégrer à la base.

Système de points pour les contributions

Pour inciter les utilisateurs à contribuer, nous voulons mettre en place un système de points pour les contributions. D'une part les utilisateurs peuvent contribuer localement sur un article et d'autre part, ils peuvent envoyer leurs propres ressources qu'ils ont développées localement chez eux.

Chaque utilisateur qui envoie ses contributions à la base une contribution validée gagne un certain nombre de points. Chaque mois, un tableau récompense les contributeurs les plus efficaces.

Affectation des points en fonction des profils

Il convient de donner d'autant plus de points à un contributeur que ses contributions sont nombreuses et pertinentes. Une solution consiste à évaluer une contribution en fonction de son type (difficulté) et de sa qualité (note donnée au moment de la validation et de l'intégration dans la base).

1.2.2. Définition d'un serveur et des différents acteurs

La base lexicale est utilisée par de nombreux utilisateurs. Il faut donc l'installer sur un serveur. Ce serveur doit être accessible par le Web, par FTP et par Telnet. Pour la communication entre utilisateurs, il faut aussi envisager un serveur de courriel et de listes de distribution.

Le serveur répond à de multiples requêtes venant de différentes applications. Il doit tourner en démon pour pouvoir répondre automatiquement et 24h/24.

Sur le serveur, il faut mettre en place un système de gestion des utilisateurs et groupes avec des droits d'accès différents. Nous distinguons plusieurs types d'utilisateurs :

L'administrateur

Il définit des tâches administratives. Il gère les droits d'accès des utilisateurs. Il ajoute les nouveaux profils d'utilisateurs ou de groupes d'utilisateurs dans la base.

Le lexicologue/lexicographe en chef

Il s'occupe de la récupération, de la manipulation et de la construction des dictionnaires.

Il récupère et convertit la structure logique des ressources existantes à intégrer dans la base. À partir de ces ressources récupérées, il construit ensuite un squelette de dictionnaire qui sera ensuite complété.

Il gère un projet collaboratif de construction de nouvelles ressources. Il définit la macrostructure et la microstructure des dictionnaires qu'il veut construire. Il répartit ensuite le travail de rédaction des articles entre les lexicographes et le révise. Ils gère leurs apports et leurs modifications. Il distribue les points aux contributeurs en fonction de la qualité et la quantité de leurs contributions.

Le lexicographe

Il participe à la construction des dictionnaires. Il rédige le corps des articles en éditant une partie du dictionnaire. Il apporte une contribution complète pour chaque article. Il doit respecter la microstructure définie par le ou les lexicologues. Il annote des articles et partage ses annotations. Le lexicographe gagne des points pour chaque article rédigé et accepté.

Le contributeur

Il participe aussi à la construction des dictionnaires mais en général de manière partielle. Ils rédige une petite partie des articles, établit ou modifie des liens monolingues ou interlingues entre différents articles. Il annote des articles et partage ses annotations. Pour toutes ses contributions, il gagne un certain nombre de points en fonction de son profil et de la qualité de ses contributions.

Le consulteur

Il consulte la base en établissant des requêtes et en personnalisant leur résultat. Il n'a pas le droit de modifier ces ressources mais il peut les annoter et partager ses annotations. Les consulteurs peuvent être aussi bien des humains que des machines. Ils visualisent et naviguent dans plusieurs dictionnaires différents générés à partir de la base ou externes.

Les consulteurs peuvent consulter gratuitement une partie de la base. S'ils veulent consulter toute la base, personnaliser le résultat de leurs requêtes ou extraire de nouvelles ressources de la base, ils doivent payer avec les points qu'ils ont gagnés au préalable avec leurs contributions.

On désire en fait que le serveur soit fait de telle sorte que les consulteurs soient incités à devenir contributeurs, et que cela soit très facile. Il faudra donc éviter de faire remplir un grand formulaire à quelqu'un qui désire contribuer. Une meilleure stratégie consiste à demander à chaque consulteur de s'inscrire, à son premier accès, exactement comme le font les serveurs de courriel, puis à lui permettre de passer en "mode contributeur" n'importe quand.

Les serveurs partenaires

Ces serveurs échangent des données avec la base pour enrichir leurs ressources. Ils se rendent des services mutuels selon les outils dont ils disposent. Par exemple, un analyseur morphologique est intéressé par

les nouveaux mots qui sont intégrés dans la base et sur lesquels il n'a pas d'information. En échange, il peut lemmatiser les mots des requêtes faites sur la base avant la consultation.

1.2.3. Gestion des contributions

Les contributions sont envoyées à la base. Elles sont gérées par un petit groupe de lexicologues spécialistes qui doivent alors réviser le travail afin soit de l'ajouter, soit de le renvoyer pour des corrections. Les contributions ne sont donc pas incluses directement dans la base. Elles sont au préalables stockées dans l'espace virtuel du contributeur en attendant leur révision par un lexicologue (voir figure C.1).

De plus, les lexicologues ont besoin d'un outil permettant de déterminer les articles incomplets. Il leur permet alors de construire un ensemble virtuel de choses à faire et de les classer éventuellement par ordre de priorité. Cet ensemble est ensuite distribué aux lexicographes et contributeurs selon leur niveau de compétence.

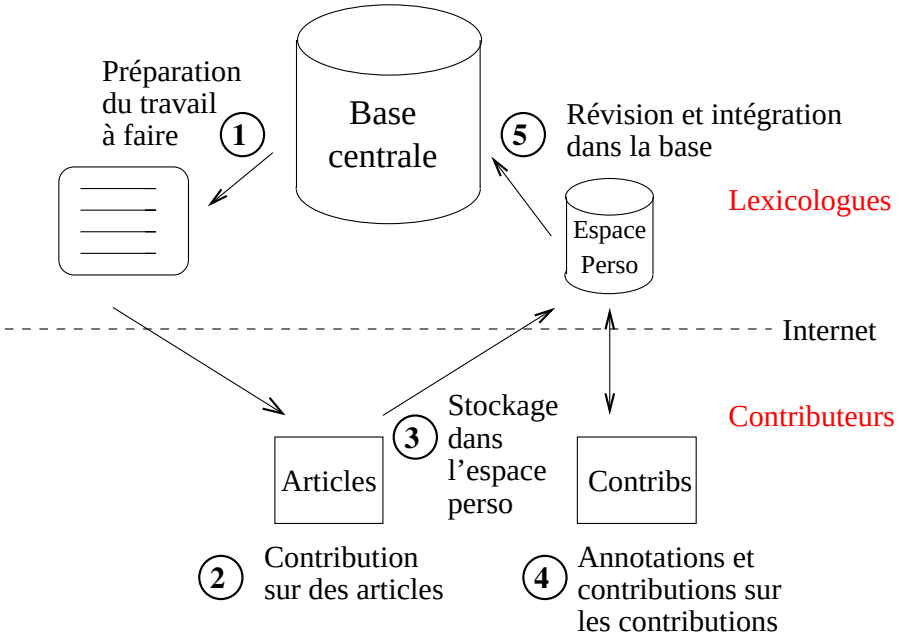


FIG. C.1 – *processus de gestion des contributions*

Il reste des problèmes à résoudre. Comment affecter les points, par exemple, comment répartir les points si une contribution a été annotée par plusieurs contributeurs; comment calculer les profils d'utilisateurs et comment automatiser le plus possible l'affectation des points et l'intégration de la base quand un certain niveau de confiance est atteint par un contributeur.

1.3. Intégration des expériences précédentes

1.3.1. Consultation des ressources

La consultation et la navigation dans une base lexicale telle que nous la définissons sont des tâches-clés de notre système. En effet, tous les utilisateurs de la base seront amenés à la consulter. De plus, il faut des outils de navigation capables de permettre à l'utilisateur de naviguer dans une grande quantité d'information sans être noyé.

L'utilisateur doit pouvoir effectuer des requêtes complexes, visualiser de grandes quantités de données et enfin déclencher des actions liées aux données visualisées.

Requêtes sur la méta-information

De nombreuses ressources sont disponibles. Il faut donc pouvoir les comparer et en sélectionner certaines que l'on veut consulter. Il faut que l'utilisateur puisse établir des requêtes sur la méta-information relative à chaque ressource présente dans la base.

Voici des exemples de requêtes possibles :

– Quels sont les dictionnaires avec du japonais en langue cible ?

– Combien d'entrées y a-t-il dans le OHD anglais-français ?

– Quelle est la microstructure complète du NODE (ou sa DTD) ?

– Quels sont les dictionnaires plus récents que 1980 ?

– Quels sont les droits d'accès et d'utilisation du dictionnaire FeM ?

Requêtes sur la macrostructure

La sélection dans la macrostructure consiste à ne sélectionner parmi tous les articles des dictionnaires sélectionnés précédemment que ceux qui intéressent l'utilisateur. Cela revient en fait à redéfinir une nomenclature personnalisée selon les propres critères de l'utilisateur.

Voici des exemples de sélection dans la macrostructure :

– seulement les verbes intransitifs du premier groupe;

– les articles classés selon l'origine des mots-vedettes (e.g. latin, esquimo, grec) puis selon l'ordre alphabétique;

– les mêmes articles que le Lexis mais réordonnés selon la phonologie (selon les deux dernières syllabes) comme un dictionnaire de rimes;

– tous les mots polonais de huit lettres finissant par "icz" (ou conformes à une expression régulière donnée), classés par ordre alphabétique mais sans aucune information (simple liste de mots);

– tous les mots d'origine latine du domaine juridique classés selon leur date d'apparition;

– tous les homophones du français classés par famille d'homophones et par ordre alphabétique (mot1 homophone 11 ... homophone 1n,...., mot m ... homophone m1 ... homophone mn);

– recherches par champs sémantiques, dérivations sémantiques, etc.

Requêtes sur la microstructure

Lorsque la liste des articles est définie, il faut sélectionner les informations dans la microstructure, si l'on ne veut pas voir toutes les informations contenues dans les articles.

Voici des exemples de sélection dans la microstructure :

– les définitions en français, des exemples d'utilisation et la traduction en anglais de l'entrée sans informations grammaticales sur le français ni sur la provenance des entrées.

– les exemples du DHO, l'étymologie du NODE et les informations de tous les dictionnaires bilingues anglais-français disponibles (en indiquant la provenance)

– la définition du Robert avec les images du Larousse si l'article en contient une.

Visualisation des données

L'utilisation d'un dictionnaire papier imprimé montre vite l'importance du contexte dans lequel un article se trouve, c'est à dire l'importance des autres articles qui l'entourent dans l'ordre défini par la nomenclature du dictionnaire (en général l'ordre alphabétique). Il est alors possible de découvrir des mots d'une même famille ou des variantes orthographiques, etc. Pour garder cet avantage, il faut pouvoir :

– accéder à tout moment aux articles précédant et suivant celui qu'on est en train de consulter, dans un ordre suivant la nomenclature précédemment définie par l'utilisateur. L'ordre par défaut suivant la nomenclature classique du dictionnaire;

– demander de visualiser une fenêtre plus ou moins grande représentant le contexte autour d'un article précis. Par exemple, ouvrir une fenêtre avec les 5 articles précédant et suivant celui qu'on consulte.

Pour visualiser des objets complexes comme des arbres, des graphes, du son ou de la vidéo, il faudra pouvoir utiliser des visualisateurs spécialisés. Ces visualisateurs seront utilisés comme des plug-ins par les interfaces de consultation/navigation. Il faut pour cela que leur interface API soit compatible avec ces interfaces. Une liste de ces plug-ins devra être mise à jour et disponible sur le serveur de la base lexicale.

Toutes les requêtes effectuées par un utilisateur devront être notées dans un historique. L'utilisateur pourra alors comparer des résultats ou relancer des requêtes déjà effectuées sans problèmes.

Personnalisation des requêtes et des résultats

La grande quantité d'informations et leur hétérogénéité nécessitent sélection et organisation de la part des utilisateurs qui consultent la base. Ceux-ci doivent pouvoir personnaliser le résultat de leurs requêtes. Les personnalisations portent sur :

– la structure. L'utilisateur peut choisir les informations qu'il veut visualiser et celles qui ne l'intéressent pas. S'il consulte plusieurs ressources, il peut choisir l'ordre d'apparition des ressources. Par exemple, si l'utilisateur ne connaît pas le malais, il peut le masquer.

– la présentation. L'utilisateur peut sélectionner un style particulier pour chaque élément d'information. Par exemple, il peut sélectionner la couleur bleue pour toutes les informations concernant le français ou l'italique pour les catégories grammaticales.

Le serveur doit donc proposer un système de préférences pour chaque utilisateur. Les préférences doivent être stockées sur le serveur pour pouvoir être utilisées lors d'une prochaine session. Il faut aussi pouvoir les changer dynamiquement en cours de consultation.

Plus généralement, les profils des utilisateurs seront stockés dans leur espace virtuel. Ces profils sont divers (préférences, compétences, intérêts) et peuvent varier selon les ressources. Une personne peut par exemple contribuer à la prononciation d'un dictionnaire français et consulter un dictionnaire bilingue anglais-japonais.

Les profils se répartissent dans plusieurs dimensions et le système doit les affiner automatiquement en fonction des actions des utilisateurs.

1.3.2. Rédaction des articles

La rédaction des articles et des liens entre les articles impose plusieurs formes de contraintes. En effet, plusieurs catégories de personnes sont amenées à contribuer à la construction d'un dictionnaire. Les types de contribution peuvent être très différents. Les plates-formes pour lexicographes seront donc nécessairement différentes pour répondre aux besoins de tous les contributeurs.

Accessibilité

La plate-forme doit être accessible et utilisable par le plus grand nombre de personnes possibles. Chaque personne souhaitant contribuer doit pouvoir le faire.

Certaines personnes ont un accès au réseau payant, d'autres un accès lent. Ces personnes ne peuvent pas travailler tout le temps en ligne. Il faut aussi pouvoir travailler en local, par exemple sur un ordinateur portable. Il faut donc faire en sorte que la totalité du travail (création, annotation, etc.) puisse se faire aussi bien en ligne qu'en local sans réseau.

Pour qu'un grand nombre de personnes puisse contribuer à la construction des ressources, il faut que les outils de rédaction puissent tourner sur des équipements de bas de gamme, bon marché et pas nécessairement puissants.

Spécifications de la plate-forme

Une plate-forme de lexicographe doit permettre :

- de travailler sur la rédaction d'un article ou sur l'élaboration de liens entre articles. Il faut donc concevoir deux types d'interfaces selon le travail à effectuer;
- de travailler sur tout l'article ou sur une partie seulement. Par exemple, un linguiste travaillera plus souvent sur la catégorie grammaticale d'un mot et un traducteur travaillera sur les traductions de ce mot dans les langues qu'il connaît;
- de voir toutes les ressources disponibles sur la base pour avoir le maximum d'informations à disposition de façon à faciliter les choix et aussi voir les contributions et annotations de certains groupes;
- d'accéder à des outils variés tels que des analyseurs morphologiques, des outils de recherche dans des corpus, etc.;
- d'utiliser des mécanismes d'aide à la rédaction comme l'affichage de listes fermées (catégories grammaticales, étiquettes, etc.) des vérificateurs de cohérence et de l'aide contextuelle.

Contraintes sur les ressources

Pour mener à bien un projet avec de multiples contributions, il ne faut intégrer aux ressources en construction que les contributions qui sont validées auparavant par un groupe d'experts. Les contributions qui n'ont pas été encore validées sont stockées dans l'espace virtuel des contributeurs en attendant la révision.

Pour chaque modification acceptée dans la base, il faut stocker le nom de la personne qui a fait cette modification et la date. Il faut gérer l'historique complet de toutes les modifications sur les ressources afin d'assurer la sécurité de la base (même le groupe "central" peut se tromper) et aussi la reconnaissance pour les contributeurs. Cela permettra par exemple de faire chaque mois un tableau de classement des contributeurs en fonction de leur mérite.

Il faut aussi gérer le travail restant à faire en proposant aux contributeurs une liste des articles les plus urgents à rédiger en leur donnant un ordre de priorité en fonction de leur fréquence d'apparition dans un corpus par exemple. Cette gestion est nécessaire d'une part pour éviter que deux contributeurs travaillent sur les mêmes données et d'autre part pour contrôler l'avancement de la construction des ressources.

1.3.3. Utilisation de modules externes

Prétraitement de la requête

Avant de lancer la requête sur la base lexicale, il faut permettre à l'utilisateur d'effectuer des prétraitements sur la requête. Ces prétraitements peuvent s'avérer très utiles surtout lorsque l'utilisateur ne maîtrise pas la langue du ou des mots-vedettes qu'il recherche. Les mots-vedettes sont dans la plupart des cas des lemmes. Il est parfois difficile d'accéder au lemme à partir d'une forme de surface. C'est pourquoi il serait intéressant de pouvoir ignorer les diacritiques, faire une analyse morphologique du mot-vedette ou lancer une correction orthographique avant interrogation.

Actions associées aux informations

Lorsque le résultat de la requête est affiché, il nous paraît intéressant de pouvoir associer des actions aux différentes informations présentes. Certaines actions peuvent nous permettre de continuer à naviguer dans la base par exemple en suivant des liens de traduction ou de synonymie.

Voici quelques exemples d'actions associées :

– Associer un conjugueur à un article représentant un verbe;

– Associer un phonétiseur à une prononciation;

– Associer un dictionnaire de synonymes à un autre dictionnaire pour créer des liens de synonymie.

Annotation des données

Tous les utilisateurs de la base peuvent avoir besoin d'annoter les parties de ressources qu'ils consultent. Les lexicographes et les contributeurs ont besoin de partager des remarques sur des parties d'articles en cours de rédaction. Les lecteurs peuvent ajouter leurs remarques personnelles et se construire leur propre dictionnaire personnel. Il faut aussi pouvoir annoter des liens interlingues.

Les annotations doivent donc pouvoir être partagées par plusieurs utilisateurs ou groupes d'utilisateurs. Ces groupes peuvent être des lexicographes travaillant sur la même ressource, des contributeurs qui ont en commun les mêmes langues, etc.

Du côté des clients, il faut une interface permettant d'annoter n'importe quelle partie d'information disponible sur la base. Du côté du serveur, il faut d'une part stocker les informations de gestion pour les utilisateurs et les groupes mais aussi stocker pour chaque utilisateur un dictionnaire personnel d'annotations.

Échange entre bases lexicales

Les bases lexicales sont en constante évolution. Certaines peuvent être réparties sur le réseau. Pour qu'elles puissent se synchroniser, il faut leur permettre d'échanger des données. Par exemple, un nouvel article rédigé sur une base sera répercuté sur les autres bases. Ces échanges de données entre bases lexicales partenaires doivent pouvoir se faire automatiquement. Il faut donc élaborer un protocole d'échange entre bases lexicales et clients/fournisseurs de données.

D'autre part, les applications clientes des bases lexicales peuvent devenir à leur tour fournisseurs de services. Par exemple, la construction d'un analyseur morphologique nécessite des catégories grammaticales que la base peut fournir. À son tour, l'analyseur morphologique une fois fini peut proposer ses services à la base. Celle-ci doit pouvoir interroger ses clients pour voir ou réutiliser les applications créées grâce aux données qu'elle a fourni. Il faut instaurer un système de rappel automatique (call back) des clients. Ceux-ci doivent implémenter une API standard définie par la base.

2. Définition du noyau de l'environnement avec SUBLIM

Définissons maintenant le noyau de notre environnement de manipulation, création et consultation de ressources lexicales. Comme nous l'avons dit plus haut, nous utilisons pour cela SUBLIM [Sérasset94e].

Mais ce système est défini avec des langages dont la syntaxe "noyau" est écrite en LISP. Pour respecter notre cahier des charges, nous redéfinissons SUBLIM en XML en y ajoutant les nouvelles fonctionnalités qu'imposent notre environnement.

2.1. Étude critique de SUBLIM

Un lexicologue utilisant le système SUBLIM décrit la structure interne de sa base lexicale en utilisant deux langages de haut niveau, LEXARD et LINGARD.

2.1.1. Architecture lexicale du système

Le langage LEXARD permet à l'utilisateur de définir la macrostructure de sa base, en spécifiant l'ensemble des dictionnaires de la base et leur type (monolingue, bilingue, interlingue). Dans l'exemple suivant, nous décrivons une architecture lexicale inspirée du projet EDR [EDR93] basée à la fois sur une approche bilingue et sur une approche interlingue. La base lexicale (voir figure C.3) comprend deux dictionnaires monolingues (anglais et japonais) reliés à la fois par deux dictionnaires bilingues unidirectionnels et par un dictionnaire interlingue (voir figure C.2).

Description des dictionnaires

On peut définir en LEXARD des dictionnaires monolingues, bilingues unidirectionnels, bilingues bidirectionnels ou interlingues. La figure C.2 montre des exemples de définition de dictionnaires.

Description de la base lexicale

On peut définir en LEXARD une base lexicale basée sur une approche par transfert ou sur une approche par pivot, avec ou sans le contrôle d'un administrateur de la base, localement ou par des pigistes travaillant chez eux, etc. La figure C.3 montre un exemple de définition d'une base lexicale.

Critiques

L'aspect de la gestion de différents utilisateurs avec des droits spécifiques pour chacun ainsi que la possibilité de créer des groupes d'utilisateurs n'ont pas été abordés dans LEXARD. Il faut ajouter la possibilité

```
-- /* définition des dictionnaires monolingues */ --
(define-monolingual-dictionary english
  :language "English"
  :owner "EDR")
-- /* définition du dictionnaire interlingue */ --
(define-interlingual-dictionary concept-dictionary
  :links (english japanese)
  :owner "EDR")
-- /* définition des dictionnaires bilingues */ --
(define-bilingual-dictionary japanese-english
  :type unidirectional
  :source japanese
  :target english
  :owner "EDR")
```

FIG. C.2 – *description de dictionnaires avec LEXARD*

```
(define-lexical-database EDR
  :owner EDR/
  :comment "Une base lexicale fondée sur une approche mixte"
  :dictionaries
    (english japanese english-japanese
    japanese-english concept-dictionary))
```

FIG. C.3 – *description d'une base lexicale avec LEXARD*

de définir un compte pour chaque utilisateur et des groupes d'utilisateurs. Pour chaque base lexicale, il faut prévoir des groupes de base : administrateurs, lexicologues, lexicographes, contributeurs et lecteurs.

Nos expérimentations menées sur des bases lexicales hétérogènes montrent l'importance de la méta-information sur les dictionnaires. Cette information permet aux utilisateurs d'avoir une meilleure idée de la qualité d'une ressource et de pouvoir sélectionner celle qui correspond le mieux à leurs besoins. Il faut donc ajouter des attributs décrivant la méta-information sur chaque dictionnaire.

LEXARD ne permet de décrire que deux niveaux dans une base lexicale : le niveau de la base lexicale regroupant tous les dictionnaires et le niveau du dictionnaire. Dans le cas d'un dictionnaire bilingue bidirectionnel, il faut donc une description avec LEXARD pour chaque partie de dictionnaire. Pourtant, ces deux descriptions partagent beaucoup d'informations en commun comme la date de création, l'auteur de la ressource, le domaine, etc.

Extension de LEXARD

Pour prendre en compte la méta-information sur les ressources ainsi que les informations sur les utilisateurs, nous étendons LEXARD de façon triviale par ajout de champs.

Pour affiner la description d'une base lexicale, nous complétons LEXARD en rajoutant un niveau dans la description avec : le niveau base lexicale qui liste les dictionnaires de la base, le niveau dictionnaire qui décrit un dictionnaire et le niveau volume qui décrit chaque volume de dictionnaire.

Les figures C.4, C.5 et C.6 montrent un exemple d'utilisation des fonctions LEXARD étendues

```
(define-lexical-database GETA-database
  :owner GETA
  :comment "base lexicale hétérogène du GETA"
  :creation-date "22/10/99"
  :users  (root MM GS CB)
  :groups  (universe administrators lexicologists)
  :partner-servers  (XRCE-analysers)
  :dictionaries  (FeM UNL-fr Homerica))
```

FIG. C.4 – *description d'une base lexicale avec LEXARD étendu*

```
(define-dictionary FeM
  :owner GETA
  :comment "French-English-Malay dictionary"
  :category "multilingual"
  :creation-date "21/01/97"
  :installation-date "21/01/97"
  :format "rtf"
  :hw-number 192460
  :bytes 9106261
  :type "unidirectional"
  :version 1
  :source-languages(French)
  :target-languages(English Malay)
  :contents "general vocabulary in 3 languages"
  :domain "general"
  :source "ML, YG, PL, Puteri, Kiki, CB, MA, Kim"
  :legal "all rights belong to ass.  Champollion"
  :administrators "Kim, ML"
  :volumes (FeM_fr_en_ms))
```

FIG. C.5 – *description du dictionnaire FeM avec LEXARD étendu*

```
(define-volume FeM_fr_en_ms
  :comment "Unique volume du FeM"
  :source-language "French"
-- /* articles composant le volume */ --
  ...
  ...
)
```

FIG. C.6 – *description du volume du FeM avec LEXARD étendu*

2.1.2. Architecture linguistique du système

Définition d'un objet linguistique

L'utilisateur définit ensuite avec LINGARD la microstructure des dictionnaires qu'il veut créer. Pour chaque dictionnaire, il décrit les structures informatiques des unités de son lexique. Pour cela, il utilise les

constructeurs de base du langage : arbre, graphe, automate, structure de traits, liste, ensemble, énumération, etc.

```
(def-linguistic-class french_entry
  (feature-structure
    (lexical_unit string)
    (Part-of-Speech (one-of "n.m" "n.f" "v.t"
      "v.i" "v.pr." "a" "adv" "loc" "prep"))
    (example (set-of string))
    (indexer string)
    (quality (one-of "manual" "auto" "reviewed"))
    (properties (set-of property))
    (uws (set-of string))))
```

FIG. C.7 – *description d'une unité lexicale avec LINGARD*

Dans la figure C.7, l'objet linguistique `french-entry` est une structure de traits `feature- structure`. Les traits ont des types différents : chaînes de caractères `string`, ensembles `set-of` ou énumération `one-of` de chaînes de caractères. Cet objet linguistique est relativement simple.

LINGARD permet aussi de représenter des structures plus complexes comme celle du DEC [Mel'tchuk84,88,92]. Chaque article du DEC étant très complet, nous ne représenterons ici qu'une partie d'article.

Les lexies du DEC sont décrites par un régime représentant les réalisations syntaxiques des arguments du prédicat. Ce régime est le reflet imprimable d'une structure complexe où l'on retrouve l'ensemble des combinaisons possibles des réalisations d'arguments. On peut représenter cette structure de manière abstraite par un automate dont chaque chemin forme un combinaison valide (voir figure C.8).

```
(def-linguistic-class régime
  (feature-structure
  ((automate automate-régime)
    (argument-order (list-of string)))
    (exemples exemples-regime))))
(def-linguistic-class automate-régime
  automaton :arcs réalisation-argument))
(def-linguistic-class exemples-régime
  (set-of ((feature-structure
    ((réalisations (list-of (string)))
    (exemple string))))))
```

FIG. C.8 – *description d'un régime du DEC avec LINGARD*

Cette structure s'exprime sous forme d'une structure de traits : le premier comporte un automate `automate-regime`, le second donne l'ordre dans lequel les arguments apparaissent dans le régime `argument-order` et le troisième donne l'ensemble des exemples `exemples-regime`.

Le langage LINGARD est très puissant puisqu'il permet de représenter un grand nombre de structures informatiques et linguistiques. L'utilisateur n'est pas limité à une théorie linguistique en particulier. Ce langage est générique.

Critique

Il est possible de représenter un grand nombre de structures de dictionnaires hétérogènes mais il n'est cependant pas possible de faire le lien entre ces structures. Lorsqu'on utilise plusieurs ressources à la fois, il semble pourtant intéressant de pouvoir les fusionner dans la mesure du possible. Nous proposons donc d'ajouter un mécanisme de fusion de structures linguistiques basé sur des objets linguistiques communs.

2.1.3. Architecture logicielle du système

Présentation

L'architecture logicielle de SUBLIM prévoit trois niveaux :

– BD pour le stockage physique des données

– Structure pour les différentes manipulations sur les entrées de dictionnaires. C'est lui qui manipule les structures que l'utilisateur a définies avec les langages LEXARD et LINGARD. Il comprend un vérificateur de cohérence sur toute la base.

– Présentation des informations à l'utilisateur.

Le fonctionnement de cette architecture est basé sur l'aller-retour entre les différents niveaux. Une requête sera formulée au niveau présentation, puis traduite en une structure qui sera elle même traduite en une requête au niveau BD.

Vérificateur de cohérence

Au niveau interne, en plus des manipulations de structures, SUBLIM comprend un vérificateur de cohérence permettant d'élaborer des contraintes définies par des linguistes sur les dictionnaires. Nous montrons ici un exemple de contrainte de cohérence globale définie au niveau de la base lexicale.

Dans l'exemple, la base lexicale est composée du dictionnaire *French* et du dictionnaire *Pivot*. Nous montrons dans la figure C.9 une partie de la structure du dictionnaire *French* :

```
(def-linguistic-class lexie
  (feature-structure
    (id integer)
    (headword string)
    (government-pattern government)
    (lexical-functions (set-of function))
    (examples (set-of example))
    (axies (set-of refaxie)))
(def-linguistic-class refaxie
    (link :source French::lexie
        :target Pivot::axie)
```

FIG. C.9 – *microstructure du dictionnaire* `French`

Nous montrons dans la figure C.10 une partie de la structure du dictionnaire *Pivot* :

Nous voulons vérifier maintenant la contrainte de cohérence globale suivante : l'élément `axie` indiqué comme cible sur un lien du dictionnaire *French* existe bien dans le dictionnaire *Pivot*. Cette contrainte porte sur tous les liens du dictionnaire *French*. L'expression booléenne de la figure C.11 vérifie l'existence de l'axie cible dans le dictionnaire *Pivot*.

```
(def-linguistic-class axie
  (feature-structure
    (id integer)
    (semantic-cat string)
    (fra (set-of reflexie)))
    (external-references (set-of reference))
(def-linguistic-class reflexie
  (link :source Pivot::axie
    :target French::lexie)
```

FIG. C.10 – *microstructure du dictionnaire* `Pivot`

```
(define-coherence-rule verification-target-link
  :applies-on (French::refaxie :target @axie)
  :verifies (exist?  Pivot::@axie)
  :error-message "L'axie cible du lien n'existe pas."
  :level :critical)
```

FIG. C.11 – *exemple de règle de cohérence en SUBLIM*

Critique

Le niveau bases de données limite le stockage à une base de données locale. Or le stockage des ressources ne se fait pas forcément en local. Il faut aussi pouvoir interagir avec des ressources distantes répondant à un protocole de communication spécifique.

L'interaction avec les clients et les fournisseurs n'a pas été abordé dans la définition de SUBLIM. Il faut spécifier des interfaces pour que le niveau interne communique avec le niveau fournisseurs de stockage des données et le niveau d'interaction avec les clients.

2.2. Passage de SUBLIM à XML

Dans le cahier des charges, nous avons choisi XML comme format de représentation interne de nos données. Dans cette section, nous allons donc redéfinir le système SUBLIM en XML en ajoutant des informations utiles à la gestion pour mettre en place un environnement de manipulation, création et consultation de bases lexicales hétérogènes multilingues.

Lors de l'implémentation, on pourra stocker tout ou une partie des documents XML représentant ces bases lexicales dans des bases de données classiques pour accélérer l'accès aux informations. Toute la base devra cependant rester disponible sous forme de fichiers XML, pour garantir la lisibilité et la portabilité des données.

2.2.1. L'espace de noms : DML

Pour noter nos documents XML, nous utiliserons un espace de noms XML [XML Namespaces]. Les espaces de noms permettent de qualifier les éléments et attributs utilisés dans les documents XML de manière unique en leur associant un espace de noms identifié par une URI (Uniform Resource Identifier).

Notre espace de noms est appelé DML pour Dictionary Markup Language. L'URI de notre espace de noms est noté `http://www-clips.imag.fr/geta/services/dml`. Cet URI est une adresse symbolique qui ne pointe pas obligatoirement sur un fichier. C'est seulement un moyen de définir un nom unique.

La déclaration d'un espace de noms sur un élément XML se fait avec l'attribut `xmlns`. On peut lui associer localement un raccourci qui sera ensuite utilisé comme préfixe par les éléments de niveau inférieur dépendants pour les qualifier. Ce raccourci est séparé de l'attribut `xmlns` par ":". Dans l'exemple de la figure C.12, il est noté par `dml`.

```
<MyElement
xmlns:dml="http://www-clips.imag.fr/geta/services/dml">
  ...
  <dml:MyDescendant/>
  ...
</myElement>
```

FIG. C.12 – *exemple d'utilisation de l'espace de noms DML*

Les espaces de noms servent à éviter les collisions entre deux éléments portant le même nom mais n'ayant pas la même sémantique, c'est à dire faisant partie de deux espaces de noms distincts. On peut par exemple trouver dans un document deux éléments préfixés de manière différente : `<dml:table>` et `<html:table>`. Ces éléments ne seront pas interprétés de la même manière.

Pour noter les liens entre nos documents XML comme des liens intradictionnaires de synonymie ou des liens interdictionnaires de traduction, nous utilisons la recommandation XLink [XLink 1.0]. L'espace de noms xlink est déclaré avec l'URI suivante : `http://www.w3.org/1999/xlink`.

Pour décrire un document XML, nous pouvons utiliser soit une DTD (Définition de Type de Document), soit un schéma XML [XML Schemas]. Les DTD sont celles de SGML, un peu simplifiées. Elles sont plus concises que les schémas mais souffrent de quelques déficiences : elles ne sont pas écrites en XML, ne supportent pas les espaces de noms et n'offrent qu'un typage très limité des données.

C'est pourquoi nous préférons utiliser des schémas XML pour décrire nos documents XML. Les schémas apportent des fonctionnalités intéressantes comme :

- un grand nombre de types de données intégrés comme les booléens, les entiers, les intervalles de temps, etc.,
- la possibilité de créer de nouveaux types par ajout de contraintes sur un type existant,
- la notion d'héritage,
- le support des espaces de noms,
- les indicateurs d'occurrences des éléments,
- la possibilité de définir les attributs et leurs valeurs par défaut en fonction du contexte d'apparition de l'élément qui les porte.

Par la suite, les descriptions seront notées dans le schéma XML DML dont l'URL est : `http://www-clips.imag.fr/geta/services/dml.xsd`.

Ce schéma est le schéma de base de notre base lexicale. La structure de tous les documents, éléments, attributs et types XML est décrite soit directement par ce schéma XML, soit par un schéma qui importe ou redéfinit ce schéma.

2.2.2. Types et attributs communs de DML

Pour certaines informations, nous définissons des types et des attributs communs à tous les éléments de DML. Cela permet de standardiser les données. Les schémas XML disposent à l'origine de types simples prédéfinis. Nous en avons sélectionné et réutilisé certains pour nos définitions. Ces types et ces attributs sont déclarés dans le schéma XML DML (voir le début de l'annexe A).

Dates et heures

Les dates sont représentées par le type de schéma XML dateType. La représentation lexicale d'une date est tirée du format étendu de la norme ISO 8601 : aaaa-mm-jjThh:mm:ss où "aaaa" représente l'année, "mm" le mois et "jj" le jour. La lettre "T" est le séparateur date/heure et "hh", "mm", et "ss" représentent respectivement les heures, les minutes et les secondes. L'attribut DML `date` est du type dateType.

Cette représentation peut être immédiatement suivie d'un "Z" pour indiquer le temps UTC (Temps Universel Coordonné). Pour indiquer le fuseau horaire, il faut indiquer la différence entre l'heure locale et l'heure UTC représentée comme hh:mm (les minutes sont obligatoires), précédée d'un signe + ou -.

Par exemple, pour indiquer 13h20 le 31 mai 1999 dans le Eastern Standard Time, qui est décalé de 5 heures avant l'UTC, il faut écrire 1999-05-31T13:20:00-05:00.

Délai de réponse

L'attribut DML `delay` porté par un élément indique le délai de réponse lorsqu'une requête a été faite sur cet élément. Ce délai est une durée exprimée avec le type simple `durationType` des schémas XML. Par exemple, 5 secondes et 10 centièmes sera indiquée : "5.10S".

Identificateur unique

L'attribut DML `id` porté par un élément est un identificateur unique dans toute la base lexicale. Il permet d'établir des liens entre plusieurs éléments. Son type reprend le type simple ID des schémas XML.

Historique des modifications

L'historique des modifications d'un élément porte un identificateur unique. L'élément référence son historique grâce à l'attribut DML `history` qui donne la valeur de l'identificateur unique de l'historique. Le type de cet attribut reprend le type simple ID des schémas XML.

Le fait de référencer un historique par un identificateur unique nous permet de stocker dans des fichiers différents les éléments et leur historique. Le fichier d'historique est référencé dans le fichier d'éléments par l'attribut DML `history-ref` qui indique l'URL du fichier d'historique. Le type de cet attribut est celui décrit par la norme xlink et utilisé pour les références : xlink:href.

Notation des langues

Pour noter les différentes langues, nous utilisons la norme ISO-639-2/T (T pour Terminologie) [ISO98] qui définit un code à 3 lettres pour chaque langue (français->fra; anglais->eng, malais->msa, etc.). Nous ajoutons aussi nos propres codes comme "unl" pour le langage UNL. Cette liste de codes représente le type lang. L'attribut DML `lang` est du type DML lang.

Encodages des documents

Pour noter l'encodage des différents documents de la base, nous définissons dans le schéma DML le type `encodingType`. Les valeurs de ce type sont celles décrites par l'IANA (Internet Assigned Number Authority) pour les encodages. Ce sont aussi les valeurs utilisées pour les types MIME (Multipurpose Internet Mail Extension). Parmi les plus utilisées nous trouvons ASCII sur 7 bits, ISO-8859-1 sur un octet (8 bits) pour les langues latines, Shift-JIS sur un ou deux octets pour le japonais, UTF-8 sur un octet pour les caractères Unicode, etc.

Statut d'un élément

L'attribut DML `status` d'un élément est utilisé pour indiquer son statut. Il peut prendre entre autres les valeurs `auto` si l'élément a été fabriqué automatiquement, `rough` si l'élément n'a pas encore été révisé, `revised` s'il a été révisé, etc.

Poids d'un élément

Nous prévoyons d'utiliser des systèmes de poids sur les éléments de notre base. Cela devrait permettre de personnaliser des préférences ou de noter des fréquences en ne mémorisant que les poids, et de transformer toute la base en une sorte de "réseau neuronal" [Véronis90] susceptible d'apprentissage.

Cependant, si nous voulons implémenter plusieurs systèmes de poids, nous devons séparer les éléments et les liens entre éléments de leur poids. Les éléments et les liens doivent porter des identificateurs uniques. Cela permettra de les référencer et de leur associer plusieurs poids provenant de théories ou de ressources différentes.

L'exemple suivant est une base constituée de 7 objets : l'article français `rivière` avec l'identificateur `fra01`, l'article français `fleuve` avec l'identificateur `fra02`, l'article français `cours d'eau` avec l'identificateur `fra03`, l'article anglais `river` avec l'identificateur `eng01`, un lien `lk01` reliant l'article français `rivière` et l'article anglais `river`, un lien `lk02` reliant l'article français `fleuve` et l'article anglais `river` et un lien `lk03` entre l'article français `cours d'eau` et l'article anglais `river`.

Pour tous ces objets, nous pouvons calculer des poids différents. Ici, nous avons noté la fréquence de consultation de ces objets par des utilisateurs de la base, la fréquence d'apparition de ces termes dans le corpus aligné anglais-français Hansard et les résultats de recherche de ces termes sur le moteur de recherche Google. Voici par exemple, dans le tableau C.1 plusieurs systèmes de poids pour les mêmes objets.

id	objet	consultation	fréq. Hansard	Google
fra01	rivière	6	1 621	314 000
fra02	fleuve	5	535	151 000
fra03	cours d'eau	2	485	28 600
eng01	river	10	6 974	12 900 000
lk01	rivière<->river		30/50	39 500
lk02	fleuve<->river		15/50	9 380
lk03	cours d'eau<->river		5/50	30 040

TAB. C.1 – *différents systèmes de poids sur les mêmes objets*

2.2.3. Sémantique du sous-ensemble CDM de DML

Définition du sous-ensemble

Pour pouvoir manipuler et fusionner certaines parties de ressources, nous avons besoin d'un formalisme commun de représentation de dictionnaires. Il existe des standards comme la TEI [Ide95], [Johnson95], MARTIF [Melby94], [ISO99b]; GENELEX/EAGLES [GENELEX93] et [GENETER] visant à l'universalité mais peu de ressources lexicales réelles les implémentent.

Nous avons donc fait un travail plus pragmatique consistant à identifier les informations apparaissant dans les ressources accumulées, ainsi que leur signification, et à les nommer de façon unique dans l'espace de noms DML.

Cet ensemble hiérarchisé est appelé Common Dictionary Markup et provient principalement de l'examen détaillé des dictionnaires FeM, DEC, DHO, OUPES, NODE, EDict, de la base ELRA-MÉMODATA, et du chapitre 12 de la TEI concernant les dictionnaires. Il contient les éléments les plus courants trouvés dans ces ressources, à savoir le mot-vedette, la prononciation, la catégorie grammaticale, le vocable, la lexie, l'étymologie, les exemples, les étiquettes, les gloses, etc. Ces éléments ont toujours la même sémantique. Par exemple, <dml:entry> réfère toujours à un article et <dml:headword> au mot-vedette de l'article.

Pour certains éléments ayant des listes fermées de valeurs, nous définissons pour chaque langue une liste représentant l'intersection des valeurs et des règles de conversion pour chaque ressource. Un exemple est la liste des catégories grammaticales d'une langue.

Lors de la récupération d'une ressource existante, nous essayons dans la mesure du possible de convertir les éléments originaux vers des éléments de cet ensemble. Si toutefois certaines informations ne sont pas représentables avec cet ensemble, les éléments originaux sont conservés. Si ces éléments se retrouvent fréquemment dans plusieurs ressources existantes, ils sont ajoutés à cet ensemble.

Les éléments de l'ensemble CDM sont utilisés comme points de référence dans un dictionnaire converti inconnu. La correspondance entre un élément de cet ensemble et un élément original lors de la récupération est effectué par un linguiste pour éviter des conflits possibles entre les éléments.

Le tableau C.2 liste une première version de l'ensemble d'éléments CDM. Les éléments ont été choisis sur la base de leur fréquence. L'ensemble lui-même évolue dès lors que de nouveaux dictionnaires sont explorés et récupérés.

La structure des éléments de l'ensemble CDM est décrite dans le schéma DML.

Exemple de correspondance

Lors de la récupération de ressources, il faut alors établir un tableau de correspondance des éléments à récupérer et des éléments de CDM. Le tableau C.3 a été utilisé pour récupérer les dictionnaires FeM et DHO décrits chacun en première partie.

Exemples de fusion

Grâce à cet ensemble d'éléments prédéfinis, nous pouvons maintenant fusionner des articles provenant de ressources hétérogènes s'ils contiennent les mêmes éléments CDM. Dans les exemples suivants, les éléments CDM sont préfixés par "dml:". Ils appartiennent à l'espace de noms DML. La sémantique de ces éléments est donc fixée par les tableaux C.2 et C.3.

Les figures C.13 et C.14 montrent un article du FeM et un article du DHO après récupération.

La fusion s'opère autour des éléments communs. Les éléments <entry> contiennent le même élément <headword>. Ils sont donc fusionnés. Les éléments <syntactic-cat> contiennent le même élément <pos>. Ils sont donc fusionnés. Cette fusion peut s'opérer par exemple grâce à un programme XSLT. La figure C.15 montre le résultat de la fusion.

<balise CDM>	**(équivalent TEI)**	**français**
<entry>	(entry)	article
<headword hn="">	(hom)(orth)	mot-vedette
<headword-var>	(oVar)	variante
<pronunciation>	(pron)	prononciation
<etymology>	(etym)	étymologie
<syntactic-cat>	(sense level="1")	vocable
<pos>	(pos)(subc)	catégorie
<lexie>	(sense level="2")	lexie
<indicator>	(usg)	glose
<label>	(lbl)	étiquette
<definition>	(def)	définition
<example>	(eg)	exemple
<translation>	(trans)(tr)	traduction
<collocate>	(colloc)	collocation
<link href="">	(xr)	lien
<note>	(note)	note

TAB. C.2 – *ensemble d'éléments CDM*

éléments CDM	**FeM**	**DHO**	**NODE**
<entry>	<fem-entry>	<se>	<se>
<headword>	<entry>	<hw>	<hw>
<pronunciation>	<french_pron>	<pr><ph>	<pr><ph>
<etymology>			<etym>
<syntactic-cat>		<sense n=1>	<s1>
<pos>	<french_cat>	<pos>	<ps>
<lexie>		<sense n=2>	<s2>
<indicator>	<gloss>	<id>	
<label>	<label>		<la>
<example>	<french_sentence>	<ex>	<ex>
<definition>			<df>
<translation>	<english_equ><malay_equ>		<tr>
<collocate>		<co>	
<link>	<cross_ref_entry>	<xr>	<xg>/<vg>
<note>		<ann>	

TAB. C.3 – *équivalents des éléments CDM dans le FeM, le DHO et le NODE*

2.2.4. Passage effectif de SUBLIM à XML

La transformation effective de SUBLIM à XML est possible pour les structures déjà définies en SUBLIM. Cette transformation peut s'effectuer automatiquement à l'aide d'un script. Les informations disponibles en SUBLIM n'étaient pas pertinentes pour nos expérimentations. Nous n'avons donc pas travaillé sur cette conversion.

```
<dml:entry>
  <dml:headword>abréger</dml:headword>
  <dml:pronunciation encoding="geta">abre-je-</dml:pronunciation>
  <dml:syntactic-cat>
    <dml:pos>v.tr.</dml:pos>
    <dml:lexie>
      <gloss lang="fra">un texte</gloss>
      <dml:translation lang="eng">to shorten</dml:translation>
<dml:translation lang="eng">to abridge</dml:translation>
<dml:translation lang="msa">memendekkan</dml:translation>
<dml:translation lang="msa">meringkaskan</dml:translation>
<french_phrase>je vous demande d'abréger votre lettre</french_phrase>
      <english_phrase>please shorten your letter</english_phrase>
<malay_phrase>sila ringkaskan surat anda</malay_phrase>
    </dml:lexie>
  </dml:syntactic-cat>
</dml:entry>
```

FIG. C.13 – *article provenant du FeM après récupération*

```
<dml:entry>
  <dml:headword>abréger</dml:headword>
  <dml:pronunciation>
    <ph>abKeZe</ph>
  </dml:pronunciation>
  <dml:syntactic-sense>
    <dml:part-of-speech>v.tr.</dml:part-of-speech>
    <dml:lexie>(
      <ic>rendre court</ic>) to shorten [
      <co>mot,expression</co>]; to summarize [
      <co>texte, discours</co>];
      <sl>  "télévision" en "télé"</sl>
to shorten "television" to "TV" (...)
    </dml:lexie>
  </dml:syntactic-sense>
</dml:entry>
```

FIG. C.14 – *article provenant du DHO après récupération*

2.3. Redéfinition des langages de SUBLIM en XML

Dans cette section, nous redéfinissons les langages LEXARD et LINGARD en XML en reprenant les mots-clés de ces langages. Cela nous permet de décrire entièrement la base sous forme de documents XML.

2.3.1. Définitions de macrostructure

Pour décrire la macrostructure de nos dictionnaires ainsi que de notre base lexicale, nous utilisons des documents XML. Le système analyse ces documents pour trouver les informations dont il a besoin pour ses

```
<dml:entry>
  <dml:headword>abréger</dml:headword>
  <dml:pronunciation encoding="geta">abre-je-</dml:pronunciation>
  <dml:syntactic-cat>
    <dml:pos>v.tr.</dml:pos>
    <dml:lexie provenance="FeM">
      <gloss lang="fra">un texte</gloss>
      <dml:translation lang="eng">to shorten</dml:translation>
<dml:translation lang="eng">to abridge</dml:translation>
<dml:translation lang="msa">memendekkan </dml:translation>
<dml:translation lang="msa">meringkaskan </dml:translation>
<french_phrase>je vous demande d'abréger votre lettre</french_phrase>
      <english_phrase>please shorten your letter</english_phrase>
<malay_phrase>sila ringkaskan surat anda </malay_phrase>
    </dml:lexie>
    <dml:lexie provenance="OHD">(
      <ic>rendre court</ic>) to shorten [
      <co>mot,expression</co>]; to summarize [
      <co>texte, discours</co>];
      <sl>\  "télévision" en "télé"</sl>
to shorten "television" to "TV" (...)
    </dml:lexie>
  </dml:syntactic-cat>
</dml:entry>
```

FIG. C.15 – *résultat de la fusion entre le FeM et le DHO*

opérations. Nous reprenons les fonctions du langage LEXARD en ajoutant des informations.

Organisation logique de la base

La figure C.16 montre l'organisation logique de la base lexicale.

Nous exprimons cette organisation en XML. L'élément <database> décrit une base lexicale. Dans cet élément sont listés les dictionnaires éventuellement hétérogènes qui composent la base.

L'élément <dictionary> décrit un dictionnaire de manière générale. Il référencie tous les volumes du dictionnaire.

L'élément <volume> décrit une partie de dictionnaire. Cet élément est composé principalement des articles de dictionnaire. Par exemple, un dictionnaire bilingue bidirectionnel anglais-français sera décrit par un seul objet <dictionary>. Les articles de dictionnaire seront ensuite répartis entre deux éléments <volume>. Les articles français->anglais seront mis dans un élément et les articles français->anglais dans un autre élément.

Description d'une base lexicale

Pour décrire une base lexicale, nous reprenons la fonction `define-lexical-database` du langage LEXARD avec l'élément <database>. La description formelle de cet élément est contenue dans le schéma XML dml en annexe de ce document. Elle est référencée par l'attribut `xsi:schemaLocation`.

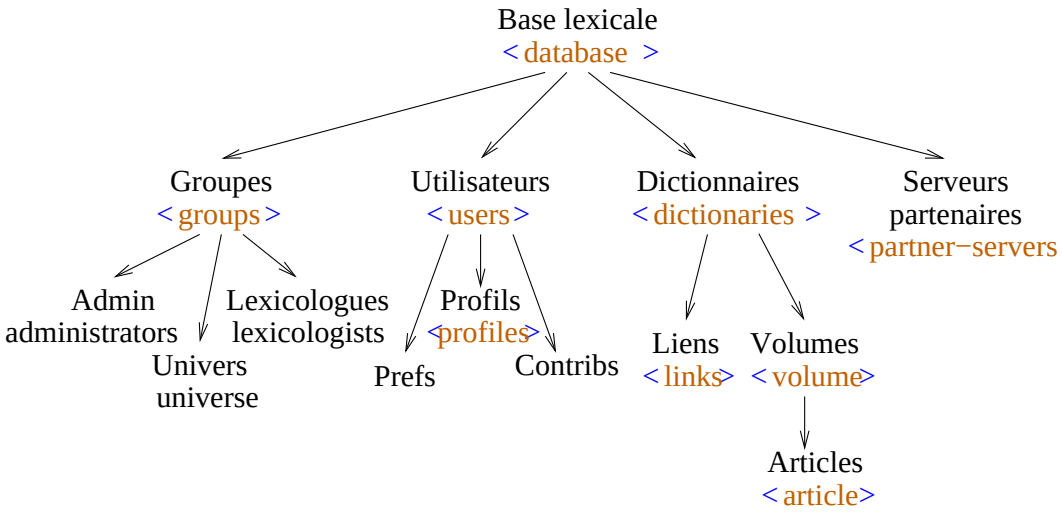


FIG. C.16 – *organisation logique d'une base lexicale*

Les modifications de l'élément <database> et de ses descendants sont stockées dans le document pointé par l'attribut history-ref.

Nous ajoutons à LEXARD la possibilité de définir différents utilisateurs et groupes de la base. Les différents rôles de ces utilisateurs sont décrits dans la première partie de ce chapitre. Au départ, trois groupes sont prédéfinis : l'univers (universe) contenant tous les utilisateurs de la base, les administrateurs de la base (administrators) et les lexicologues spécialistes (lexicologists) chargés du contre des données.

Les informations relatives à chaque utilisateur sont stockées dans un ficher à part référencé par l'élément < user-ref >.

Tous les dictionnaires sont référencés par des pointeurs sur les documents XML les décrivant. Les pointeurs sont les attributs href des éléments <dict-ref>. Ces éléments sont regroupés dans l'élément <dictionaries>.

L'exemple suivant est la version XML de la figure C.4 montrant une base lexicale :

```
<database xsi:schemaLocation="http://www-clips.imag.fr/geta/services/dml
http://clips.imag.fr/geta/services/dml/dml.xsd"
name="GETA Lexical Database"
creation-date="22/10/99"
history-ref="http://clips.imag.fr/geta/services/dml/database-his.xml"
owner="GETA">
  <partner-servers>     <user-ref name="XRCE Analyser" href="xrce.xml"/>
  </partner-servers>   <users>
    <user-ref name="Mathieu.Mangeot" href="mangeot.xml"/>
    <user-ref name="Mutsuko.Tomokiyo" href="tomokiyo.xml"/>
    <user-ref name="John.Doe" href="doe.xml"/>
 </users>
  <groups>
    <group name="universe">
      <user-ref name="Mathieu.Mangeot"/>
      <user-ref name="Mutsuko.Tomokiyo"/>
```

```
      <user-ref name="John.Doe"/>
    </group>
    <group name="lexicologists">
      <user-ref name="Mutsuko.Tomokiyo"/>
    </group>
    <group name="administrators">
      <user-ref name="Mathieu.Mangeot"/>
    </group>
  </groups>
  <dictionaries>
    <dict-ref name="FeM" href="FeM.xml"/>
    <dict-ref name="Papillon" href="papillon.xml"/>
  </dictionaries>
</database>
```

Description d'un dictionnaire

Pour décrire un dictionnaire, nous reprenons du langage LEXARD les fonctions `define-monolingual-dictionary`, `define-bilingual-dictionary` et `define-interlingual-dictionary` avec l'élément <dictionary>. La description formelle de cet élément est contenue dans le schéma XML dml en annexe de ce document. Elle est référencée par l'attribut `xsi:schemaLocation`.

Les modifications de l'élément <dictionary> et de ses descendants sont stockées dans le document pointé par l'attribut `history-ref`.

La méta-information sur la ressource est ajoutée.

Les éléments <category>, <type> et <links> décrivent la macrostructure du dictionnaire. L'élément <category> indique le type du dictionnaire. Nous distinguons quatre types de dictionnaires : monolingue, bilingue, multilingue et interlingue. L'élément <type> indique si les dictionnaires sont unidirectionnels, bidirectionnels ou à pivot. L'élément <links> indique les liens entre les lexiques qui composent le dictionnaire. Par exemple, si un dictionnaire est à structure pivot avec trois langues l'anglais, le français et le malais, il contient quatre lexiques : interlingue, anglais, français et malais, liés de la façon suivante :

```
<links>
  <link from="anglais" to="interlingue"/>
  <link from="français" to="interlingue"/>
  <link from="malais" to="interlingue"/>
</links>
```

Les volumes du dictionnaire sont référencés par leur nom unique. L'élément <volumes> regroupe toutes les références aux fichiers représentant les volumes. Ces références sont notées avec l'élément <volume-ref>.

Les langues sources et cibles sont indiquées par l'élément <languages> suivant la norme ISO 639-2/T [ISO98] avec leur code de trois lettres.

L'élément <contenu> indique le contenu du dictionnaire.

L'élément <domain> indique le domaine couvert par le dictionnaire. Un dictionnaire d'usage couvre le domaine général. Certains dictionnaires sont spécialisés dans des domaines précis comme la médecine, l'informatique, etc.

Nous indiquons aussi la taille du dictionnaire en octets, par <bytes>, et le nombre de mots-vedettes par <hw-number>.

Pour la gestion des différentes versions, nous indiquons le numéro de version (<version>), la date de création du dictionnaire (<creation-date>) et la date d'intégration du dictionnaire dans la base <installation-date>.

Pour les ressources qui ne sont pas récupérées vers notre format DML et installées localement, nous devons indiquer le format <format> et l'encodage (<encoding>). Les valeurs de l'élément (<encoding>) sont du type DML encodingType.

Nous indiquons aussi des informations sur le dictionnaire comme le fournisseur de la ressource (<source>), le propriétaire (<owner>), le responsable au niveau de la base (<responsible>), les droits sur le dictionnaire (<legal>) et des commentaires (<comments>).

La liste d'éléments de l'ensemble CDM consultables avec pour chacun le délai de réponse maximal est indiquée avec l'élément (<cdm-elements>). L'élément (<corpus>) est spécial : il permet d'indiquer que l'on recherche un mot contenu dans n'importe quel élément du dictionnaire.

L'emplacement du dictionnaire est noté avec une URI suivant la norme xlink [XLink 1.0].

Nous indiquons ensuite les droits des différents utilisateurs en suivant ces rôles. Les administrateurs (<administrators>) peuvent modifier le fichier de description du dictionnaire et son emplacement. Les lexicologues (<lexicologists>) peuvent effectuer des transformations sur tout le dictionnaire et lancer des vérificateurs de cohérence. Les lecteurs (<readers>) peuvent consulter le dictionnaire.

L'exemple suivant est la version XML de la figure C.5 décrivant le dictionnaire FeM.

```
<dictionary
xsi:schemaLocation="http://clips.imag.fr/geta/services/dml
http://clips.imag.fr/geta/services/dml/dml.xsd"
history-ref="http://clips.imag.fr/geta/services/dml/papillon-his.xml"
category="multilingual"
creation-date="21/1/97 00:00:00"
encoding="ISO-8859-1"
format="rtf"
hw-number="192460"
installation-date="23/06/99 15:04:00"
fullname="dictionnaire français-anglais-malais"
name="FeM"
owner="GETA"
type="unidirectional"
version="1">
  <languages>
    <source-language lang="fra"/>
    <target-language lang="eng"/>
    <target-language lang="msa"/>
  </languages>
  <contents>general vocabulary in 3 languages</contents>
  <domain>general</domain>
  <bytes>9106261</bytes>
  <source>ML, YG, PL, Puteri, Kiki, CB, MA, Kim</source>
  <legal>all rights belong to ass.  Champollion</legal>
  <comments>French-English-Malay dictionary</comments>
  <cdm-elements>
    <headword delay="1s"/>
    <pronunciation delay="5s"/>
```

```
    <part-of-speech delay="5s"/>
    <translation lang="eng" delay="5s"/>
    <translation lang="msa" delay="5s"/>
    <corpus delay="10s"/>
  </cdm-elements>
  <administrators>     <user-ref name="Kim, ML"/>
  </administrators>
  <volumes>
    <volume-ref name="FeM" href="fem_fr_en_ms.xml"/>
  </volumes>
</dictionary>
```

Description d'un volume

L'élément <volume> regroupe des articles de dictionnaires ayant la même langue source. La description formelle de cet élément est contenue dans le schéma DML en annexe de ce document. Elle est référencée par l'attribut xsi:schemaLocation.

Les modifications de l'élément <volume> et de ses descendants sont stockées dans le document pointé par l'attribut history-ref.

L'exemple suivant est la version XML de la figure C.6 décrivant le volume du dictionnaire FeM.

```
<volume
xsi:schemaLocation="http://clips.imag.fr/geta/services/dml
http://clips.imag.fr/geta/services/dml/dml.xsd"
history-ref="http://clips.imag.fr/geta/services/dml/fem-his.xml"
name="FeM_fr_en_ms"
source-language="fra">
  <article>
    ...
  </article>
    ...
</volume>
```

2.3.2. Définitions de microstructure

Pour représenter les microstructures des dictionnaires, nous proposons d'une part de reprendre les structures définies avec LINGARD en XML et d'autre part de définir quelques structures de traits de base. Nous reprenons ici la spécification de LINGARD point par point.

Arbres

Pour représenter un arbre de dépendances associé à la phrase "Le chat mange une souris.", par exemple, on peut utiliser un élément "nœud décoré" <nd> avec des attributs correspondant aux variables grammaticales.

```
<nd ul="manger" time="present" aspect="imperfectif">
  <nd ul="chat" determ="defini" gnr="masc" pos="-1"/>
  <nd ul="souris" determ="indefini" gnr="fem" pos="+1"/>
</nd>
```

Graphes

La figure C.17 montre un graphe UNL représentant la phrase "Monkeys eat bananas.".

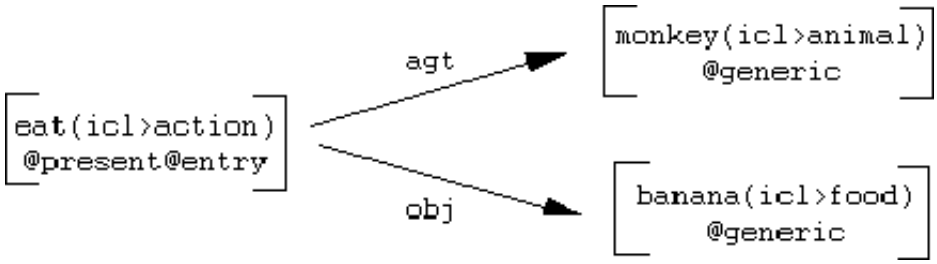


FIG. C.17 – *exemple de graphe UNL*

La définition de ce graphe se fait en XML se fait de la manière suivante :

```
<graph xmlns:xl="http://www.w3.org/1999/xlink"
xl:type="extended">
  <nodes>
    <node xl:type="locator" xl:label="n001">
     eat(icl>action).@present.@entry</node>
    <node xl:type="locator" xl:label="n002">
     monkey(icl>animal).@generic</node>
    <node xl:type="locator" xl:label="n003">
     banana(icl>food).@generic</node>
  </nodes>
  <arcs>
    <arc xl:type="arc" type="oriented" xl:from="n001"
     xl:to="n002">agt</arc>
    <arc xl:type="arc" type="oriented" xl:from="n001"
     xl:to="n003">obj</arc>
  </arcs>
</graph>
```

La norme xlink [XLink 1.0] est utilisée pour décrire les arcs. Le type des arcs est soit orienté `type="oriented"`, soit bijectif `type="bijective"`. L'origine et l'extrémité des arcs sont notés avec les identificateurs des nœuds `from="n001"` et `to="n002"` .

Liens

La définition d'un lien se fait en utilisant aussi la norme xlink [XLink 1.0]. Nous ajoutons aussi nos propres attributs :

– l'attribut type="bidirectionnal" ou type="oriented" indique si le lien est bidirectionnel ou non;

– l'attribut id est du type des schémas XML ID. Il permet d'attribuer un identificateur unique à chaque lien. Cet indentificateur sera utilisé par la suite pour implémenter des systèmes de réseaux pondérés;

– le texte de l'élément permet d'étiqueter les liens.

Voici un exemple de lien :

```
<link type="oriented" id="l001"
href="example.xml#xpointer(//node[xl:label='n002'])"/>
```

La référence à l'élément externe se fait avec l'attribut `href`. La référence est notée sous forme d'URI. Si l'objet n'a pas d'identificateur unique (id), le lien est décrit en suivant la norme XPointer [XPointer]. Sinon, il est pointé de cette façon :

```
<link type="oriented" id="l001" href="example.xml#n002"/>
```

Automates

La définition d'un automate suit celle d'un graphe. Le nœud de départ est noté avec l'attribut xl:title ="`starting-node`". Les nœuds d'arrivée sont notés avec l'attribut xl:title="`ending-node`". L'exemple de la figure C.18 représente le régime de la lexie `ENSEIGNER (X enseigne Y à Z)` [Mel'tchuk95].

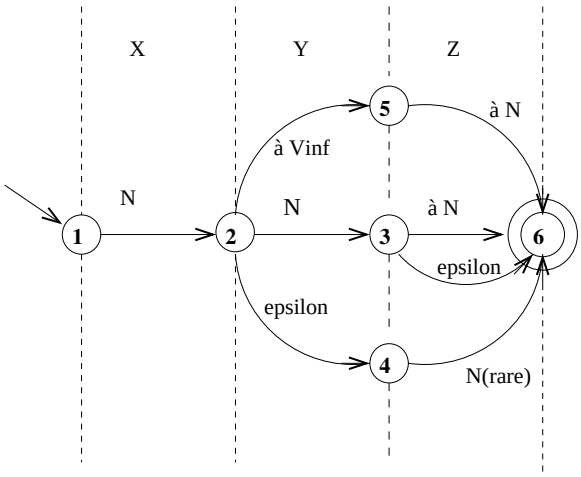


FIG. C.18 – *régime d'ENSEIGNER sous forme d'automate*

L'automate de la figure C.18 est représenté en XML de la façon suivante :

```
<automaton xmlns:xl="http://www.w3.org/1999/xlink"
  xl:type="extended">
  <nodes>
    <node xl:type="locator" xl:title="starting-node"
      xl:label="1"/>
    <node xl:type="locator" xl:label="2"/>
    <node xl:type="locator" xl:label="3"/>
    <node xl:type="locator" xl:label="4"/>
    <node xl:type="locator" xl:label="5"/>
    <node xl:type="locator" xl:title="ending-node"
      xl:label="6"/>
  </nodes>
  <arcs>
    <arc xl:type="arc" type="oriented" xl:from="1"
      xl:to="2">N</arc>
   <arc xl:type="arc" type="oriented" xl:from="2"
```

```
      xl:to=3">N</arc>
   <arc xl:type="arc" type="oriented" xl:from="2"
      xl:to="4">à V_inf</arc>
   <arc xl:type="arc" type="oriented" xl:from="2"
      xl:to="5">epsilon</arc>
   <arc xl:type="arc" type="oriented" xl:from="3"
      xl:to="6">à N</arc>
   <arc xl:type="arc" type="oriented" xl:from="3"
      xl:to="6">epsilon</arc>
   <arc xl:type="arc" type="oriented" xl:from="4"
      xl:to="6">à N</arc>
   <arc xl:type="arc" type="oriented" xl:from="5"
      xl:to="6">N(rare)</arc>
 </arcs>
</automaton>
```

Fonctions

L'exemple suivant représente la fonction lexicale `[lambda]x1 (CausOper`$_1$`x0x1)` définie par Igor Mel'tchuk [Mel'tchuk95]. Cette fonction signifie pour un mot-clé qui est un nom de sentiment "faire en sorte que quelqu'un éprouve". Les résultats de son application à la lexie DÉSESPOIR sont les suivants : pousser, réduire quelqu'un au désespoir, jeter quelqu'un dans le désespoir, frapper quelqu'un de désespoir. La fonction est notée en XML de la façon suivante :

```
<function name="CausOper_1">
  <arguments>
    <first value="desespoir"/>
  </arguments>
  <valgroup>
    <value>pousser</value>
    <value>réduire [qqun au désespoir]</value>
    <value>jeter [qqun dans le désespoir]</value>
    <value>frapper [qqun de désespoir]</value>
  </valgroup>
</function>
```

Structures de traits

Les structures de traits sont notés en XML par des éléments. Si les traits sont typés, le type est noté par un attribut, si le trait a plusieurs valeurs, l'élément est dupliqué.

```
<trait1 type="type1">valeur1</trait1>
<trait1 type="type2">valeur2</trait1>
```

Ensembles

Les ensembles sont définis au niveau de la définition des documents. Dans un schéma XML, les ensembles sont notés de la façon suivante :

```
<complexType mixed="yes" name="jours-feriés">
   <choice minOccurs="0" maxOccurs="unbounded">
```

```
        <element name="samedi" type="string"/>
        <element name="dimanche" type="string"/>
    </choice>
</complexType>
```

Cet exemple définit l'ensemble `jours-feriés` comme étant le samedi et le dimanche.

Disjonction

La disjonction est aussi définie au niveau de la définition des documents. Dans un schéma XML, la disjonction est notée avec l'élément `<xsd:choice>` de la façon suivante :

```
<complexType name="section">
    <sequence>
        <choice>
            <element name="paragraphe" type="string"/>
            <element name="illustration" type="string"/>
        </choice>
    </sequence>
</complexType>
```

Cet exemple définit une section comme étant une succession de paragraphes et d'illustrations.

Types de base

Le type de base d'un document XML est la chaîne de caractères. Grâce aux schémas XML, nous pouvons utiliser un certain nombre d'autres types de base représentés par la figure C.19.

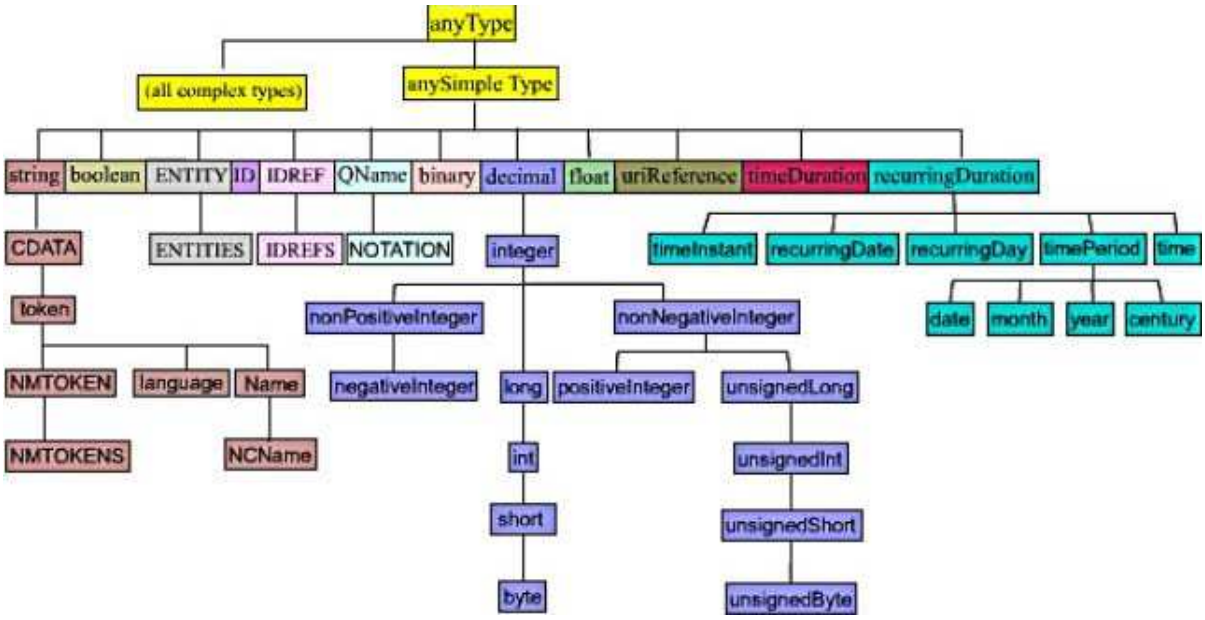


FIG. C.19 – *types simples des schémas XML*

Héritage

Le mécanisme d'héritage de LINGARD est réalisé en XML par un programme implémentant une API DOM [DOM]. À l'heure actuelle, les principaux langages de programmation répandus comme C, C++, java, Perl implémentent le DOM.

2.3.3. Vérificateurs de cohérence

Nous avons vu dans le chapitre précédent comment rédiger une contrainte de cohérence grâce au langage de vérification de cohérence inclus dans SUBLIM. Nous allons voir maintenant comment rédiger des contraintes de cohérence sur une base lexicale définie avec DML.

Comme toute l'information contenue dans la base est décrite sous forme de documents XML, il est possible d'écrire des modules de vérification avec un langage de programmation implémentant une API DOM [DOM].

Nous montrons dans cet exemple une solution simple utilisant le langage XSLT [XSLT 1.0] pour exprimer ces contraintes. Pour faciliter la compréhension, nous avons repris les exemples du chapitre précédent exprimés avec SUBLIM. Nous avons donc une base lexicale composée du dictionnaire *French* et du dictionnaire *Pivot*.

Voici une partie de la structure d'un article du dictionnaire *French* :

```
<element name="lexie">
  <complexType>
    <sequence>
      <element name="headword" type="string"/>
      <element ref="government-pattern"/>
      <element ref="lexical-functions"/>
      <element ref="examples"/>
      <element name="axies">
        <complexType>
          <sequence>
          <element ref="refaxie"/>
          </sequence>
        </complexType>
      </element>
    </sequence>
    <attribute ref="id"/>
  </complexType>
</element>
<element name="refaxie">
  <complexType>
    <attribute ref="href"/>
  </complexType>
</element>
```

Voici une partie de la structure du dictionnaire *Pivot* :

```
<element name="axie">
  <complexType>
    <sequence>
      <element name="semantic-cat" type="string"/>
      <element name="fra">
```

```
            <complexType>
              <sequence>
              <element ref="reflexie"/>
              <element ref="external-references"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
        <attribute ref="id"/>
      </complexType>
    </element>
    <element name="refaxie">
      <complexType>
        <attribute ref="href"/>
      </complexType>
    </element>
```

Nous pouvons élaborer des contraintes sur ces dictionnaires exprimées par exemple avec le langage XSLT. Les contraintes sont écrites dans des feuilles de styles XSLT. Celles-ci sont ensuite lues et exécutées sur des documents XML par un moteur XSLT. Les exemples suivants ont été élaborés en collaboration avec Marc Salvati, étudiant en première année de magistère d'informatique.

Le langage XSLT comporte quelques restrictions. Tout d'abord, nous devons fusionner tous les articles des dictionnaires différents dans un seul document pour pouvoir vérifier des contraintes de cohérence. Nous pouvons réaliser cette opération grâce à une feuille de style XSLT :

```
    <xsl:stylesheet version="1.0">
      <xsl:param name="f_source"/>
      <xsl:param name="f_ref"/>
    <!-- application du modèle sur le nœud racine -->
<xsl:template match="/">
    <!-- l'élément racine du résultat est :  fusion -->
       <fusion>
          <xsl:variable name="source" select="document($f_source)"/>
          <xsl:variable name="ref" select="document($f_ref)"/>
    <!-- copie du fichier $f_source dans le résultat -->
          <xsl:copy-of select="$source"/>
    <!-- copie du fichier $f_ref dans le résultat -->
          <xsl:copy-of select="$ref"/>
       </fusion>
      </xsl:template>
    </xsl:stylesheet>
```

Nous voulons vérifier la contrainte de cohérence globale suivante : l'élément `axie` indiqué comme cible sur un lien du dictionnaire *French* existe bien dans le dictionnaire *Pivot*. Cette contrainte porte sur tous les liens du dictionnaire *French*, elle vérifie l'existence de l'axie cible dans le dictionnaire *Pivot* :

```
    <xsl:stylesheet version="1.0">
    <!-- application du modèle sur le nœud fusion -->
<xsl:template match="fusion">
    <!-- application du modèle identification pour chaque refaxie -->
       <xsl:for-each select="dictionary/lexie/axies/refaxie">
```

```
        <xsl:call-template name="identification"/>
      </xsl:for-each>
    </xsl:template>
  <!-- modèle identification -->
   <xsl:template name="identification">
      <xsl:variable name="axicour" select="string(attribute::href)"/>
      <xsl:value-of select="$axicour"/>
  <!-- pour chaque axie -->
     <xsl:for-each select="/fusion/dictionary/axie">
  <!-- si $axicour pointe sur axie :  OK -->
        <xsl:if test="$axicour=string(attribute::id)">OK</xsl:if>
      </xsl:for-each>
    </xsl:template>
  </xsl:stylesheet>
```

Nous voulons vérifier maintenant la contrainte de cohérence locale suivante sur une lexie : l'élément <axies> ne doit comporter qu'un lien vers une axie. S'il y a deux liens, c'est une erreur. Il faudra ensuite soit fusionner les deux axies, soit créer une deuxième lexie. Cette contrainte porte sur toutes les lexies du dictionnaire *French*, elle compte le nombre de liens de l'élément <axies> :

```
  <xsl:stylesheet version="1.0">
  <!-- application du modèle sur les lexies -->
   <xsl:template match="lexie">
  <!-- si la lexie a plusieurs refaxie :  erreur -->
     <xsl:if test="count(axies/refaxie)>1">
  Error:  The lexie &lt;a href="#<xsl:value-of select="@id"/>"
  <xsl:value-of select="headword/text()"/> &lt;/a&gt; is linked to
various axies!
      </xsl:if>
    </xsl:template>
  </xsl:stylesheet>
```

Cette feuille de style produit un fichier d'erreur avec des liens activables vers les lexies posant problème.

3. Paradigme de construction coopérative

Dans ce chapitre, nous appliquons le principe de développement coopératif de LINUX à la construction de ressources lexicales. Pour cela, nous devons définir un serveur et ses différents utilisateurs. La construction des ressources se fait en coopération par une communauté de contributeurs bénévoles.

Toutes les contributions sont révisées par un groupe de spécialistes lexicographes avant d'être intégrées à la base lexicale. La base fonctionne sur un système de points : toute contribution acceptée augmente le nombre de points du contributeur et à l'inverse, exportation de la base diminue le nombre de points. La consultation demeure gratuite pour tous les utilisateurs.

3.1. Définition du serveur et ses différents utilisateurs

3.1.1. Mise en place du serveur

L'architecture logicielle de notre serveur est tirée de celle de SUBLIM qui distingue fortement les problèmes de stockage, de manipulation et de visualisation de données (voir figure C.20). Elle est basée sur trois niveaux :

– niveau fournisseurs : ce niveau rassemble les fournisseurs de ressources. Elles peuvent être stockées et accédées en local. Il est possible d'utiliser diverses formes de stockage comme le stockage dans des fichiers avec accès par index, dans des bases de données, ou tout en mémoire. Elles peuvent aussi être stockées dans un endroit distant accessible par le réseau. Ce niveau est invisible pour l'utilisateur.

– niveau interne : ce niveau est en charge des différentes manipulations sur les articles de dictionnaires ainsi que la récupération de ressources existantes. Ce niveau correspond aux structures XML/DML définies plus haut.

– niveau clients : ce niveau est en charge de l'interaction avec les clients. Le client rédige ses requêtes et spécifie ses préférences de présentation. Cette présentation n'est pas nécessairement proche de la structure interne utilisée. Il est possible d'avoir plusieurs vues différentes d'un même objet linguistique.

Le fonctionnement de cette architecture est basé sur l'aller-retour entre les différents niveaux. Une requête sera formulée par un client, puis traduite en une structure XML du niveau interne. Cette structure sera elle même traduite en une requête de fournisseur. Le résultat sera transformé en un ensemble de structures XML du niveau interne, qui sera visualisé selon les préférences du client.

Le niveau interne est une plate-forme d'échange entre clients et fournisseurs. Les clients et les fournisseurs communiquent avec la base lexicale par l'intermédiaire d'APIs. Toutes les API sont définies sous forme de documents XML. La description formelle des éléments XML est réalisée dans le schéma XML dml en annexe A de ce document.

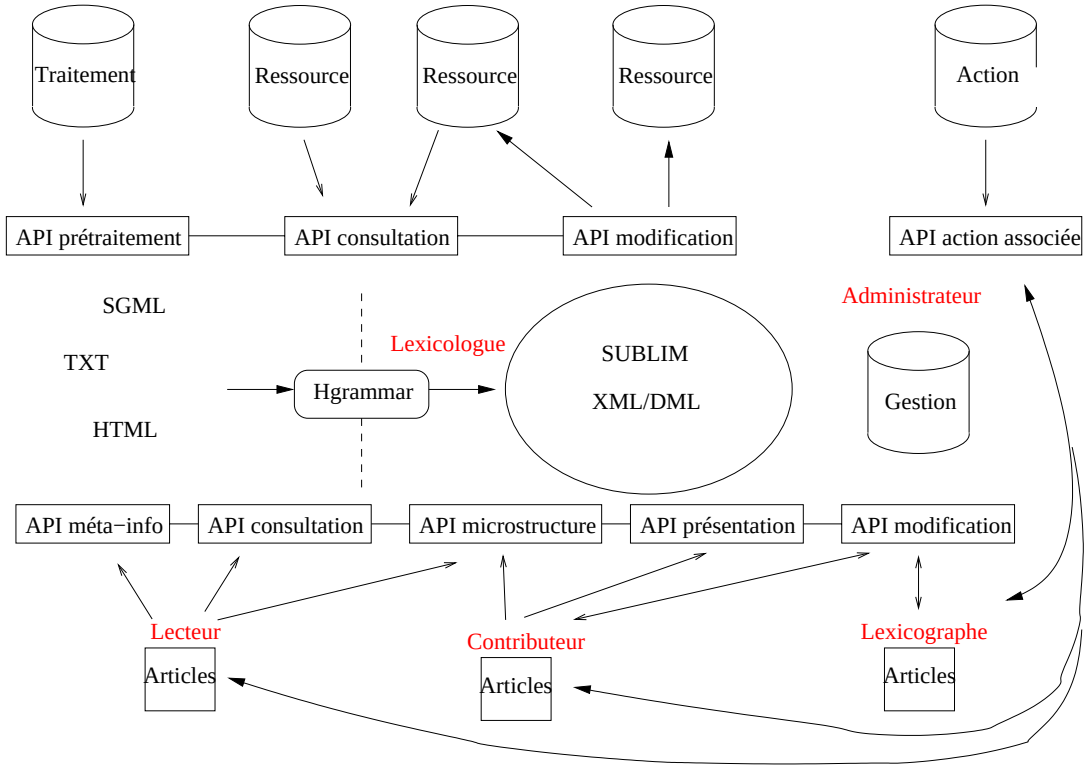


FIG. C.20 – *schéma général de l'environnement et ses API*

Les API destinées aux clients sont disponibles sur le serveur de la base. La personne qui souhaite développer une application cliente de la base doit respecter les API définies. Les API destinées aux fournisseurs sont aussi disponibles sur le serveur de la base. Lorsque l'administrateur de la base ajoute un nouveau fournisseur, il renseigne tous les éléments de l'API correspondant au type de service proposé par le fournisseur. Lorsque la base lexicale a besoin d'un service, elle lit le fichier où est décrite l'API du fournisseur et s'y connecte ensuite automatiquement.

3.1.2. Description des utilisateurs

L'élément <user> regroupe les informations disponibles pour chaque utilisateur. Cet élément représente l'espace virtuel de l'utilisateur. Nous décrivons pour chaque utilisateur un certain nombre d'informations :

- son nom avec l'attribut name;
- son login avec l'élément <login>;
- son mot de passe avec l'élément <password>;
- son adresse électronique avec l'élément <email>;
- ses différents profils avec l'élément <profiles> (on distingue pour l'instant trois types différents : compétences, intérêt et activité);
- un pointeur sur ses annotations sur le contenu de la base avec l'élément <annotations>;

- les contributions qu'il envoie à la base avec l'élément avec l'élément <contributions>;
- les points qu'il a accumulés en contribuant à la base avec l'élément avec l'élément <credits>;
- un pointeur sur l'historique de ses requêtes avec l'élément <requests>;
- des pointeurs sur ses préférences. Dans l'exemple suivant, nous avons stocké une feuille de style CSS avec l'élément <xsl:stylesheet> pour les préférences de présentation de l'utilisateur Mathieu.Mangeot;
- les groupes auxquels l'utilisateur appartient avec l'élément <groups>.

La description formelle de l'élément <user> est contenue dans le schéma XML dml en annexe de ce document. Elle est référencée par l'attribut `xsi:schemaLocation`. Les informations sont toutes représentées sous forme de documents XML. Lorsque le système a besoin d'une information, il lit le document XML correspondant et l'analyse pour trouver l'information. Les modifications de l'élément <user> et de ses descendants sont stockées dans le document pointé par l'attribut `history-ref`.

```
<user
xsi:schemaLocation="http://www-clips.imag.fr/geta/services/dml
http://www-clips.imag.fr/geta/services/dml/dml.xsd"
name="Mathieu MANGEOT"
creation-date="22/10/2001"
history-ref="mangeot-his.xml">
  <login>Mathieu.Mangeot</login>
  <password>toto</password>
  <email>Mathieu.Mangeot@imag.fr</email>
  <profiles>
   <competences>
      <cat level="good">translation</cat>
      <eng level="good">translation</eng>
      <fra level="mother tongue">phonetic, collocations, examples,
grammar</fra>
      <hun level="beginner"/>
      <jpn level="beginner"/>
      <spa level="good">translation</spa>
    </competences>
    <interests>
      <interest lang="hun,jpn"/>
    </interests>
    <activities>
      <activity dictionary="DicoSzótár">administration,
indexing</activity>
      <activity dictionary="FeM">interface</activity>
      <activity dictionary="Nihongo">administration,
indexing</activity>
      <activity dictionary="Papillon">administration</activity>
    </activities>
  </profiles>
 <credits>10</credits>
  <annotations href="mangeot-ann.xml"/>
```

```
    <contributions>
      <contribution source="French.xml" href="mangeot-cnt1.xsl"/>
    </contributions>
    <requests href="mangeot-req.xml"/>
    <xml-stylesheet type="text/css" href ="mangeot-sty.css"/>
    <groups>
      <group-ref name="universe"/>
      <group-ref name="administrators"/>
    </groups>
  </user>
```

3.2. Gestion des contributions

Lors de la première connexion au serveur, les utilisateurs doivent se créer un compte. Ils peuvent ensuite contribuer à n'importe quel moment. L'utilisateur peut contribuer spontanément sur un article s'il a par exemple constaté une erreur en consultant la base, ou consulté sur le serveur une liste de choses à faire. Cette liste est proposée par les spécialistes lexicologues de la base en fonction du profil des utilisateurs si possible avec une aide automatique (détection de schémas d'incomplétude, d'erreur, etc.).

Pour éviter que la base ne soit polluée par des contributions non vérifiées, celles-ci sont tout d'abord stockées sous forme de documents XML dans l'espace virtuel du contributeur. Elles sont ensuite vérifiées par les spécialistes lexicographes qui décident ou non de les intégrer à la base.

3.2.1. Vérification des données

Le groupe des spécialistes lexicologues a plusieurs tâches à effectuer. Il doit constamment vérifier les données présentes dans la base, puisque celle-ci n'est jamais figée dans un état stable. La vérification des données permet de préparer le travail à faire. Les articles à vérifier seront proposés aux contributeurs en fonction de leur profil de compétences.

Il doit enfin vérifier et intégrer les contributions proposées par les contributeurs et stockées dans leur espace virtuel.

Pour vérifier les données présentes sur la base, les lexicologues élaborent des contraintes de cohérence sur une interface spécialisée. La contrainte de cohérence sera ensuite traduite par exemple en feuille de style XSLT et appliquée à la base lexicale en tâche de fond, lorsque le serveur est inactif ou que le nombre de requêtes est limité. Des pointeurs sur les données posant problème sont alors générés par l'application des feuilles XSLT.

Par exemple, les lexicologues peuvent vérifier la validité des liens présents dans la base. Ils rédigeront alors des contraintes de cohérences traduites en une feuille de style du type de celle de la section 2.3.3.

Les lexicologues préparent ensuite avec ces pointeurs des ensembles de données à vérifier. Les contributeurs se connectent alors à la base pour piocher dans ces ensembles de données.

Ils vérifient aussi les diverses contributions stockées dans les espaces virtuels des contributeurs. Lorsqu'ils décident d'intégrer des nouveaux articles ou des contributions dans une ressource, ils ajoutent des informations dans un fichier d'historique des modifications. La ressource modifiée comporte un lien vers cet historique grâce à un identificateur. À chaque modification, il faut stocker le nom du modificateur et la date et éventuellement des commentaires.. Voici un exemple d'historique :

```
<administration id="h00001">
  <creation indexer="Automatic"
    date="01/11/00">fusion FeM & JMDict</creation>
```

```
  <modification indexer="MM"
    date="01/12/00">ajout du QSyn assassinat</modification>
  <modification indexer="MM"
    date="02/12/00">ajout du QSyn homicide</modification>
<modification indexer="MM"
    date="17/04/01">ajout de Example e1</modification>
  <revision indexer="CB"
    date="06/12/00">Tout OK, RAS</revision>
</administration>
```

La description formelle des historiques est définie dans le schéma XML dml en annexe de ce document.

3.2.2. Stockage des contributions

Les contributeurs rédigent leurs contributions à l'aide d'une interface spécialisée. Ils ne rédigent pas directement leurs contributions en XML. En effet, ce langage est difficile voire impossible à comprendre pour un contributeur non informaticien.

Les contributions sont stockées avec leur date de création sous forme de documents XML. Ces contributions ne sont pas visibles du public tant qu'elles ne sont pas vérifiées et intégrées dans la base. Cependant, le contributeur peut partager ses contributions avec d'autres contributeurs, au cas par cas, ou avec les groupes dont il fait partie. Les autres contributeurs peuvent alors annoter à leur tour ses contributions avant qu'elles ne soient intégrées à la base. Lors de la révision, le spécialiste lexicographe pourra visualiser toutes les contributions sur un article ainsi que les annotations sur l'article lui-même ainsi que sur les contributions.

Trois types de contribution sont possibles : l'import de lexiques ayant leur propre format, l'ajout de nouveaux articles et enfin les contributions sur une partie d'article.

Import de lexiques

Certains traducteurs développent leurs propres lexiques privés. Ils peuvent contribuer en envoyant leurs lexiques à la base. Les sociétés ou laboratoires possédant des ressources lexicales peuvent contribuer de la même manière. Ces ressources ont un format propre. Elles doivent être récupérés puis intégrés dans la base par un spécialiste lexicographe. Dans un premier temps, les ressources sont récupérées avec leur structure logique plus ou moins complète puis certaines peuvent être ajoutées à la soupe lexicale en cours de révision.

Ajout de nouveaux articles

Les lexicographes rédigent de nouveaux articles directement au format de la base de données. Ces articles sont envoyés à la base et stockés dans l'espace privé du lexicographe sous forme de document XML. Ils sont ensuite révisés par un spécialiste lexicologue puis intégrés dans la base.

Contributions sur des parties d'articles

Les contributions sur des parties d'articles sont stockées avec leur date de création dans l'espace virtuel du contributeur sous forme de feuille de style XSL. Pour visualiser la contribution, la feuille de style est appliquée sur l'article portant la contribution. Il est aussi possible de visualiser plusieurs contributions. En effet, il suffit d'appliquer les feuilles de style suivant l'ordre chronologique des dates de création.

Dans l'exemple suivant, le contributeur souhaite ajouter un exemple d'usage à cette lexie dont la structure correspond à celle décrite dans la section 2.3.3. Voici la lexie d'origine :

```
<lexie id ="meurtre$1">
```

```
    <headword>meurtre</headword>
    <government-pattern/>
    <lexical-functions/>
    <axies>
       <refaxie href="a001"/>
    </axies>
    <examples/>
  </lexie >
```

L'exemple est le suivant : "Soupçonné du meurtre de son épouse, il a été arrêté par les gendarmes mercredi". Il sera ajouté à l'élément `<exemples>` de la lexie `meurtre$1`. Voici la feuille de style XSLT permettant d'ajouter cet exemple à la lexie `meurtre$1` :

```
  <xsl:stylesheet>   <xsl:output method="xml"/>
    <xsl:template match="*|@*" priority="-1">
  <!-- modèle par défaut :  recopie l'élément et son contenu -->
      <xsl:copy>
        <xsl:apply-templates select="@*|*|text()"/>
      </xsl:copy>
    </xsl:template>
  <!-- modèle de la contribution -->
    <xsl:template match="//lexie[.@id='meurtre$1]/examples">
      <xsl:copy>
        <xsl:apply-templates select ="@*|*|text()"/>
  <!-- ajout d'un nouvel exemple -->
            <example id="e1" >Soupçonné du meurtre de son épouse, il a
été arrêté par les gendarmes mercredi</example>
          < xsl:apply-templates/>
      </xsl:copy>
    </xsl:template>
  </xsl:stylesheet>
```

Le résultat de l'application de cette feuille de style à la lexie `meurtre$1` est la même lexie contenant un exemple de plus.

4. Intégration des outils de manipulation, construction et consultation de dictionnaires

4.1. Manipulation des données

Pour manipuler les données lexicales, nous utilisons plusieurs outils. Pour la récupération de données existantes et la production de nouvelles données, il s'agit des outils définis par Haï Doan Nguyen dans sa thèse. Pour les manipulations internes de données, ce sont des outils de manipulation XML.

4.1.1. Récupération des ressources existantes

La récupération de ressources existantes se fait en suivant la méthodologie RÉCUPDIC. Les informations sont d'abord nettoyées puis marquées à l'aide d'un outil possédant un langage d'expressions régulières (Word, BBEdit, scripts Perl, etc.). Ensuite, la structure que l'on veut obtenir est décrite selon une grammaire H-grammar. L'outil H-grammar récupère ensuite la ressource et la transforme en objets structurés CLOS (Common Lisp Object System). Ces objets reflètent la structure décrite par la grammaire H-grammar.

Les données sont ensuite stockées dans des fichiers texte au format LISPO. Ce format, élaboré par Haï Doan Nguyen, permet de stocker des objets CLOS dans des fichiers texte puis ensuite de lire les fichiers texte pour reconstruire les objets.

Le format interne de notre base lexicale est basé sur XML. Il faut donc transformer le résultat de la récupération du format LISPO vers le format XML. Pour cela, nous avons écrit la fonction LISP suivante :

```
(defun list2xml (ma-liste)
(let ((string ""))
(cond
-- ma liste est une chaine :  renvoyer la chaine
 ((stringp ma-liste) ma-liste)
-- ma liste est un symbole :  construire l'élément XML
 ((and (symbolp ma-liste) (neq list nil)
  (concatenate "<" (princ-to-string ma-liste) "/>"))
-- ma liste est une liste :  la parcourir
 ((neq (cdr ma-liste) nil)
  (let ((element (princ-to-string (pop ma-liste)))
   (attribute "") (value nil))
   (if (not (listp ma-liste)) (setf ma-liste (list ma-liste)))
   (if (symbolp (first ma-liste)) (setf value (pop ma-liste)))
   (if (neq value nil)
    (setf attribute (concatenate " lispo='"
      (princ-to-string value) "'")))
```

```
-- pour chaque objet, construire l'élément XML
   (if (eq value 'LIST)
    (dolist (item ma-liste)
     (setf string (concatenate string "<" element attribute ">"))
     (setf string (concatenate string (list2xml item)))
     (setf string (concatenate string "</" element ">")))
    (progn (setf string (concatenate "<" element attribute ">"))
     (dolist (item ma-liste)
     (setf string (concatenate string (list2xml item))))
     (setf string (concatenate string "</" element ">"))))
   string))
  ((neq (first ma-liste) nil)
   (concatenate string "<" (princ-to-string (first ma-liste)) "/>")))))
```

On voit qu'il s'agit d'une opération assez simple si on la programme à ce niveau. En C++, il faudrait beaucoup plus de code et l'efficacité ne serait pas supérieure.

4.1.2. Manipulations internes des données

La manipulation interne des données se fait grâce aux outils XML équipés de parseurs. Il existe essentiellement l'API SAX (Simple Api for XML) [SAX 2.0] et l'API DOM (Document Object Model) [DOM]. Il est aussi possile d'utiliser le langage XSLT en écrivant une feuille de stype XSL pour chaque transformation.

Par exemple, pour la fusion décrite dans la section 2.2.3 de cette partie, nous utilisons une feuille de style XSL. Cette feuille de style nous permet de fusionner deux articles qui ont le même mot-vedette <headword>. S'ils ont la même catégorie <pos>, les lexies sont regroupées. Elles portent ensuite l'attribut provenance qui indique le nom de leur dictionnaire d'origine.

Voici un extrait de cette feuille de style :

```
<!--modèle pour le premier article de la fusion-->
<xsl:template match="/fusion/*[position()=1]" >
<!-- recopier le début dans le résultat -->
<xsl:copy>
    <xsl:copy-of select="dml:headword"/>
    <xsl:copy-of select="dml:pronunciation"/>
    <!-- stocker le mot-vedette -->
<xsl:variable name="hw1" select="dml:headword/text()"/>
  <!-- recopier le début dans le résultat -->
  <xsl:for-each select="dml:syntactic-cat">
      <xsl:copy>
        <xsl:copy-of select="dml:pos"/>
        <xsl:element name="dml:lexie">
          <xsl:attribute name="provenance">
  <!--on induique la provenance de la lexie -->
         <xsl:value-of select="../@provenance"/>
          </xsl:attribute>
          <xsl:copy-of select="dml:lexie/*"/>
        </xsl:element>
```

```
        <xsl:variable name="pos1" select="dml:pos/text()"/>
  <!-- pour tous les articles suivants -->
       <xsl:for-each select="../../*[position()>1]">
 <!-- s'ils ont le même mot-vedette -->
       <xsl:if test="dml:headword/text()=$hw1">
          <xsl:for-each select="dml:syntactic-cat">
 <!-- s'ils ont la même catégorie grammaticale -->
      <xsl:if test="dml:pos/text()=$pos1">
 <!--on recopie la lexie dans le résultat -->
        <xsl:element name="dml:lexie">
          <xsl:attribute name="provenance">
 <!--en indiquant sa provenance -->
         <xsl:value-of select="../@provenance"/>
          </xsl:attribute>
          <xsl:copy-of select="dml:lexie/*"/>
          </xsl:element>
          </xsl:if>
          </xsl:for-each>
        </xsl:if>
      </xsl:copy>
    </xsl:for-each>
  </xsl:copy>
</xsl:template>
```

4.1.3. Production de nouvelles ressources

Pour produire de nouvelles ressources à partir des ressources stockées dans la base lexicale, il est possible d'utiliser l'outil PRODUCDIC défini, implémenté et expérimenté par Haï Doan Nguyen dans sa thèse. Cet outil permet d'effectuer des opérations ensemblistes sur les ressources : sélection, fusion, intersection, extraction, combinaisons, etc. Le format d'entrée de cet outil est aussi le format LISPO. Nous avons donc mis au point une feuille de style XSL qui permet de transformer nos ressources du format XML vers le format LISPO. Cela permet ensuite d'utiliser PRODUCDIC.

Voici un extrait de la feuille de style transformant des documents au format XML vers le format LISPO :

```
<!-- modèle pour les commentaires -->
  <xsl:template match="comment()"> ;;
    <xsl:value-of select="."/>
    <xsl:text>
</xsl:text>
  </xsl:template>
  <xsl:template match="/*">
    <xsl:apply-templates/>
  </xsl:template>
<!-- modèle pour les éléments -->
  <xsl:template match="*">
    <xsl:apply-templates select="comment()"/>(
<!-- recopie du nom de l'élément -->
```

```
      <xsl:value-of select="name()"/>
      <xsl:if test="text()" > .  "
        <xsl:value-of select="text()"/>"
      </xsl:if>
      <xsl:if test="*">
        <xsl:text>
</xsl:text>
      </xsl:if>
      <xsl:apply-templates select="@*"/>
      <xsl:if test="*">
        <xsl:apply-templates select="*"/>
      </xsl:if>)
    </xsl:template>
<!-- modèle pour les attributs lispo -->
    <xsl:template match="@lispo">
      <xsl:value-of select="."/>
      <xsl:text>
</xsl:text>
    </xsl:template>
```

4.2. Interaction avec les serveurs partenaires

La base lexicale peut interagir avec de nombreux fournisseurs différents. Elle peut établir des partenariats avec d'autres bases ou d'autres applications en échangeant des données. Ce partenariat est basé sur un principe de dualité.

La base peut utiliser des ressources lexicales distantes. Ces ressources sont disponibles via des fournisseurs de ressources. Il faut donc standardiser les échanges entre la base et les fournisseurs de ressources.

D'autre part, la base lexicale fait aussi appel à des services externes par exemple pour lemmatiser un mot avant une recherche dans la base, ou pour conjuguer un verbe, etc. La base se connecte à des fournisseurs de services. Il faut donc aussi standardiser les échanges entre la base et les fournisseurs de services.

4.2.1. Principe de réciprocité

La base lexicale est amenée à échanger des données avec d'autres programmes partenaires. Nous avons "découvert" la possibilité d'une dualité avec ces programmes partenaires qui peuvent être à la fois clients et fournisseurs d'information. Par exemple, un partenaire lemmatiseur du français est client de la base pour tous les mots inconnus et les nouveaux mots qu'il ne sait pas traiter. Lorsqu'il essaye de lemmatiser un mot inconnu, il peut se connecter à la base pour trouver des informations sur ce mot. Il peut aussi interroger périodiquement la base en se connectant automatiquement pour collecter les nouveaux mots français qui ont été intégrés. Il actualisera ses propres dictionnaires automatiquement à partir de ces informations. Nous nous proposons, vu son intérêt de rendre cette dualité possible dans tous les cas, même dans ceux qui ne sont pas utiles a priori.

De son côté, la base lexicale peut faire appel aux services du lemmatiseur lors du prétraitement d'une requête de consultation de la part d'un utilisateur. Les mots-vedettes des articles sont en général les lemmes des mots d'une langue. Par exemple, *mangeons* est le verbe *manger* à la première personne du pluriel. Le mot-vedette correspondant sera donc l'infinitif du verbe *manger*. Les utilisateurs voulant faire du déchiffrage de texte et ne maîtrisant pas forcément la langue ont besoin d'un lemmatiseur pour trouver les lemmes

correspondant aux mots qu'ils recherchent dans la base. La base lexicale se connectera alors au lemmatiseur avant de rechercher les articles correspondant à la requête de l'utilisateur.

Le principe de points peut s'appliquer aussi aux partenaires. En effet, chaque service rendu par un partenaire fait augmenter son nombre de points. À l'inverse, chaque information demandée à la base fait baisser le nombre de points. Cependant, des difficultés surgissent : il peut être difficile de quantifier un service en nombre de points ou d'arrêter le service si un programme partenaire est débiteur, etc.

Pour communiquer avec tous les fournisseurs, nous devons standardiser les échanges. Nous proposons de les standardiser grâce à des API. La plupart du temps, il sera cependant nécessaire d'écrire un adaptateur (wrapper) pour interfacer un fournisseur avec notre environnement afin qu'il respecte l'API définie.

Les API fournisseur contiennent de la méta-information sur le fournisseur, des informations de connexion et enfin la structure des éléments utilisés en entrée et en sortie. Ces API sont décrites par l'élément <api>.

Pour la méta-information, l'attribut name indique le nom du fournisseur; l'élément <information> contient un texte explicatif sur le service proposé par le fournisseur.

Pour la connexion, l'élément <url> contient l'attribut xlink:href indiquant l'URI (Uniform Resource Identifier) du fournisseur. Cette URI doit être conforme à rfc (request for comment) 2396 de l'IETF Internet Engineering Task Force) [RFC2396]. Les URIs sont du type http, ftp, mailto, telnet, etc.

L'élément <protocol> décrit le protocole utilisé pour la communication. Les protocoles sont indiqués avec l'attribut type. Ils peuvent être du type post ou get pour les CGIs, telnet-DICT, etc. Les attributs login et password indiquent les login et mot de passe qu'il faut utiliser pour se connecter au fournisseur.

L'élément <delay> renseigne sur le délai de connexion au fournisseur. L'attribut min indique le délai minimum, max indique le délai maximum et timeout indique le temps au dela duquel la connexion est interrompue automatiquement.

L'élément <encoding> indique les encodages utilisés. L'attribut input indique l'encodage en entrée, l'attribut output indique l'encodage en sortie. Les valeurs des attributs sont du type DML encodingType.

L'élément <format> indique les formats utilisés. L'attribut input indique le format en entrée, l'attribut output indique le format en sortie. Cela peut être par exemple : texte, HTML, XML, rtf, etc.

Pour la recherche, l'élément <arguments> décrit les arguments de l'API à fournir par la base lexicale en suivant la syntaxe des schémas XML. Les arguments doivent être envoyés par la base lexicale lors de chaque connexion au fournisseur. L'élément <result> décrit le résultat de l'API en suivant la syntaxe des schémas XML. Le résultat est renvoyé par le fournisseur en réponse aux requêtes de la base lexicale.

4.2.2. Fournisseur de services

Nous distinguons deux types de services. Les services de prétraitement sont utilisés en amont de la consultation de la base pour faciliter la recherche dans la base. Ces services incluent des lemmatiseurs pour trouver le ou les lemmes correspondant au mot que l'utilisateur recherche, des correcteurs orthographiques, ou des recherches plus spécifiques avec les consonnes du mot classées par ordre alphabétique [Zock01], etc.

Les services d'actions associées permettent d'associer des actions à des parties d'informations se trouvant dans le résultat des requêtes faites sur la base. Par exemple, il est possible d'associer un conjugueur aux verbes, et d'inclure un phonétiseur permettant d'entendre la prononciation de n'importe quelle chaîne (mot-vedette, exemples, etc.).

Voici en exemple une API de prétraitement utilisant les lemmatiseurs développés par XRCE [Démos]. Ces lemmatiseurs sont utilisés par les maquettes décrites en partie B (DicoWeb, DicoSzótár, DicoFeJ, Nihongo, FeM) :

```
<api type="supplier" category="preprocessing" name="XRCE-fra-morphan">
  <info>Lemmatiseur du français de Xerox</info>
  <url href="http://www.xrce.xerox.com/research/mltt/demos/french.cgi"/>
```

```
    <protocol type="get" login="toto" password="foo"/>
    <delay min="1s" average="1s" max="2s" timeout="10s"/>
    <encoding input="ISO-8859-1" output="ISO-8859-1"/>
    <format input="txt" output="txt"/>
    <arguments>
      <element name="input" type="string"/>
    </arguments>
    <result>
      <element name="output">
        <complexType>
          <sequence maxOccurs="unbound">
            <element name="lemma" type="string"/>
          </sequence>
        </complexType>
      </element>
    </result>
  </api>
```

Voici un exemple d'argument :

```
<input>cochons</input>
```

Voici le résultat obtenu :

```
<output>
  <item>
    <lemma>cocher</lemma>
    <analysis>+Imp+PL+P1+Verb</analysis>
  </item>
  <item>
    <lemma>cocher</lemma>
    <analysis>+IndP+PL+P1+Verb</analysis>
  </item>
  <item>
    <lemma>cochon</lemma>
    <analysis>+Masc+PL+Adj</analysis>
  </item>
  <item>
    <lemma>cochon</lemma>
    <analysis>+Masc+PL+Noun</analysis>
  </item>
</output>
```

4.2.3. Fournisseur de ressources

Les ressources distantes peuvent être consultables par la base mais aussi modifiables. Nous avons donc prévu un type d'API pour chaque tâche.

Interface de consultation

Voici une API de consultation du dictionnaire JMDict japonais-anglais de Jim Breen [EDict] utilisé dans notre maquette DicoFeJ décrite en partie B.

```
<api type="supplier" category="consultation" name="JMDict_en-ja">
  <info>Dictionnaire japonais-anglais de Jim Breen</info>
  <url href="http://www.csse.monash.edu.au/cgi-bin/cgiwrap/jwb/wwwjdic"/>
  <protocol type="get"/>
  <delay min="1s" average="1s" max="2s" timeout="10s"/>
  <encoding input="UTF-8" output="EUC-JP"/>
  <format input="txt" output="html"/>
  <arguments>
    <element name="source-language">
      <complexType>
        <restriction base="string">
          <enumeration value="jpn"/>
          <enumeration value="eng"/>
        </restriction>
      </complexType>
    </element>
    <element name="headword" type="string"/>
    <element name="regex" type="boolean"/>
  </arguments>
  <result>
    <element name="output" type="string"/>
  </result>
</api>
```

Voici un exemple d'argument :

```
<source-language>eng</source-language> <headword>house</headword>
<regex>no</regex>
```

Voici le résultat de la requête :

```
<output>登院【とういん】(n) attendance at the House (Diet)<br/>
家相 【かそう】(n) construction of a house (divination term)<br/>
本家【ほんけ】(n) head house (family); birthplace;
originator<br/>
 メゾン (fr:) house (fr: maison); (P)<br/>
下院議 【かいんぎ】lower house (of Parliament, etc.)<br/>
家 【うち】(n) house (one's own); (P)<br/>
青瓦台 【せいがだい】the Blue House (South Korea's presidential
palace)<br/>
下院議長 【かいんぎちょう】Speaker of the House (US)<br/>
家屋敷 【いえやしき】(n) house and lot</output>
```

Interface de modification

Les contributions reçues par la base lexicale ne sont pas tout de suite intégrées dans les ressources. Elles sont d'abord stockées dans l'espace virtuel du contributeur puis éventuellement annotées et remodifiées par d'autres contributeurs puis finalement révisées par un groupe de spécialistes lexicologues. Lorsqu'elles sont acceptées, elles sont intégrées dans les ressources. Comme ces ressources peuvent être distantes, Il faut une API pour pouvoir se connecter aux serveurs qui les gèrent et leur envoyer les modifications acceptées.

Voici une API de modification du dictionnaire hongrois-français DicoSzótár décrit en partie B.

```
<api type="supplier" category="modification" name="DicoSzótár">
  <info>Dictionnaire hongrois-français</info>
  <url href="http://www-clips.imag.fr/geta/services/dicoszotar"/>
  <protocol type="post" login="getabase" password="toto"/>
  <encoding input="ISO-8859-1"/>
  <format input="xml"/>
  <arguments>
    <element name="article" type="string"/>
  </arguments>
  <result>
    <element name="output" type="string"/>
  </result>
</api>
```

Voici un exemple d'argument :

```
<article>
  <administration>     <indexer date="2001-05-31T16:34:29Z">Mathieu
Mangeot</indexer>
    <lesson n="7" date="31/05/01"/>
    <status>révisé par AS</status>
  </administration>   <hun>szem</hun>
  <pos>n.</pos>
  <fra>œil</fra>
  <pos>n.m.</pos>
</article>
```

Et voici le résultat :

```
<output>Article szem inséré; Article œil inséré</output>
```

4.3. Consultation de la base

Divers clients accèdent à la base pour y effectuer des tâches variées. La base dispose d'interfaces de consultation et de modification en ligne. Il est aussi possible au programmeur qui le souhaite de construire une application cliente de la base. Il devra dans ce cas respecter les interfaces API définies plus bas et correspondant aux services demandés à la base.

4.3.1. Sélection des ressources

Pour la sélection des ressources dans la base, l'utilisateur a besoin de consulter la méta-information disponible sur ces ressources. Il le fait via une API de méta-information. Il peut ensuite, grâce aux informations fournies par cette API, choisir les ressources qu'il désire consulter.

Les éléments présentés en arguments sont consultables avec des expressions régulières et les opérateurs de comparaison suivants (=, >, >=, <, <=, !=).

Il est possible de faire des recherches sur le nom <name>, le type (monodirectionnel, bidirectionnel, pivot) <type>, le domaine <domain>, la catégorie (monolingue, bilingue, multilingue) <category>, le contenu <contents>, les langues sources et cibles <source-language> et <target-language>, les dates de création et d'installation dans la base <creation-date> et <installation-date>, l'encodage <encoding>, le format <format>, le propriétaire des ressources <owner>, la version <version>, le nombre de mots-vedettes <hw-number>, le nombre d'octets du fichier source <bytes>, et les aspects légaux <legal>.

Voici l'API de méta-information de la base lexicale du GETA.

```
<api type="client" category="meta-info" name="GETA public database">
  <info>API utilisée pour consulter l'information disponible
sur les ressources de la base</info>
  <url href ="http://www-clips.imag.fr/cgi-bin/geta/dictlist
ftp://www-clips.imag.fr/geta/services/dictlist
mailto:dictlist@imag.fr
telnet://www-clips.imag.fr:2628"/>
  <protocol type="post get ftp mailto DICT" login="anonymous"/>
  <delay min="1s" average="1s" max="2s" timeout="10s"/>
  <encoding input="ASCII ISO-8859-1 UTF-8" output="UTF-8"/>
  <format input="txt xml" output="xml html txt"/>
  <arguments>
    <element name="type" type="string"/>
    <element name="domain" type="string"/>
    <element name="category" type="string"/>
    <element name="contents" type="string"/>
    <element name="source-language"type="string"/>
    <element name="target-language" type="string"/>
    <element name="creation-date" type="string"/>
    <element name="installation-date" type="string"/>
    <element name="encoding" type="string"/>
    <element name="format" type="string"/>
    <element name="owner" type="string"/>
    <element name="version" type="string"/>
    <element name="hw-number" type="string"/>
    <element name="bytes" type="string"/>
    <element name="source" type="string"/>
    <element name="legal" type="string"/>
  </arguments>
  <result>
    <element name="output">
      <complexType>
        <sequence>
          <element name="info" type="string"/>
        </sequence>
      </complexType>
    </element>
```

```
    </result>
  </api>
```

Voici un exemple de requête sur la méta-information : recherche des ressources bilingues et multilingues avec l'anglais comme langue source, créées après 1990 et avec au moins 5 000 mots-vedettes. Le texte XML des exemples est rendu lisible pour le lecteur. Les entités XML représentant les caractères "<", ">" et "&" sont donc converties.

```
<category>(bilingual)|(multilingual)</category>
<source-language>eng</source-language> <creation-date>>19900101T00:00:00Z</
<hw-number>>5000</hw-number>
```

Voici un extrait du résultat sur la base lexicale du laboratoire XRCE. Pour simplifier, nous ne présentons que les noms des ressources répondant à la requête et non les informations complètes.

```
<name>EuroWordNet</name>
<name>German Dictionary</name>
<name>Hungarian_en-hu</name>
<name>JMDict_en-ja</name>
<name>Multilingual medical dictionary</name>
<name>Oxford-Hachette French dictionary</name>
<name>Oxford Spanish Dictionary</name>
<name>Urdu English Dictionary</name>
<name>DHydro Dictionary</name>
```

4.3.2. Élaboration des requêtes

Une API de consultation de la base lexicale est disponible. Les clients de la base l'utilisent pour rédiger leurs requêtes de consultation de ressources. Ils peuvent consulter plusieurs ressources à la fois, utiliser des expressions régulières, etc. Ils configurent ensuite le résultat des requêtes grâce aux API de microstructure et de présentation.

L'utilisateur indique le nom des ressources à consulter avec l'élément <name>, l'ordre de tri du résultat avec l'élément <word-order>, les éléments CDM qu'il veut consulter avec<cdm-elements>, le nombre d'articles suivant et précédant les articles résultats avec l'élément <context>, le nom du module utilisé pour le prétraitement de l'input avec l'élément <preprocessing>, s'il utilise une expression régulière ou non avec l'élément <regex>, et la chaîne de caractères qu'il recherche ou une expression régulière avec l'élément <input>.

Le résultat est une liste d'articles correspondant aux critères de la recherche. Il est stocké dans l'élément <output>.

Voici l'API de consultation de la base lexicale du GETA :

```
<api type="client" category="consultation" name="getabase">
  <info>API de consultation de la base lexicale du GETA</info>
  <url href="http://www-clips.imag.fr/cgi-bin/geta/dicoweb
ftp://www-clips.imag.fr/geta/services/dicoweb
mailto:dicoweb@imag.fr
telnet://www-clips.imag.fr:2628"/>
  <protocol type="post get ftp mailto DICT" login="anonymous"/>
  <delay min="1s" average="1s" max="2s" timeout="10s"/>
  <encoding input="ASCII ISO-8859-1 UTF-8" output="UTF-8"/>
  <format input="txt xml" output="xml html txt"/>
  <arguments>
```

```
      <element name="name" type="string"/>
      <element name="source-language" type="lang"/>
      <element name="word-order" type="string"/>
      <element name="cdm-elements" type="string"/>
      <element name="context" type="positiveInteger"/>
      <element name="preprocessing" type="string"/>
      <element name="regex" type="boolean"/>
      <element name="input" type="string"/>
    </arguments>
    <result>
      <element name="output">
        <complexType>
          <sequence>
            <element name="article" type="articleType"/>
          </sequence>
        </complexType>
      </element>
    </result>
  </api>
```

Voici un exemple de consultation du dictionnaire FeM avec recherche du mot "essais", un contexte d'un article prédécent et suivant et un prétraitement de lemmatisation :

```
<name>FeM</name>
<source-language>fra</source-language> <context>1</context>
<preprocessing>Lemmatisation</preprocessing>
<regex>no</regex>
<input>essais</input>
```

Voici un extrait du résultat de la requête. Nous ne montrons que les mots-vedettes des articles.

```
<output>
  <article><headword>essuyer</headword>...</article>
  <article><headword>essai</headword>...</article>
  <article><headword>esssayer</headword >...</article>
</output>
```

4.3.3. Visualisation du résultat

Pour l'instant, le résultat de la requête est en format XML. Le client utilisant un navigateur classique ne peut donc pas encore visualiser directement son résultat. La visualisation du résultat se fait en deux phases.

D'abord le client redéfinit la macrostructure des articles à l'aide d'une interface spécialisée. La requête est alors traduite en une feuille de style XSLT stockée dans son espace virtuel. Il peut décider de fusionner les articles selon leurs éléments DML, par exemple les articles ayant le même mot-vedette et la même catégorie grammaticale.

Ensuite, le résultat XML est transformé pour l'essentiel en XHTML [XHTML 1.0] et la présentation du résultat est décrite dans une feuille de style CSS. Il est possible d'associer des visualisateurs sous forme d'applets java pour certains éléments XML particuliers. Pour ne pas pénaliser les clients ayant des équipements de bas de gamme, la transformation se fait sur le serveur.

Les éléments XML ne nécessitant pas de visualisateur particulier sont transformés en éléments XHTML `<span>`. L'attribut `class` de cet élément prend alors la valeur du nom de l'élément XML. Cet attribut

permet ensuite d'associer un style particulier à cet élément. Les styles sont décrits à part dans une feuille de style CSS.

Voici un exemple de conversion de quelques éléments DML :

XML **avant conversion**	**XHTML** **après conversion**
<headword>meurtre</headword>	meurtre
<pronunciation>meu+rtr(e)</pronunciation>	meu+rtr(e)
<pos>n.m.</pos>	n.m.
<example>La mésentente pourrait être le mobile du meurtre.</example>	La mésentente pourrait être le mobile du meurtre.

TAB. C.4 – *conversion de XML vers XHTML*

Certains éléments XML nécessitent un visualisateur particulier ou sont associés à une action particulière. Une applet java implémentant le visualisateur ou l'action est alors associée à l'élément lors de la transformation XSLT. Il est possible d'associer par exemple un visualisateur d'arbres hyperboliques décrit dans la partie B à des traductions multiples d'un mot.

Dans l'exemple suivant, un synthétiseur est associé à la prononciation du mot-vedette. À l'élément XML <prononciation>sera alors associé l'élément XHTML <applet>. Le contenu de l'élément sera passé en paramètre de l'applet :

XML **avant conversion**	**XHTML** **après conversion**
<pronunciation>meu+rtr(e)</pronunciation>	<applet code="phonetiseur.class" archive="phonetiseur.zip"> <param name="element" value="pronunciation"> <param name="data" value="meu+rtr(e)"> </applet>

TAB. C.5 – *conversion de XML vers une Applet HTML*

La transformation du résultat XML vers un document XHTML se fait par l'intermédiaire d'une feuille de style XSLT. Voici un extrait de la feuille de style permettant de transformer les exemples ci-dessus :

```
<!-- modèle par défaut pour tous les éléments -->
<xsl:template match="*" priority="-1">
<!-- fabrique un élément span -->
  <span>
<!-- avec un attribut class -->
    <xsl:attribute name="class">
<!-- prenant comme valeur le nom de l'élément -->
      <xsl:value-of select="name()"/>
    </xsl:attribute>
    <xsl:apply-templates/>
  </span>
</xsl:template>
<!-- modèle pour les éléments associés à une applet java -->
<xsl:template match="pronunciation">
```

```
  <!-- crée un élément HTML applet -->
    <applet code="phonetiseur.class" archive="phonetiseur.zip"> <!--
avec un paramètre contenant le nom de l'élément -->
      <param name="element">
        <xsl:attribute name="value">
          <xsl:value-of select="name()"/>
        </xsl:attribute>
      </param>
  <!-- et un paramètre contenant le texte à phonétiser -->
      <param name="data">
          <xsl:attribute name="value">
          <xsl:value-of select="text()"/>
        </xsl:attribute>
     </param>
    </applet>
  </xsl:template>
```

4.3.4. Personnalisation du résultat

L'utilisateur définit ses préférences de présentation du résultat avec une interface spécialisée. Les informations sont ensuite envoyées au serveur puis traduites en feuille de style CSS [CSS 2.0] ou XSL-FO [XSL] qui sera ensuite stockée dans l'espace virtuel de l'utilisateur puis utilisée pour afficher les résultats de ses requêtes.

Pour chaque élément d'information, l'utilisateur peut préciser son style (couleur, police, taille, emplacement). Dans l'exemple, le mot-vedette `headword` aura une taille double de la taille normale (`font-size: 2em`), les traductions anglaises `eng` seront en orange (`color: orange`), les traductions françaises `fra` en bleu (`color: blue`), les traductions japonaises `jpn` en rouge (`color: red`) et la méta-information `meta` en gris (`color: gray`). L'utilisateur ne veut pas visualiser les traductions malaises `msa`, et indique donc `display:none`.

Voici la feuille de style CSS exprimant ces préférences de présentation :

```
.headword { font-size:  2em; }
.eng { font-family:  "Arial", sans-serif; color:  orange; }
.fra { font-family:  "Helvetica", sans-serif; color:  blue; }
.jpn { font-family:  "Osaka", sans-serif; color:  red; }
.msa { display:  none; }
.meta { font-family:  "Times", serif; color:  gray; }
```

4.4. Rédaction des articles et contributions

Lorsque les lexicographes rédigent de nouveaux articles et que les autres utilisateurs de la base contribuent, ils peuvent le faire grâce à plusieurs méthodes exposées en partie B.

4.4.1. Rédaction en ligne via le Web

La rédaction en ligne via le Web est possible grâce à des formulaires HTML. Cette technique n'est valable que si la structure du dictionnaire à construire est simple. Les formulaires HTML sont adaptés à la

structure du dictionnaire. Pour les faire évoluer en même temps que la structure, il faut à chaque fois les reprogrammer.

Cette technique est cependant envisageable pour des contributions localisées, par exemple pour ajouter un élément à chaque article (prononciation, exemple d'usage, idiotisme, etc.).

4.4.2. Rédaction avec des éditeurs structurés

La rédaction des articles avec des éditeurs structurés est très avantageuse. L'éditeur s'adapte facilement à une structure évolutive d'un dictionnaire même complexe.

Le logiciel Amaya [Amaya] est un éditeur/navigateur HTML. Il permet donc à la fois de visualiser des documents HTML et de les éditer. Il semble très prometteur à plusieurs points de vue :

– il est multiplate-forme (UNIX, LINUX, Windows);

– il dispose déjà d'un mécanisme d'annotation;

– le code source est disponible. Nous pouvons donc modifier le logiciel pour l'adapter à nos besoins. Par exemple, nous pouvons restreindre les fonctionnalités de l'éditeur et en ajouter d'autres.

L'éditeur Amaya travaille avec des documents au format XHTML [XHTML 1.0]. Nous pouvons aisément établir une bijection entre le document XML de la base et le document XHTML que le contributeur ou le lexicographe édite avec Amaya. Il suffit pour cela de transformer le document XML dans le format XHTML et de lui associer une feuille de style à la sortie de la base, et inversement une fois les modifications effectuées.

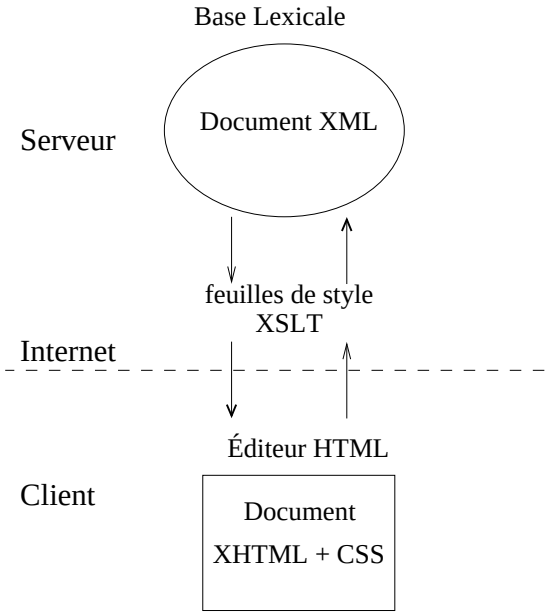


FIG. C.21 – *transformation et édition d'un document XHTML*

La transformation d'un document XML vers un document XHTML se fait avec la feuille de style XSLT décrite dans la section précédente concernant la visualisation du résultat. La transformation inverse se fait de la même manière. Le tableau C.6 montre des exemples de rétroconversion d'éléments XHTML vers les éléments XML de départ.

XHTML **avant conversion**	**XML** **après conversion**
meurtre	<headword>meurtre</headword>
meu+rtr(e)	<pronunciation>meu+rtr(e)</pronunciation>
n.m.	<pos>n.m.</pos>
La mésentente pourrait être le mobile du meurtre.	<example>La mésentente pourrait être le mobile du meurtre.</example>

TAB. C.6 – *conversion de XHTML vers XML*

Voici un extrait de la feuille de style permettant cette transformation :

```
<!-- modèle par défaut pour les éléments span -->
<xsl:template match="span" priority="-1"> <!-- crée un élément avec
la valeur de l'attribut class -->
  <xsl:element name="{@class}">
    <xsl:apply-templates select="text()"/>
    <xsl:apply-templates select="*"/>
  </xsl:element>
</xsl:template>
<!-- modèle pour les éléments utilisant une applet java -->
<xsl:template match="applet">
<!-- crée un élément avec l'attribut du paramètre element -->
  <xsl:element name="{param[./@name='element']/@value}">
<!--recopie l'attribut du paramètre data -->
    <xsl:value-of select="param[./@name='data']/@value"/>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

4.4.3. Rédaction avec des pseudo-éditeurs structurés

Suite à nos expériences avec le logiciel Word™ exposées en partie B, nous pouvons proposer aux lexicographes d'utiliser ce logiciel. Il sera considéré comme un pseudo-éditeur structuré, car la structure du document n'est pas directement visible par l'utilisateur. Celui-ci doit donc, lorsqu'il édite, faire attention à ne pas ajouter des informations parasites dans le document qui ne respecteraient pas la structure initiale.

La bijection entre la base lexicale et le client se fait alors entre le document XML de la base et le document RTF édité avec Word™. La technique est aussi valable avec tout autre éditeur utilisant un format structuré qui puisse être généré facilement en mode texte par exemple. Cette transformation peut par exemple se faire avec le programme LISP utilisé dans [Mangeot97].

Le tableau C.7 montre un exemple de conversion de quelques éléments XML vers leur équivalent RTF.

Voici l'en-tête du fichier RTF qui sera utilisé pour les éléments convertis :

```
{\rtf1\iso \deff8\deflang1033{\fonttbl
{\f0\froman\fcharset77\fprq2 Tms Rmn;}
/** définition des polices **/
{\f1\fnil\fcharset2\fprq2 Symbol;}
```

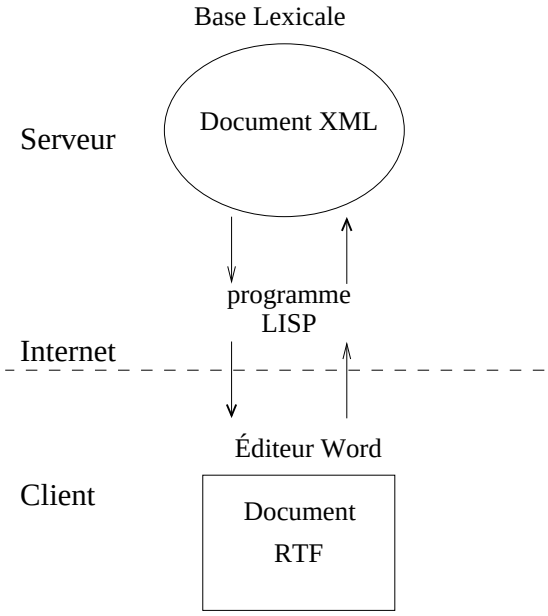


FIG. C.22 – *transformation et édition d'un document rtf*

XML **avant conversion**	**RTF** **après conversion**
<headword>meurtre</headword>	\par \pard\plain \s15 \f8 meurtre
<pronunciation>meu+rtr(e)</pronunciation>	\par \pard\plain \s16 \f3 meu+rtr(e)
<pos>n.m.</pos>	\par \pard\plain \s17 \f9 n.m.
<example>La mésentente pourrait être le mobile du meurtre.</example>	\par \pard\plain \s18 \f8 La mésentente pourrait être le mobile du meurtre.

TAB. C.7 – *conversion de XML vers RTF*

```
..f2..f3.......f52..f53..
{\f54\fnil\fcharset77\fprq2 \'96\'7b\'96\'be\'92\'a9\'81|\'82l;}}
{\stylesheet{\widctlpar \f8\lang1036 \snext0 Normal;}
{\*\cs10 \additive Default Paragraph Font;}
/** définition des styles **/
{\s15\widctlpar \b\f8\fs28\ul\lang1036 headword;}
{\s16\widctlpar \caps\f3\lang1036 \sbasedon15 pronunciation;}
{\s17\widctlpar \i\f9\lang1036 \sbasedon15 pos;}
{\s18\widctlpar \b\f8\cf4\lang1036 \sbasedon15 example;}
..s19..s20...}}
```

4.4.4. Rédaction avec des éditeurs spécialisés

L'expérience menée par Gilles Sérasset [Sérasset96,97a,97b] sur DECID, un éditeur pour le DEC d'Igor Mel'tchuk montre qu'un éditeur spécialisé pour un dictionnaire est très utile. Il simplifie grandement l'édition d'un tel dictionnaire.

Cependant, l'élaboration d'un éditeur spécialisé demande des efforts non négligeables. Sachant qu'ensuite ces éditeurs spécialisés sont difficilement adaptables à de nouvelles structures, il ne faut envisager cette élaboration que lorsque la structure du dictionnnaire en construction est déjà relativement stable. De plus, il est nécessaire de programmer cet éditeur de façon portable et il n'existe pas encore de solution totalement satisfaisante même avec java (incompatibilités entre les différentes versions de java).

D : Application à Papillon, projet de base lexicale multilingue sur Internet

Introduction

Nous disposons maintenant d'outils pour définir et utiliser un environnement de manipulation, création et consultation de dictionnaires hétérogènes. Nous allons dans cette partie appliquer nos méthodes et outils à un cas concret : le projet Papillon. Ce projet vise à construire une base lexicale multilingue avec une architecture en étoile autour d'un dictionnaire pivot d'acceptions interlingues, à stocker les données dans des base de données, puis à en extraire des dictionnaires personnalisés.

Les intérêts de cette partie sont multiples. Nous voulons démontrer l'efficacité de nos outils dans une réalisation concrète pour le projet Papillon, ajouter l'aspect collaboratif dans la construction de dictionnaires et aussi faire progresser différents aspects linguistiques. Ces aspects sont principalement l'utilisation de la lexicographie explicative et combinatoire à large échelle et appliquée à plusieurs langues, l'établissement de liens interlingues non basés sur des concepts et relier ces liens à d'autres théories extérieures.

Nous présenterons d'abord l'historique, les buts et l'architecture générale du projet Papillon. Nous définirons ensuite les macrostructures et microstructures des dictionnaires du projet à l'aide de nos outils. Enfin, nous terminerons en détaillant la méthodologie de construction de ces dictionnaires en différentes étapes.

1. Présentation du projet Papillon

1.1. Historique et buts du projet

Le projet Papillon [Papillon] a été lancé en janvier 2000 par une coopération entre le GETA-CLIPS et le National Institute of Informatics (NII) japonais avec le support actif de l'Ambassade de France à Tokyo. Depuis, des partenaires thaïlandais (Kasetsart University & NECTEC à Bangkok) se sont joints au projet, et un doctorant du GETA-CLIPS, spécialiste de l'informatisation du lao, travaille à l'intégration de cette langue. Le projet devrait s'étendre à court terme au vietnamien et au malais.

La première motivation de ce projet est le manque de ressources lexicales entre le français et le japonais gratuites et disponibles au format électronique.

La seconde est que les dictionnaires existants indiquent rarement à la fois l'écriture japonaise en kanji et son écriture en alphabet romain romaji. Les articles japonais omettent aussi souvent d'indiquer les spécificateurs numériques. Ce manque est également criant pour bien d'autres langues importantes, mais le japonais est particulièrement intéressant.

En troisième lieu, le manque de ressources bilingues est aussi un obstacle au développement d'applications linguistiques pour lesquelles existe un fort besoin en dictionnaires adaptés. Par exemple, NTT (Nippon Telegraph and Telephone) au Japon ou Lexiquest en France doivent développer leurs propres dictionnaires séparément. Dans le monde académique, les applications créées pour le français et le japonais offrent une couverture réduite alors que de très bonnes ressources existent entre le japonais et l'anglais.

Il est maintenant envisageable de construire ces ressources par Internet grâce à des linguistes, lexicologues, lexicographes, traducteurs, informaticiens, etc. travaillant en coopération.

Un projet similaire concernant l'anglais et le japonais est actif depuis plus de dix ans, et a permis la construction effective d'un dictionnaire gratuit japonais-anglais disponible sur Internet. C'est le projet EDict, dirigé par le professeur Jim Breen de Monash University en Australie [EDict]. Le dictionnaire actuel JMDict comprend actuellement plus de 70 000 articles de vocabulaire commun, un dictionnaire spécifique de kanji et une vingtaine de dictionnaires spécialisés (biologie, droit, etc.).

Enfin, le projet SAIKAM [Ampornaramveth00] en coopération entre le NII (Tokyo, Japon) et NECTEC (Bangkok, Thaïlande) est actif de puis environ cinq ans. Les étudiants thaïlandais travaillant ou ayant travaillé au Japon ont construit un dictionnaire japonais-thaï d'environ 4 000 articles sur Internet en validant des articles construits automatiquement.

En août 2000, le premier séminaire Papillon a eu lieu au NII à Tokyo. Il a porté sur des discussions autour de la structure et du contenu de la base lexicale et des décisions sur les aspects techniques liés au développement de la base.

Le séminaire Papillon 2001 a eu lieu en juillet à Grenoble. Les participants ont décidé d'adapter l'organisation du W3C au projet Papillon en élisant un comité directeur de 8 à 12 membres et en définissant des tâches avec pour chacune un groupe de coordination, un groupe de travail et un comité consultatif.

Un aspect novateur et essentiel du projet est que la construction du contenu se fera sur la base d'informations libres de droits, produites par des chercheurs (FeM, JMDict, Saikam) ou par des internautes bénévoles

coopérant à travers Internet. Sur le serveur Papillon, chaque contributeur pourra voir la base active, et aura son espace de contribution privé. Seul un petit groupe d'experts aura les droits nécessaires pour intégrer les contributions dans la base active, après validation et correction.Les données produites seront disponibles publiquement selon les termes d'une licence de logiciel libre (Open Source). Cela signifie que les données ne peuvent être réutilisées qu'à des fins non commerciales.

Les enjeux scientifiques de cette recherche sont d'ordre conceptuel, technique et ergonomique. Il s agit d'articuler :

- des macrostructures (structures linguistiques) et microstructures (structures lexicales) pour représenter les données ;
- des outils permettant la manipulation des ressources lexicales ;
- un environnement de construction de dictionnaires en coopération et de navigation/consultation dans une base lexicale.

1.2. Architecture générale du projet

La base lexicale devra résider sur un serveur relié à Internet. Le développement des ressources se fera à distance par les contributeurs. Le scénario est le suivant : ceux-ci envoient leurs contributions. Elles sont stockées dans leur espace virtuel avant d'être révisées par les lexicologues. Une fois révisées, les contributions sont intégrées à la base lexicale. La base est ensuite consultée via Internet par les utilisateurs, qui peuvent configurer le résultat de leurs requêtes.

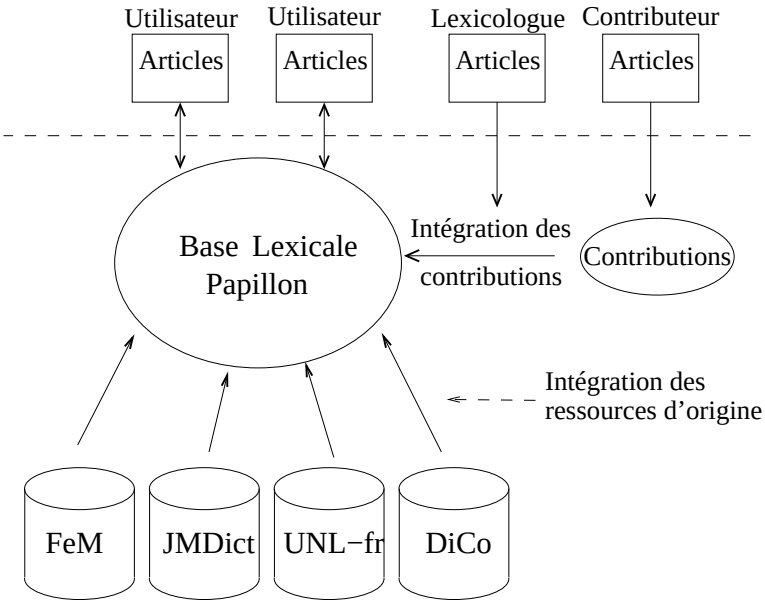


FIG. D.1 – *vue globale de la base lexicale Papillon*

La méthode retenue est de construire une base lexicale multilingue à "pivot" et d'en extraire des dictionnaires bilingues ou multicibles à la volée ou hors ligne, avec ou sans filtrage, et dans des formats variés, pour usages humains et machinaux. Pour chaque langue, on construit un dictionnaire strictement monolingue au format DiCo de Polguère et Mel'tchuk [Polguère00], où l'unité essentielle est la "lexie" (sens de mot), avec

une description très riche et systématique (collocations, fonctions lexicales, exemples, etc.) permettant des usages fins et variés.

On relie les lexies dans un "pivot" central de "liens interlingues" dits aussi "acceptions interlingues" ou "axies", en utilisant pour cela les équivalences traductionnelles attestées par les sources disponibles. Ces axies ont donc un statut plus linguistique que conceptuel, et sont motivées de façon pragmatique et testable, ce qui évite les problèmes de construction d'ontologie.

1.3. Points forts du projet

Le projet présente au moins neuf "points forts".

1) L'architecture de la base lexicale est symétrique, "à pivot" : N dictionnaires monolingues accédés par vocables et par lexies (sens de mots) et un ensemble d'axies (liens interlingues). On aimerait que les dictionnaires soient consultables de façon plus riche (ex: trouver les mots masculin en *-ion*). Les langues de départ sont le français, le japonais, l'anglais (utilisé pour lier les dictionnaires existants) et le thaï. Cette architecture rend possible le point suivant.

2) De la base, il est possible d'extraire des dictionnaires bilingues ou multicibles à usage humain par exemple au format LAF (Lexique Actif du Français), au format FeM ou "machinal" pour la traduction automatique, les outils d'aide aux traducteurs, de correction, de paraprasage, ou pour l'indexation, la synthèse de parole, etc.

3) La consultation des données se fait de manière interactive par le Web, avec possibilités de contrôler la présentation et de filtrer les informations, comme dans l'expérience réalisée avec le serveur du FeM en partie B.

4) La structure des dictionnaires utilise un fondement linguistique en pointe. Elle reprend en effet la structure du "DiCo" élaborée par Igor Mel'tchuk et Alain Polguère et est basée sur la lexicologie explicative et combinatoire, branche de la théorie sens-texte.

5) La construction des ressources est coopérative et se fait sur le Web. Le serveur Papillon rend la base accessible par tous en lecture. Tout client voulant consulter la base est un contributeur potentiel. Il s'inscrit lors de sa première connexion et reçoit un "compte" avec mot de passe et zone de travail dans laquelle seront mis ses profils d'intérêt et de compétence, l'historique de ses consultations et surtout ses contributions (fragments d'entrées, corrections, annotations, remarques générales).

Il est possible de construire des groupes et définir les règles d'accès associées. Un groupe prédéfini (groupe central de spécialistes lexicologues) sera seul habilité à modifier la base. Le rôle de ses membres sera de filtrer, corriger, valider et intégrer les contributions de tous, en fonction bien sûr des compétences de chacun.

6) La base est alimentée au départ par la réutilisation de ressources lexicales informatisées libres de droits. Il est ainsi prévu de récupérer en 2001-2002 les ressources informatisées existantes suivantes : le JMDict de Jim Breen composé de 70 000 articles japonais->anglais, le FeM composé de 20 000 articles et 50 000 lexies français->anglais, le dictionnaire du projet SAIKAM d'environ 4 000 articles japonais<->thaï. Ensuite, les foncionnalités de contribution lexicale généralisée seront mises en route quand un noyau suffisant aura été réalisé.

7) La base fonctionne en "source ouverte". Les utilisateurs sont encouragés à contribuer. Chaque contribution effective validée augmente un capital de points initial. Chaque extraction d'un dictionnaire sous forme de fichier diminue ce capital de points. La consultation reste gratuite. Les utilisateurs peuvent de cette façon contribuer en mettant à disposition de tous leurs propres lexiques personnels sous forme de contributions.

8) Les liens sont construit de façon pragmatique. Une axie n'est pas un concept, mais a vocation à le devenir. Les axies (liens interlingues) reflètent au départ des relations de traduction. Si un contributeur s'aperçoit qu'il s'agit de synonymes (quasi-)parfaits, donc qu'ils correspondent au même "concept", on

décidera de les fusionner. Par exemple, la lexie *AFFECTION* au sens médical est synonyme de *MALADIE*. C'est le même concept. Il est donc possible de fusionner ces deux lexies. Cela peut aussi arriver si un contributeur établit un lien de ces deux lexies avec la même lexie dans une autre langue. Par exemple, *AFFECTION* au sens médical et *MALADIE* se traduisent toutes deux en japonais par *BYOUKI* (). C'est une autre raison pour fusionner les deux lexies françaises.

9) Une liaison est prévue avec tous les projets de lexicographie multilingue pour autant que les contributeurs apportent l'information. Toute axie aura pour chaque système (WordNet, EDR, UNL, ONTOS, LexiGuide) un champ contenant une liste de symboles de ces systèmes (synset, concept, UW).

2. Cahier des charges

2.1. Aspects coopératifs

2.1.1. Langues présentes au départ

Au lancement du projet, les langues visées étaient le français et le japonais. Pour faire le lien entre des dictionnaires existants, nous avons rajouté l'anglais. Cela permet de croiser des dictionnaires français-anglais et anglais-japonais. Le projet a été lancé en coopération entre le GETA côté français et le NII côté japonais. Ensuite, les coopérations entre le NII et des organismes thaï comme le NECTEC et Kasetsart University à Bangkok sur le projet SAIKAM de dictionnaire japonais-thaï d'un côté, et entre le GETA et d'autres partenaires thaï sur la construction d'un dictionnaire français-thaï de l'autre, ont amené le NECTEC et Kasetsart university à coopérer au projet Papillon. Nous avons donc rajouté le thaï aux langues de départ. Enfin, la proximité du thaï et du lao d'une part et les travaux de Vincent Berment du GETA sur le lao d'autre part nous ont amenés à rajouter le lao.

Pour l'instant, il y a six langues dans le projet Papillon : l'anglais, le français, le japonais, le lao, le thaï et le vietnamien. Le malais devrait être rajouté à brève échéance. À moyen terme, le projet devrait s'élargir au coréen et au chinois. Les partenaires ne parlant pas tous la même langue, la langue de travail sera l'anglais. Les documentations seront écrites au moins en anglais pour être compréhensibles du plus grand nombre. C'est pourquoi nos commentaires dans les schémas XML en annexe sont rédigés en anglais.

2.1.2. Utilisateurs visés

Au début, les utilisateurs visés sont ceux susceptibles d'enrichir rapidement et efficacement la base lexicale. Les traducteurs français-japonais, français-thaï et thaï-japonais sont les premiers concernés. Ceux-ci ont d'ailleurs souvent leurs propres lexiques personnels qu'ils pourront mettre en commun et intégrer dans la base lexicale. Les contributeurs du projet SAIKAM, et en particulier l'ATPIJ (Association of Thai Professionals in Japan), seront aussi concernés par le projet Papillon.

Une fois que la base lexicale enrichie par les traducteurs offrira une couverture suffisante, le grand public pourra consulter la base à travers tout navigateur Web.

2.1.3. Élaboration du serveur

Le serveur permettant d'accéder à la base lexicale doit être accessible par Internet. Il doit principalement implémenter un serveur Web/Http. De plus, pour pouvoir être accessible au plus grand nombre, il doit aussi implémenter des interfaces pour les protocoles telnet/DICT [DICT], ftp, mail.

Le serveur doit aussi pouvoir être accédé par différents utilisateurs et groupes d'utilisateurs ayant des droits d'accès variables et différents. Il doit permettre à chaque utilisateur de créer un compte virtuel où seront stockées ses données personnelles comme les préférences, les contributions, les annotations, les crédits, etc.

Pour faciliter la communication entre les différents utilisateurs et groupes de la base, le serveur doit implémenter un système de listes de distribution de courrier électronique avec possibilité d'archivage et de consultation par le Web.

2.2. Principes lexicologiques

2.2.1. Architecture pivot de la base

Nous avons choisi de baser l'architecture de notre base lexicale sur NADIA, un système spécialisé de gestion de bases lexicales à l'interlingue par acceptions. Le système NADIA a été décrit en SUBLIM dans la thèse de Gilles Sérasset [Sérasset94e] et expérimenté par Etienne Blanc dans PARAX [Blanc96].

Chaque langue du projet sera décrite dans un dictionnaire monolingue. Ces dictionnaires seront reliés entre eux par un dictionnaire pivot de liens interlingues appelés acceptions interlingues (axi). Ces acceptions seront aussi reliées entre elles par des liens de raffinement.

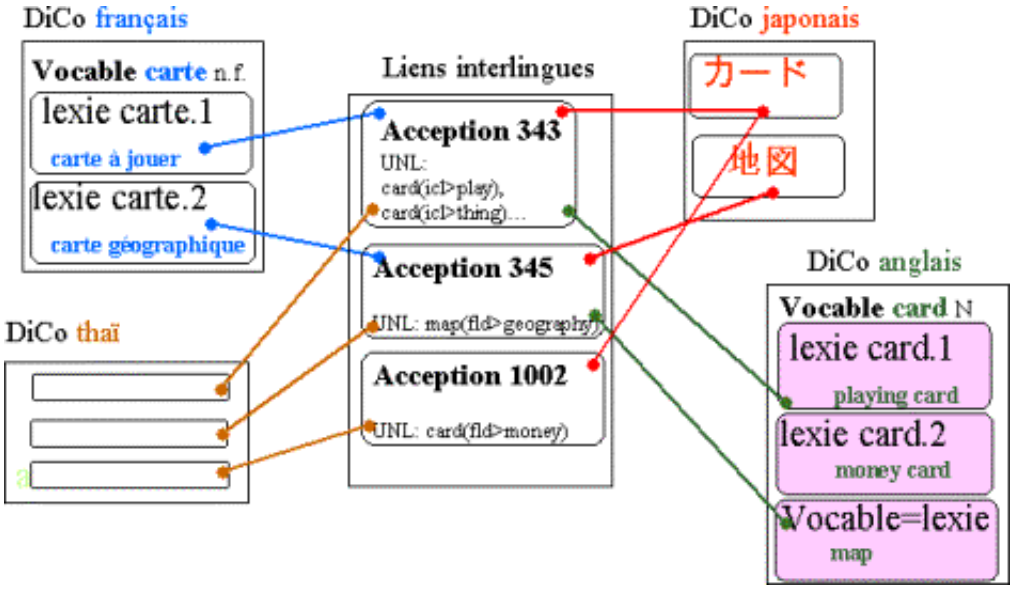


FIG. D.2 – *macrostructure du dictionnaire Papillon*

Les mots sont représentés dans les dictionnaires monolingues par des vocables. À chaque vocable correspond un ou plusieurs sens de mots appelés lexies. Chaque lexie est liée à une acception du dictionnaire pivot.

Dans l'exemple de la figure D.2, le vocable français *CARTE* a deux sens ou lexies : *CARTE À JOUER* et *CARTE GÉOGRAPHIQUE*. La lexie *CARTE À JOUER* est reliée à l'acception interlingue 343 et l'autre lexie *CARTE GÉOGRAPHIQUE* est reliée à l'acception interlingue 345. L'acception 343 est à son tour liée à plusieurs lexies de différents dictionnaires monolingues : la lexie *KAADO* du dictionnaire japonais, la lexie *CARD.1* (*playing card*) du dictionnaire anglais, la lexie *CARTE À JOUER* du dictionnaire français, etc.

Pour trouver la traduction en japonais de la lexie française *CARTE À JOUER* , il faut donc passer par l'acception interlingue 343 reliée à cette lexie. Cette acception est reliée à la lexie japonaise *KAADO* qui est donc la traduction japonaise de la lexie *CARTE À JOUER*.

2.2.2. Articles monolingues : les lexies de la base DiCo

Igor Mel'tchuk et ses collègues ont mis au point la théorie sens-texte d'abord en Russie (en particulier avec le laboratoire du professeur Rosenzweig) puis à l'université de Montréal. Cette théorie fournit les informations nécessaires pour passer d'une idée (le sens) à sa réalisation dans une langue donnée (le texte). La lexicologie explicative et combinatoire [Mel'tchuk95] est issue de la théorie sens-texte. Elle décrit une méthode de construction d'articles de dictionnaire basés sur cette théorie. Cette méthode est indépendante des langues. Elle permet donc de repésenter n'importe quelle langue.

La lexicologie combinatoire a permis de construire le Dictionnaire Explicatif et Combinatoire du français contemporain [Mel'tchuk92]. Son usage est expérimental. Il comporte peu de vocables mais chacun est très détaillé. Les vocables sont divisés en lexies qui représentent les unités de base du dictionnaire. La microstructure du DEC est trop complexe pour être utilisée à grande échelle. C'est pourquoi Alain Polguère [Polguère00] a simplifié les structures utilisées dans le DEC pour construire la base DiCo.

La microstructure des dictionnaires monolingues du projet Papillon se base sur celle des lexies de la base DiCo.

2.2.3. Articles interlingues : les axies

Les articles du dictionnaire interlingue relient les lexies monolingues des différentes langues ayant le même sens. Ce sont des acceptions interlingues ou axies. Les axies ont une catégorie sémantique pouvant prendre quatre valeurs différentes : entité, processus, résultat et état.

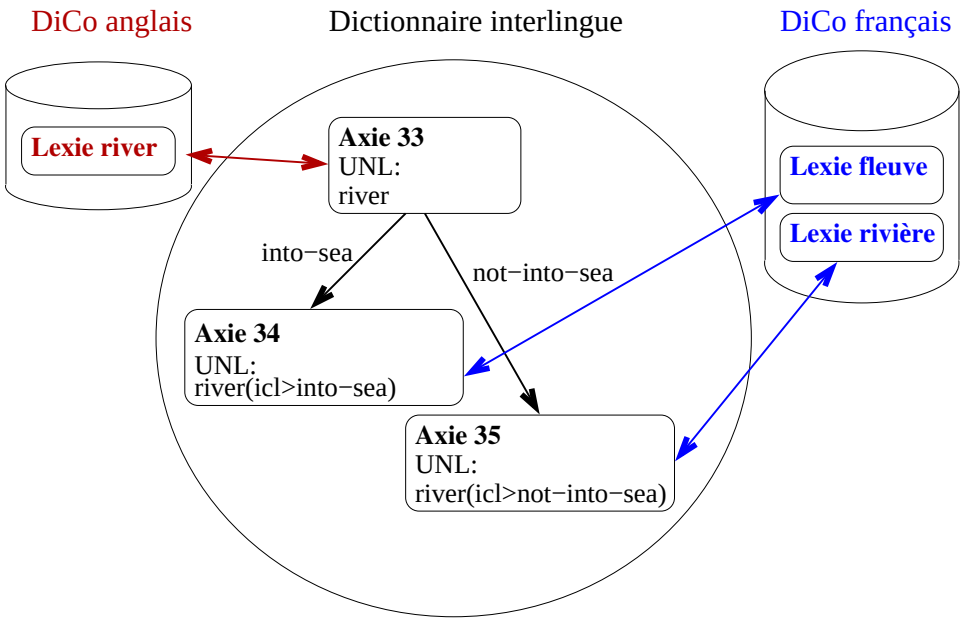


FIG. D.3 – *axies reliées par des liens de raffinement*

Les axies sont reliées entre elles par des liens de raffinement et de quasi-synonymie hérités des fonction lexicales de la lexicologie explicative et combinatoire. Les liens de raffinement sont éventuellement décorés par une glose en anglais expliquant ce lien. Dans l'exemple de la figure D.3, l'axie 33 liée à la lexie anglaise *RIVER* est reliée par des liens de raffinement aux axies 34 et 35 liées aux lexies françaises *RIVIÈRE* et *FLEUVE* . Le lien de raffinement de l'axie 33 vers l'axie 34 est décoré par une glose `not-into-sea` (pas

dans la mer) et l'autre lien de raffinement de l'axie 33 vers l'axie 35 est décoré par la glose `into-sea` (dans la mer). Ces gloses sont traduites puis utilisées pour générer la partie contrastive des dictionnaires bilingues.

Les articles interlingues peuvent aussi relier des exemples, des tournures, des citations ou des proverbes avec leurs traductions ou correspondances dans les autres langues. Les liens entre exemples sont appelés "exies"; les liens entre tournures "idiom axies"; les liens entre citations "citation axies", et les liens entre proverbes "proverb axies".

2.3. Ressources à récupérer et calendrier

2.3.1. Types de données à récupérer

Afin de faciliter la construction de la base lexicale, nous récupérons des données pour construire des squelettes d'articles qui seront ensuite complétés par les contributeurs et revus par les lexicologues.

- La base DiCo comportera à terme 3 000 vocables français encodés en Macintosh Western et stockés sous forme de champs textuels dans une base de données FileMaker. Nous avons déjà testé la conversion vers XML et UTF-8 d'environ 400 lexies avec succès.

- La partie français-anglais du dictionnaire FeM [Gut96] comporte environ 20 000 vocables français correspondant à 70 000 lexies françaises et 30 000 lexies anglaises. Ce dictionnaire encodé en Macintosh Western a été récupéré par Haï Doan-Nguyen au format LISPO. Nous l'avons converti en XML et UTF-8.

- Le dictionnaire japonais->anglais JMDict [EDict] de Jim Breen comporte environ 70 000 articles. Il est encodé en XML. De plus, un autre projet en cours piloté par Jean-Marc Desperrier a pour but d'y ajouter des traductions françaises. À l'heure actuelle, 10 000 articles sont déjà traduits.

- Le projet SAIKAM [SAIKAM] comporte environ 4 000 articles japonais-thaï encodés en XML.

Les ressources sont d'abord à récupérer pour les transformer au format XML/DML dans les structures définies pour le projet. Il faudra ensuite calculer des liens entre différentes langues automatiquement puis les faire réviser.

2.3.2. Étapes de la récupération

Nous avons défini une méthodologie de construction de la base à partir des ressources existantes. Nous distinguons trois étapes successives, chacune constituée de tâches pouvant être réalisées en parallèle.

- Étape 1 : récupération "primaire" de toutes les ressources disponibles complètes ou non avec transformation du format source vers XML/DML et de l'encodage d'origine vers UTF-8.

- Étape 2 : fusion et intégration des données dans les dictionnaires monolingues de Papillon. Le dictionnaire français contiendra les données de la base DiCo et de la partie française du FeM. Le dictionnaire anglais contiendra les parties anglaises du FeM, de JMDict, et de SAIKAM. Le dictionnaire japonais contiendra les parties japonaises de JMDict et de SAIKAM. LE dictionnaire thaï contiendra la partie thaïlandaise de SAIKAM.

- Étape 3 : évolution par travail coopératif sur le Web.

2.4. Description des interactions et sorties

2.4.1. Types de sorties à produire

Les formats cibles sont en priorité ceux des dictionnaires qui auront été récupérés puis intégrés dans la base lexicale. Pour pouvoir regénérer les dictionnaires intégrés, il faut donc adopter un principe clair : garder toutes les informations des ressources que l'on récupère.

Nous devons aussi générer des dictionnaires monolingues d'usage comme le LAF, fabricable à partir du format DiCo. Nous devons aussi produire des dictionnaires multilingues furcoïdes du type du FeM avec une langue source et plusieurs langues cibles dont l'anglais (FeT, FeJ, JeT, TeJ, etc.) qui peut servir de point de référence pour beaucoup d'utilisateurs. Cependant, il est peu utile de conserver l'anglais dans un dictionnaire imprimé. Les formats du dictionnaire JMDict et du projet SAIKAM sont aussi à produire.

Nous devons aussi proposer des outils pour produire d'autres types de sorties comme des bases terminologiques, des dictionnaires d'unités de vocabulaire virtuel (UW), etc.

2.4.2. Types de consultation de la base

Au départ, la base sera principalement consultable par des humains. Les utilisateurs auront des profils très différents. Le linguiste spécialiste d'une langue particulière s'intéressera aux données monolingues sur cette langue. Le terminologue et le traducteur consulteront les liens interlingues. L'apprenant d'une nouvelle langue cherchera des informations sur cette langue et aussi des liens de traduction entre sa langue et la nouvelle langue. Les personnes navigant sur le Web et les touristes ont besoin d'informations minimales pour pouvoir décoder quelques éléments de texte dans une langue inconnue.

Tous ces utilisateurs accéderont principalement à la base via un navigateur Web. Le serveur Web doit donc répondre aux requêtes des utilisateurs via un serveur http. Les temps de réponses aux requêtes simples (recherche d'un article par son mot-vedette) ne doivent pas excéder une demi-seconde pour que le serveur puisse être utilisable

La base lexicale peut être aussi consultable directement par des applications. Celles-ci peuvent se connecter au serveur via les différents protocoles disponibles : http, ftp, telnet/Dict, mail. Les besoins des applications peuvent être très divers. Une application de traduction automatique peut se connecter au serveur de la base lexicale pour le traitement d'un mot inconnu en cours de traduction. Le temps de réponse doit être très bref pour ne pas gêner le processus. Par contre, une application utilisant un dictionnaire se mettant à jour en tâche de fond n'a pas besoin d'une réponse immédiate.

2.4.3. Ouvertures possibles à d'autres modules

Notre serveur doit pouvoir s'ouvrir vers d'autres modules extérieurs permettant d'aider les utilisateurs lors de la consultation de la base en amont ou en aval. Nous souhaitons réutiliser des lemmatiseurs comme lors des expériences de la partie B et connecter tout autre module suscceptible d'aider la consultation. Une API sera disponible afin d'ajouter des modules mais pour l'instant, aucun module spécifique n'est prévu.

3. Spécifications externes

3.1. Serveur Papillon

3.1.1. Scénarios type

Accueil

La figure D.4 montre la page d'accueil du serveur Papillon. Lorsque les utilisateurs se connectent pour la première fois, ils doivent s'enregistrer dans la base en cliquant sur le menu de gauche. Lors des sessions suivantes, ils doivent s'identifier en cliquant sur le menu de gauche.

Ensuite, les utilisateurs choisissent une section dans la barre des menus horizontale en haut. La section "informations" contient l'archivage de la liste de distribution du projet Papillon, la section "consultation" permet de consulter les dictionnaires de la base Papillon, et la section "édition" permet de rédiger de nouveaux articles ou de contribuer localement sur des articles. Les parties sont pour l'instant en cours de développement.

Consultation

Les utilisateurs qui consultent la base Papillon ont accès à l'état actuel de ce qui est disponible dans la base Papillon. Ils éditent leurs préférences grâce à une interface spécialisée répondant à l'API de personnalisation décrite en partie C. Elles sont traduites ensuite en feuilles de style XSLT ou CSS puis stockées sur le serveur Papillon dans leur espace virtuel.

La consultation de la base Papillon se fait avec un navigateur Web classique. L'utilisateur compose sa requête et visualise le résultat dans son navigateur.

Annotation

Les utilisateurs peuvent annoter toutes les données de la base Papillon. Il est possible d'annoter les articles lors de la consultation ou directement le travail d'autres contributeurs lors de la correction d'articles existants. Les annotations sont stockées sur le serveur dans l'espace virtuel des utilisateurs. Elles peuvent être partagées entre des groupes d'utilisateurs.

Contribution directe

Les personnes souhaitant contribuer doivent impérativement s'enregistrer la première fois sur le serveur Papillon. Par la suite, il se connectent et s'identifient. Dans la partie édition du serveur, un classement récompense les contributeurs du mois. Ensuite, les contributeurs ont accès à un panneau d'articles et de liens à réviser. Ils téléchargent les articles sur lesquels il souhaitent travailler et les éditent ensuite localement sur leur machine. Il est aussi possible de contribuer et d'annoter des contributions d'autres contributeurs.

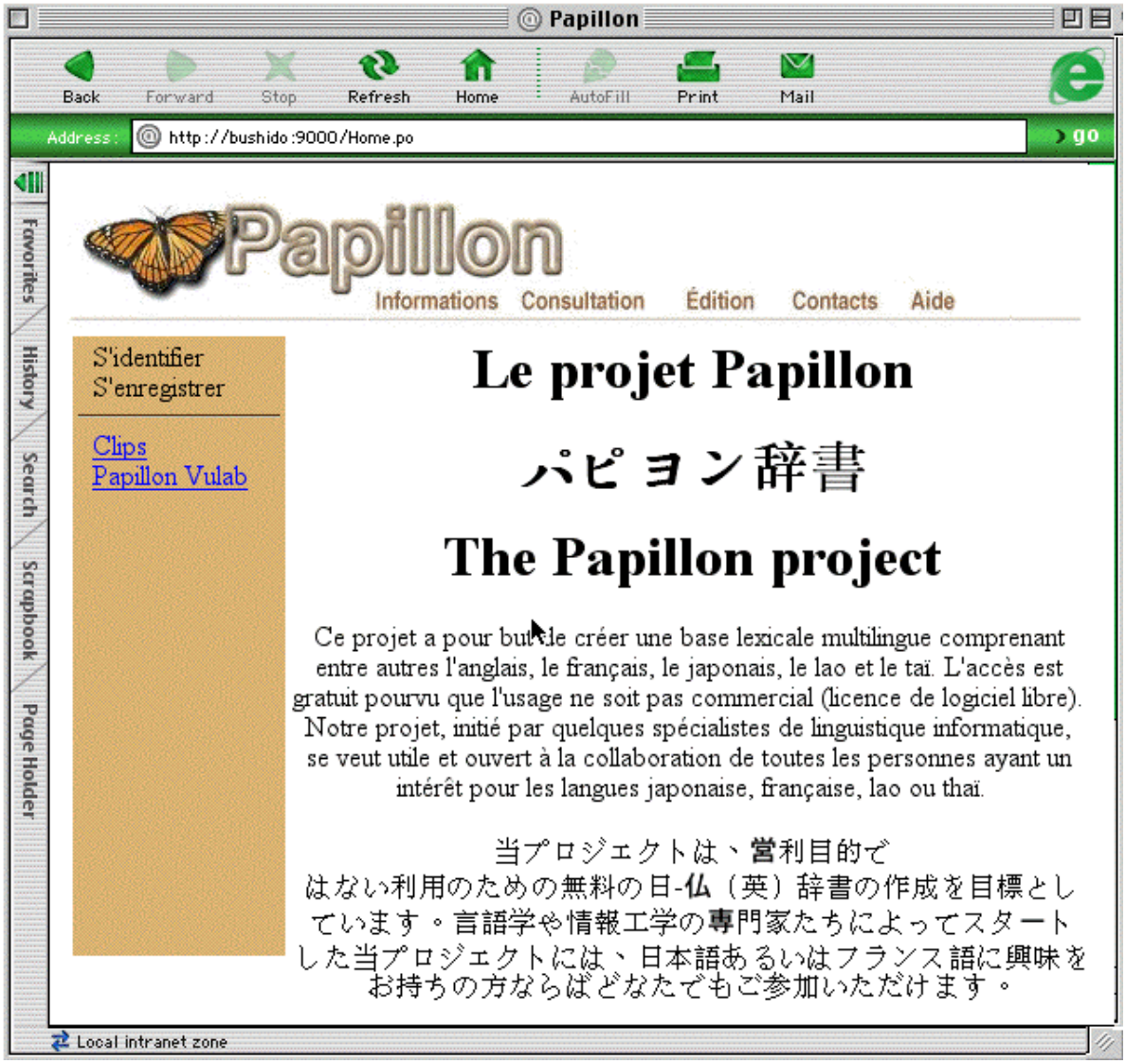


FIG. D.4 – *page d'accueil du serveur Papillon*

Lorsque les lexicographes et contributeurs ont fini leur travail, ils se reconnectent au serveur Papillon et envoient les articles modifiés et les nouveaux articles. Ces données sont stockées dans leur espace virtuel en attente de révision.

Extraction de données

Chaque utilisateur peut extraire de la base Papillon de nouveaux dictionnaires au format qu'il souhaite. Pour cela, il doit spécifier la structure de son dictionnaire à l'aide d'une interface spécialisée. Cette structure est ensuite convertie en une feuille de style XSLT qui est envoyée au serveur Papillon. Le système génère automatiquement son dictionnaire à partir de la feuille de style.

Validation/correction des contributions

Les spécialistes lexicologues ou lexicographes en chef révisent les contributions avant de les intégrer à la base. Pour cela, ils se connectent au serveur. Dans la partie réservée aux lexicologues, ils ont accès à un panneau des contributions non intégrées à la base. Ils sélectionnent des contributions, les révisent et décident de les intégrer ou non à la base. Pour chaque contribution intégrée, son auteur est gratifié de crédits.

Ajout d'une nouvelle langue

Pour chaque nouvelle langue, un spécialiste lexicologue doit décrire dans un schéma XML les parties spécifiques de cette langue (catégories grammaticales, liste de quantificateurs, etc.). Il se connecte ensuite au serveur Papillon et envoie le schéma XML.

Sur le serveur, le document est analysé. Les informations sont utilisées pour créer dans la base de données une nouvelle entrée pour la langue nouvellement définie.

Intégration de données existantes

Les données existantes sont intégrées par un spécialiste lexicologue. Il faut ajouter un nouveau dictionnaire pour chaque ressource récupérée. Si le dictionnaire contient déjà des données, le lexicologue les envoie au serveur sous forme de document XML en même temps que le fichier de description du dictionnaire. Ensuite il écrit une grammaire de récupération des données existantes dans le formalisme H-grammar. Il envoie ensuite le fichier représentant la grammaire H-grammar. Le serveur récupère automatiquement ces données existantes et les intègre à la base.

Ensuite, le lexicologue peut souhaiter intégrer ce dictionnaire récupéré au dictionnaire Papillon pour constituer un squelette. Pour cela, il décrit l'algorithme de manipulation du dictionnaire en partie avec le langage PRODUCDIC et en partie avec des commandes du système Papillon. Il envoie ensuite cet algorithme au système. Ce dernier manipule alors automatiquement le dictionnaire récupéré pour compléter le dictionnaire Papillon. Si le dictionnaire récupéré est envoyé par un contributeur, celui-ci obtient des points de crédit pour sa contribution.

3.1.2. Utilisateurs et groupes

Utilisateurs

Lors de leur première connexion, les utilisateurs sont invités à s'enregistrer dans la base. Ils doivent fournir leur nom, prénom, un login et un mot de passe ainsi qu'une adresse courriel. Ils peuvent aussi indiquer leurs intérêts, leurs compétences, et éventuellement leur désir d'apporter à la base le contenu de fichiers dont ils disposent. Un espace virtuel leur est ensuite alloué avec un capital de points de départ pour les contributions. Ils peuvent ensuite stocker leurs préférences, annotations, contributions, etc.

Groupes

Au départ, trois groupes prédéfinis sont enregistrés sur le serveur : l'univers (ensemble de tous les utilisateurs), les administrateurs et les lexicologues. Les administrateurs créent des comptes utilisateurs et des groupes, administrent les serveurs Web, ftp et les listes de discussion par courrier électronique. Les lexicologues gèrent les contributions. Ils les révisent et les intègrent aux ressources existantes. Ils proposent aussi des listes de choses à faire par les contributeurs.

Les utilisateurs peuvent se constituer en groupes pour partager des annotations et des contributions, créer une liste de distribution par courrier électronique, etc. Ils doivent être au minimum deux pour constituer un groupe. Ils envoient leur demande à un aministrateur qui crée le groupe.

Liste des commandes

Pour gérer les différents utilisateurs et groupes, le serveur interprète les commandes suivantes :

– création d'un nouvel utilisateur

– modification des données d'un utilisateur

– suppression d'un utilisateur existant

– création d'un nouveau groupe

– ajout d'un utilisateur dans un groupe

– suppression d'un utilisateur dans un groupe

– suppression d'un groupe

Le serveur Web de Papillon doit proposer des interfaces sous forme de formulaires HTML permettant d'accéder à ces commandes. Par exemple, pour l'ajout d'un nouvel utilisateur dans la base, celui-ci devra renseigner cinq champs du formulaire HTML : nom, prénom, login, mot de passe et adresse électronique. Le formulaire HTML envoie ensuite les données au serveur qui exécute la commande "création d'un nouvel utilisateur" avec ces données comme paramètres.

3.1.3. Outils utilisés pour construire le serveur

Le paradigme de développement de LINUX appliqué à la base Papillon d'une part et le budget limité pour le projet Papillon d'autre part nous incitent à choisir des outils gratuits ou avec une licence de logiciel libre (Open Source). Toutefois, pour pouvoir utiliser les outils RÉCUPDIC et PRODUCDIC, nous avons besoin d'une licence commerciale de MCL (Macintosh Common Lisp).

Les données lexicales, les données nécessaires au fonctionnement du serveur ainsi que les données relatives aux utilisateurs et groupes sont toutes stockées sur le serveur. Nous utiliserons donc un SGBD libre et compatible Unicode pour ce stockage.

Le serveur doit implémenter un serveur Web pour être accessible via Internet par http.

Pour faire le lien entre la base de données et le serveur Web d'une part et aussi pour manipuler les documents XML, il faut un langage implémentant une API DOM ainsi qu'un pilote de base de données.

3.2. Structures de données

3.2.1. Description des structures

Toutes les données de la base lexicale sont décrites sous forme de documents XML. Tout se passe come si chaque "collection" (lexies, axies, poids, utilisateurs, contributions, profils) était un grand fichier XML. La structure de ces documents est décrite par le schéma DML en annexe A.

On définit ensuite la granularité de la représentation dans le SGBD choisi pour le stockage.

Chaque cellule du SGBD du type "`balise`" contient un fragment du "fichier total XML" compris entre <balise> et <balise>. En ce qui concerne les attributs liés à <balise>, ils sont stockés dans un autre champ de la même table du SGBD. Ce champ porte le nom de l'attribut.

On peut ensuite recréer le fichier total XML à tout moment.

La microstructure des dictionnaires du projet est décrite par des schémas XML spécifiques. Ces schémas redéfinissent l'élément <article> du schéma DML. Ils utilisent les éléments et les types définis dans le schéma DML. Par exemple, on doit redéfinir la liste des catégories morphosyntaxiques pour chaque langue : le thaï n'a pas d'adjectifs, le lao a cette catégorie, le japonais en distingue plusieurs, etc. Le schéma spécifique au français redéfinit pour cet exemple le type des catégories `posType` :

```
<redefine schemaLocation="papillon.xsd">
  <simpleType name="posType">
    <restriction base="dml:posType">
      <enumeration value="adj"/>
      <enumeration value="adv"/>
      <enumeration value="nom"/>
      ...
      <enumeration value="verbe pron"/>
    </restriction>
  </simpleType>
</redefine>
```

3.2.2. Principe de poids sur les éléments

La base lexicale complète peut être vue comme un seul ensemble pondéré. Cela permet d'implémenter des systèmes de "dictionnaires neuronaux" utilisables pour la désambiguïsation lexicale en contexte [Dolan96] ou d'indiquer des fréquences d'apparition dans des corpus.

Pour représenter cet ensemble pondéré, les unités de base des lexiques, les lexies et les axies doivent pouvoir porter des poids. Ces unités de base forment alors les nœuds d'un graphe pondéré. Les liens entre ces nœuds doivent aussi pouvoir porter des poids. Ce sont les arcs du graphe.

Pour que les utilisateur puissent implémenter plusieurs théories différentes pour pondérer les objets de la base, les poids ne sont pas stockés sur les objets mais à part. Les objets portent alors tous des identificateurs qui les relient à leur liste de poids. Les pods sont stockés dans une matrice à deux dimensions. En abscisse sont indiqués les identificateurs d'objets portant des poids et en ordonnées, les différents contextes d'utilisation de ces poids.

3.2.3. Manipulation des structures

Les spécialistes lexicologues forment un groupe qui gère la construction des dictionnaires. Ils ont besoin de manipuler les unités de base de ces dictionnaires, les lexies et les axies. Ils peuvent en créer et en fusionner. Ils ont aussi besoin de créer des ensembles virtuels de lexies ou d'axies selon des critères particuliers pour pouvoir vérifier le contenu de la base et contrôler sa qualité.

Si un lexicologue s'aperçoit entre autres qu'un contributeur fait systématiquement la même faute sur les lexies qu'il envoie au serveur, par exemple qu'il indique systématiquement comme catégorie "adj." au lieu de "a." pour les adjectifs, il peut créer un ensemble virtuel de toutes les lexies envoyées par ce contributeur et corriger la faute de façon systématique.

D'autre part, les ensembles virtuels peuvent servir pour construire un lexique et ensuite l'exporter selon un format particulier. Par exemple, exporter la liste des verbes pronominaux français.

Voici la liste des commandes auxquelles le système doit répondre :

– créer une lexie

– fusionner deux lexies

– créer une axie

– fusionner deux axies

– créer un ensemble virtuel de lexies

– créer un ensemble virtuel d'axies

3.3. Récupération

La récupération de ressources existantes se fait en deux étapes en suivant un principe de "traçabilité" utile pour la notoriété des contributeurs et aussi pour la "dépollution" extérieure comme dans l'exemple précédent. Tout élément d'information doit être traçable à une granularité définie par la structure XML : chaque élément XML peut porter un attribut qui référencie l'historique de ses modifications.

La première étape consiste à convertir le format et la structure des ressources. Le format d'origine est converti vers le format XML/DML. La structure est récupérée et un maximum d'informations est balisée de façon à pouvoir ensuite la réutiliser. Cette étape s'effectue selon la méthode RÉCUPDIC.

La deuxième étape consiste à répartir chaque ressource convertie en lexies et axies et à intégrer ensuite les lexies et les axies dans la base lexicale. Cette étape se fait automatiquement par le système à l'aide d'un script de commandes PRODUCDIC propre à chaque ressource.

Voici un exemple concret de répartition sur le FeM après récupération :

```
/* pour tous les articles du FeM */
for-all entry in FeM do
   /* pour toutes les catégories syntaxiques */
   for-all sense in entry do
     /* pour tous les sens français */
     for-all sense1 in sense1*(sense) do
       /* créer une lexie française */
       create-obj MaFra from entry
       /* créer une axie reliée à la lexie MaFra */
       create-obj MonAxie from MaFra
       /* pour tous les sens anglais */
       for-all eng in eng*(sense1) do
          /*créer une lexie anglaise MaEng */
          create-obj MaEng from sense1
         /* relier l'axie à la lexie MaEng */
          link-obj(MonAxie,MaEng)
          /* stocker la lexie dans la base */
          store-database MaEng
     /* stocker la lexie MaFra dans la base */
     store-database MaFra
     /* stocker la lexie MaFra dans la base */
     store-database MonAxie
```

La fonction store-database n'est pas encore implémentée. Il est possible dans un premier temps d'effectuer la répartition en stockant les objets résultats dans un fichier qui sera ensuite converti en XML/DML puis envoyé au serveur Papillon. Le système assurera lui-même automatiquement le stockage dans la base.

3.4. Consultation

La consultation sur le site de Papillon suit les principes de la partie C. Les utilisateurs consultent la base avec n'importe quel navigateur Web. Pour ne pas pénaliser les utilisateurs, la transformation des documents XML avec des feuilles de style XSL se fait sur le serveur. Le résultat final est entièrement au format HTML. Il est ensuite envoyé à l'utilisateur.

Les utilisateurs peuvent définir leurs préférences de présentation à l'aide d'une interface spécialisée. Ces préférences sont ensuite envoyées au serveur puis stockées sous forme de feuilles de style XSL (XSLT et XSL-FO) et CSS dans l'espace virtuel des utilisateurs. Elles sont ensuite appliquées au résultat de chaque requête de ces utilisateurs. Il est aussi possible de partager des préférences entre groupes d'utilisateurs.

4. Analyse générale et implémentation

4.1. Définition des structures avec DML

4.1.1. Organisation des schémas XML

La définition formelle de toutes les structures utilisées dans le projet Papillon est faite par des schémas XML. Le langage des schémas XML permet d'importer ou de redéfinir des parties de schémas avec les clauses <import> et <redefine>. Nous avons donc organisé nos schémas XML en redéfinissant des parties d'autres schémas. La figure D.5 montre l'organisation des schémas XML utilisés dans le projet Papillon.

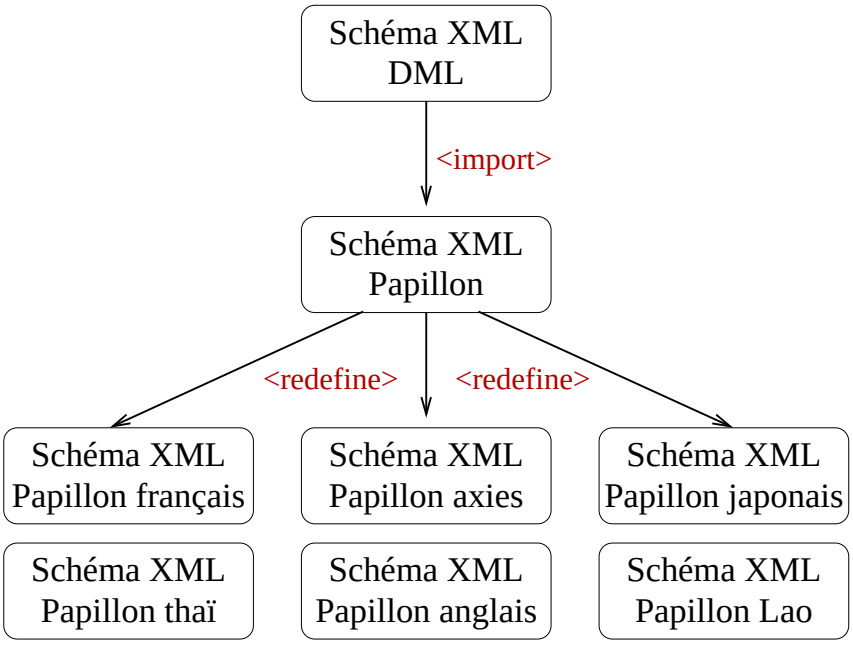


FIG. D.5 – *organisation des schémas XML dans le projet Papillon*

Au départ, nous utilisons le schéma DML dans lequel sont définis les éléments communs DML, les bases lexicales, les dictionnaires et les lexiques. Ensuite, le schéma Papillon importe le schéma DML et définit les éléments communs au projet comme les lexies et les axies. Enfin, les schémas spécifiques aux langues ou aux lexiques comme papillon-fra pour le français, papillon-jpn pour le japonais et papillon-axi pour les liens interlingues redéfinissent les éléments spécifiques du schéma général papillon.

4.1.2. Macrostructure des dictionnaires

La base lexicale

La base lexicale Papillon est décrite par l'élément DML <database>. La description formelle de cet élément est réalisée par le schéma DML donné en annexe A. Pour l'instant, la base contient quatre dictionnaires référencés par l'élément <dictionaries>. Le FeM, le JMDict et le DiCo sont utilisés pour construire des squelettes d'articles du dictionnaire Papillon.

Voici la description en LEXARD++ de la base lexicale Papillon :

```
(define-lexical-database GETA-database
  :owner GETA
  :comment "Papillon Lexical Database"
  :creation-date "25/06/01"
  :users  (root VB FL MM AP GS MT)
  :administrators   (MM GS)
  :lexicologists  (FL MT)
  :partner-servers  (XRCE-analyser)
  :dictionaries  (DiCo FeM JMDict Papillon SAIKAM))
```

Voici le document XML équivalent :

```
<database
 xsi:schemaLocation="http://www-clips.imag.fr/geta/services/dml
http://www-clips.imag.fr/geta/services/dml/dml.xsd"
 history-ref="database-his.xml"
 name="Papillon Lexical Database"
 creation-date="25/06/01"
 owner="GETA">
  <partner-servers>     <user-ref name="XRCE Analyser" href="xrce.xml"/>
  </partner-servers>   <users>
    <user-ref name="Vincent.Berment" href="berment.xml"/>
    <user-ref name="François.Lareau" href="lareau.xml"/>
    <user-ref name="Mathieu.Mangeot" href="mangeot.xml"/>
    <user-ref name="Alain.Polguère" href="polguere.xml"/>
    <user-ref name="Gilles.Sérasset" href="serasset.xml"/>
    <user-ref name="Mutsuko.Tomokiyo" href="tomokiyo.xml"/>
  </users>
  <groups>
  <group name="lexicologists" >
      <user-ref name="Francis.Bond"/>
      <user-ref name="Mutsuko.Tomokiyo"/>
      <user-ref name="François.Lareau"/>
      <user-ref name="Alain.Polguère"/>
   </group>
   <group name="administrators">
     <user-ref name="Mathieu.Mangeot"/>
     <user-ref name="Gilles.Serasset"/>
   </group>
  </groups>
  <dictionaries>
```

```
    <dict-ref name="DiCo" href="DiCo.xml"/>
    <dict-ref name="FeM" href="FeM.xml"/>
    <dict-ref name="JMDict" href="JMDict.xml"/>
    <dict-ref name="papillon" href="papillon.xml"/>
    <dict-ref name="SAIKAM" href="SAIKAM.xml"/>
  </dictionaries>
</database>
```

Ce fichier est en évolution constante, tout comme le contenu de la base elle-même.

Le dictionnaire Papillon

Le dictionnaire Papillon est décrit par l'élément DML <dictionary>. La description formelle de cet élément est réalisée par le schéma DML donné en annexe A. Ce dictionnaire est ensuite réparti en plusieurs volumes référencés par l'élément <volume> . On trouve un volume interlingue Axies et un volume pour chaque langue : English pour l'anglais, French pour le français, Japanese pour le japonais, Lao pour le lao et Thai pour le thaï. Les liens entre les volumes sont notés par l'élément <links>. Les articles des volumes de chaque langue sont reliés aux articles du volume interlingue.

Voici la description en LEXARD++ du dictionnaire Papillon :

```
(define-dictionary Papillon
  :owner GETA
  :comment "Papillon Multilingual Dictionary"
  :category "multilingual"
  :creation-date "16/05/2001"
  :installation-date "16/05/2001"
  :encoding "UTF-8"
  :format "XML"
  :hw-number 381
  :type "pivot"
  :version 1
  :source-languages (Axies)
  :target-languages (English French Japanese Lao Thai)
  :contents "general vocabulary"
  :domain "general"
  :legal "all rights belong to GETA and NII"
  :cdm-element (headword pronunciation pos corpus)
  :volumes (English French Japanese Lao Thai Axies)
  :links (from Axies
     (to English French Japanese Lao Thai)))
```

Voici le document XML équivalent :

```
<dictionary
 xsi:schemaLocation="http://www-clips.imag.fr/geta/services/dml
 http://www-clips.imag.fr/geta/services/dml/dml.xsd"
 history-ref="papillon-his.xml"
 category="multilingual"
 creation-date="21/06/01"
 encoding="UTF-8"
 format="XML"
```

```
hw-number="381"
installation-date="16/05/2001"
fullname="Papillon Multilingual Dictionary"
name="Papillon"
owner="GETA"
type="pivot"
version="1">
 <languages>
   <source-language lang="axi"/>
   <target-language lang="eng"/>
   <target-language lang="fra"/>
   <target-language lang="jpn"/>
   <target-language lang="lao"/>
   <target-language lang="tha"/>
 </languages>
 <contents>general vocabulary</contents>
 <domain>general</domain>
 <legal>all rights belong to GETA and NII</legal>
 <cdm-elements>
   <headword delay="1s"/>
   <pronunciation delay="5s"/>
   <pos delay="5s"/>
   <corpus delay="10s"/>
 </cdm-elements>
 <administrators>     <user-ref name="Mathieu.Mangeot"/>
 </administrators>   <volumes>
   <volume-ref name="English" lang="eng" href="papillon-eng.xml"/>
   <volume-ref name="French" lang="fra" href="papillon-fra.xml"/>
   <volume-ref name="Japanese" lang="jpn" href="papillon-jpn.xml"/>
   <volume-ref name="Lao" lang="lao" href="papillon-lao.xml"/>
   <volume-ref name="Thai" lang="tha" href="papillon-tha.xml"/>
   <volume-ref name="Axies" lang="axi" href="papillon-axi.xml"/>
 </volumes>
 <links>
   <link from="Axies" to="English" type="bijective"/>
   <link from="Axies" to="French" type="bijective"/>
   <link from="Axies" to="Japanese" type="bijective"/>
   <link from="Axies" to="Lao" type="bijective"/>
   <link from="Axies" to="Thai" type="bijective"/>
 </links>
</dictionary>
```

Les volumes

Chaque langue est représentée par un volume monolingue qui regroupe les articles de la langue. Les articles monolingues sont des lexies. Les vocables sont construits automatiquement à la volée à partir des lexies. Il y a en plus un volume pivot dont les articles sont les liens interlingues ou axies reliant les articles

des volumes monolingues. L'attribut history-ref fait référence à un fichier où sont stockés les historiques des modifications de tous les articles contenus dans les volumes. La description formelle de l'élément <volume> est réalisée par le schéma Papillon en annexe B.

Voici la description en LEXARD++ du volume du français :

```
(define-volume French
  :comment "Volume du français"
  :source-language "French"
  -- /* articles composant le volume */ --
  ...
  ...
)
```

Voici le document XML correspondant :

```
<volume
 xsi:schemaLocation="http://www-clips.imag.fr/geta/services/dml
 http://www-clips.imag.fr/geta/services/dml/papillon-fra.xsd"
 history-ref="French-his.xml"
 name="French"
 source-language="fra">
```

4.1.3. Microstructure des dictionnaires

Article monolingue : une lexie

Les articles monolingues sont représentés par les éléments <lexie>. Leur structure est une adaptation en XML des lexies dans la structure dictionnairique DiCo [Polguère00] définie par Alain Polguère. La définition formelle de la partie commune à toutes les langues des lexies est représentée par le schéma XML Papillon en annexe B.

Chaque lexie contient un identificateur unique dans la base, porté par l'attribut id. Cet attribut est du type DML ID. Il est construit en concaténant le mot-vedette de la lexie avec un numéro. Si la lexie est détruite, l'identificateur n'est pas réaffecté. Il est interne à la base et caché aux utilisateurs. L'attribut basic indique si la lexie est l'unité lexicale de base du vocable. Cet attribut est du type booléen. L'attribut history est un attribut DML. C'est un identificateur permettant de faire référence à l'historique des changements ayant eu lieu dans la lexie. L'origine des informations est aussi stockée dans l'historique. Si une modification a ensuite lieu, un attribut history est automatiquement créé sur l'élément XML le plus proche contenant toute la modification.

Les lexies contiennent 10 éléments principaux : le nom du vocable, la prononciation, les éléments spécifiques aux langues, la catégorie grammaticale, la formule sémantique, le régime, les fonctions lexicales, les exemples, les idiotismes et les liens vers les axies.

Le nom du vocable est une représentation graphique de la lexie trouvée dans les textes. Pour représenter ce nom, nous utilisons l'élément DML <headword>. Cela nous permet de donner une définition sémantique précise à cet élément.

La prononciation est représentée par l'élément DML <pronunciation>. Les encodages peuvent être différents selon les langues : alphabet phonétique international, transcriptions phonétiques, encodages "maison", etc.

Les informations spécifiques à chaque langue sont décrites par le groupe <language-specific> dans le schéma Papillon. Ce groupe est ensuite redéfini dans les schémas XML spécifiques aux langues.

La catégorie grammaticale est représentée par l'élément DML <pos>. Les valeurs possibles de cet élément sont redéfinies dans les schémas XML spécifiques aux langues.

La formule sémantique est représentée par l'élément <semantic-formula>. C'est un substitut de la définition lexicographique de la lexie. Elle est formée d'une étiquette sémantique suivie d'une structure décrivant les actants de l'unité lexicale. La formule sémantique de l'exemple suivant dit que le sens principal de MEURTRE est l'action de tuer et qu'elle comprend deux actants : celui qui tue (X) et celui qui est tué (Y).

Le régime de la lexie est représenté par l'élément <government-pattern>. Le régime indique les valences actives de l'unité lexicale. Cette information est présentée au moyen d'une table présentée dans le DEC [Mel'tchuk84,88,92]. Le régime ci-dessous indique que le tueur et la victime d'un meurtre peuvent être exprimés par un complément prépositionnel précédé de *de* ou par des adjectifs possessifs.

Les fonctions lexicales sont regroupées dans l'élément <lexical-functions>. Elles sont ordonnées selon la méthodologie standard utilisée dans le DEC [Mel'tchuk84,88,92] : les fonctions paradigmatiques qui correspondent aux dérivations sémantiques sont suivies des fonctions syntagmatiques qui encodent les collocations. Enfin viennent les fonctions lexicales non standard. Les fonctions lexicales de base sont au nombre de 52. Ce sont les mêmes pour chaque langue. Certaines sont rarement utilisées dans certaines langues, mais toutes sont théoriquement possibles. Ces fonctions lexicales de base peuvent être combinées. Dans l'exemple suivant, la fonction lexicale "VO"représente le verbe associé au nom *MEURTRE* (*TUER*).

Chaque fonction lexicale est représentée par l'élément DML <function>. Chaque valeur de fonction est représentée par un élément <value>. Les valeur ayant une distance sémantique proche sont regroupés dans un élément <valgroup>. Lorsque la valeur d'une fonction lexicale est une autre lexie, sa référence est indiquée avec un lien. Cela permet de construire un véritable réseau entre les lexies monolingues.

Les exemples d'usage de la lexie sont représentés par l'élément <examples>. Chaque exemple porte aussi un identificateur unique dans la base. Cet identificateur est représenté par l'attribut id du type DML ID. Il nous permet de relier aussi les traductions des exemples via le dictionnaire interlingue.

Les idiotismes contenant la lexie sont représentés par l'élément <full-idioms>. Chaque idiotisme porte aussi un identificateur unique qui nous permet de relier ses traductions via le dictionnaire interlingue.

Enfin, les liens interlingues vers les axies sont regroupés dans l'élément <axies>. Chaque lien est ensuite représenté par l'élément <refaxie>. La référence est notée avec l'attribut href qui est du type Xlink. En théorie, une lexie n'est reliée qu'à une seule axie. Cependant, nous laissons la possibilité aux contributeurs de la relier à plusieurs axies. Ces liens multiples seront ensuite détectés par des programmes de vérification automatique et traités à part.

Voici un extrait de la lexie française MEURTRE :

```
<lexie id="meurtre$1" history="h01" basic="true" >
  <headword hn="1">meurtre</headword>
  <pronunciation encoding="GETA">meu+rtr(e)</pronunciation>
  <language-specific/>
  <pos>n.m.</pos>
  <semantic-formula>action de tuer:     PAR L'
    <actor><sem-label>individu</sem-label>
      <sem-variable>X</sem-variable></actor>DE L'
    <actor><sem-label>individu</sem-label>
      <sem-variable>Y</sem-variable></actor>
  </semantic-formula>
  <government-pattern>
    <mod nb="1">
      <actor>
        <sem-actant>X</sem-actant>=
```

```
            <synt-actant>I</synt-actant>=
            <surface-group>
              <surface>
                <reflexie href="#de">de</reflexie>N
              </surface>,
              <surface>A-poss</surface>
            </surface-group>
          </actor>
          <actor>
            <sem-actant>Y</sem-actant>=
            <synt-actant>II</synt-actant>=
            <surface-group>
              <surface>
              <reflexie href="#de">de</reflexie>N
              </surface>,
              <surface>A-poss</surface>
            </surface-group>
          </actor>
        </mod>
      </government-pattern>
      <lexical-functions>
        <function name="V0">
          <valgroup>
              <reflexie href="#tuer$1">tuer</reflexie>
          </valgroup>
        </function>
      </lexical-functions>
      <examples>
        <example id="e1">C'est ici que le double meurtre a été
commis.</example>
      </examples>
      <full-idioms>
        <idiom id="i1" href="papillon-axi.xml#i04">_appel au
meurtre_</idiom>
      </full-idioms>
      <axies>
        <refaxie href="papillon-axi.xml#a01"/>
      </axies>
    </lexie>
```

Cette lexie n'a pas encore été modifiée. Elle ne porte donc pas d'autres attributs `history` que celui de l'élément <lexie> indiquant sa provenance. Aucun poids n'a encore été calculé. La lexie ne porte donc pas non plus d'attributs `id` sur tous les éléments susceptibles de porter un poids.

Lorsque nous voudrons intégrer des informations plus fines provenant d'autres dictionnaires, en particulier des informations qui ne sont pas dans les schémas de Papillon, nous ajouterons pour chaque dictionnaire un élément supplémentaire portant son nom regroupant ces informations directement dans l'élément <lexie>. Ces informations pourront être réutilisées afin de regénérer les dictionnaires d'origine.

Spécificités du français

Les spécificités du français sont décrites formellement par le schéma Papillon français donné en annexe B de ce document. Ce schéma redéfinit le groupe spécifique aux langues de la lexie ainsi que la liste des catégories grammaticales. Pour le français, il n'y a pas d'informations spécifiques à ajouter. Le groupe <language-specific> est donc vide.

Les catégories grammaticales utilisées pour le français sont pour l'instant au nombre de 29. La liste provient du travail lexicographique réalisé au GETA sur les dictionnaires FeM et UNL-français. Nous n'avons pas utilisé la liste des catégories grammaticales de DiCo, car elle est moins précise. Les catégories sont décrites dans le schéma Papillon français donné en annexe B.

Spécificités du japonais

Les spécificités du japonais sont décrites formellement par le schéma Papillon japonais donné en annexe B de ce document.

Les catégories grammaticales utilisées pour le japonais sont tirées principalement de la liste des 29 catégories du dictionnaire Gakken Kokugo Daijiten édité en 1985 par Gakushûkenkyûsya, Tokyo. Elles sont décrites dans le schéma Papillon japonais donné en annexe B.

Nous avons ajoutés plusieurs informations spécifiques au japonais la transcription des kanji, les quantificateurs et les niveaux de langue (politesse et déférence).

Tout d'abord, les lexies japonaises sont souvent écrites avec des kanji (idéogrammes). Ces kanji ont plusieurs prononciations possibles. Pour les distinguer, nous ajoutons une lecture de la lexie à l'aide des syllabaire japonais hiragana et katakana. Cette lecture est stockée dans l'élément <yomigana>.

Les objets en japonais sont comptés de manière différente selon leur forme, leur taille, etc. Par exemple, pour compter les fruits ronds ou les ballons, on utilisera "ko"; pour compter des machines comme des voitures, des télévisions, on utilisera "dai", etc. Les quantificateurs appropriés sont notés comme valeurs des fonctions lexicales `Sing` et `Mult` (`Sing`(*riz*) = *grain*; `Mult`(*chien*) = *meute*). La liste des valeurs possibles provient en majorité de celle définie par Senko K. Maynard pour le Japan Times de Tokyo en 1990. Elle est définie dans le schéma Papillon japonais donné en annexe B.

Les niveaux de langue sont représentés par l'élément <language-levels>. La politesse est définie par quatre degrés majeurs : neutre, respect, humilité et politesse simple (l'humilité est équivalente à la déférence). Elle est représentée par l'attribut grade de l'élément <politeness grade ="neutral"/>.

La référence note la situation dans laquelle se trouve le locuteur. Par exemple, s'il parle de sa mère, la référence est cotextuelle. S'il parle d'une autre mère, la référence est contextuelle. Cet élément est important car selon la situation, le locuteur japonais n'emploiera pas le même mot. Elle est représentée par l'attribut `grade` de l'élément <reference grade="cotextuel" />.

Voici un extrait de la lexie japonaise ARAU correspondant au verbe français LAVER :

```
<lexie id="洗う $1" basic="true">
  <headword hn="1">  洗う  </headword>
  <kun-yomi>あらう</kun-yomi>
  <pos>他動詞</pos>
  <language-levels>     <politeness grade="neutral"/>
    <usage grade="NA"/>
    <reference grade="NA"/>
  </language-levels>   ...
</lexie>
```

Articles interlingues

Les articles interlingues sont des acceptions ou liens interlingues. Ce sont des axies représentées par l'élément <axie>. Les axies ne sont que des liens entre les lexies monolingues. Elles ne contiennent pas de définition. La description formelle de cet élément est notée par le schéma Papillon donné en annexe B.

Chaque axie contient un identificateur unique dans la base porté par l'attribut `id`. Cet attribut est du type DML ID. Si l'axie est détruite, l'identificateur n'est pas réaffecté. Il est interne à la base et caché des utilisateurs. L'attribut `history` est un attribut DML. C'est un identificateur permettant de faire référence à l'historique des changements ayant eu lieu dans l'axie.

L'axie est composée de quatre éléments principaux : la catégorie sémantique de l'axie, les liens vers les lexies monolingues, les liens vers d'autres axies et enfin les références externes. Certains éléments de l'axie sont susceptibles d'être modifiés au cours du projet avec l'ajout de liens vers une nouvelle langue ou vers une nouvelle référence externe. Les liens vers les lexies monolingues et les liens vers les références externes sont donc décrits formellement par le schéma Papillon axies donné en annexe B. Ce schéma redéfinit les deux éléments précédents du schéma Papillon.

La catégorie sémantique est représentée par l'élément <semantic-cat>. Les axies reliant des lexies peuvent prendre quatre valeurs possibles : entité `entity`, processus `process`, état `state` et résultat `result`. Les axies peuvent aussi relier des phrases entières. La valeur de la catégorie sémantique indique alors le type de phrase : `example` pour un exemple, `idiom` pour un idiotisme, `citation` pour une citation et `proverb` pour un proverbe.

Les liens vers les lexies sont représentés en fonction des langues. Le nom de l'élément regroupant les liens vers une langue donnée est construit avec le code à trois lettres ISO-639-2/T représentant cette langue. Par exemple, les liens vers des lexies françaises seront regroupés dans l'élément <fra>, pour des lexies anglaises, ce sera l'élément <eng>, etc.

Chaque référence vers une lexie est ensuite représentée par l'attribut `href` de type Xlink. Par exemple, un lien vers la lexie anglaise RIVER sera noté par l'élément `<reflexie href="river$1"/>`.

Les axies peuvent aussi être reliées entre elles par des liens de synonymie regroupés dans l'élément <synonyms>, des liens de raffinement regroupés dans l'élément <refinements>, et des liens inverses de généralisation regroupés dans l'élément <generalization>.

Chaque référence vers une axie est ensuite représentée par l'attribut href de type Xlink. Ces liens peuvent être étiquetés. Par exemple, un lien d'une axie ayant le sens de cours d'eau vers une autre axie ayant le sens de fleuve sera un lien de raffinement. Il sera étiqueté par exemple avec une glose anglaise expliquant que ce cours d'eau débouche dans la mer : `<refaxie tag-type="gloss:eng" tag="into-sea" href="a009"/>`.

Pour pouvoir relier nos données à celles d'autres projets et construire des dictionnaires les utilisant, nous relions nos axies à des références externes. Ces liens externes sont notés par l'élément <external-references>.

Pour l'instant, il est prévu des liens vers des UW du projet UNL [UNL96, UNL97], notés avec l'élément <UNL> ; des liens vers des sens du projet WordNet [Fellbaum98], notés avec l'élément <WordNet> ; des liens vers les catégories sémantiques du dictionnaire NTT, notés avec l'élément <NTTsemcat> et des liens vers des concepts du projet LexiGuide de la société LexiQuest, notés avec l'élément <LexiGuideConcept>.

Voici par exemple l'axie reliant la lexie française MEURTRE :

```
<axie id="a01">
  <semantic-cat>entity</semantic-cat>
  <fra>
    <reflexie href="meurtre$1"/>
  </fra>
```

```
    <eng>
      <reflexie href="murder$1"/>
    </eng>
    <external-references>
      <UNL resource="UNL-fr.unl">
        <refuw href="murder(icl>action,agt>human,obj>human)"/>
      </UNL>
      <WordNet resource="Wordnet.txt">
        <refsynset href="00143589"/>
      </WordNet>
    </external-references>
  </axie>
```

Les axies d'idiome ou de définition pourront aussi contenir comme référence externe leur représentation comme graphe UNL. Celui-ci sera noté avec l'élément <UNL-graph>.

4.2. Implémentation du serveur

4.2.1. Architecture générale du serveur

La figure D.6 montre l'architecture générale du serveur Papillon. Le cœur du serveur est constitué d'un SGBD en logiciel libre. Nous avions d'abord choisi MySQL [MySQL] car c'est un outil très répandu. Nous avons cependant renoncé à l'utiliser à cause d'incompatibilité avec la représentation des documents Unicode. Nous avons donc finalement choisi PostgreSQL [PostgreSQL].

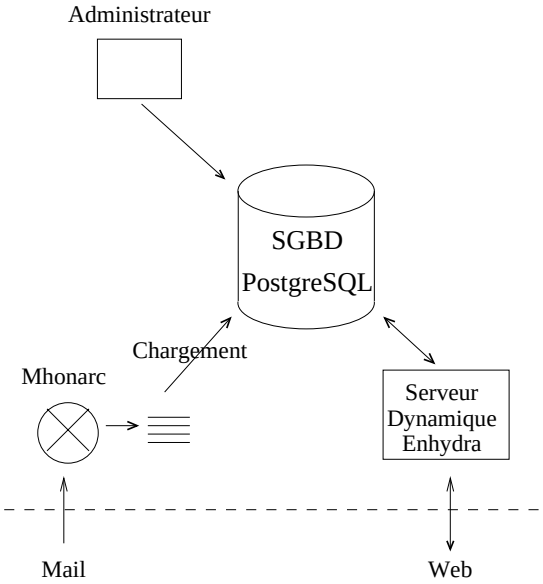


FIG. D.6 – *architecture du serveur Papillon*

Les listes de distribution de courrier électronique sont archivées et stockées dans la base de données du serveur Papillon. Les archives sont ensuite consultables directement via le Web. Pour gérer l'archivage des courriers, nous utilisons MHonArc [MHonArc]. Il a fallu modifier le code source pour pouvoir convertir tous les courriers en Unicode UTF-8 avant de les stocker dans la base.

Enfin, le serveur Papillon est accessible par le Web. Le serveur est une combinaison des serveurs Apache pour les objets statiques et Enhydra pour les objets dynamiques. Enhydra [Enhydra] est un serveur Web dynamique java disponible selon les termes d'une licence de logiciel libre (OpenSource).

4.2.2 Organisation de la base de données

Le choix d'une base de données relationnelle traditionnelle a été fait en attendant que se généralisent les bases de données XML intégrant des outils de manipulation comme DOM, XLink, XPointer et Xpath. Des projets sont en cours comme XML:DB [XML:DB], X-Hive [X-Hive] ou encore Tamino [Tamino].

Poids

identif.	Fréq1	fréq2	fréq3
Arau$1	5002	250	3000
Voir$1	3220	352	155

Historiques

ident	texte
Arau$1	aaa
Voir$1	aaa

Base lexicale

groups	users	Dicos	Volumes	éléments CDM	Articles Contributions Annotations
Tous	Tous	FeM	FeM	Meurtre	Texte
Admin	Guest	EDict	EDict	Arau	XML
Lexg	Root	DiCo	Dico−fra	Sentaku	XSL
Lexph	CB	Pap	Pap−fra	Verbe	
Guests	GS		Pap−jpn	Nom	
	MZ		Pap−eng	Adj	
	MM		Pap−tha		
	MT		Pap−lao		

Users

nom	login	pass	mail	credit
MM	matt	***	a@b	250
CB	xan	***	b@b	352

Dictionnaires

nom	orig	domain	hw	volumes
FeM	geta	gal	70k	FeM
Pap	G+N	gal	3	fra

FIG. D.7 – *tables de la base de données de Papillon*

La figure D.7 représente la structure des tables de la base de données de Papillon. La base est organisée pour l'instant en cinq tables : les données lexicales, les dictionnaires, les utilisateurs, les historiques et les poids.

La table des données lexicales est la table principale. Chaque unité de lexique (lexies, axies, articles) est stockée sous forme de texte XML. Les annotations et les contributions XSL sont aussi stockées dans cette table. La clé de chaque entrée est l'identificateur unique porté par l'élément. Ensuite, pour chaque élément, on stocke les index de ses éléments communs de l'ensemble CDM, son lexique et son dictionnaire. Si c'est une annotation ou une contribution, on stocke les utilisateurs et les groupes ayant l'autorisation de la voir.

La table des dictionnaires permet de stocker les informations contenues dans l'élément DML <dictionary>. La clé de chaque entrée est le nom du dictionnaire. On trouve entre autres les langues, le domaine, le type, le nombre de mots-vedette, les lexiques, etc.

La table des utilisateurs permet de stocker les informations contenues dans l'élément DML <user>. La clé de chaque entrée est le nom de l'utilisateur. On trouve entre autres le login, le mot de passe, l'adresse électronique, les crédits et les groupes auxquels appartient l'utilisateur et ses préférences stockées sous forme de feuilles de style XSL et CSS.

La table des historiques permet de stocker les historiques de tous les éléments DML ayant un attribut history. La clé de chaque entrée est l'identificateur de l'historique.

Pour l'instant, les poids sont stockés dans une table à part. Cette table reprend les principes du poids sur les éléments exposés en partie C. Cette méthode a cependant ses limites. En effet, il ne sera pas possible de stocker chaque poids si la base compte par exemple 300 000 éléments et 3 000 utilisateurs différents qui personnalisent ces poids. On arrive alors à un volume de données dépassant le téraoctet. Il faudra alors trouver un autre moyen de stockage comme des listes de poids pour chaque élément ou des matrices creuses.

4.2.3. Utilisation de la base lexicale

Les données linguistiques sont stockées dans la base de données sous forme de texte XML. Ces données sont accessibles selon plusieurs clés. Ces clés correspondent aux éléments communs de l'ensemble CDM qui se trouvent dans les données. La liste des éléments de cet ensemble est définie en partie C. Un article sera par exemple directement accessible selon le mot-vedette, la prononciation, la catégorie grammaticale, les traductions, les idiotismes. Sinon, les autres informations sont accessibles en parcourant directement le texte XML des articles.

Les annotations et les contributions sont triées puis appliquées selon l'ordre chronologique de leur date de création. Le mécanisme de la table des données linguistiques permet à chaque personne de voir la base lexicale avec la forme et les données qu'il souhaite. Chaque utilisateur peut voir les données auxquelles il a accès selon ses droits. Un contributeur peut visualiser en plus des données lexicales de la base, toutes ses annotations et contributions ainsi que celles de ses groupes.

Lorsqu'un utilisateur établit une requête, le serveur sélectionne la liste des éléments qui correspondent à la requête. Il applique ensuite par ordre chronologique de leur date de création les feuilles de style représentant ses contributions ainsi que celles des membres de ses groupes. Ensuite, les annotations correspondantes sont ajoutées. Enfin, les feuilles de style définissant les préférences de présentation sont appliquées. Le résultat final est ensuite envoyé à l'utilisateur.

Il peut arriver que certaines contributions ne soient plus valides car la base a été modifiée et les contributions ont été acceptées puis intégrées dans la base. Dans ce cas, le système envoie un message d'avertissement à l'utilisateur. D'autre part, lorsque des éléments fusionnent, les objets reliés aux identificateurs des deux éléments sont ensuite reliés à l'identificateur du nouvel élément résultant de la fusion.

4.3. Implémentation des interfaces

4.3.1. Consultation de la base

La consultation de la base peut s'effectuer directement avec un navigateur Web en suivant les principes décrits en partie C. L'interface est du type de celle de l'expérience DicoWeb décrite en partie B.

Les utilisateurs ont la possibilité de spécifier leurs préférences de visualisation grâce à une interface spécialisée. Ces préférences sont ensuite envoyées au serveur et stockées sous forme de feuilles de style XSL et CSS. Les utilisateurs peuvent aussi partager des feuilles de style entre groupes d'utilisateurs.

4.3.2. Contribution sur les articles monolingues

Les contributions sur les articles peuvent se faire de deux manières. Pour des contributions localisées et systématiques, une interface avec un formulaire HTML peut être créée. Par exemple, un linguiste travaillant sur la prononciation aura besoin d'une interface pour pouvoir entrer la prononciation de chaque mot-vedette.

Pour les autres contributions, l'utilisateur modifie l'article et le renvoie au serveur. Le système calcule les éléments modifiés par rapport à l'article initial. Il génère ensuite automatiquement une feuille de style XSL en indiquant les endroits modifiés sur l'article par des pointeurs XPointer (voir partie C section 3.2.2) et en y ajoutant le nom du contributeur et la date. La feuille de style est ensuite stockée dans l'espace virtuel du contributeur.

4.3.3. Contribution sur les liens interlingues

La contribution sur les liens interlingues est spécifique car elle ne demande pas de rédaction. Il faut simplement relier deux lexies de langues différentes entre elles. De plus, les contributions sur les liens interlingues sont faites par des personnes aux compétences différentes de celle des contributeurs sur les articles monolingues. Ce sont pour la plupart des traducteurs ou interprètes qui ont une bonne connaissance des deux langues qu'ils veulent relier.

Pour la création et la révision des liens interlingues entre lexies, une interface spécifique (voir figure D.8) a été programmée en Java en collaboration avec Magaly Drant. Cette interface possède trois fenêtres principales. La fenêtre de gauche permet d'afficher des lexies d'une langue, la fenêtre de droite, des lexies d'une autre langue et la fenêtre du milieu des axies représentant les liens entre ces deux langues.

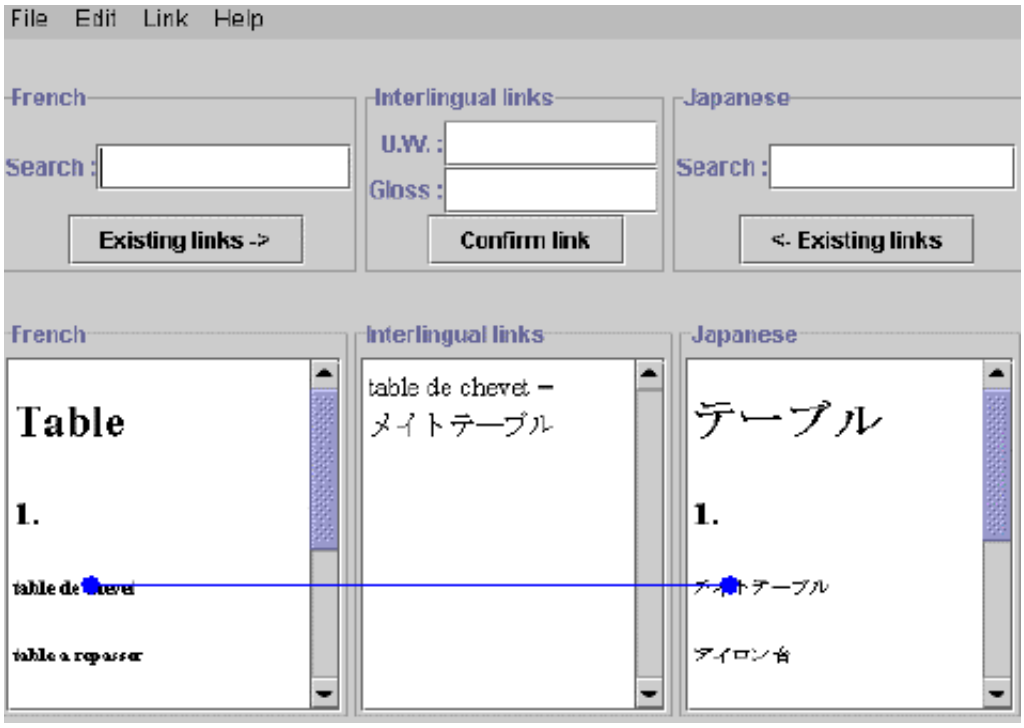


FIG. D.8 – *interface java permettant de créer des liens entre lexies*

Les liens entre lexies sont materialisés par un trait. L'utilisateur peut créer ou supprimer des liens directement à l'aide de la souris. Lorsque l'utilisateur crée un lien en traçant un trait bleu entre deux lexies, une

axie est générée automatiquement au milieu. Cet outil est encore à l'état de maquette. Il sera amélioré et mis à disposition des contributeurs sur le serveur de Papillon lorsque l'étape de révision des liens sera atteinte. Le visualisateur d'arbres hyperboliques étudié en partie B n'a pu être utilisé ici, car il ne permet pas d'éditer.

Cette interface est encore à l'état de maquette. Il faudrait encore l'améliorer pour la rendre plus fonctionnelle, en ajoutant par exemple la possibilité de visualiser et manipuler un contexte plus global que deux lexies (plusieurs vocables et plusieurs langues).

4.3.4. Pseudo-éditeur structuré

Les lexicographes rédigeant des articles monolingues peuvent souhaiter travailler chez eux en local sans connexion au réseau. Pour cela, nous générons des squelettes d'articles au format RTF depuis la base en suivant la méthode décrite en partie C. Si ces squelettes proviennent d'articles de la base à compléter, ces articles sont marqués pour éviter la duplication des efforts de rédaction. Les fichiers RTF sont ensuite envoyés avec un modèle de document contenant des macros d'aide à la rédaction aux lexicographes qui travaillent chez eux avec l'éditeur Word (voir figure D.9).

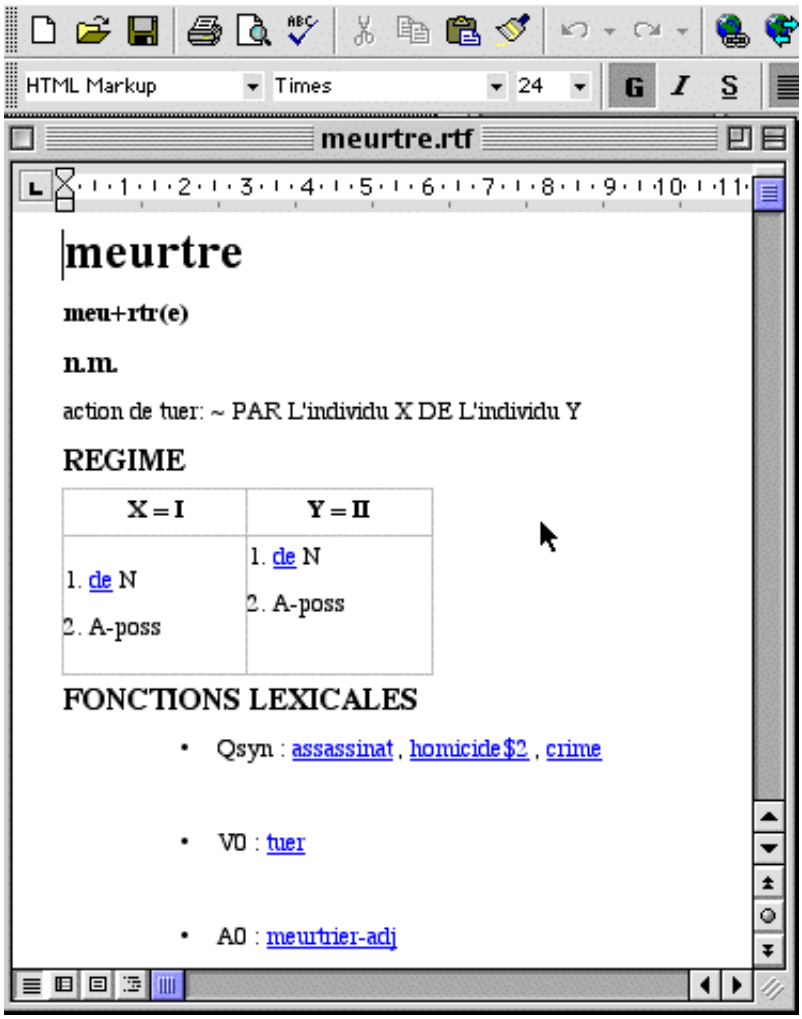


FIG. D.9 – *édition de la lexie MEURTRE avec Word*

Lorsque le lexicographe a fini de rédiger un fichier, il le renvoie à la base. Là le fichier est reconverti du format RTF vers le format original XML/DML, puis un spécialise lexicologue révise les articles avant de les intégrer dans la base. Il enlève ensuite les marques sur les articles intégrés.

4.3.5. Éditeur structuré

Nous proposons aussi aux lexicographes rédigeant des articles monolingues de travailler directement avec un éditeur structuré XHTML. Pour cela, nous transformons les squelettes d'articles du format XML vers le format XHTML au moyen d'un feuille de style XSLT [XSLT 1.0] en suivant la méthode décrite dans la partie C. Si ces squelettes proviennent d'articles de la base à compléter, ces articles sont marqués pour éviter la duplication des efforts de rédaction. Les fichiers XHTML sont ensuite envoyés aux lexicographes qui travaillent chez eux (voir figure D.10).

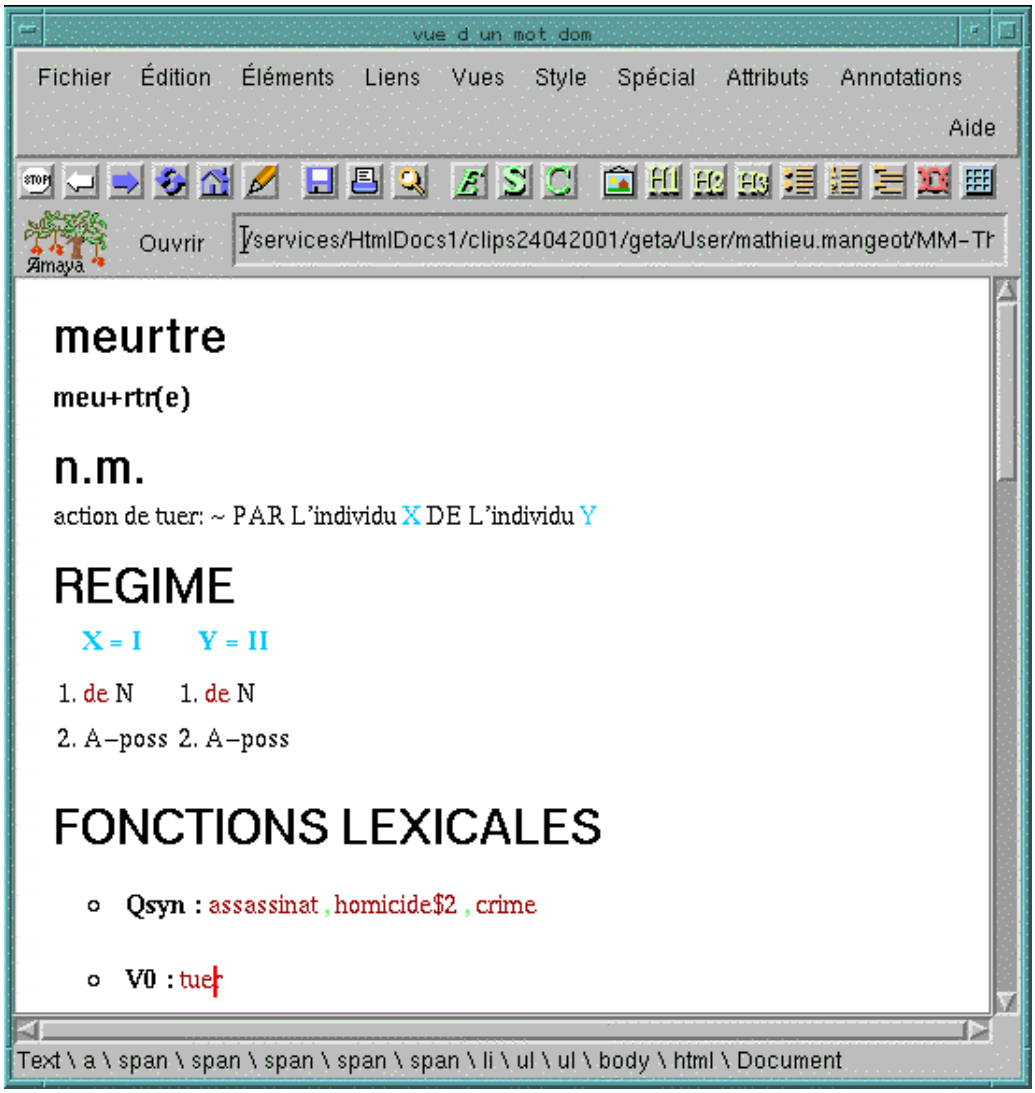


FIG. D.10 – *édition de la lexie MEURTRE avec Amaya*

Lorsque le lexicographe a fini de rédiger un fichier, il le renvoie à la base. Le fichier est ensuite reconverti

du format XHTML vers le format original XML/DML au moyen d'une autre feuille de style XSLT en suivant la méthode décrite en partie C. Puis un spécialise lexicologue révise les articles avant de les intégrer dans la base. Il enlève ensuite les marques sur les articles intégrés.

Les lexicographes utilisent un éditeur structuré XHTML pour travailler. Ils peuvent par exemple utiliser Amaya [Amaya]. Cependant, cet éditeur comporte une restriction importante. En effet, même s'il est possible de travailler avec des documents encodés en UTF-8, les caractères n'appartenant pas à la norme ISO-8859-1 ne sont pas affichés. Il n'est donc pas encore possible d'utiliser Amaya pour éditer un article japonais par exemple. L'équipe de développement du logiciel Amaya travaille actuellement sur cette limitation.

4.3.6 Interfaces pour les spécialistes lexicologues

Le groupe des spécialistes lexicologues a accès à toutes les données de la base et en particulier aux axies qui sont cachées lors de la consultation. Pour mener à bien leur travail de vérification, ils peuvent lancer des requêtes sur tout le contenu de la base et en extraire des statistiques sur les données (langues de la base, lexies, axies, contributeurs, etc.).

La figure D.11 montre une interface permettant d'effectuer des statistiques sur les lexies françaises. Dans cet exemple, le lexicologue a demandé toutes les lexies ayant la chaîne "nom" contenue dans la catégorie grammaticale.

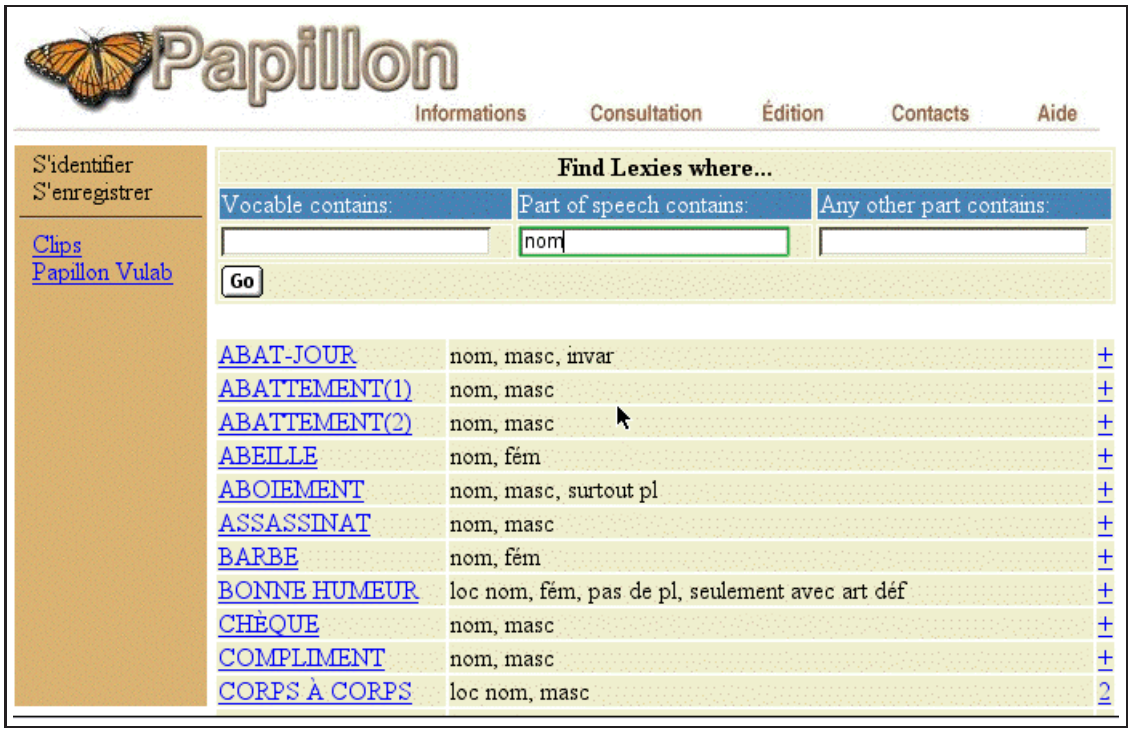


FIG. D.11 – *requête sur la base Papillon*

Les lexicologues ont aussi besoin de vérifier la cohérence et la complétude de la base afin de détecter des erreurs éventuelles ou de préparer un tableau de choses à faire. Pour cela, ils rédigent des scripts de vérification de cohérence et les envoient au serveur Papillon qui les exécute ensuite en tâche de fond.

5. Évaluations préliminaires et exemples

5.1. Récupération du FeM

5.1.1. Exemple d'article après récupération

Le dictionnaire FeM [FeM] a été récupéré du format original au format LISPO selon la méthode RÉCUPDIC par Haï Doan-Nguyen durant sa thèse [Doan-Nguyen98a]. Nous avons converti le résultat du format LISPO vers XML avec le programme lisp donné en partie C. Voici un extrait de l'article `abandonner` après cette conversion :

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

```
    </BODY>
  </HFEM>
```

Cet article est réparti automatiquement en lexies et axies qui sont ensuite intégrées dans la base lexicale (voir figure D.12). La répartition se fait selon le script décrit dans la section 3.3. de cette partie.

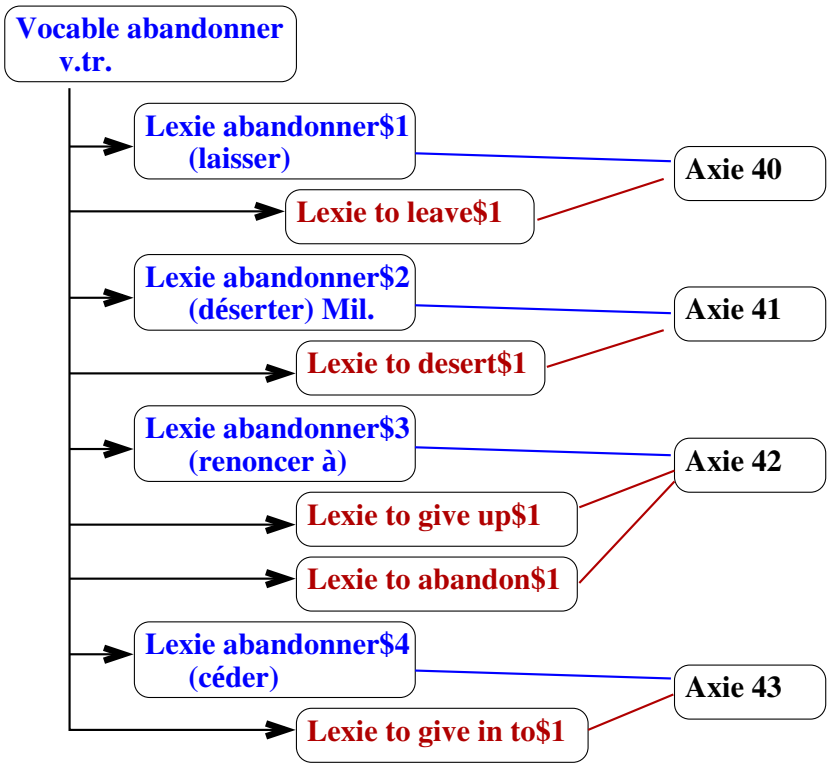


FIG. D.12 – *répartition d'un article du FeM en lexies et axies*

5.1.2. Lexies françaises provenant de cet article

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

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

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

5.1.3. Lexies anglaises provenant du même article

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

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

5.1.4. Axies provenant du même article

Le traitement de l'article du FeM a généré automatiquement cinq axies reliant chacune une lexie française et une lexie anglaise. Voici en exemple l'axie `a5` reliant les deux lexies prédécentes.

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

5.2. Récupération de JMDict

Le dictionnaire JMDict est déjà encodé en XML. Nous pouvons donc l'utiliser directement pour l'intégration dans la base lexicale. La répartition en lexies et axies se fait selon le schéma de la figure D.13.

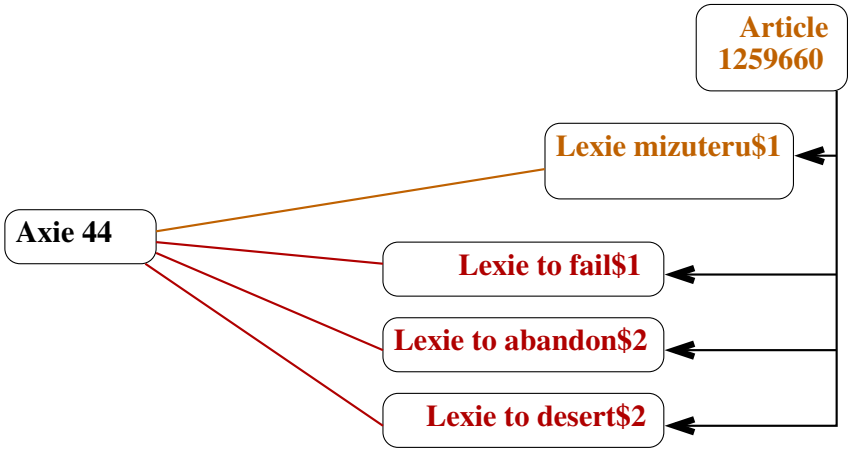


FIG. D.13 – *répartition d'un article de JMDict en lexies et axies*

5.2.1. Exemple d'article

Voici un exemple d'article de ce dictionnaire. L'élément <ent_seq> est un identificateur unique de l'article. L'élément <k_ele> regroupe les informations concernant l'écriture en kanji du mot-vedette. L'élément <r_ele> regroupe les élément concernant l'écriture en kana du mot-vedette.

```
<entry>
```

```
    <ent_seq>1259660</ent_seq>
    <k_ele>
      <keb> 見捨てる </keb>
      <ke_pri>ichi1</ke_pri>
      <ke_pri>jdd1</ke_pri>
    </k_ele>
    <r_ele>
      <reb> みすてる </reb>
      <re_pri>ichi1</re_pri>
      <re_pri>jdd1</re_pri>
    </r_ele>
    <sense>
      <gloss>to abandon</gloss>
      <gloss>to fail</gloss>
      <gloss>to desert</gloss>
    </sense>
  </entry>
```

5.2.2. Lexie japonaise provenant de l'article

Cet article est réparti en une lexie japonaise :

```
<lexie id="mizuteru$1" basic="yes">
  <headword> 見捨てる </headword>
  <kun-yomi> みすてる </kun-yomi>
  <jmdict-data>
    <ent_seq>1259660</ent_seq>
    <ke_pri>ichi1</ke_pri>
    <ke_pri>jdd1</ke_pri>
    <re_pri>ichi1</re_pri>
    <re_pri>jdd1</re_pri>
  </jmdict-data>
  <axies>
    <refaxie href="a44"/>
  </axies>
</lexie>
```

Cette lexie est reliée à l'axie a44.

5.2.3. Lexies anglaises provenant de l'article

L'article génère trois lexies anglaises dont le mot-vedette est une traduction anglaise contenue dans l'article. L'identificateur de la lexie suivante porte le numéro 2 car une précédente lexie a déjà été créée avec ce mot-vedette lors de la récupération du FeM.

```
<lexie id="to abandon$2" basic="no">
  <headword>to abandon</headword>
  <axies>
    <refaxie href="a44"/>
  </axies>
```

```
</lexie>
```

Cette lexie est aussi reliée à l'axie `a44`.

5.2.4. Axies provenant de l'article

L'article génère une axie reliant la lexie japonaise et les lexies anglaises. Voici l'axie `a44` reliant les deux lexies prédécentes :

```
<axie id="a44">
 <eng>
  <reflexie href="to fail$1"/>
  <reflexie href="to abandon$2"/>
  <reflexie href="to desert$2"/>
 </eng>
 <jpn>
  <reflexie href="mizuteru$1"/>
 </jpn>
</axie>
```

5.3. Fusion éventuelle de lexies anglaises

Le travail automatique est terminé. Le contenu de la base est maintenant révisé par des spécialistes lexicologues qui décident de fusionner ou de séparer des lexies ou des axies. Dans la suite, nous imaginons qu'un linguiste spécialiste de l'anglais décide de fusionner les deux lexies anglaises `to desert$1` et `to desert$2` d'une part, puis les deux lexies anglaises `to abandon$1` et `to abandon$2` d'autre part (figure D.14).

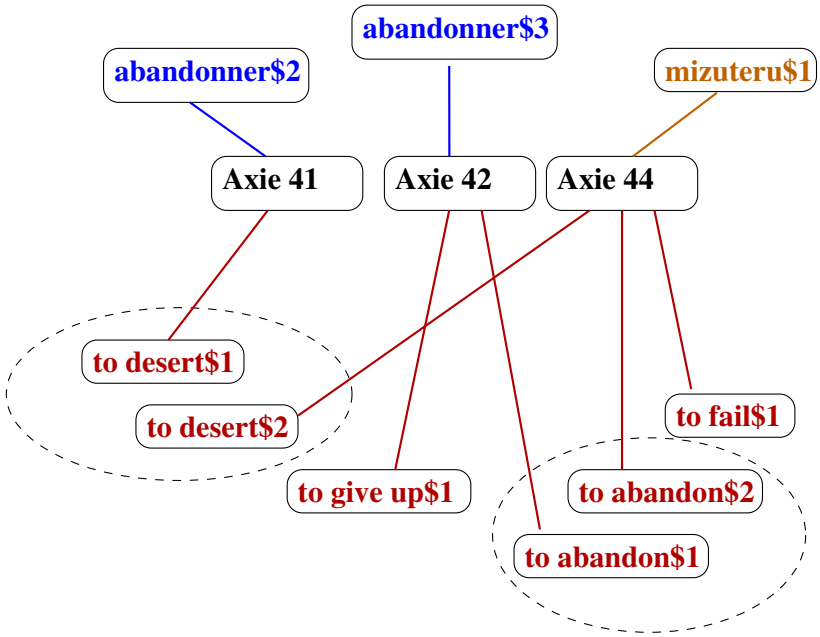


FIG. D.14 – *fusion manuelle de certaines lexies anglaises*

5.3.1. Lexies après fusion

Dans l'exemple suivant, le linguiste a décidé de fusionner les deux lexies anglaises `to abandon$1` et `to abandon$2` ayant le même mot-vedette "to abandon". La lexie `to abandon$2` est supprimée mais son id ne sera jamais réaffecté.

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

5.3.2. Axies après fusion

Maintenant, le linguiste doit gérer les anciennes axies qui étaient reliées aux lexies qu'il vient de fusionner. Il se trouve alors dans la situation de la figure D.15. Pour obtenir une configuration normale, il faut résoudre les conflits générés par la fusion des lexies anglaises, à savoir qu'une lexie ne peut pointer vers deux axies différentes. Dans la figure D.15, `to desert$1` pointe sur les axies a41 et a42, et `to abandon$1` pointe sur les axies a42 et a44.

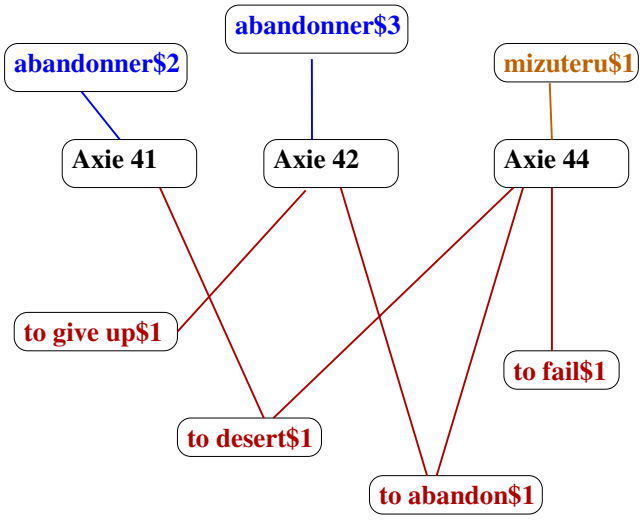


FIG. D.15 – *axies après fusion manuelle de certaines lexies anglaises*

S'il n'a aucune information sur les autres langues, le lexicologue ajoutera une axie intermédiaire pointant sur chaque lexie fusionnée (figure D.16).

Par contre, si ses connaissances lui permettent de décider que deux axies peuvent être reliées par des liens de raffinement, il modifie les axies en ajoutant ces liens entre les axies existantes (voir figure D.17).

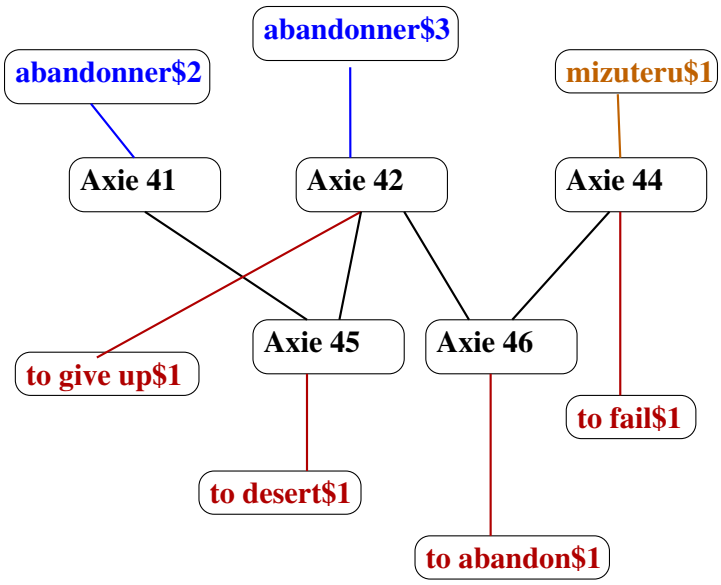


FIG. D.16 – *ajout d'axies intermédiaires*

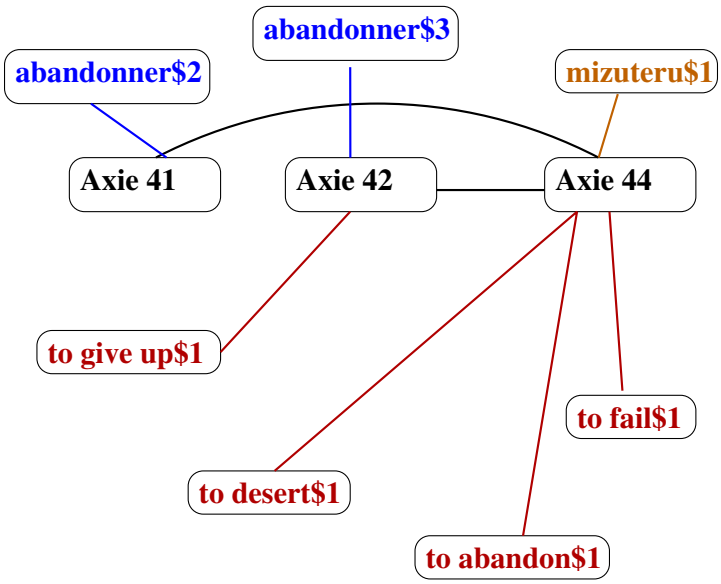


FIG. D.17 – *ajout de liens de raffinement entre axies*

Conclusion

Nous avons présenté dans cette thèse un environnement centralisé et distribué de récupération, manipulation, construction et consultation de ressources lexicales hétérogènes et multilingues. Cet environnement répond aux problèmes complexes de structuration et manipulation de données hétérogènes, de visualisation d'une grande quantité de données et de construction en collecticiel par des personnes aux compétences diverses contrôlée par un groupe central de lexicologues.

Nous avons d'abord résolu séparément ces problèmes grâce à des expérimentations variées sur la consultation de ressources hétérogènes, l'enrichissement et personnalisation du résultat ainsi que la construction de ressources.

Notre environnement répond à l'ensemble de ces problèmes en ajoutant un niveau d'abstraction qui domine les bases de données utilisées pour le stockage et en intégrant un serveur pour la construction coopérative. Son noyau inclut un formalisme générique de définition des structures. Il permet de concevoir une véritable "plate-forme lexicale" générique et extensible.

Nous avons appliqué cet environnement au projet Papillon de développement par des bénévoles sur Internet d'une base lexicale multilingue dont l'architecture est constituée d'un dictionnaire monolingue de sens (lexies) pour chaque langue et d'un dictionnaire pivot d'acceptions interlingues (axie) reliant les articles monolingues. Les expériences préliminaires ont été concluantes.

Principes dégagés devant ce travail

Au cours de notre travail, plusieurs principes se sont dégagés, et ont été tantôt affinés, tantôt généralisés, mais toujours expérimentés et validés. Nous proposons ci-dessous une liste des "dix commandements" de la construction d'une base de données lexicales idéale.

Principes de structuration logique

1) Le *principe d'exhaustivité* reprend le principe d'œcuménisme provient de la thèse de Gilles Sérasset. Il s'agit de la volonté d'accueillir dans une base lexicale toutes les théories linguistiques, et en particulier celles relatives au niveau lexical, grâce à un formalisme générique permettant de représenter un grand nombre de structures de dictionnaires sans imposer leur conversion en une seule structure particulière. Cela autorise l'utilisation de données provenant de théories linguistiques différentes. Nous avons expérimenté ce principe avec DicoWeb où nous utilisons des données ayant des structures très différentes. En partie C, DML est basé sur ce principe puisqu'il reprend le système SUBLIM de la thèse de Gilles Sérasset. Ce principe est observé dans le projet Papillon avec d'une part l'utilisation d'une structure complexe (celle de DiCo) et d'autre part, la possibilité de référencer des données externes au projet comme les UW du projet UNL, les catégories sémantiques du dictionnaire de NTT, les synonymes du projet WordNet, etc.

2) Le *principe d'abstraction du niveau données* consiste à différencier le niveau de stockage des informations du niveau de manipulation. Nous avons expérimenté ce principe avec la maquette DicoWeb dans laquelle nous utilisons des ressources stockées directement sous forme de fichiers texte et d'autres provenant de serveurs Web distants. en partie C, nous spécifions ce principe avec l'API de fournisseur de ressources. Ensuite, dans le projet Papillon, nous réalisons ce principe avec l'utilisation d'un SGBD pour le stockage et de programmes en DOM pour la manipulation.

Principes liés à l'aspect collaboratif

3) Le *principe de mutualisation* consiste à mettre en commun les ressources lexicales apportées par chaque utilisateur de la base. Ce principe est spécifié en partie C avec le système de crédit de points accordé pour chaque contribution à la base. Ce principe est réalisé dans le projet Papillon dès le départ avec la

récupération de ressources provenant d'horizons divers : le dictionnaire JMDict de Jim Breen, la base DiCo d'Alain Polguère, le FeM du GETA et les données du projet SAIKAM.

4) Le *principe de consultation gratuite* consiste à toujours laisser la possibilité au public de consulter la base gratuitement. Ce principe a été observé avec le premier serveur du FeM wAlex construit par Mathieu Lafourcade et ensuite expérimenté avec les maquettes DicoWeb, DicoSzótár, DicoFeJ, Nihongo et le FeM. Il est spécifié en partie C et réalisé dans le projet Papillon.

5) Le *principe de personnalisation général* consiste à laisser à chaque utilisateur de la base lexicale la possibilité de personnaliser les requêtes, les résultats, les propositions de travail de la base, etc. Le résultat des requêtes est personnalisé principalement par l'utilisation de feuilles de style. Ce principe a été expérimenté en premier lieu avec la nouvelle maquette du serveur du FeM qui permet de configurer le résultat à la volée. En partie C, nous avons proposé d'implémenter ce principe en créant un espace virtuel pour chaque utilisateur, où il peut stocker ses feuilles de style et en laissant la possibilité d'annoter les informations de la base. Ce principe a été réalisé dans le projet Papillon avec l'utilisation de profils d'utilisateurs ainsi que des préférences personnalisées, modifiables strictement via des interfaces appropriées, et si possible évoluant automatiquement par suivi et apprentissage du système.

Principes liés aux données

6) Le *principe d'héritage* s'applique de façon variée. En ce qui concerne les groupes d'utilisateurs, il consiste à utiliser une hiérarchie de groupes d'utilisateurs qui héritent de plusieurs propriétés comme les feuilles de style, les droits d'accès, les poids. Ces propriétés sont définies une seule fois pour l'univers des utilisateurs. Par défaut chaque groupe et chaque utilisateur hérite de ces propriétés. Chacun peut ensuite définir au niveau d'un groupe ou d'un utilisateur d'autres propriétés qui seront à leur tour héritées. Ce principe est spécifié en partie C et réalisé dans le projet Papillon à différents endroits, là aussi par les groupes d'utilisateurs, les poids, les définitions des schémas, etc.

7) Le *principe de traçabilité* consiste à noter tous les changements effectués sur les informations lexicales et être capable de tracer tous les changements successifs subis par ces informations depuis leur création ou leur importation dans la base. Ce principe a été expérimenté en partie B dans les maquettes DicoSzótár et Nihongo dans lesquelles nous notons des informations de gestion pour chaque ajout d'information dans ces dictionnaires. Nous avons ensuite généralisé et spécifié ce principe en l'appliquant à toutes les informations de la base lexicale en partie C. Pour cela, nous utilisons les attributs DML history-ref et history ainsi que des fichiers d'historique. Nous appliquons ce principe au projet Papillon en créant une table dans la base de données réservée aux historiques des modifications.

8) Le *principe de protection des données* communes consiste à n'intégrer dans la base commune que des données révisées par des spécialistes. Ce principe vient d'une constatation réalisée sur des projets comme SAIKAM. En effet, lorsque beaucoup de contributeurs apportent des données nouvelles directement dans la base, celle-ci se retrouve, même sans mauvaise intention, polluée par des contributions erronées. Il est très difficile ensuite de les corriger. De ce fait, la base n'est jamais dans un état stable. En partie C, nous avons donc spécifié que les contributions sont d'abord stockées dans l'espace virtuel du contributeur puis sont révisées par des spécialistes avant d'être intégrées à la base. En partie D, nous observons ce principe dans le projet Papillon.

Principes de mise en œuvre

9) Le *principe de récupération totale* intervient lors de la récupération d'une ressource lexicale. Il consiste à récupérer toutes les informations de la ressource de façon à pouvoir regénérer cette ressource à partir de la forme récupérée. Nous spécifions ce principe en partie C et l'appliquons à la récupération des

dictionnaires FeM, JMDict et DiCo dans le projet Papillon de façon à pouvoir regénérer des dictionnaires à partir de la base dans ces formats.

10) Le *principe de réciprocité* consiste à considérer que la base devrait échanger des informations avec des programmes partenaires. Ce principe a été expérimenté dans les maquettes DicoWeb avec les lemmatiseurs et DicoSzótár avec les conjugueurs. Nous avons spécifié ce principe en partie C avec les API de fournisseurs de services et de ressources. Le projet Papillon implémente ces API.

Problèmes complexes restant à résoudre

L'analyse et la mise au point d'un environnement de création, manipulation et consultation de ressources lexicales a fait surgir des problèmes complexes restant à résoudre. Ces problèmes appartiennent à des domaines variés de l'informatique.

Le stockage et le calcul des poids ne posent pas de problèmes lorsqu'on a peu de données et peu d'utilisateurs. Par contre, nous pensons arriver au bout de peu de temps à plus de 100 000 articles dans la base. Envisageons que ces articles soient reliés par environ 300 000 liens et que la base comporte plus de 3 000 utilisateurs. Si un poids différent est associé à chaque élément, cela représente plus d'un téraoctet de données à stocker. Une base de données ordinaire ne peut gérer cette taille. Il faut alors imaginer un autre moyen pour stocker les poids comme par exemple instaurer un système d'héritage de poids entre les groupes et les stocker sous forme de listes ou de matrices creuses ou encore utiliser les techniques de compression de séquences d'images.

Le problème de *calcul automatique de profils d'utilisateurs* est important pour savoir qui contribue à quoi dans la base, avec quelle fréquence, et quelle qualité de contribution. Les profils sont utiles pour établir des statistiques, optimiser la répartition du travail à faire, accorder un degré de confiance aux contributeurs, etc.

Le problème de *gestion de charge importante sur un serveur* est provoqué par des connexions simultanées multiples, des téléchargements très fréquents, une activité continue (connexions depuis le Japon ou le Canada, etc.), et des opérations en tâche de fond et une sauvegarde tous les jours.

Le problème de la *gestion des conflits et de la synchronisation sur les annotations et les contributions* survient lorsqu'un article est supprimé de la base ou que deux articles sont fusionnés. Que deviennent alors les annotations et les contributions associées à ces articles ? Lorsqu'une contribution est acceptée, que deviennent les annotations et les autres contributions faites sur cette contribution ?

Perspectives de recherche

Nous n'avons pas encore pu tester notre environnement pour la construction de nouvelles ressources dans des conditions réelles d'utilisation permettant de mettre au point et de vérifier l'utilisabilité de nos outils. Le projet Papillon lancé en collaboration entre le GETA-CLIPS, le National Institute of Informatics de Tokyo au Japon, et de nombreux autres partenaires, nous permettra de tester notre environnement pour la construction de dictionnaires multilingues avec entre autres le français, le japonais, le thaï, le lao et le vietnamien. Nous prévoyons d'ajouter à court terme le malais, puis le coréen et le chinois.

Un financement post-doctoral de la JSPS (Japanese Society for the Promotion of Science) nous a été accordé pour travailler deux ans sur le projet Papillon au NII à Tokyo. Nous mettrons en place un serveur qui implémente notre environnement de création de nouvelles ressources et réaliserons ensuite les tâches d'administration nécessitées par un tel serveur.

Nous prévoyons de mettre en place les différentes interfaces pour la consultation et la personnalisation du résultat des données et aussi des interfaces pour préparer le travail des lexicologues sur la vérification

et le contrôle des données. Nous testerons nos différentes méthodes de construction de dictionnaires auprès des contributeurs bénévoles.

Enfin, le cadre de ce projet nous permettra de nous attaquer aux problèmes restant à résoudre : stockage des poids, calcul automatique des profils, et gestion des conflits sur les contributions.

Bibliographie

[Aarts85] J. Aarts & T. V. D. Heuvel (1985) *Computational Tools for the Syntactic Analysis of Corpora.* Linguistics, 23/1, pp. 303-335.

[Adriaens90] G. Adriaens & M. Lemmens (1990) *The Self Extending Lexicon : Off-line and On-line Defaulting of Lexical Information in the METAL Machine Translation System.* Proc. Coling-90, Helsinki, 20-25 August 1990, H. Karlgren ed. vol. 3/3: pp. 305-307.

[Aho86] A. Aho, R. Sethi & J. Ullman (1986) *COMPILATEURS Principes, techniques et outils.* ed. Intereditions, Paris, 875 p.

[Ampornaramveth98] Vutichai Ampornaramveth (1998) *SAIKAM: An online dictionary development project.* Proc. of the 4th Workshop on Academic Information Networks and Systems, février 98, NACSIS seminar house, Karuizawa, Japon.

[Ampornaramveth00] Vutichai Ampornaramveth, Akiko Aizawa, Keizo Oyama & Tanasee Methapisit (2000) *An Internet-Based Collaborative Dictionary Development Project: SAIKAM.* First International Symposium on Advanced Informatics, Proc. AdInfo 2000, 9-10 mars 2000, NACSIS, Tokyo, Japon, 4 p.

[Antoine92] F. Antoine (1992) *Dictionnaire(s) mode(s) d'emploi.* La maison du dictionnaire, 120 p.

[Atkins92] B. T. Sue Atkins (1992) *Tools for computer-aided corpus lexicography: the Hector Project.* Proc. COMPLEX'92, Conference on Computational Lexicography and text research, Budapest, Hongrie, Linguistics Institute, Hungarian Academy of Sciences, pp. 3-59.

[Atkins94] B. T. Sue Atkins & Antonio Zampolli (1994) *Computational Approaches to the Lexicon.*Oxford University Press, 480 p.

[Bauer94] Daniel Bauer, Frédérique Segond & Annie Zaenen (1994) *Enriching a SGML-tagged bilingual dictionary for machine-aided comprehension.* Technical Report Xerox Research Center Europe, 21 p.

[Bachut84a] Daniel Bachut (1984) *ATLAS, manuel d'utilisation.* GETA, rapport interne, 37 p.

[Bachut84b] Daniel Bachut & Nelson Verastegui (1984) *Software tools for the environment of a computer aided translation system.* Proc. COLING-84, Stanford, GETA, 4 p.

[Blanc96] Etienne Blanc (1996) *Une maquette de base lexicale multilingue à pivot lexical: PARAX.* Lexicomatique et Dictionnairique, Actes du colloque LTT, Lyon septembre 1995, ed. AUPELF-UREF, Montréal, Canada, pp. 43-58.

[Blanc99] Etienne Blanc (1999) *PARAX-UNL: a Large Scale Hypertextual Multilingual Lexical Database.* Proceedings 5th Natural Language Processing Pacific Rim Symposium 1999, Tsinghua University Press, Beijing, 1999, pp. 507-510.

[Boguraev89] Brian Boguraev et al. (1989) *Computational lexicography for natural language processing.* Brian Boguraev & Ted Briscoe, ed., Longman, Londres & New York, 310 p.

[Boitet82a] Christian Boitet (1982) *Le point sur ARIANE-78 début 82 (DSE 1).* GETA-CHAMPOLLION, CAP SOGETI FRANCE, 252 p.

[Boitet88] Christian Boitet (1988) *Hybrid Pivots using m-structures for multilingual Transfer- based systems.* Japanese Institute of Electronic Information and Communication Engineering, NLC, 88/3: pp. 17-22.

[Boitet90] Christian Boitet (1990) *Towards Personnal MT: general design, dialogue structure, potential role of speech.* Proc. Coling-90, Helsinki, 20-25 August 1990, H. Karlgren ed., vol. 3/3: pp. 30-35.

[Boitet93a] Christian Boitet (1993a) *Crucial open problems in Machine Translation &Interpretation.* Proc. BKK'93, Bangkok, Thailand, 17-20 March 1993 vol. 1/1.

[Boitet93b] Christian Boitet (1993b) *La TAO comme technologie scientifique : le cas de la traduction automatique fondée sur le dialogue.* In "La traductique", P. Bouillon & A. Clas ed., Les Presses de l'Université de Montréal (PUdM), AUPELF/UREF: pp. 109- 148.

[Boitet93c] Christian Boitet (1993c) *TA et TAO à Grenoble : 32 ans déjà !* TAL (revue semestrielle de l'ATALA), 33/1/2, Spécial Trentenaire: pp. 45-84.

[Boitet95a] Christian Boitet (1995a) *Factors for success (and failure) in Machine Translation - some lessons of the first 50 years of R&D.* 5th Machine Translation Summit, Luxemburg, 1995, 18 p.

[Boitet95b] Christian Boitet (1995b) *Machine-Aided Human Translation.* Sections 8.3 & 8.4, "Survey of the State of the Art in Human Language Technology", A. Cole & al eds, 1995, pp. 288-294.

[Boitet97] Christian Boitet (1997) *GETA's methodology and its current development towards personal networking communication and speech translation in the context of the UNL and C-STAR projects.* Proc. PACLING, Ohme, Tokyo, Japon, PACLING, vol. 1/1, pp. 23-57.

[Boitet98] Christian Boitet, Etienne Blanc, Mathieu Mangeot-Lerebours, Pierre Guillaume, Nicolas Nédeau, Mutsuko Tomokiyo & Jerzy Sitko (1998) *Processing of French in the UNL Project (Year 1).* Final Report, The United Nations University and Université Joseph Fourier, Grenoble, mars 1998, 216 p.

[Boitet82b] Christian Boitet, Pierre Guillaume & Maurice Quezel-Ambrunaz (1982) *ARIANE-78: an integrated environment for automatic translation and human revision.* Proc. COLING-82, Prague, July 1982: pp. 19-27.

[Boitet82c] Christian Boitet & Nicolas Nedobejkine (1982) *Base lexicale : organisation générale et indexage.*Rapport final, projet ESOPE ADI, partie D, GETA, Grenoble, 1982, 30 p.

[Boitet86a] Christian Boitet & Nicolas Nedobejkine (1986a) *Toward Integrated Dictionary for M(A)T : Motivations and Linguistic Organisation.* Proc. COLING-86, Bonn, 25- 29 août 1986 vol. 1/1: pp. 423-428.

[Boitet86b] Christian Boitet & Nicolas Nedobejkine (1986b) *Vers une base lexicale intégrée pour la T(a)O : motivations et organisation linguistique.* Proc. Journées francophones de l'informatique, bases de données et bases de connaissances, Grenoble, janvier 1986, vol. 1/1: pp. 151-169.

[Boitet94] Christian Boitet & Marc Seligman (1994) *The "Whiteboard" Architecture: A Way to Integrate Heterogeneous Components of NLP Systems.* Proc. COLING-94, Kyoto, Japan, 5-9 August 1994, M. Nagao ed. vol. 1/2: pp. 426-430.

[Boufaida98] Mahmoud Boufaïda & Zizette Boufriche-Boufaïda (1998) *On extending a Semantic Data Model with Some Aspects of Rules and Objects.* Proceedings of the 5th KRDB Workshop, Seattle, WA, 7 p.

[Bonhomme98] Stéphane Bonhomme (1998) *Transformation de documents structurés, une combinaison des approches explicites et automatique.* Thèse de nouveau doctorat, Spécialité Informatique, Université Joseph Fourier Grenoble 1, 212 p.

[Briscoe93] Ted Briscoe, Valeria De Paiva & Anne Coperstake (1993) *Inheritance, Defaults and the Lexicon.*Cambridge University Press, Cambridge, 298 p.

[Buseman96] A. Buseman et al. (1996) *The Linguist's Shoebox.* Summer Institute of Linguistics, 111 p.

[Byrd87] R. J. Byrd et al. (1987) *Tools and Methods for Computational Lexicology.* Journal of Computational Linguistics, 13/3-4, pp. 219-240.

[Calzolari90] Nicoletta Calzolari (1990) *Acquisition of Lexical Information from a Large Textual Italian Corpus.* Proc. COLING-90, Helsinki, H. Karlgren ed., vol. 3/3: pp. 54-59.

[Calzolari89b] Nicoletta Calzolari & R. Bindi (1989) *Lexical Databases and Textual Corpora : Perspectives of Integration for a Lexical Knowledge Base.* Technical Report, Universita di Pisa, Dipartimento di linguistica, 1989, 6 p.

[Calzolari88] Nicoletta Calzolari &Eugenio Picchi (1988) *Acquisition of Semantic Information from an Online Dictionary.* Proc. COLING-88, Budapest, 22-27 August 1988, D. Várgha ed.: pp. 87-92.

[Calzolari94] Nicoletta Calzolari &Eugenio Picchi (1994) *A Lexical Workstation : From Textual Data to Structured Database.* Computational Approaches to the Lexicon, Atkins, B. T. Sue & Zampolli, Antonio ed., Oxford University Press, 480 p.

[Church94] K. W. Church (1994) *Unix (TM) for Poets.* Proc. ELSNET, European Summer School, Utrecht, Pays Bas, 53 p.

[Connoly97] Dan Connolly (1997) *XML Principles, Tools and Techniques* World Wide Web Journal, Volume 2, Issue 4, Fall 1997, O'REILLY & Associates, 250 p.

[Correard94] Marie-Hélène Corréard &Valérie Grundy (1994) *Le dictionnaire Hachette-Oxford.* Oxford University Press & Hachette, 1950 p.

[Correard99] Marie-Hélène Corréard &Mathieu Mangeot-Lerebours (1999) *XML- A Solution For LDBs, Eds and MRDs?* Proc. COMPLEX'99, Pécs, Hongrie, vol. 1/1, 6 p.

[Coutaz88] Joëlle Coutaz (1988) *Interface Homme-ordinateur : Conception et Réalisation.* Thèse d'État, Université Joseph Fourier, 402 p.

[Cunningham96] H. Cunningham, R. J. Gaizauskas & Yorick Wilks (1996) *GATE: A General Architecture for Text Engineering.* ILASH & DCS, University of Sheffield, Royaume Uni, décembre 95, 53 p.

[Cunningham97] H. Cunningham et al. (1997) *Interface Homme-ordinateur : Conception et Réalisation.*DCS, University of Sheffield, Royaume Uni, 10 février 1997, 9 p.

[Curbow95] D. Curbow & E. Dykstra-Erickson (1995) *The OpenDoc User Experience.* MacTech, Volume 22, juin 1995, pp. 83-97.

[Descotte00a] Sylvianne Descotte, Jean-Luc Husson, Laurent Romary, Marc Van Campenhoudt & Nadia Viscogliosi (2000) *Dhydro : a generic environment developed to edit and access multilingual terminological data on the Internet.* 2e Ccnférence internationale sur la terminologie maritime, Turku, Finlande, mai 2000, 11 p.

[Descotte00b] Sylvianne Descotte, Jean-Luc Husson, Laurent Romary, Marc Van Campenhoudt & Nadia Viscogliosi (2000) *From specialised lexicography to conceptual databases : which format for a multilingual maritime dictionary ?* 2e conférence internationale sur la terminologie maritime, Turku, Finlande, mai 2000, 17 p.

[Doan-Nguyen96a] Haï Doan-Nguyen (1996a) *Transformations in Dictionary Resources Accumulation - Towards a Generic Approach.* Papers in Computational Lexicography, COMPLEX'96, Linguistics Institute, Hungarian Academy of Sciences, Budapest, Hongrie, 1996, pp. 29-38.

[Doan-Nguyen96b] Haï Doan-Nguyen (1996b) *Towards a Generic Approach to the Problem of Dictionary Resources Accumulation.* Informatique et Langue Naturelle, ILN'96, Nantes, 1996, pp. 209-218.

[Doan-Nguyen98a] Haï Doan-Nguyen (1998a) *Techniques génériques d'accumulation d'ensembles lexicaux structurés à partir de ressources dictionnairiques informatisées multilingues hétérogènes.* Thèse de nouveau doctorat, Spécialité Informatique, Institut National Polytechnique de Grenoble, 168 p.

[Doan-Nguyen98b] Haï Doan-Nguyen (1998b) *Accumulation of Lexical Sets: Acquisition of Dictionary Resources and Production of New Lexical Sets.* 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, Proc. COLING-ACL'98, vol 1/1, Montréal, Canada, 10-14 août 1998, pp 330-335.

[Dolan96] William B. Dolan & Stephen D. Richardson (1996) *Interactive Lexical Priming for Disambiguation.* Proc. MIDDIM'96, Post-COLING seminar on Interactive Disambiguation, C. Boitet ed. Le Col de Porte, Isère, France. 12-14 août 1996. vol. 1/1 : pp. 54-56.

[Dutoit92] Dominique Dutoit (1992) *A Set-Theoretic Approach to Lexical Semantics.* Proc. COLING-92, C. Boitet ed., Nantes, France, 18-21 juillet 1992, pp. 982-987.

[EDR93] EDR (1993) *EDR Electronic Dictionary Technical Guide.*Project Report, n°-042, Japan Electronic Dictionary Research Institute Ltd., 16 août 1993, 144 p.

[Farwell92] David Farwell, Louise Guthrie & Yorick Wilks (1992) *The Automatic Creation of Lexical Entries for a Multilingual MT System.* Proc. COLING-92, C. Boitet ed. vol. 2/4, Nantes, France, 18-21 juillet 1992, pp. 532-538.

[Fedder91] L. Fedder, J. McNaught & S. Smith (1991) *Typed Feature Logic and its role in MULTILEX.*Centre for Computational Linguistics, UMIST, novembre 1991, 30 p.

[Fellbaum98] Christiane Fellbaum (1998) *WordNet, an Electronic Lexical Database,* MIT press, Cambridge (MA), 500 p.

[Fischer98] Laurent Fischer & Georges Fafiotte (1998) *BLAK, un assistant de découverte des caractères chinois fonctionnant par accès dynamique à des ressources lexicales.* Proc. NLP+IA 98, Moncton, N.B., Canada, 18-21 août 1998, vol. 1/2, pp. 13-17.

[Gaschler94a] Jean Gaschler & Mathieu Lafourcade (1994a) *Manipulating Human-Oriented Dictionaries with Very Simple Tools.* Proc. COLING-94, Kyoto, Japon, vol. 1/2, pp 283-286.

[Gaschler94b] Jean Gaschler & Mathieu Lafourcade (1994b) *A Case of Building and Manipulating a Dictionary with Very Simple Tools: the FEM Dictionary.* Proc. Proc. ICLA, Penang (Malaysia), 26-28 July 1994 vol. 1/1: pp. 34-37.

[GENELEX93] GENELEX (1993) *Projet Eureka Genelex, modèle sémantique.* Rapport Technique, Projet Eureka, Genelex, mars 1994, 185 p.

[Gsi93] GSI-ERLI (1993) *Le dictionnaire AlethDic* Version 1.5, 62 p.

[Gut96] Yvan Gut, Puteri Rashida Megat Ramli, Zaharin Yusoff, Chuah Choy Kim, Salina A. Samat, Christian Boitet, Nicolas Nédobejkine, Mathieu Lafourcade et al. (1996) *Kamus Perancis-Melayu Dewan, dictionnaire français-malais.* Dewan Bahasa Dan Pustaka, Kuala Lumpur, 667 p.

[Heid92] Ulrich Heid, M. Hein & O. Christ (1992) *Extracting linguistic information from machine-readable versions of traditional dictionaries, a metalexicographic method and some tools.*Proc. COMPLEX'92, Conference on Computational Lexicography an Text Research, Budapest, Hongrie, Linguistics Institute, Hungarian Academy of Sciences, pp. 161-174.

[Ide95a] Nancy Ide, Jacques Le Maitre &Jean Véronis (1995) *Outline of a Model for Lexical Databases.* Current Issues in Computational Linguistics: In Honour of Don Walker. Linguistica Computazionale IX, X (Pisa, 1995), pp 283-320.

[Ide95b] Nancy Ide & Jean Veronis (1995) *Text Encoding Initiative, background and context.* Kluwer Academic Publishers, 242 p.

[ISO86] ISO (1986) *IISO 8879 (SGML) Information processing — Text and office systems — Standard Generalized Markup Language.* Genève, 155 p.

[ISO93] ISO (1993) *ISO/IEC 10646 (UNICODE) Information technology — Universal Multiple-Octet Coded Character Set (UCS).* Genève, 754 p.

[ISO96] ISO (1996) *ISO/IEC 10179 (DSSSL) Information technology — Processing languages — Document Style Semantics and Specification Language.* Genève, 292 p.

[ISO98] ISO (1998) *ISO 639-1 & 2 Code for the representation of names of languages Part 1 & 2 Alpha-3 code.* Genève, Partie 1 : 17 p., Partie 2 : 90 p.

[ISO99a] ISO (1999a) *ISO/IEC 8859-1 à 15 — 8-bit single-byte coded graphic character sets, Latin alphabet.* Genève.

[ISO99b] ISO (1999b) *ISO DIS 12200 (MARTIF) Computer applications in terminology - Machine-readable terminology interchange format - Negotiated interchange.*ISO TC 37/SC 3/WG I, Genève, 118 p.

[ISO99c] ISO (1999c) *ISO DIS 12620 Terminology - Computer Applications Data Categories.*ISO TC 37/SC 3/WG I, Genève, 71 p.

[Johnson95] E. Johnson (1995) *The Text Encoding Initiative.* TEXT Technology vol. 5, n°3, Autumn 1995, pp 174-175.

[Keene89] Sonia Keene (1989) *Object-Oriented Programming in Common LISP: A Programmer's Guide to CLOS.* Addison-Wesley 1989, 266 p.

[Larcheveque96] Jean-Marie Larchevêque (1996) *Requirement analysis and solution proposals for the management of bilingual dictionaries*. Rapport interne XRCE, 18 juin 1996.

[Lafourcade94] Mathieu Lafourcade (1994) *Génie logiciel pour le génie linguiciel.* Thèse de nouveau Doctorat, IMAG-UJF, Grenoble 1, décembre 1994, 318 p.

[Lafourcade96a] Mathieu Lafourcade (1996) *Serveurs de dictionnaires - Etude de cas avec l'outil ALEX et le projet de dictionnaire français-anglais-malais.* Proc. Séminaire LEXIQUE, Grenoble, 13-14 novembre 1996, CLIPS-IMAG, Pôles langage naturel et parole du GDR-PRC CHM., vol. 1/1, pp. 185-192.

[Lafourcade96b] Mathieu Lafourcade (1996) *Structured Lexical data: how to make them widely available, useful and reasonably protected? - a practical example with a trilingual dictionary.* Proc. COLING-96, Copenhagen, Denmark, Vol 2/2, pp. 1106-1110.

[Lafourcade97] Mathieu Lafourcade (1997) *Construction et services dictionnaires n-lingues, exemple des projets Fe*.* Quatrième conférence annuelle sur Le traitement Automatique du Langage Naturel (TALN), 12-13 juin 1997, Grenoble, France, vol. 1/1, pp. 162-168.

[Lamping95] John Lamping, Ramana Rao &Peter Pirolli (1995) *A Focus Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies.* Proc. CHI95, 7-11 mai 1995, Denver, Colorado, États-Unis, pp. 401-408.

[Langlois97] Lucie Langlois, David Megginson & Roda p. Roberts (1997) *SGMLizing the Bilingual Canadian Dictionary.* Proc. Joint International Conference of the Association for Computers and the Humanities and the Association for Literary & Linguistic Computing, ACH-ALLC 1997, Queen's University, Kingston, Ontario, Canada, 3-7 juin 1997, 5 p.

[LREC98] LREC (1998) *Proceedings of the 1st International Conference on Language Resources & Evaluation.* Édité par A.Rubio, N.Gallardo, R.Castro, A. Tejada, Grenade, Espagne, 28-30 mai 1998, 1380 p.

[Mangeot97] Mathieu Mangeot-Lerebours (1997) *Outils pour lexicographes naïfs (en informatique).*DEA Informatique Systèmes et Communications, GETA-CLIPS-IMAG, Université Joseph Fourier Grenoble 1, 19 juin 1997, 58 p.

[Mangeot98] Mathieu Mangeot-Lerebours (1998) *Conception, implémentation et indexation de BaLeM, une base lexicale multilingue.* Proc. TALN'98, Traitement Automatique des Langues Naturelles, Paris, vol 1/1, 10-12 juin 1998, pp. 215-217.

[Mangeot99a] Mathieu Mangeot-Lerebours (1999a) *Visualisation et Navigation dans les bases de données dictionnairiques.* Journée ANRT : Les CIFRE dans le domaine de l'audiovisuel, Institut National de l'Audiovisuel, 23 septembre 99, Paris, 4 p.

[Mangeot99b] Mathieu Mangeot-Lerebours (1999b) *Accès unique à des dictionnaires hétérogènes.* Proc. LTT'99, VIe Journées scientifiques du Réseau thématique de l'AUF Lexicologie, Terminologie, Traduction, éditeurs A. Clas, H. Awaiss et J. Hardane, Beyrouth, Liban, 11-13 novembre 1999, pp 311-316.

[Mangeot00] Mathieu Mangeot-Lerebours (2000) *Papillon Lexical Database Project: Monolingual Dictionaries & Interlingual Links.* WAINS'7, 7th Workshop on Advanced Information Network and System, 7-8 décembre 2000, Kasetsart University, Bangkok, Thailande (à paraître).

[Meijs92] Willem Meijs (1992) *Computers and Written Texts.*Chapitre 6, Butler, C. editor, Oxford: Basil Blackwell, Ltd, pp. 141-165.

[McCord89] Michael C. McCord. (1989) *Design of LMT: A Prolog-based machine translation system.* Computational Linguistics, 1989, Vol 15(1), pp. 33-52.

[Melcuk92] Igor Mel'tchuk (1984, 1988, 1992) *DEC : Dictionnaire Explicatif et Combinatoire du français contemporain, recherches lexico-sémantiques I, II et III.* Presses de l'Université de Montréal, Montréal (Québec), Canada, 172 p., 332 p. et 323 p.

[Melcuk95] Igor Mel'tchuk, André Clas &Alain Polguère (1995) *Introduction à la lexicologie explicative et combinatoire.* Louvain-la-neuve, ed. Duculot, 256 p.

[Melby96] Allan Melby et al. (1996) *The Machine Readable Terminology Interchange Format (MARTIF), Putting Complexity in Perspective.* Termnet News, vol.54/55, pp. 11-21.

[Nedobejkine94] Nicolas Nedobejkine (1994) *Dictionnaire naturel russe-français issu des fichiers codés ARIANE.* Document interne GETA, Grenoble, 8 p.

[Pearsall98] Judy Pearsall (1998) *The New Oxford Dictionary of English.* Clarendon Press, Oxford, 2154 p.

[Perennou92] Guy Pérennou et al. (1992) *Le Projet BDLEX de base de données lexicales du français écrit et parlé.* Rapport technique IRIT, UMR CNRS 5505, Groupe IHM-PT Université Paul Sabatier de Toulouse, 1992, 21 p.

[Perennou97] Guy Pérennou et Martine de Calmès (1997) *Lexique de formes fléchies représentées aux plans morpho-syntaxique, phonologique et orthographique.* Lisez moi, Equipe IHMPT, IRIT - UMR 5505, Université Paul Sabatier de Toulouse, octobre 1997, 15 p.

[Pocock99] Randall J. Pocock (1999) *MRDs and LDBs.* School of Computer Studies, University of Leeds, Royaume-Uni, 8 p.

[Polguere98] Alain Polguère (1998) *La théorie Sens-Texte.* Dialangue, Vol. 8-9, Université du Québec à Chicoutimi, pp. 9-30.

[Polguere00] Alain Polguère (2000) *Towards a theoretically-motivated general public dictionary of semantic derivations and collocations for French.* Proceedings of EURALEX'2000, Stuttgart, pp. 517-527.

[Proszeky97] Gábor Proszéky (1997) *MoBiDic: A New Language Technology Tool for Translators.* Kinga Klaudy, János Kohn (eds.): Transferre necesse est, Scholastica, Budapest, 1997, pp. 558-568.

[Quint87] Vincent Quint (1987) *Une approche de l'édition structurée des documents.* Thèse d'État, Spécialité Mathématiques, Université Joseph Fourier Grenoble 1, 281 p.

[Roberts99] Roda p. Roberts & Lucie Langlois (1999) *L'apport de l'informatique à la recherche lexicographique.* Proc. de l'Association canadienne-française pour l'avancement des sciences, ACFAS, Université d'Ottawa, Ottawa, Canada, 10-14 mai 1999.

[Selva00] Thierry Selva (2000) *Ressources et activités pédagogiques dans un environnement informatique d'aide à l'apprentissage lexical du français langue seconde.* Nouvelle thèse, Spécialité : Automatique et Informatique, Université de Franche-Comté, 210 p.

[Serasset93] Gilles Sérasset & Étienne Blanc (1993) *Une approche par acceptions pour les bases lexicales multilingues.* Proc. T-TA-TAO 93, Montréal, 30 septembre-2 octobre 1993, A. Clas ed. vol. 1/1, pp 65-84.

[Serasset94a] Gilles Sérasset (1994a) *Approche œcuménique au problème du codage des structures linguistiques.* Proc. TALN-94 : Le traitement automatique du langage naturel en France aujourd'hui, Marseille, 7-8 avril 1994, Ph. Blache ed. vol. 1/1: pp. 109-118.

[Serasset94b] Gilles Sérasset (1994b) *An Interlingual Lexical Organisation Based on Acceptions: From the PARAX Mock-up to the NADIA System.*Proc. ICLA-94, Penang, 26-28 juillet 1994 vol. 1/1: pp. 21-33.

[Serasset94c] Gilles Sérasset (1994c) *Interlingual Lexical Organisation for Multilingual Lexical Databases in NADIA*, COLING-94, Kyoto, Japon, 5-9 August 1994, M. Nagao ed., vol. 1/2 : pp. 278-282.

[Serasset94d] Gilles Sérasset (1994d) *Recent Trends of Electronic Dictionary Research and Development in Europe.* Technical Memorandum TM 038, EDR, Japon, 1994, 89 p.

[Serasset94e] Gilles Sérasset (1994e) *SUBLIM: un Système Universel de Bases Lexicales Multilingues et NADIA: sa spécialisation aux bases lexicales interlingues par acceptions.* Thèse de nouveau doctorat, Spécialité Informatique, Université Joseph Fourier Grenoble 1, 194 p.

[Serasset96] Gilles Sérasset (1996) *Un Editeur pour le DEC du francais contemporain.* Proc. Séminaire Lexique, Grenoble, CLIPS, IMAG, pp. 131-138.

[Serasset97a] Gilles Sérasset (1997a) *Le projet NADIA-DEC : vers un dictionnaire explicatif et combinatoire informatisé ?* La mémoire des mots, Ve journées scientifiques du réseau LTT, AUPELF-UREF, Tunis, pp. 149-159.

[Serasset97b] Gilles Sérasset (1997b) *Informatisation du Dictionnaire Explicatif et Combinatoire.* Actes de la quatrième conférence sur le Traitement Automatique du Language Naturel (TALN97), Grenoble, 12-13 juin, pp. 194-198.

[Serasset97c] Gilles Sérasset & Alain Polguère (1997) *Outils pour lexicographes : application à la lexicologie explicative et combinatoire.* Proceedings de RIAO'97, Montréal, Canada, pp. 701-708.

[Serasset98] Gilles Sérasset & Mathieu Mangeot-Lerebours (1998) *L'édition lexicographique dans un système générique de gestion de bases lexicales*. NLP-IA'98, traitement automatique des langues et ses applications industrielles, Moncton, NB, Canada, vol 1/2, 18-21 août 1998, pp. 110-116.

[Shieber86] Stuart M. Shieber (1986) *An Introduction to Unification-Based Approaches to Grammar.* CSLI Notes, Center for the Study of Language and Information, Menlo Park, 105 p.

[Silberztein93] Max Silberztein (1993) *Dictionnaires électroniques et analyse automatique de textes : le système INTEX*. ed Masson, Paris, 234 p.

[Sitko97] Jerzy Sitko (1997) *Manuel destiné aux lexicographes.* Manuel d'indexage UNL, GETA-CLIPS, IMAG, novembre 1997, 15 p.

[Steele90] G. I. Jr. Steele (1990) *COMMON LISP. The language.* Digital Press, 1030 p.

[Tanaka94] K. Tanaka & K. Uemura (1994) *Construction of a Bilingual Dictionary Intermediated by a Third Language.* 15th International Conference on Computational Linguistics, COLING-94, Kyoto, 1994, pp. 297-303.

[Tomasino90] Igor Tomasino (1990) *ODILE, un Outil d'Intégration Extensible de Dictionnaires et Lemmatiseurs.* Mémoire d'ingéneiur CNAM, GETA-CLIPS-IMAG, 150 p.

[Tomokiyo00] Mutsuko Tomokiyo, Mathieu Mangeot-Lerebours & Emmanuel Planas (2000) *Papillon : a Project of Lexical Database for English, French and Japanese, using Interlingual Links.* Journées des Sciences et Techniques de l'ambassade de France au Japon, Tokyo, 12 novembre 2000, 3 p.

[UNL96] UNL (1996) *Universal Networking Language.* UNL center, Institute of Advanced Studies, The United Nations University, 1996, 74 p.

[UNL97] UNL (1997) *DeConverter Specification*. UNL center, Institute of Advanced Studies, The United Nations University, Tokyo, Japan, April 1, 1997, UNL-TR1997-010, Version 1.0, 25 p.

[Veronis90] Jean Véronis & Nancy Ide (1990) *Word Sense Disambiguation with Very Large Neural Networks extracted from Machine Readable Dictionaries*. In Proceedings of 13rd International Conference of Computational Linguistics (ICCL), COLING-90, Helsinki, Finlande, 19-25 août 1990, vol 2, pp 389-394.

[Vitali00] Fabio Vitali (2000) *The XMLC Browser.* ERCIM News n° 41, avril 2000. URL :http://www.cs.unibo.it/projects/

[Vossen97] Piek Vossen (1997) *EuroWordNet: a Multilingual Database for Information Retrieval*. Proc. DELOS Workshop on Cross-Language Information Retrieval, Zurich, mars 1997.

[Wall91] Larry Wall & Randal L. Schwartz (1991) *Programming PERL*. O'Reilly and Associates.

[Wilks96] Yorick Wilks, Brian M. Slator & Louise M. Gutrie (1996) *Electric Words : Dictionaries, Computers, and Meaning.* The MIT Press, 290 p.

[Zajac97] Rémi Zajac, M. Casper & N. Sharples (1997) *An Open Distributed Architecture for Reuse and Integration of Heterogeneous NLP Components*. Proc. ANLP'97, 7 p.

[Zampolli91] Antonio Zampolli (1991) *Linguistic Tools for Multifunctional Applications in Natural Language Processing.* International Symposium for Chinese Information Processing Application ISCIPA'91, Beijing, 1991, pp. 4-21.

[Zock01] Michael Zock & Jean-Pierre Fournier (2001) *Proposal for a customizable, psycholinguistically motivated dictionary to enhance word access.* Proc. 7th Symposium on Social Communication, janvier 2001, Santiago de Cuba, Cuba, 4 p. (à paraître).

Signets

[AllianceWeb] AllianceWeb édition coopérative sur le web
http://www.inrialpes.fr/opera/Alliance.html

[Amaya] Amaya éditeur/navigateur HTML
http://www.w3.org/Amaya/

[ArbresHyperboliques] Arbres hyperboliques :
http://www.inxight.com/products_wb/ht_server

[CSS] CSS 2 *Cascading StyleSheet Language, level 2.* Recommandation du W3C.
http://www.w3.org/TR/REC-CSS2/

[MLTT] Démos linguistiques MLTT http://www.xrce.xerox.com/research/mltt/toolhome

[DHYDRO] DHYDRO *Dictionnaire Hydrographique Multilingue Normalisé.* Projet MLIS.
http://www.loria.fr/projets/MLIS/DHYDRO/

[DicoWeb] DicoWeb consultation de dictionnaires :
http://www-clips.imag.fr/geta/services/dicoweb/

[DicoPro] DicoPro Projet MLIS.
http://issco-www.unige.ch/projects/dicopro_public/

[DicoSzotar] DicoSzótár dictionnaire pour apprenants du hongrois
http://www-clips.imag.fr/geta/services/dicoszotar/

[DicoFeJ] DicoFeJ dictionnaire français-anglais-japonais
http://www-clips.imag.fr/geta/services/dicofej/

[DICT] DICT Development Group
http://www.dict.org/

[dictionary.com] Dictionary.com dictionnaires en ligne
http://www.dictionary.com

[DCB] Dictionnaire Canadien Bilingue *Projet de lexicographie comparée du français et de l'anglais au Canada*.
http://balzac.sti.uottawa.ca/

[DUF] Dictionnaire Universel Francophone développé par Hachette et l'AUPELF-UREF : http://www.francophonie.hachette-livre.fr/

[DOM] DOM *Document Object Model*. Recommandation du W3C. http://w3.org/DOM

[DSSSL] DSSSL *Document StyleSheet Specification Language.* Standard ISO/IEC 10179. http://www.jclark.com/dsssl/

[EDICT] EDICT dictionnaire japonais-anglais de Jim Breen. http://www.csse.monash.edu.au/ jwb/wwwjdicinf.html

[ELRA] ELRA *European Language Ressource Association.* http://www.icp.inpg.fr/ELRA/

[Enhydra] Enhydra Serveur Web dynamique java http://www.enhydra.org/

[EURODICAUTOM] EURODICAUTOM *Multilingual terminological database of the European Commission's Translation Service.* http://eurodic.ip.lu/

[FeM] FeM dictionnaire français-anglais-malais http://www-clips.imag.fr/geta/services/fem/

[GENETER] GENETER *modèle GENErique pour la TERminologie.* http://www.uhb.fr/Langues/Craie/balneo/demo_geneter.pl?langue=1

[Hachette] Hachette dictionnaire Hachette francophone en ligne. http://www-clips.imag.fr/geta/services/dicofej/

[HTML] HTML 4.0 *HyperText Markup Language 4.0 Specification* Recommandation du W3C. http://www.w3.org/TR/REC-html40

[INTERLEX] INTERLEX Diffusion de dictionnaires électroniques via Internet ou cédéroms, projet MLIS. http://interlex.uax.es/

[MHonArc] MHonArc convertisseur mel vers HTML. http://www.mhonarc.org/

[MySQL] MySQL SGBD http://www.mysql.com/

[Nihongo] Nihongo dictionnaire pour apprenants du japonais. http://www-clips.imag.fr/geta/services/nihongo/

[OLIF] OLIF *Open Lexicon Interchange Format*. http://www.olif.net/

[MobiDic] MobiDic dictionnaires hongrois-anglais-allemand.
http://www.mobidictionary.com/

[Papillon] Papillon base lexicale français-japonais-thaï.
http://vulab.ias.unu.edu/papillon/

[PostgreSQL] PostgreSQL SGBD.
http://www.postgresql.org/

[RDF] RDF *Resource Description Framework*. Recommandation du W3C.
http://w3.org/RDF

[RFC2396] RFC 2396 *Request For Comments for Uniform Resource Identifiers (URI): Generic Syntax*. Reqête de l'IETF.
http://www.ietf.org/rfc/rfc2396.txt

[SILFIDE] SILFIDE *Serveur Interactif pour la Langue Française, son Identité, sa Diffusion et son Étude.*
http://www.loria.fr/projets/Silfide et http://silfide.imag.fr

[SALT] SALT *Standards-based Access to Lexicons and Terminologie*.
http://www.ttt.org/salt/ et http://www.loria.fr/projets/SALT/

[SAIKAM] SAIKAM *Serveur de dictionnaire japonais-thaï.*
http://saikam.nii.ac.jp/

[SAX] SAX 2.0 *Simple API for XML version 2*. interface standard pour l'analyse XML.
http://www.w3.org/DOM

[Tamino] Tamino SGBD XML.
http://www.softwareag.com/tamino

[ThirdVoice] ThirdVoice annoteur sur le web.
http://www.thirdvoice.com/

[XHTML] XHTML 1.0 Recommandation du W3C.
http://www.w3.org/TR/xhtml1

[XLINK] XLink 1.0 Recommandation du W3C.
http://www.w3.org/TR/NOTE-xlink-req/

[XMaster] XMaster AlphaWorks *Use XML Master (XMas) to design and generate custom JavaBeans for working with a particular XML document.*
http://www.alphaworks.ibm.com/tech/xmas

[XML] XML 1.0 *eXtended Markup Language 1.0.* Recommandation du W3C.
http://www.w3.org/TR/REC-xml

[XML:DB] XML:DB Développement de spécifications pour utiliser des bases de données XML.
http://www.xmldb.org/

[XML-namespaces] XML Namespaces *XML Namespaces*. Recommandation du W3C.
http://www.w3.org/TR/REC-xml-names

[XML-schemas] XML Schemas *XML Schemas*. Recommandation du W3C.
http://www.w3.org/TR/xmlschema-0

[XPath] XPath *XPath Language*. Recommandation du W3C.
http://www.w3.org/TR/xpath

[XPointer] XPointer *XML Pointer Language* Recommandation du W3C.
http://www.w3.org/TR/xptr

[XSL] XSL 1.0 *eXtended Stylesheet Language 1.0*. Recommandation du W3C.
http://www.w3.org/TR/xsl

[XSLT] XSLT 1.0 *eXtended Stylesheet Language Transformation 1.0*. Recommandation du W3C.
http://www.w3.org/TR/xslt

[X-Hive] X-Hive SGBD XML.
http://www.x-hive.com/

Annexe A : schéma XML pour DML

1. Organisation de DML

Les éléments du schéma DML permettent de décrire un environnement complet de base lexicale. Voici l'organisation de ces éléments :

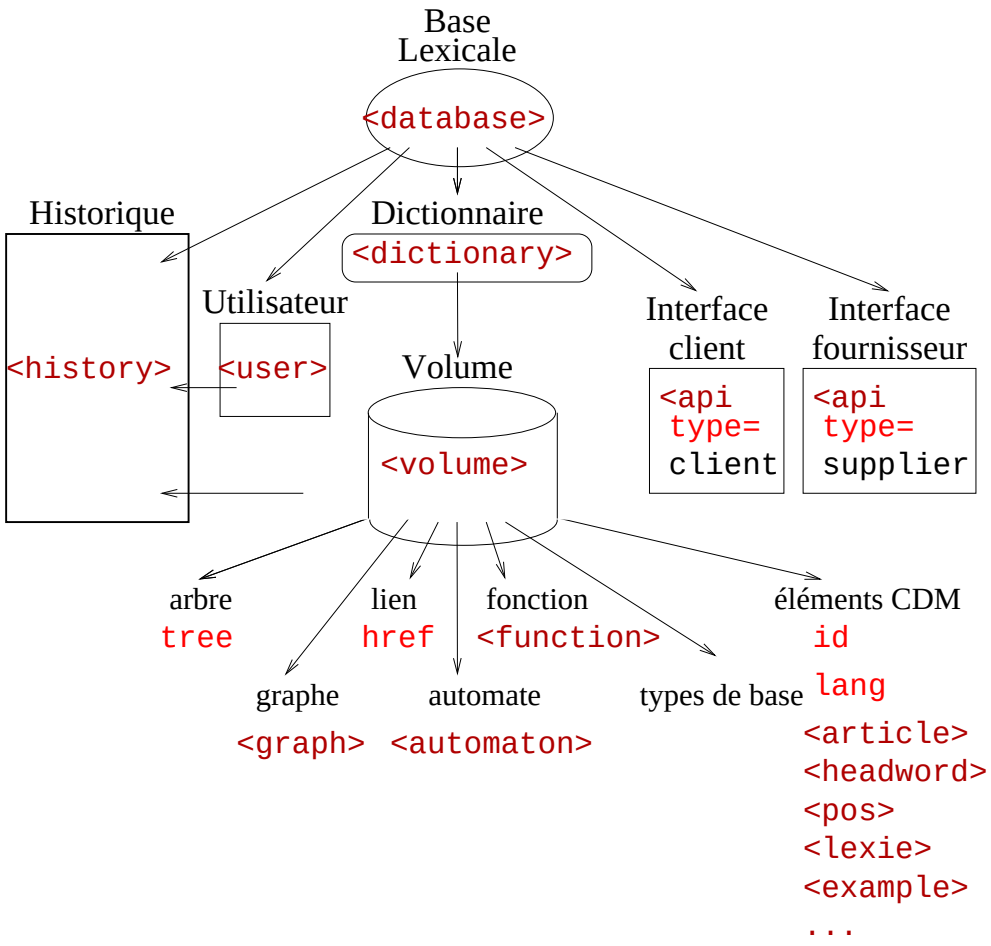


FIG. A.1 – *organisation des éléments de DML*

2. Schéma XML de DML

La langue de travail commune aux membres du projet Papillon est l'anglais. Le schéma DML est utilisé dans le projet Papillon. Les explications sont donc rédigées en anglais pour permettre une compréhension de la part de tous les membres du projet.

```
   <!-- XML Schema for common elements of Dictionary Markup Language.
These elements are used to encode heterogeneous lexical databases
   Namespace = http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/dml.xsd
   $Author:  mangeot $ Mathieu MANGEOT-LEREBOURS Mathieu.Mangeot@imag.fr
$Date:  2001/09/15 09:37:10 $ $Revision:  1.6 $
   -->
   <schema targetNamespace="http://www-clips.imag.fr/geta/services/dml">
     <annotation>
       <documentation xml:lang="en"> XML Schema for common elements
of Dictionary Markup Language.  These elements are used to encode
heterogeneous lexical databases
   Namespace = http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/dml.xsd </documentation>
     </annotation>
   <!--================ importing other schemas ================-->
   <!-- importing parts of xlink recommendation for dml links -->
     <import namespace="http://www.w3.org/1999/xlink"
   schemaLocation="http://www-clips.imag.fr/geta/services/dml/xlink.xsd"/>
   <!--================ common DML attributes and types
================-->
   <!-- Note:  the attributes are sorted in alphabetical order -->
   <!-- dateType type -->
   <!-- Used for all the dates in DML. Equals to dateTime from
XML schema basic type.  The following definition is taken for
REC-xmlschema-2:  A single lexical representation, which is a subset
of the lexical representations allowed by ISO 8601, is allowed for
dateTime.  This lexical representation is the ISO 8601 extended format
CCYY-MM-DDThh:mm:ss where "CC" represents the century, "YY" the year,
"MM" the month and "DD" the day, preceded by an optional leading "-"
sign to indicate a negative number.  If the sign is omitted, "+" is
assumed.  The letter "T" is the date/time separator and "hh", "mm",
```

```
"ss" represent hour, minute and second respectively.  Additional digits
can be used to increase the precision of fractional seconds if desired
i.e the format ss.ss...  with any number of digits after the decimal
point is supported.  The fractional seconds part is optional; other
parts of the lexical form are not optional.  To accommodate year values
greater than 9999 additional digits can be added to the left of this
representation.  Leading zeros are required if the year value would
otherwise have fewer than four digits; otherwise they are forbidden.
The year 0000 is prohibited.
   The CCYY field must have at least four digits, the MM, DD, SS, hh,
mm and ss fields exactly two digits each (not counting fractional
seconds); leading zeroes must be used if the field would otherwise have
too few digits.
   This representation may be immediately followed by a "Z" to indicate
Coordinated Universal Time (UTC) or, to indicate the time zone, i.e.
the difference between the local time and Coordinated Universal Time,
immediately followed by a sign, + or -, followed by the difference from
UTC represented as hh:mm (note:  the minutes part is required).  See
ISO 8601 Date and Time Formats (chapter D) for details about legal
values in the various fields.  If the time zone is included, both hours
and minutes must be present.
   For example, to indicate 1:20 pm on May the 31st, 1999 for Eastern
Standard Time which is 5 hours behind Coordinated Universal Time (UTC),
one would write:  1999-05-31T13:20:00-05:00.  -->
     <simpleType name="date">
       <restriction base="dateType"/>
     </simpleType>
   <!-- delay attribute -->
   <!-- indicates the delay when querying the element wearing this
attribute eg:  5 seconds.  Maybe, the type could be a time type -->
     <attribute name="delay" type="d:durationType"/>
   <!-- durationType type -->
   <!-- indicates a duration eg:  5 seconds and 10 cents= "5.10S" I
took the duration type of sxml schema.  PB: if > to 24H, it takes days.
It must be revised...  -->
     <simpleType name="durationType">
       <restriction base="duration"/>
     </simpleType>
   <!-- id attribute -->
   <!-- the elements with the ID attribute have a unique ID for all the
lexical database -->
     <attribute name="id" type="ID"/>
   <!-- history attribute -->
   <!-- The history attribute is used to link an element with its
history log file where all the changes are stored -->
     <attribute name="history" type="ID"/>
   <!-- history-ref attribute -->
```

```
   <!-- The history-ref attribute is used to reference the file where
all the changes are stored -->
     <attribute name="history-ref" type="xlink:hrefType"/>
   <!-- href attribute -->
   <!-- this attribute is used for all the links between DML elements
it is the simple definition of xlink -->
     <complexType name="hrefType">
       <attribute ref="xlink:href" use="required"/>
     </complexType>
   <!-- lang attribute -->
   <!-- the DML lang attribute is based on ISO 639-2/T standard which
uses 3 letters code instead of two letters code to indicate the name of
the languages.  We add also our proper codes for special purpose -->
     <attribute name="lang" type="d:lang"/>
     <simpleType name="lang">
       <restriction base="string">
         <enumeration value="aar"/>
   <!-- Afar; 639-1:  aa -->
         <enumeration value="abk"/>
   <!-- Abkhazian; 639-1:  ab -->
         <enumeration value="ace"/>
   <!-- Achinese -->
         <enumeration value="ach"/>
   ...
   <!-- German; 639-1:  de -->
         <enumeration value="dgr"/>
   ...
   <!-- English; 639-1:  en -->
         <enumeration value="enm"/>
   ...
   <!-- French; 639-1:  fr -->
         <enumeration value="frm"/>
   ...
   <!-- Hungarian; 639-1:  hu -->
         <enumeration value="hup"/>
   ...
   <!-- Indonesian; 639-1:  id/in -->
         <enumeration value="ine"/>
   ...
   <!-- Italian; 639-1:  it -->
         <enumeration value="jaw"/>
   ...
   <!-- Japanese; 639-1:  ja -->
         <enumeration value="jpr"/>
   ...
   <!-- Korean; 639-1:  ko -->
         <enumeration value="kos"/>
```

```
   ...
   <!-- Lao; 639-1:  lo -->
         <enumeration value="lat"/>
   ...
   <!-- Malay; 639-1:  ms -->
         <enumeration value="mul"/>
   ...
   <!-- Thai; 639-1:  th -->
         <enumeration value="tig"/>
   ...
   <!-- Vietnamese; 639-1:  vi -->
         <enumeration value="vol"/>
   ...
   <!-- Chinese; 639-1:  zh -->
         <enumeration value="znd"/>
   <!-- Zande -->
         <enumeration value="zul"/>
   <!-- Zulu; 639-1:  zu -->
         <enumeration value="zun"/>
   <!-- Zuni -->
   <!-- DML additions to the ISO 639-2/T for special purpose -->
         <enumeration value="axi"/>
         <enumeration value="unl"/>
       </restriction>
     </simpleType>
   <!-- refType type -->
   <!-- references to another object with an xlink.  The link can be
tagged with a gloss.  lang is the language of the gloss.  -->
     <complexType name="refType" mixed="true">
       <attribute ref="xlink:href" use="required"/>
       <attribute ref="d:lang"/>
       <attribute name="tag" type="string"/>
       <attribute name="tag-type" type="string"/>
     </complexType>
   <!-- status attribute -->
   <!-- The status attribute is used to indicate the status of a
linguistic element -->
     <attribute name="status">
       <simpleType>
         <restriction base="string">
           <enumeration value="auto"/>
           <enumeration value="rough"/>
           <enumeration value="revised"/>
         </restriction>
       </simpleType>
     </attribute>
   <!-- weight attribute -->
```

```
   <!-- Indicates the weight of a linguistic element.  It can be a
frequency score, etc.  This weight can be used to choose between
various elements of the same type eg between two translations/ The
possible values are between 0.0 and 1.0.

   It is better to use the IDs and to store the different weights in
a table.  It allows to store various weights (frequency in corpora,
neuronal weigth, frequency in search engines, etc.).
   -->
     <attribute name="weight" type="d:weightType"/>
     <simpleType name="weightType">
       <restriction base="float">
         <maxExclusive value="1.0"/>
         <minExclusive value="0.0"/>
       </restriction>
     </simpleType>
   <!--================ DML definitions for a database
================-->
   <!-- database element -->
   <!--The database element is the top element of the database.  It
describes the whole database with the dictionaries, the various groups
and pointers to the users file -->
     <element name="database">
       <complexType>
         <sequence>
           <element ref="d:users"/>
           <element ref="d:groups"/>
           <element ref="d:partner-servers"/>
           <element ref="d:dictionaries"/>
         </sequence>
         <attribute name="creation-date" type="d:dateType" use="optional"/>
         <attribute name="name" type="string" use="optional"/>
         <attribute name="owner" type="string" use="optional"/>
         <attribute ref="d:history" use="optional"/>
         <attribute ref="d:history-ref" use="optional"/>
       </complexType>
     </element>
   <!-- partner-servers element -->
   <!-- Lists all the users or groups that have the rights to exchange
some data with the database.  The partners are other programs
```

```
(databases, lemmatizers, etc.), not humans.  -->
     <element name="partner-servers">
       <complexType>
         <choice minOccurs="0" maxOccurs="unbounded">
           <element ref="d:group-ref"/>
           <element ref="d:user-ref"/>
         </choice>
       </complexType>
     </element>
   <!-- users element -->
   <!-- Lists all the various users of the database.  -->
     <element name="users">
       <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
           <element ref="d:user-ref"/>
         </sequence>
       </complexType>
     </element>
   <!-- group-ref element -->
   <!-- This element is used to make a reference to a group of the
database.  -->
     <element name="group-ref">
       <complexType>
         <attribute name="name" type="string" use="optional"/>
       </complexType>
     </element>
   <!-- groups element -->
   <!-- Lists all the various groups of the database.  -->
     <element name="groups">
       <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
           <element ref="d:group"/>
         </sequence>
       </complexType>
     </element>
   <!-- group element -->
   <!-- describes a group of the database by listing its users -->
   <!-- 3 groups exist in every database by default:  administrators,
lexicologists and the universe of all users -->
     <element name="group">
       <complexType>
         <sequence minOccurs="1" maxOccurs="unbounded">
           <element ref="d:user-ref"/>
         </sequence>
         <attribute name="name" type="string" use="optional"/>
       </complexType>
     </element>
```

```
  <!-- user-ref element -->
  <!-- This element is used to make a reference to a user of the
database.  -->
    <element name="user-ref">
      <complexType>
        <attribute name="name" type="string" use="optional"/>
        <attribute ref="xlink:href" use="optional"/>
      </complexType>
    </element>
  <!-- dictionaries element -->
  <!-- lists all the heterogeneous dictionaries available locally or
through the network from the database.  -->
    <element name="dictionaries">
      <complexType>
        <sequence minOccurs="0" maxOccurs="unbounded">
          <element ref="d:dict-ref"/>
        </sequence>
      </complexType>
    </element>
  <!-- dict-ref element -->
  <!-- links to a dictionary element that describes a dictionary -->
    <element name="dict-ref">
      <complexType>
        <attribute name="name" type="string" use="optional"/>
        <attribute ref="xlink:href" use="optional"/>
      </complexType>
    </element>
  <!--================ DML definitions for a user ================-->
  <!-- user element -->
  <!--The user element describes all information relative to each user
```

```
of the database with his/her settings, preferences, etc.  -->
     <element name="user">
       <complexType>
         <sequence>
           <element ref="d:login"/>
           <element ref="d:password"/>
           <element ref="d:email"/>
           <element ref="d:profiles"/>
           <element ref="d:credits"/>
           <element ref="d:annotations"/>
           <element ref="d:contributions"/>
           <element ref="d:requests"/>
           <element ref="d:xml-stylesheet"/>
           <element ref="d:groups"/>
         </sequence>
         <attribute ref="d:history" use="optional"/>
         <attribute ref="d:history-ref" use="optional"/>
         <attribute name="creation-date" type="d:dateType" use="optional"/>
         <attribute name="name" type="string" use="optional"/>
       </complexType>
     </element>
   <!-- login element -->
   <!-- used by a user to log into the database -->
     <element name="login" type="string"/>
   <!-- password element -->
   <!-- used by a user to log into the database.  Has to be encrypted
-->
     <element name="password" type="string"/>
   <!-- email element -->
   <!-- email address -->
     <element name="email" type="d:emailType"/>
     <simpleType name="emailType">
       <restriction base="string">
   <!-- regular expression:  at least on char followed by a "@"
followed by at least one char followed by a "." followed by at least
one char -->
         <pattern value=".+@.+[.]+.+"/>
       </restriction>
     </simpleType>
   <!-- profiles element -->
```

```
  <!-- User profiles -->
    <element name="profiles">
      <complexType>
        <sequence>
          <element ref="d:competences" maxOccurs="1"/>
          <element ref="d:interests" maxOccurs="1"/>
          <element ref="d:activities" maxOccurs="1"/>
        </sequence>
      </complexType>
    </element>
  <!-- competences element -->
  <!-- Indicates the linguistic competences of a user/contributor -->
    <element name="competences">
      <complexType>
        <sequence>
  <!-- The declaration of these elements should be more structured and
systematic ie:  1 element for each language of the dml:lang type.-->
          <element name="cat" maxOccurs="1" type="string"/>
          <element name="eng" maxOccurs="1" type="string"/>
          <element name="fra" maxOccurs="1" type="string"/>
          <element name="spa" maxOccurs="1" type="string"/>
          <element name="hun" maxOccurs="1" type="string"/>
          <element name="jpn" maxOccurs="1" type="string"/>
          <element name="ita" maxOccurs="1" type="string"/>
          <element name="spa" maxOccurs="1" type="string"/>
          <element name="tha" maxOccurs="1" type="string"/>
        </sequence>
      </complexType>
    </element>
  <!-- interests element -->
  <!-- Indicates the interests of a user/contributor -->
    <element name="interests">
      <complexType>
        <sequence>
          <element name="interest" type="string"/>
        </sequence>
      </complexType>
    </element>
  <!-- activities element -->
  <!-- Indicates the activities of a user/contributor -->
    <element name="activities">
      <complexType>
        <sequence>
          <element name="activity" type="string"/>
        </sequence>
      </complexType>
    </element>
```

```
   <!-- credits element -->
   <!-- contributions credits of a contributors.  If a contributor
sends a contribution to the database, his/her credits increase.  If
s/he extracts a customised dictionary from the database, his/her
credits decrease.  -->
     <element name="credits" type="Integer"/>
   <!-- contributions element -->
   <!-- groups the contributions of a contributors.  These
contributions are stored in a virtual space before being reviewed and
integrated into the database by a specialist in lexicology.  -->
     <element name="contributions">
       <complexType>
         <sequence>
           <element ref="d:contribution"/>
         </sequence>
       </complexType>
     </element>
   <!-- contribution element -->
   <!-- Links to a contribution of a contributor.  These contributions
are represented by an XSLT stylesheet on the source file.  -->
     <element name="contribution">
       <complexType mixed="true">
         <attribute name="source" type="xlink:hrefType"/>
         <attribute ref="xlink:href"/>
       </complexType>
     </element>
   <!-- requests element -->
   <!-- Links to a file where all the requests of a user are stored.
-->
     <element name="requests">
       <complexType mixed="true">
         <attribute ref="xlink:href"/>
       </complexType>
     </element>
   <!-- xml-stylesheet element -->
   <!-- links to an XML stylesheet used by a user to indicate its
preferences.  -->
     <element name="xml-stylesheet">
       <complexType>
         <attribute name="type" type="string" use="optional"/>
         <attribute ref="xlink:href" use="optional"/>
       </complexType>
     </element>
   <!--================ DML definitions for a dictionary
================-->
   <!-- dictionary element -->
```

```
   <!-- This elements describes a dictionary.  It notes the
meta-information available on a dictionary:  languages, content,
domain, size, dates, encoding, format, number of headwords, etc.  It
describes also the macrostructure of the dictionary, ie monolingual,
bilingual, multilingual, etc.  -->
     <element name="dictionary">
       <complexType>
         <sequence>
           <element ref="d:languages"/>
           <element ref="d:contents"/>
           <element ref="d:domain"/>
           <element ref="d:bytes"/>
           <element ref="d:source"/>
           <element ref="d:legal"/>
           <element ref="d:comments"/>
           <element ref="d:cdm-elements"/>
           <element ref="d:administrators"/>
           <element ref="d:volumes"/>
           <element ref="d:links"/>
         </sequence>
         <attribute ref="d:history" use="optional"/>
         <attribute ref="d:history-ref" use="optional"/>
         <attribute name="category" type="categoryType" use="optional"/>
         <attribute name="creation-date" type="d:dateType" use="optional"/>
         <attribute name="encoding" type="d:encodingType" use="optional"/>
         <attribute name="format" type="d:formatType" use="optional"/>
         <attribute name="hw-number" type="positiveInteger" use="optional"/>
         <attribute name="installation-date" type="d:dateType" use="optional"/>
         <attribute name="name" type="string" use="optional"/>
         <attribute name="nickname" type="string" use="optional"/>
         <attribute name="owner" type="string" use="optional"/>
         <attribute name="type" type="d:dictType" use="optional"/>
         <attribute name="version" type="string" use="optional"/>
       </complexType>
     </element>
   <!-- element languages -->
   <!-- lists the languages present in a dictionary -->
     <element name="languages">
       <complexType>
         <choice minOccurs="0" maxOccurs="unbounded">
           <element ref="d:source-language"/>
           <element ref="d:target-language"/>
         </choice>
       </complexType>
     </element>
   <!-- element source-language -->
```

```
  <!-- source-language present in a dictionary -->
    <element name="source-language">
      <complexType>
        <attribute ref="d:lang"/>
      </complexType>
    </element>
  <!-- element target-language -->
  <!-- target-language present in a dictionary -->
    <element name="target-language">
      <complexType>
        <attribute ref="d:lang"/>
      </complexType>
    </element>
  <!-- type categoryType -->
  <!-- macrostructure of a dictionary -->
    <simpleType name="categoryType">
      <restriction base="string">
        <enumeration value="monolingual"/>
        <enumeration value="bilingual"/>
        <enumeration value="multilingual"/>
      </restriction>
    </simpleType>
  <!-- type dictType -->
  <!-- macrostructure type of a multilingual dictionary -->
    <simpleType name="dictType">
      <restriction base="string">
        <enumeration value="monodirectional"/>
        <enumeration value="bidirectional"/>
        <enumeration value="pivot"/>
        <enumeration value="mixed"/>
      </restriction>
    </simpleType>
  <!-- type encodingType -->
  <!-- encoding type of a dictionary.  The values are
taken from the Internet Assigned Number Authority IANA
Character Set registry.  For more info, please refer to:
http://www.iana.org/assignments/character-sets These encoding types
```

```
are also used for MIME types -->
     <simpleType name="encodingType">
       <restriction base="string">
         <enumeration value="Big5"/>
         <enumeration value="EUC-JP"/>
         <enumeration value="EUC-KR"/>
         <enumeration value="GB2312"/>
         <enumeration value="ISO-2022-JP"/>
         <enumeration value="ISO-2022-KR"/>
         <enumeration value="ISO-8859-1"/>
         <enumeration value="ISO-8859-2"/>
         <enumeration value="ISO-8859-3"/>
         <enumeration value="ISO-8859-4"/>
         <enumeration value="ISO-8859-5"/>
         <enumeration value="ISO-8859-6"/>
         <enumeration value="ISO-8859-7"/>
         <enumeration value="ISO-8859-8"/>
         <enumeration value="ISO-8859-9"/>
         <enumeration value="ISO-8859-10"/>
         <enumeration value="ISO-8859-15"/>
         <enumeration value="KOI8-R"/>
         <enumeration value="US-ASCII"/>
         <enumeration value="Shift_JIS"/>
         <enumeration value="UTF-7"/>
         <enumeration value="UTF-8"/>
         <enumeration value="UTF-16"/>
       </restriction>
     </simpleType>
   <!-- type formatType -->
   <!-- format of a dictionary -->
     <simpleType name="formatType">
       <restriction base="string">
         <enumeration value="rtf"/>
         <enumeration value="xml"/>
         <enumeration value="html"/>
         <enumeration value="sgml"/>
         <enumeration value="txt"/>
       </restriction>
     </simpleType>
   <!-- element contents -->
   <!-- It describes with a text the contents of a dictionary -->
     <element name="contents" type="string"/>
   <!-- element domain -->
   <!-- It describes the domain of a dictionary e.g.  :  general,
medicine, computer science, etc.  Maybe it could be a closed list ...
-->
   <!-- element bytes -->
```

```
   <!-- Size of all the files of a dictionary in bytes -->
     <element name="bytes" type="positiveInteger"/>
   <!-- element source -->
   <!-- describes from where does the dictionary come from, who gave it
-->
     <element name="source" type="string"/>
   <!-- element legal -->
   <!-- describes the legal rights attached to the use of this
dictionary e.g.  :research purpose only, public, open source, etc.  -->
     <element name="legal" type="string"/>
   <!-- element comments -->
   <!-- general comments on a dictionary, text -->
     <element name="comments" type="string"/>
   <!-- element cdm-elements -->
   <!-- lists all the common dictionary markup (CDM) elements presents
in a dictionary.  The CDM elements have a fixed semantics.  It allows
one to merge two dictionaries following their CDM elements or to query
these elements -->
     <element name="cdm-elements">
       <complexType>
         <choice minOccurs="0" maxOccurs="unbounded">
   <!-- all CDM elements -->
           <element ref="d:headword" maxOccurs="1"/>
           <element ref="d:pronunciation" maxOccurs="1"/>
           <element ref="d:pos" maxOccurs="1"/>
           <element ref="d:translation"/>
           <element name="corpus" maxOccurs="1" type="d:cdmType"/>
         </choice>
       </complexType>
     </element>
   <!-- type cdmType -->
   <!-- dml type for cdm elements -->
     <complexType name="cdmType">
       <attribute ref="d:delay"/>
       <attribute ref="d:lang"/>
     </complexType>
   <!-- element volumes -->
   <!-- Lists all the volumes/files of a dictionary with an xlink.  -->
     <element name="volumes">
       <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
           <element ref="d:volume-ref"/>
         </sequence>
       </complexType>
     </element>
   <!-- element volume-ref -->
```

```
   <!-- references a volume/file with an xlink.  -->
     <element name="volume-ref">
       <complexType mixed="true">
         <attribute name="name" type="xlink:label"/>
         <attribute ref="xlink:href"/>
       </complexType>
     </element>
   <!-- element links -->
   <!-- indicates the links between the volumes files in the
dictionary.  -->
     <element name="links">
       <complexType>
         <sequence>
           <element ref="d:arcType" minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
       </complexType>
     </element>
   <!--================ DML definitions for CDM elements
================-->
   <!-- headword element -->
   <!-- This is the headword of the articles of the monolingual
dictionaries.  It is the name of the lexies and vocables.  -->
     <element name="headword">
       <complexType mixed="true">
   <!-- hn attribute -->
   <!-- Homograph number of the headword.  -->
         <attribute name="hn" type="string" use="optional"/>
         <attribute ref="d:delay"/>
       </complexType>
     </element>
   <!-- pos element -->
   <!-- Part-of-speech of the headword The type has to be redefined in
the schemas for the volumes.  -->
     <element name="pos" type="d:posType"/>
     <simpleType name="posType">
       <restriction base="string"/>
     </simpleType>
   <!-- pronunciation element -->
   <!-- pronunciation of the headword.  -->
     <element name="pronunciation">
       <complexType mixed="true">
         <attribute name="encoding" type="string" use="optional"/>
         <attribute ref="d:delay"/>
       </complexType>
     </element>
   <!-- translation element -->
```

```
  <!-- translation of the headword.  -->
    <element name="translation">
      <complexType mixed="true">
        <attribute ref="d:lang"/>
        <attribute ref="d:delay"/>
      </complexType>
    </element>
  <!--================ DML definitions for a volume file
================-->
  <!-- volume element -->
  <!-- This element describes a volume.  It is a list of articles
sorted following the nomenclature of the dictionary -->
    <element name="volume">
      <complexType>
        <sequence>
          <group ref="d:article" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute ref="d:history" use="optional"/>
        <attribute ref="d:history-ref" use="optional"/>
        <attribute name="creation-date" type="d:dateType" use="optional"/>
        <attribute name="encoding" type="d:encodingType" use="optional"/>
        <attribute name="format" type="d:formatType" use="optional"/>
        <attribute name="hw-number" type="positiveInteger" use="optional"/>
        <attribute name="installation-date" type="d:dateType" use="optional"/>
        <attribute name="name" type="string" use="optional"/>
        <attribute name="source-language" type="d:lang" use="optional"/>
        <attribute name="version" type="string" use="optional"/>
      </complexType>
    </element>
  <!-- article group -->
  <!-- Its content is an article of a dictionary.  It has to be
redefined in other schemas for specific volumes -->
  <!-- due to a bug in XSV (redefinition is not implemented) I
change the content of the group article for validation <group
name='article'><all /></group> -->
    <group name="article">
      <sequence>
        <element ref="d:lexie" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </group>
  <!--================ DML definitions for APIs ====-->
  <!--== API element ==-->
  <!-- This element encodes the APIs used by clients and suppliers of
the database to exchange data with it.  API = Application Programming
```

```
Interface -->
     <element name="api">
       <complexType>
         <sequence maxOccurs="unbounded">
   <!-- provides general information on the API -->
           <element name="info" type="string"/>
   <!-- indicates the URLs used for connection -->
           <element ref="d:url"/>
   <!-- indicates the protocols used for connection -->
           <element ref="d:protocol"/>
   <!-- indicates the connection delays -->
           <element ref="d:delay"/>
   <!-- indicates the input and output encodings -->
           <element ref="d:encodings"/>
   <!-- indicates the input and output formats -->
           <element ref="d:formats"/>
   <!-- format of the arguments -->
           <element name="arguments" type="d:argumentsType"/>
   <!-- format of the result -->
           <element name="result" type="d:resultType"/>
         </sequence>
         <attribute name="creation-date" type="d:dateType" use="optional"/>
         <attribute name="name" type="string" use="optional"/>
         <attribute name="type">
           <simpleType>
             <restriction base="string">
               <enumeration value="supplier"/>
               <enumeration value="client"/>
             </restriction>
           </simpleType>
         </attribute>
   <!-- indicates the input and output formats -->
         <attribute name="category">
           <simpleType>
             <restriction base="string">
               <enumeration value="preprocessing"/>
               <enumeration value="meta-info"/>
               <enumeration value="consultation"/>
               <enumeration value="modification"/>
             </restriction>
           </simpleType>
         </attribute>
       </complexType>
     </element>
   <!-- url element -->
```

```
  <!-- indicates the URLs used for connection -->
    <element name="url">
      <complexType>
        <attribute ref="xlink:href"/>
      </complexType>
    </element>
  <!-- protocol element -->
  <!-- indicates the protocol used for connection -->
    <element name="protocol">
      <complexType>
        <attribute name="type" type="string"/>
        <attribute name="login" type="string"/>
        <attribute name="password" type="string"/>
      </complexType>
    </element>
  <!-- delay element -->
  <!-- indicates the connection delays for the API -->
    <element name="delay">
      <complexType>
        <attribute name="min" type="d:durationType"/>
        <attribute name="average" type="d:durationType"/>
        <attribute name="max" type="d:durationType"/>
      </complexType>
    </element>
  <!-- encodings element -->
  <!-- indicates the input and output encodings.  The type used is
encodingType defined before.  -->
    <element name="encodings">
      <complexType>
        <attribute name="input" type="encodingType"/>
        <attribute name="output" type="encodingType"/>
      </complexType>
    </element>
  <!-- formats element -->
  <!-- indicates the input and output formats.  The type used is
formatType defined before.  -->
    <element name="formats">
      <complexType>
        <attribute name="input" type="formatType"/>
        <attribute name="output" type="formatType"/>
      </complexType>
    </element>
  <!-- argumentsType type -->
```

```
   <!-- indicates the format of the arguments.  The XML schema syntax
is used.  The type has to be redefined in another schema.  -->
     <simpleType name="argumentsType">
       <restriction base="string"/>
     </simpleType>
   <!-- resultType type -->
   <!-- indicates the format of the result.  The XML schema syntax is
used.  The type has to be redefined in another schema.  -->
     <simpleType name="resultType">
       <restriction base="string"/>
     </simpleType>
   <!--================ DML definitions for the history file
================-->
   <!-- element history -->
   <!-- The history file contains the logs of the modifications
performed on every element of the database.  A log is referenced with
the DML history attribute -->
     <element name="history">
       <complexType>
         <sequence maxOccurs="unbounded">
           <element ref="d:administration"/>
         </sequence>
         <attribute name="creation-date" type="d:dateType" use="optional"/>
         <attribute name="name" type="string" use="optional"/>
       </complexType>
     </element>
   <!-- element administration -->
   <!-- The element administration contains the administration
information and the history of the changes performed on the element
with the history attribute which references this element through this
id attribute -->
     <element name="administration">
       <complexType>
         <sequence>
           <element ref="d:creation" minOccurs="1" maxOccurs="1"/>
           <element ref="d:modification" minOccurs="0" maxOccurs="unbounded"/>
           <element ref="d:revision" minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute ref="d:id"/>
       </complexType>
     </element>
   <!-- type annotation -->
   <!-- to describe all the changes on one element of the database.  It
indicates the name of the person which has changed the element and the
```

```
date of the change.  It can contain some text for various comments -->
     <complexType name="annotationType" mixed="true">
       <attribute name="indexer" type="string"/>
       <attribute name="date" type="d:dateType"/>
     </complexType>
   <!-- element creation -->
   <!-- describes the creation of an element -->
     <element name="creation" type="d:annotationType"/>
   <!-- element modification -->
   <!-- describes the modification of an element -->
     <element name="modification" type="d:annotationType"/>
   <!-- element revision -->
   <!-- describes the revision of an element -->
     <element name="revision" type="d:annotationType"/>
   <!--================ DML definitions for common elements and
structures ====-->
   <!--== Tree structure ==-->
   <!-- DML element to represent a tree -->
     <element name="nd">
       <complexType mixed="true">
         <sequence>
           <element ref="d:nd"/>
         </sequence>
       </complexType>
     </element>
   <!--== Graph structure ==-->
   <!-- DML element to represent a graph -->
     <element name="graph">
       <complexType>
         <sequence>
           <element ref="d:nodes" minOccurs="1" maxOccurs="1"/>
           <element ref="d:arcs" minOccurs="1" maxOccurs="1"/>
         </sequence>
         <attribute ref="xlink:type" fixed="extended"/>
       </complexType>
     </element>
   <!-- DML element to represent a list of nodes -->
     <element name="nodes">
       <complexType>
         <sequence>
           <element ref="d:node" minOccurs="1" maxOccurs="unbounded"/>
         </sequence>
       </complexType>
     </element>
```

```
<!-- DML element to represent a node -->
  <element name="node">
    <complexType mixed="true">
      <attribute ref="xlink:type" fixed="locator"/>
      <attribute ref="xlink:label"/>
      <attribute name="xlink:title">
        <simpleType>
          <restriction base="xlink:titleType">
<!-- to note the starting node of an automaton -->
            <enumeration value="starting-node"/>
<!-- to note an ending node of an automaton -->
            <enumeration value="ending-node"/>
          </restriction>
        </simpleType>
      </attribute>
    </complexType>
  </element>
<!-- DML element to represent a list of arcs -->
  <element name="arcs">
    <complexType>
      <sequence>
        <element ref="d:arcType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
<!-- DML element to represent an arc -->
  <element name="arc" type="d:arcType"/>
<!-- DML type to represent an arc -->
  <complexType name="arcType" mixed="true">
    <attribute ref="xlink:type" fixed="arc"/>
    <attribute ref="xlink:from"/>
    <attribute ref="xlink:to"/>
<!-- an oriented arc has the attribute oriented='true' -->
    <attribute name="oriented" type="boolean"/>
  </complexType>
<!--== Automaton structure ==-->
<!-- DML element to represent a graph -->
  <element name="automaton">
    <complexType>
      <sequence>
        <element ref="d:nodes" minOccurs="1" maxOccurs="1"/>
        <element ref="d:arcs" minOccurs="1" maxOccurs="1"/>
      </sequence>
      <attribute ref="xlink:type" fixed="locator"/>
    </complexType>
  </element>
<!--== Function structure ==-->
```

```
  <!-- DML element to represent a function -->
    <element name="function">
      <complexType mixed="true">
        <sequence>
          <element ref="d:arguments" minOccurs="0" maxOccurs="1"/>
          <choice>
            <element ref="d:value" maxOccurs="1"/>
            <element ref="d:valgroup" minOccurs="0" maxOccurs="unbounded"/>
          </choice>
        </sequence>
        <attribute name="name" type="string"/>
      </complexType>
    </element>
  <!-- DML element for arguments of a function -->
  <!--It has to be redefined in schemas specific to the dictionaries
-->
    <element name="arguments">
      <complexType>
        <sequence/>
      </complexType>
    </element>
  <!-- DML element for groups of values of a function -->
    <element name="valgroup">
      <complexType mixed="true">
        <sequence>
          <element name="comment" type="string" minOccurs="0" maxOccurs="1"/>
          <element ref="d:value" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
  <!-- DML element for value of a function -->
  <!--It has to be redefined in schemas specific to the dictionaries
-->
    <element name="value" type="d:valueType"/>
    <complexType name="valueType"/>
  </schema>
```

Annexe B : schémas XML pour Papillon

1. Schéma général de Papillon

```
<!-- XML Schema for common elements of Papillon dictionary.  These
elements are used to encode Papillon monolingual dictionaries and
Papillon axies dictionary Namespace =
   http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/papillon.xsd
   $Author:  mangeot $ Mathieu MANGEOT-LEREBOURS Mathieu.Mangeot@imag.fr
   $Date:  2001/09/15 09:37:10 $
   $Revision:  1.14 $ -->
   <schema targetNamespace="http://www-clips.imag.fr/geta/services/dml">
     <annotation>
       <documentation xml:lang="en"> XML Schema for common elements
of Papillon dictionary.  These elements are used to encode Papillon
monolingual dictionaries and Papillon axies dictionary Namespace =
   http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/papillon.xsd
</documentation>
     </annotation>
   <!--================ Redefining elements of other schemas
================-->
   <!-- including dml schema for common DML elements used in the
dictionary -->
     <redefine
   schemaLocation="http://www-clips.imag.fr/geta/services/dml/dml.xsd">
   <!-- valueType type -->
   <!-- redefinition of the value of a function -->
       <complexType name="valueType" mixed="true">
         <complexContent>
           <extension base="d:valueType">
             <choice>
               <element ref="d:reflexie" minOccurs="0"/>
             </choice>
           </extension>
         </complexContent>
       </complexType>
     </redefine>
```

```
   <!--================ common definitions for monolingual dictionaries
=======-->
   <!-- Note:  the elements specific to a dictionary/language have to
be redefined in a specific schema -->
   <!-- lexie element -->
   <!-- A lexie is an entry of a Papillon monolingual dictionary.  The
structure of the articles, that is microstructure of the monolingual
dictionaries, is based on the structure used for the formal lexical
database DiCo of the OLST laboratory at Universite de Montreal.  The
encoding methodology is directly borrowed from the explanatory and
combinatorial lexicology, which is part of the meaning-text theory
elaborated by Igor Melc'uk and his colleagues.  -->
     <element name="lexie">
       <complexType>
         <sequence>
           <element ref="d:headword" minOccurs="1" maxOccurs="1"/>
           <group ref="d:language-specific" minOccurs="0" maxOccurs="1"/>
           <element ref="d:pronunciation" minOccurs="0" maxOccurs="1"/>
           <element ref="d:pos" minOccurs="1" maxOccurs="1"/>
           <element ref="d:language-levels" minOccurs="0" maxOccurs="1"/>
           <element ref="d:semantic-formula" minOccurs="1" maxOccurs="1"/>
           <element ref="d:government-pattern" minOccurs="1" maxOccurs="1"/>
           <element ref="d:lexical-functions" minOccurs="0" maxOccurs="1"/>
           <element ref="d:examples" minOccurs="0" maxOccurs="1"/>
           <element ref="d:full-idioms" minOccurs="0" maxOccurs="1"/>
           <element ref="d:axies"/>
         </sequence>
   <!-- The attribute id is an internal unique id.  It is hidden from
the users.  If the lexie is deleted, its id remains.  It can't be
reused.  It is metalinguistic information.  It has to be discussed if
it is necessary to write it in capital letters knowing that it does not
exist in Japanese.  -->
         <attribute ref="d:id" use="required"/>
   <!-- The attribute basic indicates if this lexie is the basic
lexical unit of the vocable.  It's boolean.  Its value is true or
false.  Information taken from DiCo.  -->
         <attribute name="basic" type="boolean" use="optional"/>
   <!-- The frequency is noted in another part and referenced with the
attribute id -->
       </complexType>
     </element>
   <!-- headword element -->
   <!-- is a common DML element already declared in the DML schema -->
   <!-- language-specific group -->
   <!-- This group has to be redefined in the schemas for the
monolingual dictionaries.  It contains all the language-specific
information.  -->
```

```
   <!-- due to a bug in XSV (redefinition is not implemented) I
change the content of the group article for validation <group
name='language-specific'> <all /> </group> -->
     <group name="language-specific">
       <sequence>
         <element ref="d:kun-yomi" minOccurs="0" maxOccurs="1"/>
         <element ref="d:on-yomi" minOccurs="0" maxOccurs="1"/>
       </sequence>
     </group>
     <element name="kun-yomi" type="string"/>
     <element name="on-yomi" type="string"/>
   <!-- pronunciation element -->
   <!-- is a common DML element already declared in the DML schema -->
   <!-- pos element -->
   <!-- is a common DML element already declared in the DML schema -->
   <!-- language-levels element -->
   <!-- It has to be redefined into the language specific schemas -->
     <element name="language-levels">
       <complexType>
         <sequence>
           <element ref="d:politeness" minOccurs="0" maxOccurs="1"/>
           <element ref="d:usage" minOccurs="0" maxOccurs="1"/>
           <element ref="d:reference" minOccurs="0" maxOccurs="1"/>
         </sequence>
       </complexType>
     </element>
   <!-- politeness element -->
   <!-- language level -->
     <element name="politeness">
       <complexType>
         <attribute name="grade" type="d:politenessType"/>
       </complexType>
     </element>
   <!-- politeness grade type -->
   <!-- has to be redefined into the language specific schemata -->
     <simpleType name="politenessType">
       <restriction base="string"/>
     </simpleType>
   <!-- usage element -->
   <!-- language level -->
     <element name="usage">
       <complexType>
         <attribute name="grade" type="d:usageType"/>
       </complexType>
     </element>
   <!-- usage grade type -->
   <!-- has to be redefined into the language specific schemata -->
```

```
    <simpleType name="usageType">
      <restriction base="string"/>
    </simpleType>
  <!-- reference element -->
  <!-- language level -->
    <element name="reference">
      <complexType>
        <attribute name="grade" type="d:referenceType"/>
      </complexType>
    </element>
  <!-- reference grade type -->
  <!-- has to be redefined into the language specific schemata -->
    <simpleType name="referenceType">
      <restriction base="string"/>
    </simpleType>
  <!-- semantic-formula element -->
  <!-- comes from the meaning-text theory -->
    <element name="semantic-formula">
      <complexType mixed="true">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="d:sem-label"/>
          <element ref="d:sem-actant"/>
          <element ref="d:actor"/>
        </choice>
      </complexType>
    </element>
  <!-- government-pattern element -->
  <!-- comes from the meaning-text theory -->
    <element name="government-pattern">
      <complexType>
        <sequence minOccurs="1" maxOccurs="unbounded">
          <element ref="d:mod"/>
        </sequence>
      </complexType>
    </element>
  <!-- mod element -->
  <!-- There might be more than one government pattern (we call them
"modifications") for the same lexical unit.  We need a way to encode
that.  -->
    <element name="mod">
      <complexType>
        <sequence minOccurs="1" maxOccurs="unbounded">
          <element ref="d:actor"/>
        </sequence>
  <!-- number of the modification in the government pattern -->
        <attribute name="nb" type="positiveInteger"/>
      </complexType>
```

```
    </element>
  <!-- sem-label element -->
  <!-- semantic label comes from the meaning-text theory.  Used to
tag the semantic formula.  We should define a closed list of possible
values -->
    <element name="sem-label" type="string"/>
  <!-- sem-actant element -->
  <!-- semantic actant comes from the meaning-text theory.  Used to
tag the semantic actant of a formula -->
    <element name="sem-actant" type="string"/>
  <!-- sem-variable element -->
  <!-- semantic variable comes from the meaning-text theory.  Used to
tag the semantic variable of a formula -->
    <element name="sem-variable" type="string"/>
  <!-- synt-actant element -->
  <!-- syntactic actant comes from the meaning-text theory.  Used to
tag the syntactic actant of a formula -->
    <element name="synt-actant" type="string"/>
  <!-- actor element -->
  <!-- comes from the meaning-text theory.  Used to tag the actors of
the semantic formula and the government pattern -->
    <element name="actor">
      <complexType mixed="true">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="d:sem-label"/>
          <element ref="d:sem-actant"/>
          <element ref="d:sem-variable"/>
          <element ref="d:synt-actant"/>
          <element ref="d:surface-group"/>
        </choice>
      </complexType>
    </element>
  <!-- surface-group element -->
  <!-- comes from the meaning-text theory.  Used to tag the syntactic
actant of a formula -->
    <element name="surface-group">
      <complexType mixed="true">
        <sequence minOccurs="1" maxOccurs="unbounded">
          <element ref="d:surface"/>
        </sequence>
      </complexType>
    </element>
  <!-- surface element -->
  <!-- comes from the meaning-text theory.  Used to tag the syntactic
actant of a formula -->
    <element name="surface">
      <complexType mixed="true">
```

```
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="d:pos"/>
          <element ref="d:reflexie"/>
        </choice>
      </complexType>
    </element>
  <!-- reflexie element -->
  <!-- reference to another lexie with an xlink -->
    <element name="reflexie" type="d:refType"/>
  <!-- lexical-functions element -->
  <!-- comes from the meaning-text theory Lists all the lexical
functions of the lexie -->   <element name="lexical-functions">
      <complexType>
        <sequence maxOccurs="unbounded">
          <element ref="d:function"/>
        </sequence>
      </complexType>
    </element>
  <!-- function element -->
  <!-- is a common DML element already declared in the DML schema -->
  <!-- comment element -->
  <!-- comment in a lexical function -->
    <element name="comment" type="string"/>
  <!-- fct-example element -->
  <!-- example in a lexical function -->
    <element name="fct-example" type="string"/>
  <!-- examples element -->
  <!-- lists some usage examples of a lexie -->
    <element name="examples">
      <complexType>
        <sequence maxOccurs="unbounded">
          <element ref="d:example"/>
        </sequence>
      </complexType>
    </element>
  <!-- example element -->
  <!-- a usage example of a lexie -->
    <element name="example">
      <complexType mixed="true">
        <attribute ref="d:id" use="optional"/>
        <attribute ref="d:lang"/>
        <attribute ref="xlink:href" use="optional"/>
      </complexType>
    </element>
  <!-- examples element -->
  <!-- lists some full idioms containing the lexie -->
    <element name="full-idioms">
```

```
      <complexType>
        <sequence maxOccurs="unbounded">
          <element ref="d:idiom"/>
        </sequence>
      </complexType>
    </element>
  <!-- idiom element -->
  <!-- a full idioms containing the lexie -->
    <element name="idiom">
      <complexType mixed="true">
        <attribute ref="d:id" use="optional"/>
        <attribute ref="xlink:href" use="optional"/>
      </complexType>
    </element>
  <!-- axies element -->
  <!-- lists all the references to axies Normally, a lexie should be
linked to only one axie.  Non conforming cases should be signalled to
the lexicologists.  -->
    <element name="axies">
      <complexType>
        <sequence maxOccurs="unbounded">
          <element ref="d:refaxie"/>
        </sequence>
      </complexType>
    </element>
  <!-- refaxie element -->
  <!-- references to an axie with an xlink.  The link can be tagged
with a gloss.  -->
    <element name="refaxie" type="d:refType"/>
  <!--================ definitions for the axie dictionary
==================-->
  <!-- Note:  elements whose content can vary e.g.:
external-references, have to be redefined in the schema specific to
the axie volume -->
  <!-- axie element -->
  <!-- An axie is an interlingual link between lexies of different
languages.  It consists also in links to other set of semantic symbols
following other theories like WordNet, UNL, etc.  -->
    <element name="axie">
      <complexType>
        <sequence minOccurs="0">
          <element ref="d:semantic-cat" minOccurs="0" maxOccurs="1"/>
          <group ref="d:language-links" minOccurs="0" maxOccurs="1"/>
          <element ref="d:refinements" minOccurs="0" maxOccurs="1"/>
          <element ref="d:generalizations" minOccurs="0" maxOccurs="1"/>
          <element ref="d:synonyms" minOccurs="0" maxOccurs="1"/>
          <element ref="d:external-references" minOccurs="0" maxOccurs="1"/>
```

```
        </sequence>
        <attribute ref="d:id" use="required"/>
        <attribute ref="d:history" use="optional"/>
        <attribute ref="d:status" use="optional"/>
      </complexType>
    </element>
  <!-- semantic-cat element -->
  <!-- semantic category of an axie -->
    <element name="semantic-cat">
      <simpleType>
        <restriction base="string">
          <enumeration value="entity"/>
          <enumeration value="process"/>
          <enumeration value="result"/>
          <enumeration value="state"/>
          <enumeration value="gloss"/>
          <enumeration value="idiom"/>
          <enumeration value="citation"/>
          <enumeration value="proverb"/>
        </restriction>
      </simpleType>
    </element>
  <!-- language-links group -->
  <!-- This group contains the links from the axie to the monolingual
lexies.  All the links to lexies of one language are grouped into a
language element.  It has to be redefined in the schema specific to the
axies volume -->
    <group name="languages-links">
      <all/>
    </group>
  <!-- refinements element -->
  <!-- lists all the axies that refines this axie -->
    <element name="refinements" type="d:refaxiesType"/>
  <!-- generalizations element -->
  <!-- lists all the axies that generalize this axie -->
    <element name="generalizations" type="d:refaxiesType"/>
  <!-- synonyms element -->
  <!-- lists all the axies synonyms of this axie -->
    <element name="synonyms" type="d:refaxiesType"/>
  <!-- refs type -->
  <!-- lists all the axies synonyms of this axie -->
    <complexType name="refsType">
      <sequence>
        <element ref="d:reflexie" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="d:refexample" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="d:refidiom" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
```

```
    </complexType>
    <element name="refexample" type="d:refType"/>
    <element name="refidiom" type="d:refType"/>
  <!-- refaxies type -->
  <!-- type that refers to another axie with an xlink -->
    <complexType name="refaxiesType">
      <sequence minOccurs="0" maxOccurs="unbounded">
        <element ref="d:refaxie"/>
      </sequence>
    </complexType>
  <!-- external-references group -->
  <!-- lists the external references of an axie.  It has to be
redefined in the schema specific to the axies volume -->
    <group name="external-references">
      <all/>
    </group>
  </schema>
```

2. Schéma du volume Papillon axies

```
<!-- XML Schema for Papillon axies volume used as pivot dictionary for
the Papillon lexical database.  Namespace =
   http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/papillon-axi.xsd
   $Author:  mangeot $ Mathieu MANGEOT-LEREBOURS Mathieu.Mangeot@imag.fr
   $Date:  2001/09/15 09:37:10 $
   $Revision:  1.14 $ -->
   <schema targetNamespace="http://www-clips.imag.fr/geta/services/dml">
     <annotation>
       <documentation xml:lang="en"> XML Schema for Papillon axies
volume used as pivot dictionary for the Papillon lexical database.
Namespace = http://www-clips.imag.fr/geta/services/dml This schema is
identified by the location:
   http://www-clips.imag.fr/geta/services/dml/papillon-axi.xsd
</documentation>
     </annotation>
   <!--================ Redefining elements of Papillon common schema
========-->
   <!-- including schema for common papillon elements used in the
dictionary and redefining some groups The content of these groups can
evolve.  -->
     <redefine
   schemaLocation="http://www-clips.imag.fr/geta/services/dml/papillon.xsd">
   <!-- article group -->
   <!-- An article of the Papillon interlingual pivot dictionary is an
axie -->
       <group name="article">
         <sequence>
           <element ref="d:axie" minOccurs="1" maxOccurs="1"/>
         </sequence>
       </group>
   <!-- languages-links group -->
   <!-- this group contains the links from the axie to the monolingual
lexies.  All the links to lexies of one language are grouped into
a language element.  If a new language is added to the Papillon
dictionary, a new group will be added here.  -->
```

```
      <group name="languages-links">
        <sequence>
          <element ref="d:eng" minOccurs="0" maxOccurs="1"/>
          <element ref="d:fra" minOccurs="0" maxOccurs="1"/>
          <element ref="d:jpn" minOccurs="0" maxOccurs="1"/>
          <element ref="d:lao" minOccurs="0" maxOccurs="1"/>
          <element ref="d:tha" minOccurs="0" maxOccurs="1"/>
        </sequence>
      </group>
   <!-- this group links all the external references of an axie.  If
a new external reference is added, a new group will be defined there.
-->
      <group name="external-references">
        <sequence>
          <element ref="d:UNL-graph" minOccurs="0" maxOccurs="unbounded"/>
          <element ref="d:UNL" minOccurs="0" maxOccurs="1"/>
          <element ref="d:WordNet" minOccurs="0" maxOccurs="1"/>
          <element ref="d:NTTsemcat" minOccurs="0" maxOccurs="1"/>
          <element ref="d:LexiGuideConcept" minOccurs="0" maxOccurs="1"/>
        </sequence>
      </group>
    </redefine>
   <!--================ definitions for the language-links
==================-->
   <!-- eng element -->
   <!-- links to english lexies corresponding to this axie -->
     <element name="eng" type="d:refsType"/>
   <!-- fra element -->
   <!-- links to french lexies corresponding to this axie -->
     <element name="fra" type="d:refsType"/>
   <!-- jpn element -->
   <!-- links to japanese lexies corresponding to this axie -->
     <element name="jpn" type="d:refsType"/>
   <!-- lao element -->
   <!-- links to lao lexies corresponding to this axie -->
     <element name="lao" type="d:refsType"/>
   <!-- tha element -->
   <!-- links to thai lexies corresponding to this axie -->
     <element name="tha" type="d:refsType"/>
   <!-- vie element -->
   <!-- links to vietnamese lexies corresponding to this axie -->
     <element name="vie" type="d:refsType"/>
   <!--================ definitions for the external-references
===============-->
   <!-- resource attribute -->
   <!-- To locate the file where the external reference is stored -->
     <attribute name="resource" type="xlink:href"/>
```

```
   <!-- UNL-graph element -->
   <!-- To encode a UNL-graph representing an example -->
   <!-- eg:  La mésentente pourrait être le mobile du
meurtre.  <UNL-graph> obj(be(icl>state).@entry,mobile.@def)
aoj(mobile.@def,murder.@def) agt(be(icl>state).@entry,misunderstanding.@def)
mod(can.@conditional,be.@entry) </UNL-graph> -->
     <element name="UNL-graph" type="string"/>
   <!-- UWs element -->
   <!-- List of UNL UWs corresponding to the axie.  -->
     <element name="UWs">
       <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
           <element ref="d:refuw"/>
         </sequence>
         <attribute ref="d:resource"/>
       </complexType>
     </element>
   <!-- refuw element -->
   <!-- Represents a UW corresponding to the axie.  -->
   <!-- eg:  <refuw xlink:href="river(icl>not-into-sea)" /> -->
     <element name="refuw" type="d:refType"/>
   <!-- WordNet element -->
   <!-- List of WordNet synsets corresponding to the axie.  -->
     <element name="WordNet">
       <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
           <element ref="d:refsynset"/>
         </sequence>
         <attribute ref="d:resource"/>
       </complexType>
     </element>
   <!-- refsynset element -->
   <!-- Represents a synset corresponding to the axie.  -->
   <!-- eg:  synset for murder <refsynset xlink:href="00143589">00143589
04 n 03 murder 0 homicide 0 slaying 0 013 @ 00142607 n 0000   00143925
n 0000   00144191 n 0000   00145467 n 0000   00145676 n 0000   00808183
n 0000   00809697 n 0000   00812126 n 0000   00812230 n 0000   00812369
n 0000   00812446 n 0000   00812548 n 0000   00812622 n 0000 | unlawful
premeditated killing of a human being</refsynset> -->
     <element name="refsynset" type="d:refType"/>
   <!-- NTTsemcat element -->
   <!-- List of Nippon Telegraph and Telephone semantic categories
corresponding to the axie.  -->
     <element name="NTTsemcat">
       <complexType>
         <sequence minOccurs="0" maxOccurs="unbounded">
           <element ref="d:refsc"/>
```

```
        </sequence>
        <attribute ref="d:resource"/>
      </complexType>
    </element>
  <!-- refsc element -->
  <!-- Represents a semantic category corresponding to the axie.  -->
    <element name="refsc" type="d:refType"/>
  <!-- NTTsemcat element -->
  <!-- List of LexiGuide concepts from LexiQuest corresponding to the
axie.  -->
    <element name="LexiGuideConcepts">
      <complexType>
        <sequence minOccurs="1" maxOccurs="unbounded">
          <element ref="d:reflx"/>
        </sequence>
        <attribute ref="d:resource"/>
      </complexType>
    </element>
  <!-- reflx element -->
  <!-- Represents a LexiGuide concept corresponding to the axie.  -->
    <element name="reflx" type="d:refType"/>
  </schema>
```

3. Schéma de Papillon français

```
<!-- XML Schema for Papillon French lexies volume for the Papillon
lexical database.  In this schema are define all the language specific
elements like parts-of-speech, etc.  Namespace =
   http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/papillon-fra.xsd
   $Author:  mangeot $ Mathieu MANGEOT-LEREBOURS Mathieu.Mangeot@imag.fr
   $Date:  2001/09/15 09:37:10 $
   $Revision:  1.14 $ -->
   <schema targetNamespace="http://www-clips.imag.fr/geta/services/dml">
     <annotation>
       <documentation xml:lang="en"> XML Schema for
Papillon French lexies volume for the Papillon lexical
database.  In this schema are define all the language
specific elements like parts-of-speech, etc.  Namespace =
http://www-clips.imag.fr/geta/services/dml This schema is identified by
the location:  http://www-clips.imag.fr/geta/services/dml/papillon-fra.xsd
</documentation>
     </annotation>
   <!--================ Redefining elements of Papillon common schema
========-->
     <redefine
   schemaLocation="http://www-clips.imag.fr/geta/services/dml/papillon.xsd">
   <!-- article group -->
   <!-- An article of the Papillon French volume is an lexie -->
       <group name="article">
         <sequence>
           <element ref="d:lexie" minOccurs="1" maxOccurs="1"/>
         </sequence>
       </group>
   <!-- language-specific group -->
   <!-- Here are defined the elements specific to the French language
-->
       <group name="language-specific">
         <sequence/>
       </group>
   <!-- posType type -->
```

```
   <!-- Here are defined the parts-of-speech of the French language -->
       <simpleType name="posType">
         <restriction base="d:posType">
   <!-- nom commun masculin (noun masculine) -->
           <enumeration value="n.m."/>
   <!-- nom commun masculin invariable (noun masculine invariable) -->
           <enumeration value="n.m.  inv."/>
   <!-- nom commun masculin pluriel (noun masculine plural) -->
           <enumeration value="n.m.  pl."/>
   <!-- nom commun masculin et/ou féminin ???  (noun masculine and/or
feminine ???)  -->
           <enumeration value="n.m./f."/>
   <!-- nom commun masculin et/ou féminin ???  (noun masculine and/or
feminine ???)  -->
           <enumeration value="n.m., f."/>
   <!-- nom commun féminin (noun feminine) -->
           <enumeration value="n.f."/>
   <!-- nom commun féminin pluriel (noun feminine plural) -->
           <enumeration value="n.f.  pl."/>
   <!-- nom propre masculin (proper name masculine) -->
           <enumeration value="Pr.m."/>
   <!-- nom propre féminin (proper name feminine) -->
           <enumeration value="Pr.f."/>
   <!-- nom propre masculin pluriel (proper name masculine plural) -->
           <enumeration value="Pr.m.pl."/>
   <!-- nom propre féminin (proper name feminine plural) -->
           <enumeration value="Pr.f.pl."/>
   <!-- abréviation masculin (abbreviation masculine) -->
           <enumeration value="abr.m."/>
   <!-- abréviation féminin (abbreviation feminine) -->
           <enumeration value="abr.f."/>
   <!-- verbe transitif (transitive verb) -->
           <enumeration value="v.tr."/>
   <!-- verbe intransitif (intransitive verb) -->
           <enumeration value="v.intr."/>
   <!-- verbe pronominal (pronominal verb) -->
           <enumeration value="v.pr."/>
   <!-- adjectif (adjective) -->
           <enumeration value="a."/>
   <!-- adverbe (adverb) -->
           <enumeration value="adv."/>
   <!-- déterminant -->
           <enumeration value="det."/>
   <!-- conjonction -->
           <enumeration value="conj."/>
   <!-- pronom -->
           <enumeration value="pron."/>
```

```
<!-- interjection -->
        <enumeration value="intj."/>
<!-- préposition -->
        <enumeration value="prep."/>
<!-- locution -->
        <enumeration value="loc."/>
<!-- locution adjectivale -->
        <enumeration value="loc.  adj."/>
<!-- locution prépositionnelle -->
        <enumeration value="loc.  prep."/>
<!-- locution adverbiale -->
        <enumeration value="loc.  adv."/>
<!-- locution nominale -->
        <enumeration value="loc.  nom."/>
      </restriction>
    </simpleType>
  </redefine>
</schema>
```

4. Schéma de Papillon japonais

```
<!-- XML Schema for Papillon Japanese lexies volume for the Papillon
lexical database.  In this schema are define all the language-specific
elements like parts-of-speech, numeric specifiers, etc.  Namespace =
   http://www-clips.imag.fr/geta/services/dml
   This schema is identified by the location:
   http://www-clips.imag.fr/geta/services/dml/papillon-jpn.xsd
   $Author:  mangeot $ Mathieu MANGEOT-LEREBOURS Mathieu.Mangeot@imag.fr
   $Date:  2001/09/15 09:37:10 $
   $Revision:  1.14 $ -->
   <schema targetNamespace="http://www-clips.imag.fr/geta/services/dml">
     <annotation>
       <documentation xml:lang="en"> XML Schema for Papillon
Japanese lexies volume for the Papillon lexical database.  In
this schema are define all the language-specific elements
like parts-of-speech, numeric specifiers, etc.  Namespace =
http://www-clips.imag.fr/geta/services/dml This schema is identified by
the location:  http://www-clips.imag.fr/geta/services/dml/papillon-jpn.xsd
</documentation>
     </annotation>
   <!--================ Redefining elements of Papillon common schema
========-->
     <redefine
   schemaLocation="http://www-clips.imag.fr/geta/services/dml/papillon.xsd">
   <!-- article group -->
   <!-- An article of the Papillon Japanese volume is an lexie -->
       <group name="article">
         <sequence minOccurs="1" maxOccurs="1">
           <element ref="d:lexie"/>
         </sequence>
       </group>
   <!-- language-specific group -->
   <!-- Here we define the elements specific to the Japanese language
-->
       <group name="language-specific">
         <sequence>
           <element ref="d:kun-yomi" maxOccurs="unbounded"/>
           <element ref="d:on-yomi" maxOccurs="unbounded"/>
```

```
   <!--Maybe this should be put in the general lexie structure -->
           <element ref="d:language-levels" minOccurs="0" maxOccurs="1"/>
         </sequence>
       </group>
   <!-- posType type -->
   <!-- Here are defined the parts-of-speech of the Japanese language
-->
       <simpleType name="posType">
         <restriction base="d:posType">
   <!-- settôgo,prefix-->
           <enumeration value="接頭語"/>
   <!-- setsubigo, suffix-->
           <enumeration value=" 接尾語 "/>
   <!-- joshûshi, numeral-->
           <enumeration value=" 助数詞 "/>
   <!-- zôgoseibun, productive element-->
           <enumeration value="造語成分"/>
   <!-- meishi, noun-->
           <enumeration value="  名詞  "/>
   <!-- keishikimeishi, formal noun-->
           <enumeration value="形式名詞"/>
   <!-- daimeishi, pronoun-->
           <enumeration value=" 代名詞 "/>
   <!-- rentaishi, demonstrative-->
           <enumeration value=" 連体詞 "/>
   <!-- fukushi, adverb-->
           <enumeration value="  副詞  "/>
   <!-- setsuzokushi, conjunction-->
           <enumeration value=" 接続詞 "/>
   <!-- kandôshi, interjection-->
           <enumeration value=" 感動詞 "/>
   <!-- jidôshi, intransitive verb-->
           <enumeration value=" 自動詞 "/>
   <!-- tadôshi, transitive verb-->
           <enumeration value=" 他動詞 "/>
   <!-- keiyôshi, adjective -->
           <enumeration value=" 形容詞 "/>
   <!-- keiyôdôshi, adjectival verb-->
           <enumeration value="形容動詞"/>
   <!-- jodôshi, auxiliary-->
           <enumeration value=" 助動詞 "/>
   <!-- kakujoshi, case postposition-->
           <enumeration value=" 格助詞 "/>
```

```
   <!-- setsuzokujoshi, conjonctive postposition-->
          <enumeration value="接続助詞"/>
   <!-- fukujoshi, adverbial postposition-->
          <enumeration value="副助詞"/>
   <!-- kakarijoshi, topic postposition-->
          <enumeration value="係助詞"/>
   <!-- sûjoshi, sentence final postposition-->
          <enumeration value="終助詞"/>
   <!-- kantôjoshi, emotional postposition-->
          <enumeration value="間投助詞"/>
   <!-- heiretsujoshi, connective postposition-->
          <enumeration value="並列助詞"/>
   <!-- juntaijoshi, nominalisation postposition-->
          <enumeration value="準体助詞"/>
   <!-- hojodôshi, complementary verb-->
          <enumeration value="補助動詞"/>
   <!-- hojokeiyôshi, complementary adjective-->
          <enumeration value="補助形容詞"/>
   <!-- makuraji, head word-->
          <enumeration value="枕詞"/>
   <!-- rengo, mot-valise-->
          <enumeration value="連語"/>
   <!-- ku, clause-->
          <enumeration value="句"/>
        </restriction>
      </simpleType>
    </redefine>
   <!--================ Special elements of Papillon Japanese schema
========-->
   <!-- kun-yomi element -->
   <!-- element specific to the Japanese language note the writing of
the kanjis used in the headword -->
    <element name="kun-yomi" type="string"/>
   <!-- on-yomi element -->
   <!-- element specific to the Japanese language note the writing of
the kanjis used in the headword -->
    <element name="on-yomi" type="string"/>
   <!-- numerical specifiers -->
   <!-- Here are defined the numerical specifiers of the Japanese
language.  It has to be checked by Francis Bond, Yves Lepage, Jim
Breen, etc.  Specifiers are no longer an element.  They are noted as
values of the lexical function "synt" -->
    <simpleType name="numSpecifiersType">
      <restriction base="string">
```

```
<!--かい、 kai|Objects : for the stories of a building -->
      <enumeration value="階"/>

<!--こ、 ko|Used for a broad category of small and compact objects,
including round fruit, balls, boxes, etc. -->
      <enumeration value="個"/>

<!--さつ、 satsu|Objects : for bound objects such a books, notebooks,
magazines, etc. -->
      <enumeration value="冊"/>

<!--そく、 soku|Objects : for pairs of shoes, socks, stockings, etc. -->
      <enumeration value="足"/>

<!--だい、 dai|Objects : for vehicles, machines and things such as bicycles
and televion sets -->
      <enumeration value="台"/>

<!--つう、 tû|Objects : for letters and documents -->
      <enumeration value="通"/>

<!--はい、 hai|Objects : for liquide in cups, glasses, bowls, buckets, etc.
-->
      <enumeration value="杯"/>

<!--ほん、 hon|Objects : for long cylindrical objects including trees,
sticks, pens, bananas, fingers, etc. -->
      <enumeration value="本"/>

<!--まい、 mai|Objects : for flat, thin objects including paper, dishes,
stamps, blakets, boards, etc. -->
      <enumeration value="枚"/>


<!--えん、 yen|Currency  -->
      <enumeration value="円"/>

<!--cent|Currency  -->
      <enumeration value="セント"/>

<!--German mark|Currency  -->
      <enumeration value="ドイツマルク"/>

<!--dollar|Currency  -->
      <enumeration value="ドル"/>

<!--pound|Currency  -->
      <enumeration value="ポンド"/>

<!--franc|Currency  -->
      <enumeration value="フラン"/>

<!--kiro|Measuring units : used for both kilometers and kilograms  -->
      <enumeration value="キロ"/>

<!--gram|Measuring units -->
      <enumeration value="グラム"/>

<!--centimeter|Measuring units -->
      <enumeration value="センチ"/>
```

```
<!--litter|Measuring units -->
      <enumeration value="リットル"/>

<!--ひき、 Hiki|Animal world : for insects, fish, small animals such as
cets and dogs -->
      <enumeration value="匹"/>

<!--とう、 tô|Animal world : for large animals such as horses, bears, deer,
etc. -->
      <enumeration value="頭"/>

<!--わ、 wa|Animal world : for birds -->
      <enumeration value="羽"/>

<!--かい、 kai|Frequency : times -->
      <enumeration value="回"/>

<!--ど、 do|Frequency : times -->
      <enumeration value="度"/>

<!--ばん、 ban | order : times -->
      <enumeration value="番"/>

<!-- ばんめ、 banme|order : -th -->
      <enumeration value="番目"/>

<!--とう、 tô|order  -->
      <enumeration value="等"/>



<!--じかん、 jikan|duration : hour -->
      <enumeration value="時間"/>

<!--しゅうかん、 sûkan|duration : week -->
      <enumeration value="週間"/>

<!--ふん、 funkan|duration : minute -->
      <enumeration value="分"/>

<!--ふんかん、 funkan|duration : during x minute -->
      <enumeration value="分間"/>

<!--びょう、 byô|duration : second -->
      <enumeration value="秒"/>

<!--びょうかん、 byôkan|duration : during x second -->
      <enumeration value="秒 間"/>

<!--ねんかん、 nenkan|duration : during x year -->
      <enumeration value="年 間"/>

<!--ひとり、 hitori|People : one person -->
      <enumeration value="一人"/>

<!--ふたり、 futari|People  -->
      <enumeration value="二人"/>

<!--にん、 nin|People : two persons -->
      <enumeration value="人"/>
    </restriction>
  </simpleType>
</schema>
```

Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue

Résumé - Les besoins croissants en ressources lexicales et le succès des projets de développement coopératif comme LINUX convergent vers l'idée d'accumuler des données lexicales multilingues de grande taille et de grande richesse par construction coopérative sur la Toile et utilisation "mutualisée". Les contributeurs fourniraient eux-mêmes ces informations sous une forme standardisée grâce à un environnement adapté.

L'étude du contexte actuel de la dictionnairique nous a conduit à l'identification de problèmes difficiles tels que la structuration et la manipulation de données hétérogènes, la visualisation d'une grande quantité de données lexicales multilingues et la construction en coopération par des personnes aux compétences diverses.

Des prototypages et des expérimentations portant sur la consultation de ressources hétérogènes, l'enrichissement et personnalisation du résultat, la construction de ressources en ligne et la rédaction d'articles avec un éditeur standard nous ont permis de résoudre séparément ces problèmes.

Cela nous a permis de concevoir un environnement complet de "bases lexicales" répondant à tous ces problèmes se plaçant au dessus des SGBD utilisés pour le stockage et intégrant un serveur pour la construction coopérative. Son noyau inclut un formalisme générique de définition de structures lexicales inspiré de SUBLIM de G. Sérasset, mais complété et réexprimé en XML.

Cet environnement est actuellement appliqué au projet Papillon de développement par des bénévoles sur Internet d'une base lexicale comprenant cinq langues. L'architecture de la base est constituée d'un dictionnaire monolingue pour chaque langue et d'un dictionnaire pivot d'acceptions interlingues reliant les articles monolingues (lexies) dont la structure provient de la lexicologie combinatoire.

Enfin, l'architecture du serveur assez générique devrait être réutilisée dans d'autres contextes (mémoires de traduction, outils pour traducteurs, communication et RI multilingue, annotations multimédia).

Mots-Clés : Lexicologie, lexicographie, dictionnairique, bases lexicales multilingues, schémas XML, serveurs lexicaux

Centralised and Distributed Environments for Lexicographers & Lexicologists in Multilingual Context

Abstract - The growing needs in lexical resources and the success of the cooperative development projects such as LINUX lead to the idea of accumulating large amounts of very rich multilingual lexical data by cooperative construction on the Web and "mutualized" use. Contributions to data improvement would be standardized and made available thanks to an adapted environment.

While studying of the current context of the dictionaries domain, we were led to identifying difficult problems such as heterogeneous data structuring and manipulation, as well as large amount of multilingual lexical data and visualization or construction in cooperation by people with different skills.

Prototypes and experiments on consultation of heterogeneous resources, enrichment and personalization of the result, on-line resource building, and entries writing with a standard editor enabled us to solve these problems separately.

It allowed us to design a complete lexical databases environment addressing all these problems as a specific layer directly above the DBMS tools that integrates a server for cooperative building. Its kernel includes a generic formalism for the definition of lexical structures derived from SUBLIM of G. Sérasset, but extended and translated in XML.

This environment is currently applied to the Papillon project which aims at building/developing of a five-language lexical database by voluntary contributors on the Internet. The architecture of the database is made up of a monolingual dictionary for each language and a pivot dictionary of interlingual acceptions (axies) linking the monolingual entries (lexies) which structure comes from the domain of explanatory and combinatory lexicology. Extra languages are planned to be added soon.

The architecture of the server is quite generic and could be reused rapidly in other contexts (translation memories and tools for translators, communication and multilingual IR, multimedia annotations).

Keywords: Lexicology, Computational Lexicography, Dictionaries, Multilingual Lexical Databases, XML Schemata, Lexical Servers