Outils pour lexicographes naïfs (en informatique). Mémoire de DEA en informatique

Mathieu Mangeot

Université Joseph Fourier
U.F.R Informatique &
Mathématiques Appliquées

Institut National Polytechnique
de Grenoble

ENSIMAG

I M A G

DEA D'INFORMATIQUE

SYSTÈMES ET COMMUNICATIONS

Projet présenté par

Mathieu MANGEOT LEREBOURS

OUTILS POUR LEXICOGRAPHES NAÏFS (EN INFORMATIQUE)

EFFECTUE AU LABORATOIRE CLIPS (Équipe GETA)

Date : 19 juin 1997
Jury : Didier Bert
Christian Boitet
Damien Genthial
Sacha Krakowiak
Gilles Sérasset

Remerciements

Je remercie tout d’abord Christian Boitet de m’avoir proposé d’effectuer mon stage dans son laboratoire. J’espère avoir été digne de la confiance qu’il m’a accordé.

Je remercie aussi Gilles pour toute l’aide qu’il m’a apportée sans faillir quand j’en avais besoin et pour toute la patience dont il a fait preuve à mon égard.

Je remercie Mathieu pour l’aide technique qu’il m’a fournie grâce à ses travaux sur les projets Fe*.

Enfin, je remercie toute l’équipe du GETA qui a du me supporter durant tout au long de mon stage.

Table des matières

Introduction

En Traitement Automatique des Langues Naturelles (TALN), le problème de la gestion des ressources linguistiques est crucial. Le volume des données manipulées, leur grande variété et la vitesse de traduction sont autant de paramètres qui font de la construction de dictionnaires un élément clé de tout système de TALN. La dispersion des outils sur des plateformes hétérogènes dont ont besoin les lexicographes lors de l'indexation et le coût élevé qui en résulte freinent les avancées dans ce domaine.

Beaucoup d'efforts ont été faits pour essayer de créer une plateforme unique qui réduirait les coûts de production des dictionnaires mais assez peu de résultats ont été obtenus. D'autre part, pour le projet Universal Networking Language (UNL), nous devons faire face à des besoins très forts. À court terme des outils d'indexation pour construire les dictionnaires seront nécessaires au projet. Nous pensons qu'il est possible de résoudre les problèmes de dispersion des outils en proposant une application générique multi-outils. Nous pourrons l'expérimenter avec les besoins du projet UNL.

Dans une première partie, nous présenterons les besoins auxquels nous sommes confrontés et définissons le cahier des charges pour y répondre. La deuxième partie est consacrée à un état de l'art des différentes structures de dictionnaires et de quelques outils pour lexicographes puis à une étude sur des outils d'intégration d'application. Nous décrirons dans la troisième partie les différentes solutions envisagées ainsi que les spécifications de la partie que nous avons décidé d'implémenter. Dans la dernière partie, nous expliquerons d'abord plus en détail le projet UNL puis nous présenterons la composition de notre application.

Partie 1 : Notre Contexte

1. Généralités

Pour la construction et l'utilisation d'un poste de travail pour lexicographes, on trouve différentes personnes :

• le lexicologue définit les informations qui seront contenues dans le lexique, spécifie leur forme et donne les critères permettant de définir les unités du lexique.

• l'informaticien crée les outils spécifiques au lexique ainsi défini et met au point la méthodologie qui sera utilisée lors de la construction du lexique. Il construit de plus les interfaces nécessaires au lexicographe.

• le lexicographe construit le lexique selon les spécifications ainsi faites. Il va construire les unités du lexique et/ou compléter des unités déjà existantes.

La création d'outils pour le lexique pose des problèmes informatiques forts du fait de la masse 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 pour 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 tenir compte de l'aspect distribué du travail de lexicographie.

Pour construire une entrée de dictionnaire, le lexicographe doit rechercher les différents usages de l'entrée donnée. Il ne peut se contenter de son intuition linguistique. Cette validation passe par une recherche d'informations différentes dans des sources différentes (corpus, dictionnaires existants, locuteur natif...). À l'heure actuelle, ces sources sont souvent dispersées sur différentes plateformes logicielles et matérielles (lorsqu'elles sont disponibles sous forme informatique). Un outil idéal de lexicographie doit pouvoir intégrer les différentes sources d'information brutes sur la plateforme d'édition.

Enfin, 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 ajout ou un retrait d'information, ou la modification de valeurs possibles. Cette évolution de la structure du dictionnaire oblige à une reprise de tout ou une partie des données déjà indexées. L'informaticien doit pouvoir fournir des outils permettant d'automatiser (au moins partiellement) ce travail de révision. Enfin, la modification de la structure du dictionnaire peut se traduire par un changement des interfaces d'édition dont dispose le lexicographe 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.

2. Nos besoins

À l'heure actuelle, la construction de dictionnaires usuels ou d'applications nécessite des outils pour fabriquer de très grandes Bases de Données Lexicales Multilingues (BDLM). Le besoin de ce genre d'outils est général sur la planète comme en témoigne la vivacité du groupe de discussion e-lex. Beaucoup de propositions sont implémentées à grands frais mais peu acceptées en pratique.

L'expérience du GETA en lexicographie professionnelle a débouché sur la conception de nombreux outils spécialisés (ARIANE, ATLAS, BDTAO). Par la suite, l'expérience de l'indexation lors du projet FeM nous a montré qu'un lexicographe non spécialisé en informatique peut indexer avec des outils simples. C'est donc un point positif pour réduire les coûts de production. Enfin, la décision récente de construire de grosses BDLM pour le projet UNL par exemple montre bien l'évolution des besoins en lexicographie.

D'après C. Boitet, « Sachant que si l'on part de zéro, l'indexage d'un terme dans une base lexicale prend environ trente minutes en moyenne, il est crucial d'améliorer la productivité des lexicographes et de leur fournir un outil convivial pour enrichir à plusieurs une base lexicale multilingue accessible par réseau. » Il est maintenant devenu nécessaire d'industrialiser le travail de lexicographie et indispensable de réduire les coûts de production.

Pour cela, nous utiliserons des outils de travail sur postes de bas de gamme (PC avec Word95™). De plus, même si d'importants progrès ont été faits dans la récupération de données, il faudra toujours les compléter à la main.

3. Présentation du projet UNL

Pas moins de 3000 langues sont parlées sur terre aujourd'hui. Plus d'une trentaine sont parlées par plus de 100 millions de personnes incluant le chinois, l'anglais, l'espagnol, le portugais, l'arabe, l'indonésien, le français, le russe, l'allemand, et le japonais.

Avec les récents progrès des technologies de l'information, le volume des communications internationales augmente. Cependant, les différences de langues sont une barrière pour le flot d'informations dans notre société.

Le projet Universal Networking Language (UNL) [21, 22] est un projet de l'Université des Nations Unies (UNU) à Tokyo et de l'institut des études avancées (IAS). UNL est la langue pivot de tous les systèmes de traduction automatique du projet. Le principe de ce projet est la communication multilingue interpersonnelle par Internet.

Un auteur s'exprime dans sa langue. Son texte est transformé en une description sémantique avant d'être envoyé sur le réseau. Lorsqu'un lecteur souhaite lire ce texte, il est généré dans sa langue.

L'information encodée en UNL est convertie en une contrepartie équivalente écrite en langue cible à l'aide d'un générateur pour chaque langue. Un éditeur appelé « convertisseur » traduit l'information écrite dans une langue donnée en information au format UNL. La conversion en UNL est réalisée par la désambiguisation interactive avec l'usager. Le langage existe seulement sur le réseau d'information.

L'information décrite en UNL apparaîtra dans les pages d'accueil ou sera stockée dans des archives. Les documents UNL seront échangés à travers le réseau et stockées sur les serveurs WWW. Ils peuvent aussi être échangés par FTP.

Dans les communications à l'aide d'internet, HTML est le format dominant. De plus, le document entier est constitué de texte intégral. Pour que l'information codée en UNL soit manipulée par n'importe quel système dans le monde, un format de description pour les expressions UNL est proposé comme une extension de la convention HTML. L'information UNL peut être inclue dans un document HTML avec des étiquettes qui spécifient le début et la fin de l'information UNL attachée.

4. Cahier des charges de l'outil nécessaire

4.1. Objectifs

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 soucis de rentabilité. Sans contrainte temporelle, on peut donc attacher plus d'importance à la conception d'un tel outil.

Il devra être capable, par contre, 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 de l'outil nous permet de rester à un niveau générique et de ne pas se spécialiser pour une tâche particulière.

4.2. Construire l'outil autour d'un noyau

Pour répondre au besoin d'un poste de travail générique, il nous a paru logique d'étendre les possibilités du système de gestion de bases lexicales SUBLIM conçu dans cette optique de généricité.

Il est donc nécessaire de considérer ce système comme noyau d'un poste de travail plus complet et aisément paramétrable, et de lui ajouter une interface utilisable par un linguiste.

4.3. Concevoir un outil modulable

SUBLIM ne parle pas d'intégration d'autres outils nécessaires à l'indexation d'une base lexicale. Pour répondre à cet autre besoin, le poste de travail doit être conçu de manière à gérer l'intégration d'outils tels que l'analyse de corpus, la récupération de données ou des modules plus spécifiques.

4.4. Adapter l'outil aux changements

Cet outil devra servir essentiellement à des travaux de recherche en lexicologie et lexicographie. Il faut donc penser à l'adapter à des changements fréquents de formats de données ou de types de structures de représentation par exemple et inclure la possibilité de gérer des versions différentes.

4.5. L'outil doit pouvoir être géré par un non informaticien

Le système SUBLIM se situe à un niveau encore trop théorique et informatique pour pouvoir être utilisé en tant que système par des linguistes. Il faut donc adapter les interfaces pour que les linguistes puissent effectuer les changements qu'ils estiment nécessaires sans l'aide d'un informaticien.

4.6. Tenir compte de l'aspect distribué

Nous l'avons vu plus haut, le travail d'indexation est rarement effectué par une seule personne. C'est souvent une équipe de plusieurs lexicographes dirigée par un lexicologue qui réalisent ce travail. Notre outil devra permettre au lexicologue de répartir le travail entre les différents lexicographes et de le réintégrer dans sa base lexicale une fois complété.

Conclusion

Pour répondre à la demande du projet UNL, il nous faut concevoir un outil pratique qui puisse être utilisé sur un poste de bas de gamme. À l'opposé, si nous ne voulons pas proposer un énième outil spécialisé qui ne servira que pour le projet, il va falloir penser à un outil plus générique.

Nous avons donc étudié différentes structures de dictionnaires pour comprendre les besoins puis quelques outils pour lexicographes pour voir comment satisfaire ces besoins et enfin des outils d'intégration d'application pour étendre le concept de généricité à un concept de modularité.

Partie 2 : État de l'art

1. Différentes organisations de dictionnaires

1.1. Dictionnaires classiques

1.1.1. Une encyclopédie : le Larousse Universel

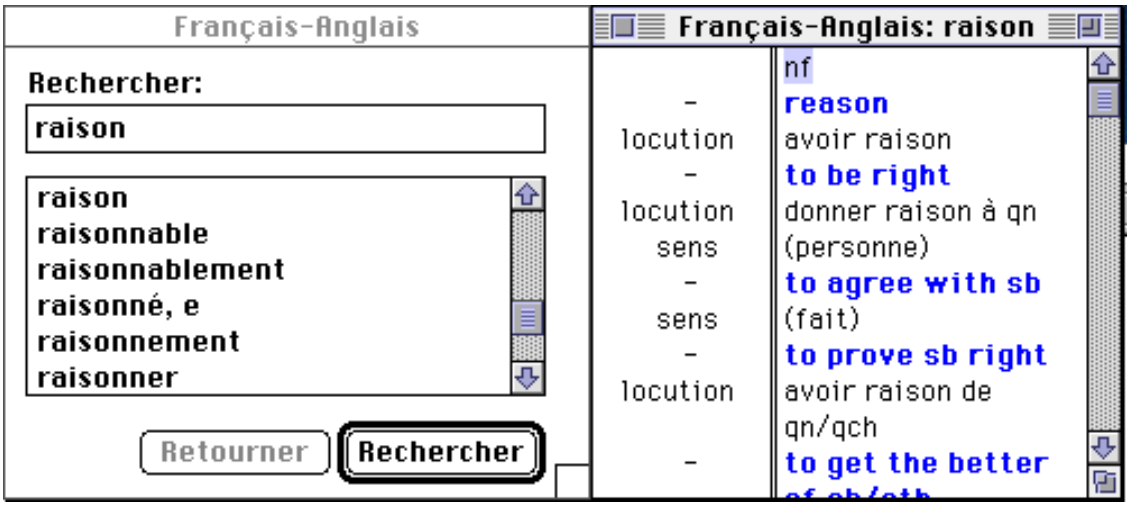
Le Larousse universel est une encyclopédie. Il contient donc à ce titre, des entrées correspondant au vocabulaire de la langue française mais aussi de nombreux noms propres et noms de lieu. On y trouve beaucoup d'illustrations ainsi que des planches couvrant une page entière.

Un article est composé comme suit. L'entrée est écrite en caractères gras avec des entrées différentes pour les homonymes. Elle est éventuellement suivie de sa prononciation ou d'une partie entre crochets et de sa catégorie grammaticale. L'origine du mot est parfois indiquée entre parenthèses. Les définitions alternent avec les exemples en italique. Pour certains articles, il y a une partie nommée ENCYCL qui cite des informations historiques ou géographiques ayant rapport avec le mot d'entrée.

1.1.2. Un dictionnaire bilingue : le Collins On-Line

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.

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. A la fin de la liste sont indiquées les sous entrées du dictionnaire.



L'entrée « raison » du Collins On -Line

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

1.1.3. Une base de données multilingue : Eurodicautom

EURODICAUTOM a été développé par la commission européenne comme un outil efficace à l'usage des traducteurs et autres personnes travaillant dans les institutions européennes. Il aide ces personnes dans leur recherche pour les termes dans les langues officielles et de travail de la communauté européenne. Ces langues sont le danois, le hollandais, l'anglais, le français , l'allemand, le grec, l'italien, le portugais et l'espagnol.

EURODICAUTOM est une base de données terminologique multilingue comprenant 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.

Cette base est consultable sur le web. Voici un exemple d'entrée de EURODICAUTOM avec l'entrée raison en français en langue source et en portugais en langue cible.

EURODICAUTOM Query Results

French	*Keyword*	raison;rapport d'echelonnement des vitesses;rapport des gradins
Portuguese	*Keyword*	razao;razao de escalonamento das velocidades DATE = 950426 CF= 4
French	*Keyword*	but du voyage;motif du voyage;raison du voyage
Portuguese	*Keyword*	motivo da viagem DATE = 941003 CF= 5
French	*Keyword*	raison speciale
Portuguese	*Keyword*	razao especial

L'entrée « raison » de EURODICAUTOM

1.1.4. Une base de données lexicales : BDLEX

BDLEX [18] est une base de données lexicales du français écrit et parlé. Elle est développée dans le cadre du GDR-PRC Communication Homme-Machine par le groupe IHM-PT de l'IRIT à l'université Paul Sabatier de Toulouse. Elle contient des matériaux lexicaux utilisés pour l'assistance linguistique au traitement de textes ou oral. On y trouve des lexiques, des composantes phonologiques et prosodiques ainsi que des linguisticiels.

BDLEX-1 comprend 23 000 entrées soit 270 000 formes fléchies. A 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 pieds.

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

Voici un extrait de cette base. Tous les champs ne sont pas répertoriés :

GRAPH_ACC PHON_SYLL FPH CS PHONOGRAMMES

```
aigre-doux E/gr™/du s" J (ai,E) (g,g) (r,r) (e,™) (-,Ÿ) (d,d) (ou,u) (x,s")
amygdale A/mi/dAl ™ N (a,A) (m,m) (y,i) (g, Ÿ ) (d,d) (a,A) (l,l) (e,™)
axe Aks ™ N (a,A) (x,ks) (e,™)
bahut /bA/y N (b,b) (a,A) (h, Ÿ) (u,y) (t,Ÿ)
chat-huant /_A/y/ã N (ch,_) (a,A) (t,Ÿ) (-,Ÿ) (h,Ÿ) (u,y) (an,ã) (t,Ÿ)
dix-huit /di/zÁi t' J (d,d) (i,i) (x,z) (-,Ÿ) (h,Ÿ) (ui,Ái) (t,t')
exact eg/zA kt" J (e,e) (x,gz) (a,A) (ct,kt")
iceberg Ajs™/b¤rg N (i,Aj) (c,s) (e,™) (b,b) (e,¤) (r,r) (g,g)
hautbois /*O/bwA N (h,*) (au,O) (t,Ÿ) (b,b) (oi,wA) (s,Ÿ)
homme Øm ™ N (h,Ÿ) (o,Ø) (mm,m) (e,™)
onze /*õz ™ J (Ÿ,*) (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)
```

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

Voici 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
```

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 travers des outils dont dispose ORACLE.

1.2. Dictionnaires à structures connues

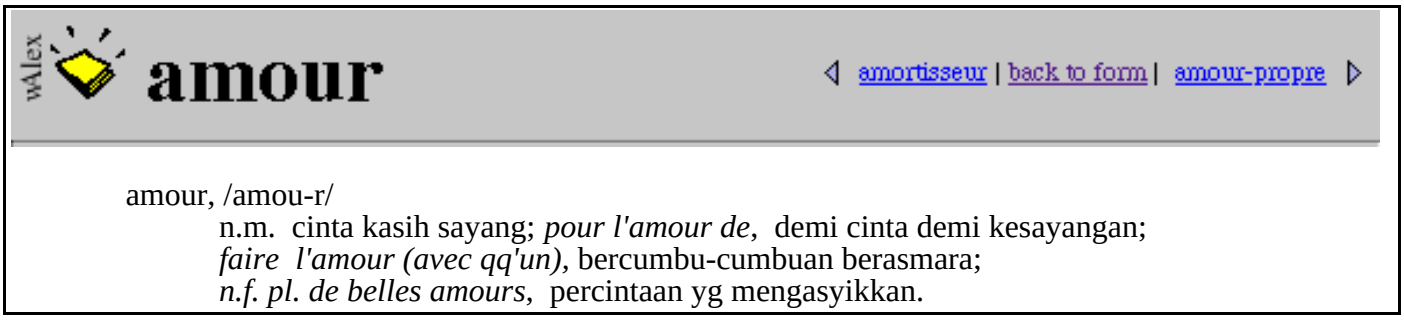
1.2.1. Le DEC

Le Dictionnaire Explicatif et Combinatoire du Français Contemporain est un dictionnaire monolingue sur papier. La version électronique est actuellement en cours de réalisation. La structure du DEC est relativement complexe. Il est d'ailleurs plus utilisé comme une base lexicale détaillée que comme un simple dictionnaire. Le DEC a été réalisé pour le cadre d'un travail de lexicologie. Il subit donc régulièrement des modifications dans sa structure.

Pour chaque article, un mot suivi de sa catégorie grammaticale (ex: coeur nom masc.), il y a une subdivision en lexies qui sont en fait les unités de base du dictionnaire. Chaque lexie comprend des connotations, le régime associé à la lexie, les fonctions lexicales qui lui sont associées et des exemples à la fin.

1.2.2. Le FeM (français, anglais, malais)

Le FeM [16, 17] est un dictionnaire édité sur papier et électroniquement (logiciels et accès par la toile). Le but était de disposer de versions électroniques de ce dictionnaire trilingue anglais, français, malais et aussi d'éditer un dictionnaire bilingue français-malais. Pour faciliter le travail des lexicographes, il a été décidé de rajouter des équivalents anglais afin que les sens soient mieux compris par tout le monde. On peut aussi le consulter à travers la toile. Pour chaque lexème, on a sa prononciation et sa traduction en anglais suivie de la langue cible. L'interface web ne permet pas de voir la structure complète. L'exemple suivant est l'entrée correspondant au mot « amour », l'équivalent anglais n'est pas affiché.



L'entrée « amour » du FeM à travers le web

Les données du FeM sont stockées à partir des formes logiques compressées. Les applications utilisent ces formes de données telles quelles. Voici la forme logique correspondant à l'entrée visualisée plus haut :

```
(FEM_ENTRY
  (:ENTRY "amour")
  (:FRENCH_PRON "/amou-r/")
  (:FRENCH_CAT "n.m.")
  (:ENGLISH_EQU "love")
  (:MALAY_EQU "cinta")
  (:MALAY_EQU "kasih")
  (:MALAY_EQU "sayang")
  (:FRENCH_PHRASE "pour l'amour de")
  (:ENGLISH_PHRASE "for the sake of")
  (:MALAY_PHRASE "demi cinta")
  (:MALAY_PHRASE "demi kesayangan")
  (:FRENCH_PHRASE "faire l'amour (avec qq'un)")
  (:ENGLISH_PHRASE "make love")
  (:MALAY_PHRASE "bercumbu-cumbuan")
  (:MALAY_PHRASE "berasmara")
  (:FRENCH_CAT "n.f. pl.")
  (:FRENCH_PHRASE "de belles amours")
  (:ENGLISH_LABEL "(poét.)")
  (:ENGLISH_PHRASE "wonderful love")
  (:MALAY_PHRASE "percintaan yg mengasyikkan")
)
```

L'entrée « amour » sous forme logique

1.2.3. Les dictionnaires d'ARIANE-G5

ARIANE-G5 [7] est un système de traduction de seconde génération développé au sein du laboratoire GETA. 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 :

<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>.

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). Ainsi, aucun délimiteur n'est nécessaire. On accède aux dictionnaires par les morphes ou les tournures.

Le langage TRANSF utilise un dictionnaire bilingue pour la traduction. Il est constitué d'une suite d'articles où chacun est caractérisé par :

- un nom d'unité lexicale source noté entre deux apostrophes,
- un séparateur "==",
- 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.

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

Un exemple tiré du dictionnaire RUS-FRA

1.3. Conclusion

Les dictionnaires ont tous des formats très différents les uns des autres, selon qu'ils sont monolingues, bilingues ou multilingues, qu'ils sont destinés à être imprimés ou qu'ils soient utilisés sous forme électronique. Un outil générique pour lexicographes doit donc s'adapter aux différentes structures. Pour des travaux de recherche en lexicologie, il faut pouvoir gérer différentes versions d'un dictionnaire et laisser la possibilité de créer des nouveaux outils pour ces versions. Le système devra gérer des formats, des structures de données et des outils différents.

2. Aperçu de différents outils pour lexicographes

Le processus d'indexation d'un mot dans un dictionnaire comprend plusieurs étapes. D'abord le lexicographe consulte d'autres dictionnaires ou une version précédente de celui qu'il remplit. Il a donc besoin d'outils de visualisation de dictionnaires de types variés. Ensuite, il fait des recherches dans un corpus pour trouver

des collocations de mots à indexer avec des outils d'analyse de corpus. Pour terminer, il entre le mot et sa lexie dans une structure particulière définie au préalable à l'aide d'un outil d'édition de dictionnaire ou de base lexicale.

2.1. Des outils d'analyse de corpus : UNIX for poets

UNIX for poets [10] a été écrit par Kenneth Ward Church en juillet 1994 pour l'European Summer School d'Utrecht aux Pays Bas. Il présente des outils d'analyse de corpus qui sont en fait directement des combinaisons de commandes UNIX comme grep, sort, uniq, tr, wc etc. Ces outils permettent de rechercher des collocations et de trier des corpus selon des schémas définis.

Ces outils ont l'avantage d'être rapides et simples. Les possibilités de combinaisons de ces outils sont infinies car ces outils sont programmés directement à l'aide du shell d'UNIX qui offre tous les avantages d'un langage de programmation. D'autre part, le système UNIX est relativement répandu dans le monde de la recherche. Cependant le maniement des commandes UNIX n'est pas très aisé pour un lexicographe néophyte. De plus, on ne peut manier que des textes représentés en caractères ASCII. On pourrait envisager l'intégration de tels outils comme des processus tournant sur une machine UNIX à part et échangeant les données et les résultats selon le modèle client-serveur.

2.2. ATLAS, un outil d'indexage des dictionnaires d'ARIANE-G5

Le logiciel ATLAS [4, 5] conçu par Daniel Bachut permet d'introduire des mots nouveaux et les codes associés dans un dictionnaire. 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 qui se trouve lui aussi sur la machine IBM.

Les manuels d'indexage représentent les arbres de décision auxquels sont confrontés les linguistes lors de l'indexation d'une entrée. Ils décrivent la structure des entrées. Le linguiste édite son manuel avec un éditeur de textes quelconque. Il le compile ensuite avec ATLAS. Lorsqu'ATLAS détecte des erreurs, il signale leur position à l'utilisateur et lui permet de la corriger.

```
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 .
```

un exemple de manuel d'indexage en 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 une partie de son manuel. Le manuel peut être aussi visualisé sous forme arborescente.

L'interpréteur de menus permet d'indexer interactivement un mot dans un dictionnaire en suivant le format du manuel d'indexage compilé au préalable.

ATLAS propose un menu relativement complet de fonctions de manipulations de manuels d'indexage. L'indexation dans un dictionnaire est cependant trop lente pour être utilisée. La conception d'ATLAS date de 1983, c'est pourquoi, il est nécessaire de l'adapter aux progrès de ces dernières années en informatique.

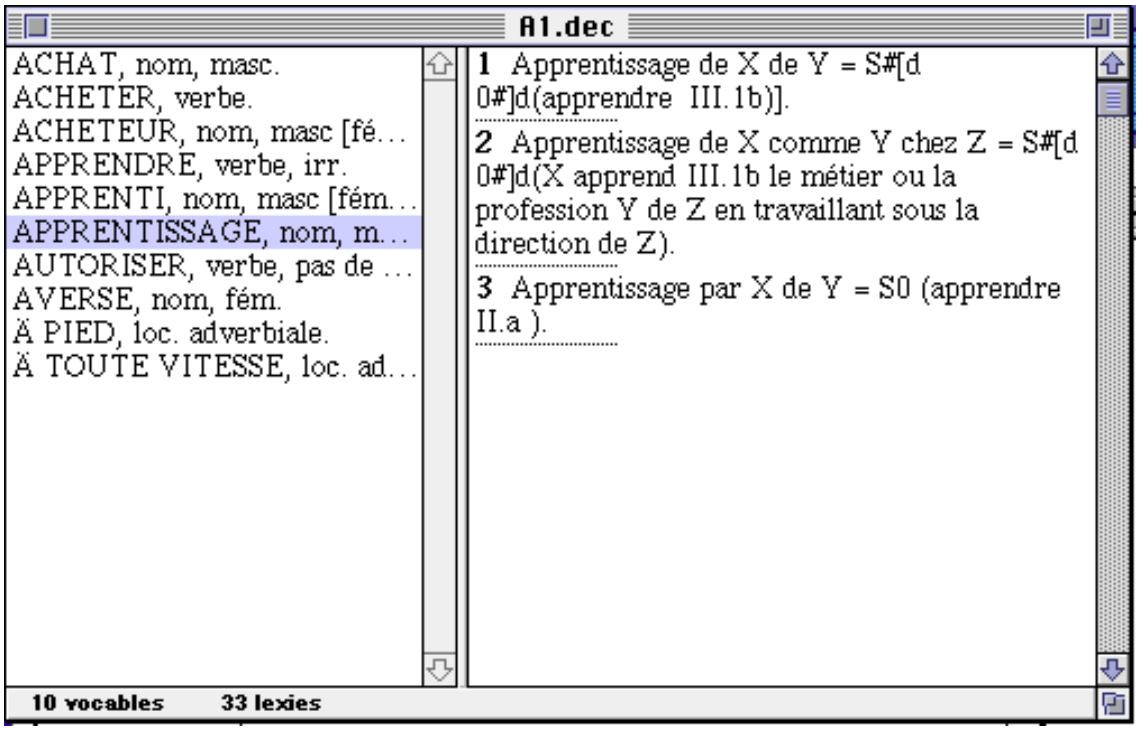
2.3. Outils d'édition

2.3.1. DECID, un éditeur pour le DEC

DECID [19] a été élaboré par Gilles SÉRASSET pour simplifier le travail d'indexation du DEC par l'équipe d'Igor Melcuk à Montréal.

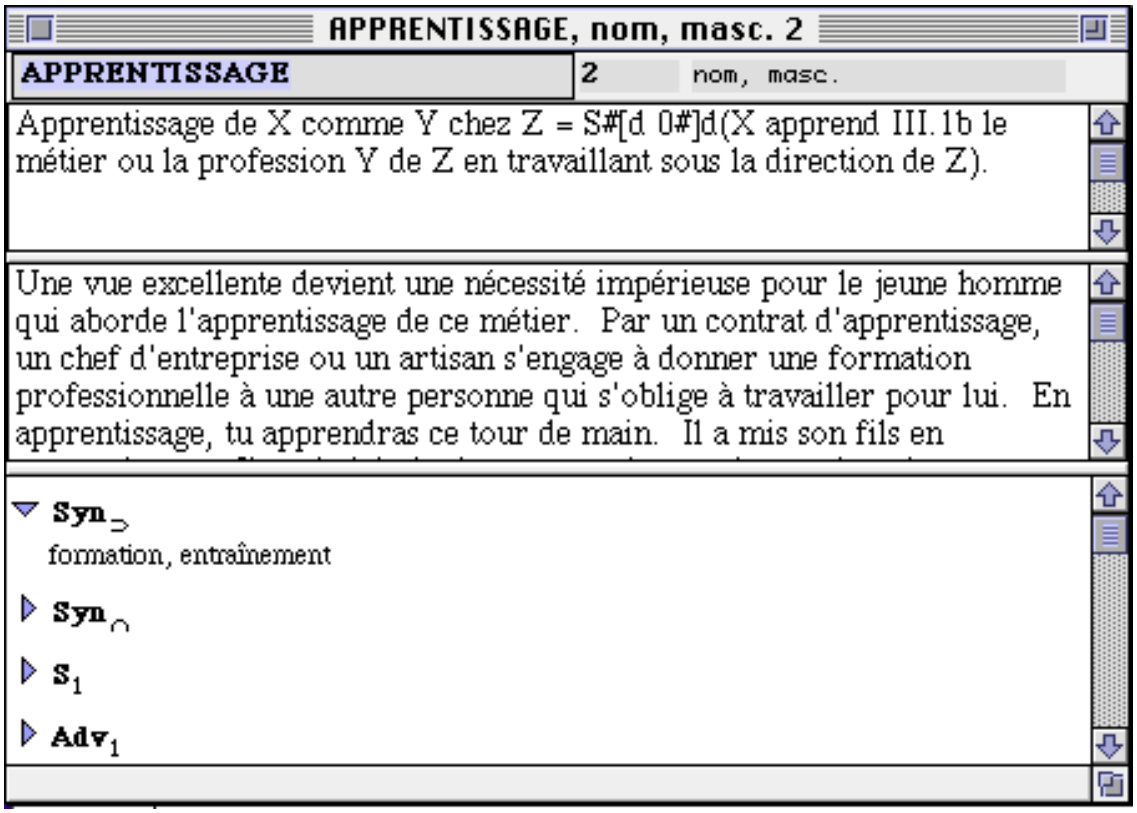
L'interface a été conçue pour simplifier au maximum le travail du rédacteur. De plus, elle est à la fois une interface de navigation et d'édition. Les dictionnaires peuvent être gérés à la discrétion de l'utilisateur. Il est possible d'utiliser un ou plusieurs fichiers dictionnaire. Afin de permettre de gérer des fichiers variés, il a été rajouté à l'éditeur DECID des fonctionnalités d'édition générales de fichiers textes.

L'éditeur DECID a été élaboré selon la technique du « wysiwyg » afin d'obtenir une visualisation du contenu la plus proche de celle du dictionnaire final.



La fenêtre principale de DECID

Dès que l'on crée un dictionnaire, la fenêtre principale du dictionnaire apparaît. dans la première partie, il y a la liste des vocables et si l'on clique sur l'un, 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.



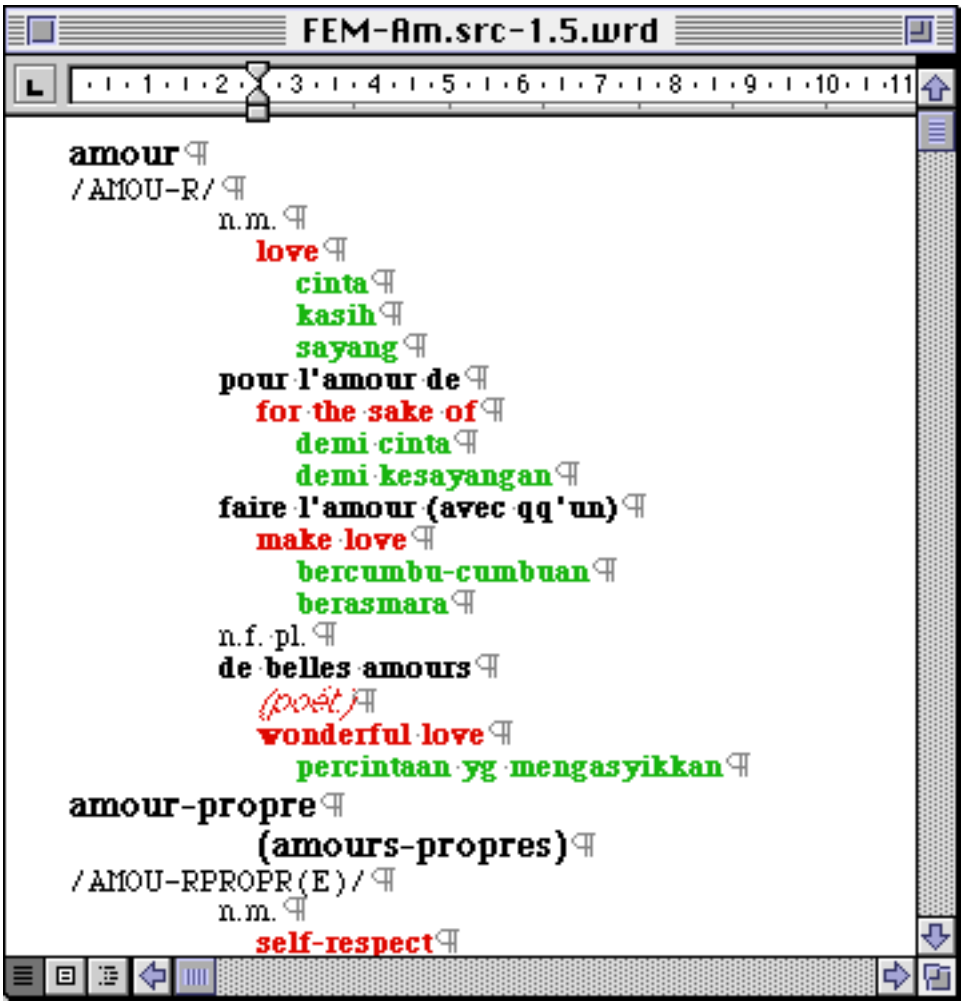
La fenêtre de lexie de DECID

La fenêtre de lexie comporte quatre parties. La première contient un texte éditable contenant la graphie, suivi d'un autre contenant le numéro d'homonyme et d'un dernier contenant les informations morphologiques. Les vocables n'ont pas d'existence propre. Chaque vocable est identifié par sa graphie et sa morphologie. La deuxième partie contient la définition dans un champ de texte éditable. Les exemples se trouvent dans la troisième et les fonctions lexicales composent la dernière partie. Elles sont présentées sous forme de liste. On peut obtenir leur valeur en cliquant sur le triangle situé à côté de leur nom. Il est possible d'éditer les valeurs et aussi le nom des fonctions lexicales.

Cet éditeur n'est qu'une première étape vers un outil véritablement utilisable par les lexicographes. Il faudrait le doter d'un système de vérification de contraintes de cohérence et l'intégrer à des outils plus généraux utilisés par le lexicographe. Le DEC est un travail de lexicologie. La structure des entrées est donc en permanente évolution. Ce qui rend impossible la maintenance d'un outil ad-hoc. DECID a été construit à des fins d'expérimentation d'interfaces pour lexicographes.

2.3.2. La méthode d'édition des projets Fe*

Les dictionnaires des projets Fe* [16, 17] sont en fait édités avec l'éditeur de textes Microsoft Word™. Cela permet, étant donnée la notoriété de Word™ dans le monde entier, d'indexer les dictionnaires des projets Fe* avec le minimum d'outils. Cette méthode a été utilisée en Malaisie pour le dictionnaire FeM.



indexation des entrées « amour » et « amour-propre » du dictionnaire FeM

Les projets Fe* utilisent une structure de traits pour représenter leur données. Avec Word™, une correspondance entre les styles et les champs permet d'indexer facilement le dictionnaire et de récupérer à partir des fichiers RTF les données enregistrées ensuite en format logique. Cette récupération n'est possible que si la structure de données est assez simple, du type structure de traits.

2.4. SUBLIM, un concept de gestion de dictionnaires

SUBLIM [19], projet décrit dans la thèse de Gilles SÉRASSET, fait suite à une volonté d'unification et de généralisation des architectures lexicales et linguistiques de différentes bases lexicales. C'est donc un système de gestion de bases lexicales multilingues qui permet de spécifier l'architecture lexicale (organisation des dictionnaires) et l'architecture linguistique (organisation des informations linguistiques des unités des dictionnaires) sans imposer de contraintes ni sur les types de dictionnaires choisis, ni sur les structures linguistiques utilisées.

Stuart M. Shieber a défini le critère de formalisme grammatical qu'il appelle félicité linguistique : le degré auquel les descriptions de phénomènes linguistiques peuvent être exprimés, directement ou indirectement, de la manière où le linguiste voudrait les exprimer.

Pour la conception d'un système de gestion de bases lexicales, il faut retenir trois points :

- garantir la possibilité de coder des unités lexicales en utilisant différentes structures linguistiques;

- la taille d'une base lexicale est telle qu'elle nécessite plusieurs personnes pour l'indexage et la maintenance de cette information;

- le mécanisme de présentation doit être suffisamment général pour permettre de masquer la structure interne de l'information lexicale.

SUBLIM répond à ces exigences en contrôlant la vue qu'on a de la structure interne, en abstrayant les informations linguistiques de leur codage informatique et en définissant différentes vues d'une même structure selon l'usage que l'on fait de la base lexicale. C'est un outil qui permet de créer et gérer une base lexicale pour laquelle l'utilisateur a déclaré les dictionnaires, leur structure et les manières de présenter dictionnaires et structures.

Le langage LEXARD permet de définir l'architecture lexicale de la base tandis que l'architecture linguistique à l'intérieur des dictionnaires est décrite grâce au langage LINGUARD. Les structures logiques de base implémentées sont les structures de traits, les arbres, les automates, les graphes et les ensembles. En combinant ces structures, on peut définir des structures linguistiques complexes d'une manière naturelle.

SUBLIM a été défini selon le paradigme de la programmation à objets. Cela permet donc de définir l'héritage entre les classes ainsi que la disjonction et l'unification.

L'architecture logicielle se répartit sur trois niveaux :

- le niveau Base de Données qui régit le stockage des informations linguistiques, différents systèmes peuvent être utilisés;

- le niveau interne qui permet des manipulations sur les entrées des dictionnaires, il utilise les structures de LINGUARD;

- le niveau présentation.

Le fonctionnement est basé sur l'aller/retour entre ces différents niveaux. Une requête est formulée au niveau présentation, traduite en une structure de niveau interne, traduite ensuite en requête de base de données. Le résultat est transformé en un ensemble de structures de niveau interne et visualisée au niveau présentation.

Le niveau interne contient des fonctions d'interrogation et de manipulation des structures. Pour interroger la base, on définit une structure patron.

Le niveau présentation est composé d'un éditeur et d'un navigateur.

Un vérificateur de cohérence vérifie que les entrées d'un dictionnaire sont conformes à des contraintes définies a priori. Les contraintes sont statiques ou dynamiques. On distingue trois niveaux : alerte, délai et critique; ainsi que trois types : intégrité, cohérence globale et cohérence locale.

Un défauteur permet de donner des valeurs par défaut aux éléments de structures qui n'ont pas été renseignés par le lexicographe.

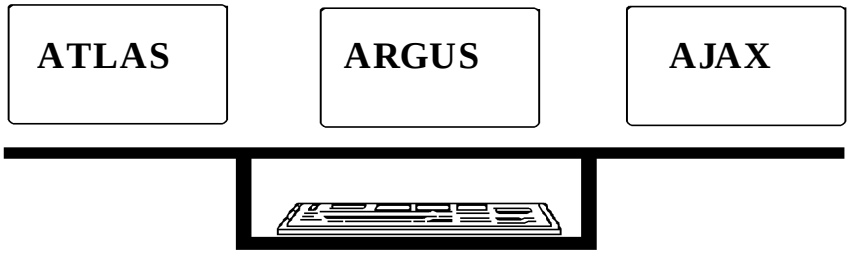
Le mécanisme d'Import/Export permet d'exporter des informations de la base lexicale et d'importer des dictionnaires existants. Le format utilisé est le SGML (Standard Generalized Markup Langage) et le type des dictionnaires est codé selon les règles de la Text Encoding Initiative (TEI).

Le projet SUBLIM présente une architecture très complète qui devrait permettre de coder un grand nombre d'informations linguistiques et de définir des architectures diverses de bases lexicales. Il faut cependant définir une interface simple pour une utilisation de la base par des linguistes non informaticiens.

2.5. Postes de travail

2.5.1. The Hector Project

Ce projet est expliqué par B. T. Sue Atkins [3] dans COMPLEX'92. Il avait un objectif lexicographique d'une part, celui de compiler les entrées d'une base de données lexicale et en même temps étiqueter les sens d'une partie du corpus électronique utilisé comme preuve pour le lexicographe et un objectif informatique d'autre part, construire des outils améliorés pour faciliter le travail du lexicographe.



Le poste de travail de Hector Project

Le poste de travail est constitué d'un clavier et de trois écrans, un pour chaque grand groupe d'outils. Ceux qui accèdent au matériel de référence sont utilisés sur l'écran de gauche appelé Atlas. Ceux qui affichent les textes de corpus structurés et réalisent des opérations dessus sont utilisés sur celui du milieu appelé Argus et enfin, ceux qui reçoivent, structurent, enregistrent et affichent les entrées lexicales et leurs amendements sont utilisés sur celui de droite appelé Ajax.

Le moniteur de gauche, Atlas est l'écran de référence. Il y a des outils qui présentent au lexicographe des données linguistiques déjà traitées, l'aidant dans la recherche dans les dictionnaires, les statistiques sur les occurrences de formes de mots et les fréquences des collocations et une liste de modèles de verbes. De plus, les lexicographes utilisent le moniteur Atlas pour demander de l'information sur des textes dans le corpus ou un rapport de progrès sur le projet, et pour vérifier la consistance de leurs entrées compilées. C'est aussi l'écran utilisé pour des tâches de routine comme l'édition de documents et le traitement du courrier électronique. L'écran Atlas est entièrement flexible et est configuré par les lexicographes à leur goût, à l'inverse d'Argus et d'Ajax qui ont chacun leur écran adapté. Les commandes dans Atlas sont simplement tapées dans une fenêtre shell d'UNIX. Il y a trois groupes correspondant au type d'information voulue : les entrées de travail de référence, l'information prétraitée comme les statistiques de corpus, et les documents d'organisation lexicographique.

Argus regroupe les outils d'étiquetage et de recherche de corpus. Les lexicographes peuvent spécifier les conditions de recherche à travers des facilitées de dialogue offertes. Ces outils opèrent sur des données linguistiques brutes. Argus utilise les étiquettes existantes dans la sélection des lignes concordant avec les conditions

établies par les lexicographes et les affiche pour l'étiquetage des sens. Argus enregistre aussi les étiquettes de sens assignées à chaque occurrence. Le système de commandes d'Argus et Ajax est très flexible. Quelques boutons de commande exécutent la commande directement et d'autres s'ouvrent sur un menu de sous commandes et d'options pouvant se séparer en plusieurs niveaux et d'autres ouvrent une fenêtre de dialogue dans laquelle le lexicographe doit entrer des informations.

Ajax abrite le logiciel de construction d'entrées, c'est ici que les lexicographes remplissent les squelettes d'entrées pour en faire des entrées complètes. Les outils fournissent un environnement structuré avec lequel les lexicographes traitent les entrées lexicales. Ils peuvent partir de zéro, avec un patron vide ou adapter une entrée du dictionnaire COD8.

2.5.2. The Linguist's Shoebox

La première "boîte à chaussures" du linguiste a été écrite comme programme DOS par John Wimbish [8]. Cette version 3.0 est disponible sur Windows et Macintosh.

C'est un programme de gestion de bases de données conçue pour bien gérer le type de bases de données que le linguiste utilise. Il est spécialement destiné à l'interlinéarisation de textes et au développement de lexiques.

La barre des menus contient les menus classiques **Fichier**, **Edition**, **Ecran** et les menus **Base de données** et **Projet** pour manipuler les bases et les projets. La barre d'outils contient les outils habituels Naviguer, Sauver, Couper, Copier et Coller et des outils plus spécifiques de navigation dans la base de données, de changement de vue et de déclenchement d'interlinéarisation. La barre des status affiche des informations diverses.

Chaque article de la base de données est divisé en champs. On peut créer une hiérarchie entre les champs et leur associer un certain style et une certaine police. On peut donc indexer des langues avec un alphabet non roman si on installe le script correspondant dans le système.

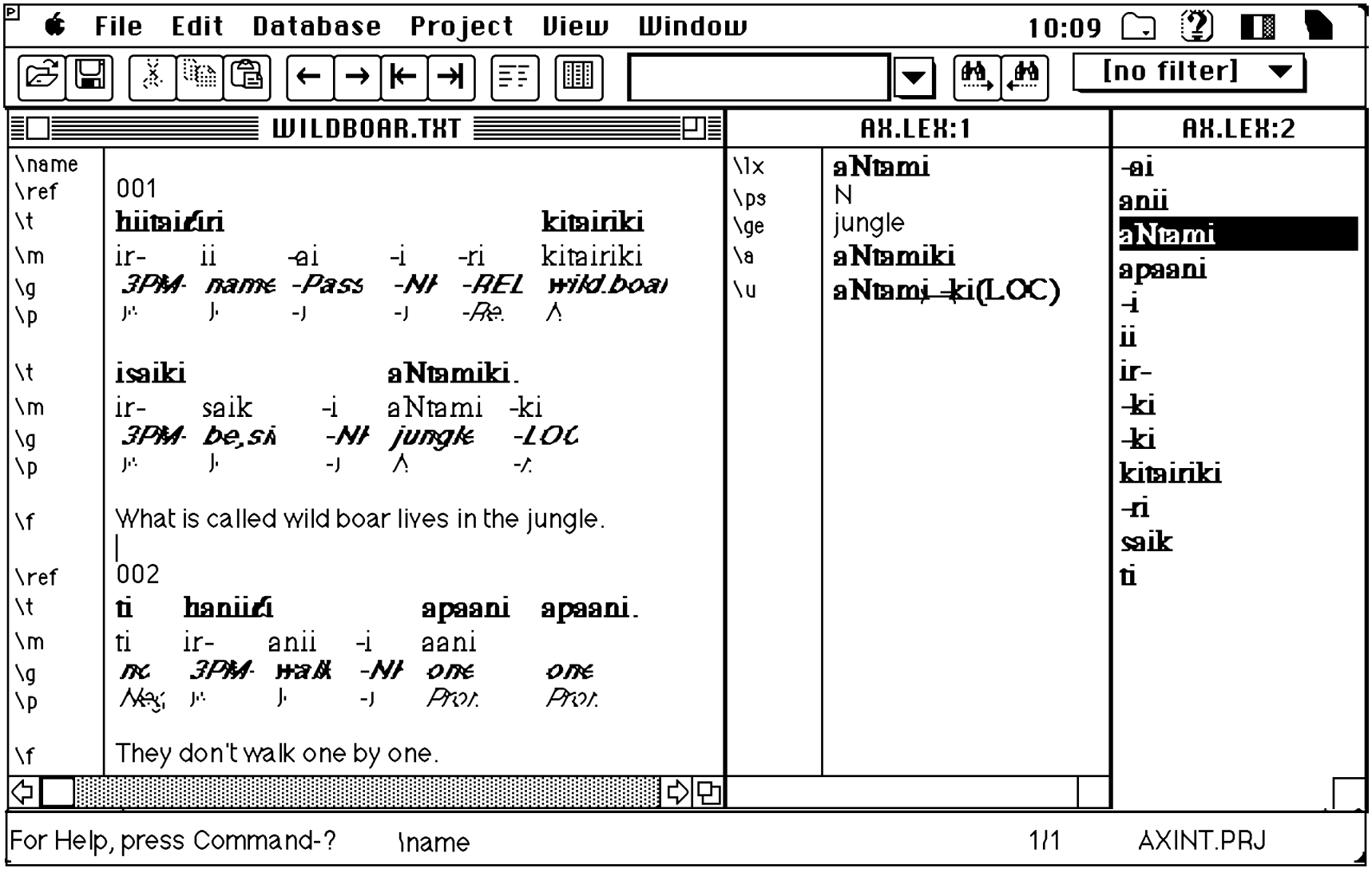
Dans ce poste de travail, la fenêtre **WILDBOAR.TXT** contient un texte interlinéarisé avec les mots de la base **AX.LEX**. On peut avoir deux vues différentes de cette base.

Les bases lexicales peuvent avoir de multiples représentations et être triées selon n'importe lequel des champs qui la composent. Il existe aussi de multiples fonctions de recherche à l'intérieur des bases.

Le grand avantage de ce logiciel est de pouvoir interlinéariser automatiquement un texte à partir d'une base de données ou, inversement, indexer une base de données à partir des mots d'un texte. Les échanges entre les deux bases sont automatisés. Il propose des solutions pour les problèmes d'import/export à partir des formats internes de Shoebox

Ce poste n'est malgré tout pas multi-utilisateurs et surtout, il n'offre qu'une seule structure logique possible : la structure de champs.

L'exemple suivant est tiré d'une analyse d'un texte en axininca, langue amérindienne. Nous avons installé une nouvelle police permettant de représenter des caractères propres à cette langue.


File Edit Database Project View Window
10:09
[no filter]
WILDBOAR.TXT
AX.LEX:1
AX.LEX:2
\name
\ref 001
\t hiitaiɾiri kitairiki
\m ir- ii -ai -i -ri kitairiki
\g 3PM- name -Pass -NF -REL wild.boar
\p
\t isaiki aNtamiki.
\m ir- saik -i aNtami -ki
\g 3PM- be_sit -NF jungle -LOC
\p
\f What is called wild boar lives in the jungle.
\ref 002
\t ti haniiɾi apaani apaani.
\m ti ir- anii -i aani
\g neg 3PM- walk -NF one one
\p Neg Prox Prox
\f They don't walk one by one.
\lx aNtami
\ps N
\ge jungle
\a aNtamiki
\u aNtami -ki(LOC)
-ai
anii
aNtami
apaani
-i
ii
ir-
-ki
-ki
kitairiki
-ri
saik
ti
For Help, press Command-? \name 1/1 AXINT.PRJ

2.5.3. Le Lexicaliste

Le Lexicaliste est un générateur de bases lexicales monolingues. Il est développé et commercialisé par la société SITE.

Il s'appuie sur une description des entrées du lexique. Un article est un arbre décoré dont la racine correspond à l'entrée du dictionnaire et les nœuds aux différents sens de l'article. Le lexicologue a à sa disposition un langage de contraintes lui permettant de spécifier les attributs syntaxiques qu'il désire gérer dans sa base lexicale particulière. Il peut aussi donner des propriétés sur les attributs de la base. Le système peut déduire la ou les valeurs possibles d'un certain nombre d'attributs prenant en compte ceux déjà saisis.

Le système gère aussi d'autres types de relations qui définissent des réseaux différents à l'intérieur du dictionnaire. Les relations lexicales sont définies sur un ensemble de sens de mots et les relations sémantiques sur un ensemble de concepts. On peut aussi mettre en oeuvre des liens hypertextuels.

2.6. Conclusion

Tous ces outils ont été créés pour une utilisation ad hoc, ils ne permettent pas l'intégration d'autres outils. Les structures de données ne peuvent pas être modifiées non plus. Il est donc nécessaire de concevoir une architecture à un niveau supérieur qui pourrait englober certains de ces outils.

3. Des outils d'intégration d'applications

Pour mettre en place une architecture ouverte et permettant la connexion de modules différents, il nous a semblé intéressant d'étudier des systèmes générateurs ou d'intégration d'application comme GATE et OpenDoc.

3.1. GATE : A General Architecture for Text Engineering

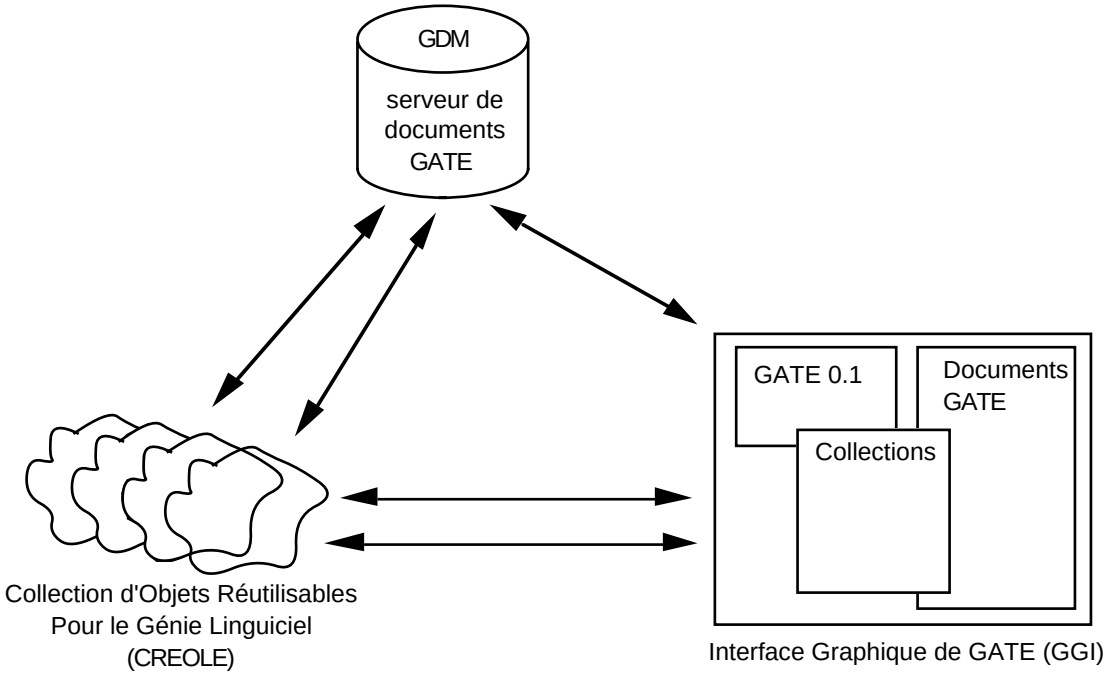
GATE [11] est une architecture dans le sens où ce système fournit une infrastructure commune pour construire des systèmes de traitement du langage (Langage Engineering) plutôt que la structure d'une construction ou des spécifications d'interfaces. C'est aussi un environnement de développement qui fournit de l'aide pour la construction, le test et l'évaluation de systèmes de traitement du langage.

GATE offre aux chercheurs et développeurs de traitement du langage un environnement dans lequel ils peuvent utiliser facilement et en combinaison des outils et des bases de données linguistiques, lancer différents processus comme des étiqueteurs ou analyseurs sur le même texte et comparer les résultats ou, au contraire, utiliser le même module sur différentes collections de textes et analyser les différences; le tout dans une interface agréable à utiliser.

GATE comprend trois éléments principaux :

- une base de données pour stocker de l'information sur des textes, GDM;
- une interface graphique pour lancer des outils sur les données et visualiser et évaluer les résultats, GGI;

• une collection de ressources algorithmiques qui interopèrent avec la base de données et l'interface et constituent une collection d'objets réutilisables pour le traitement du langage, CREOLE.



Les trois éléments de GATE

GDM est basé sur le gestionnaire de documents TIPSTER. C'est un serveur central qui stocke toutes les informations que génère le système sur les textes qu'il traite. Toute la communication entre les composants d'un système de traitement du langage passe par GDM, isolant les composants entre eux et fournissant une interface API (applications programmer interface) uniforme pour manipuler les données produites par le système.

Tout le travail réel d'analyse de textes est fait par les modules de CREOLE. Typiquement, un objet CREOLE est une enveloppe autour d'un module de traitement de langage ou une base de données, un analyseur ou un étiqueteur, un lexique ou un index de ngram par exemple. Les objets fournissent une interface API standardisée pour les ressources sous-jacentes qui permet un accès via GGI et les E/S via GDM.

Bien sûr, GATE ne résout pas tous les problèmes impliqués dans l'interconnection de diverses modules ensemble. Il y a deux barrières à cette intégration:

• l'incompatibilité de la représentation de l'information sur les textes entre les mécanismes de stockage, ceux de recherche et la communication inter-modules;

• l'incompatibilité du type de l'information, utilisée et produite par les différents modules.

3.2. OpenDoc

OpenDoc [13] est une architecture multiplateforme pour des composants logiciels. Le projet est soutenu par CI Labs, une association à but non lucratif d'environ 2000 membres dont 300 leaders de l'industrie comme APPLE et IBM. OpenDoc fournit

un modèle utilisateur orienté objet ou les documents sont des objets et contiennent d'autres objets et ou chaque objet peut avoir des comportements distincts.

Chaque document est composé d'une ou plusieurs parties (objets) : une seule partie au niveau le plus haut appelée partie racine et d'autres encapsulées dans la partie racine. L'utilisateur assemble un documents avec des parties encapsulées avec les commandes *couper* , *coller* et d'autres commandes d'insertion ou la méthode du "drag and drop".

La partie racine contient des caractéristiques telles que la taille de la zone de travail, les options d'impression ou la métaphore d'édition de base (texte, dessin...). Les parties n'ont pas besoin de connaître la structure interne du document ni la sémantique des parties incluses. Quand les utilisateurs interagissent avec les parties OpenDoc, le comportement résultant est déterminé par les éditeurs et visualiseurs.

Un éditeur de parties a toutes les caractéristiques d'un composant logiciel OpenDoc qui permet la création, l'édition et la visualisation de parties d'un type particulier, tout comme les applications conventionnelles permettent la manipulation de documents actuellement.

Un visualiseur de partie est un type spécial limité d'éditeur de partie qui peut afficher et imprimer un type particulier de partie.

Toutes les parties ont un ensemble de base de propriétés; elles incluent le type de la partie, sa catégorie, le type de vue, quel éditeur utiliser, qui a modifié le dernier le contenu et quand. Le type des parties réfère au format de données du contenu intrinsèque d'une partie. La catégorie d'une partie réfère à un ensemble de types de parties qui sont conceptuellement similaires (texte stylé, vidéo...).

Le contenu d'une partie peut être affiché dans plus d'une structure à la fois et peut avoir des représentations multiples. Par exemple, une partie tabulaire peut être visualisé comme un diagramme dans une structure et une table de texte dans une autre.

Lorsqu'on clique sur une partie encapsulée dans un document, celle-ci devient active : le menu change. Il est remplacé par le menu de l'éditeur de la partie correspondante. Les palettes d'instruments de l'éditeur s'affichent. Une partie est inactive lorsque l'utilisateur travaille ailleurs. On peut aussi sélectionner une partie avec la souris pour la redimensionner ou l'afficher en entier dans une autre fenêtre bien qu'elle reste une partie et pas un document à part entière. Dans un document, on peut ne pas afficher la partie en entier.

L'utilisateur peut copier et bouger n'importe quel contenu avec les fonctions *couper, copier* et *coller*. L'éditeur de parties choisit de mélanger ou d'inclure la partie à coller selon son type. Par exemple, il choisira de mélanger deux parties texte et de les inclure dans une partie graphique.

Les menus OpenDoc fournissent les menus de base lorsqu'un document est ouvert : **Document, Edit, Help**, **Application**. Les autres menus varient en fonction de la partie active. Les éditeurs OpenDoc sont vidés de la mémoire dès que l'on a plus besoin d'eux. La commande **Quitter** n'est donc pas implémentée.

Le projet OpenDoc est diffusé. Il est cependant peu adopté car très gourmand en ressources mémoire. Il est cependant intéressant du point de vue de l'idée que l'on essayera de conserver.

Conclusion

L'inventaire des outils pour lexicographes nous a montré qu'il existe une grande diversité d'outils. Il semble cependant que la majorité de ces outils est conçue pour une tâche spécialisée. Ce sont des outils figés que l'on ne peut utiliser que dans un contexte précis et avec un certain type de structure de données.

Les postes de travail sont relativement complets mais leur structure empêche leur modification ou l'ajout d'un module. Ils sont eux aussi basés sur des structures de données figées.

Dans la recherche en lexicologie, il est important de pouvoir manipuler aisément différents types de structures de données et gérer différentes versions d'un dictionnaire en même temps.

De plus, étant donnée la grande diversité d'outils déjà disponibles et les besoins évolutifs en lexicographie, il est indispensable de pouvoir créer un poste de travail qui puisse intégrer différents modules.

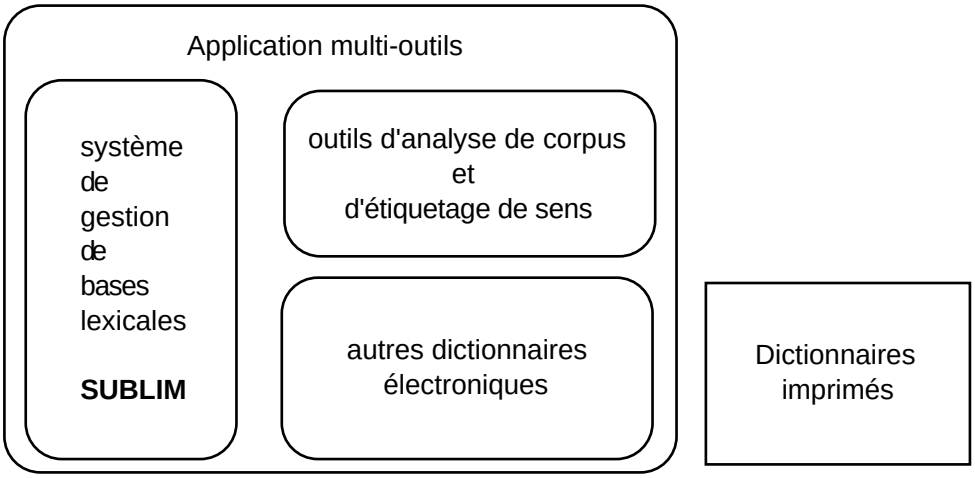
Pour répondre à ces deux critères, nous avons essayé de définir un outil générique modulaire.

Partie 3 : Méthodologie générale retenue

1. Description des solutions envisageables

Dans les temps qui nous sont impartis, il nous a fallu choisir un aspect particulier de ce poste de travail à implémenter. Nous décrivons ici les différentes possibilités qui nous sont apparues.

1.1. Intégrer SUBLIM dans chaque poste de travail



Poste de travail du lexicographe

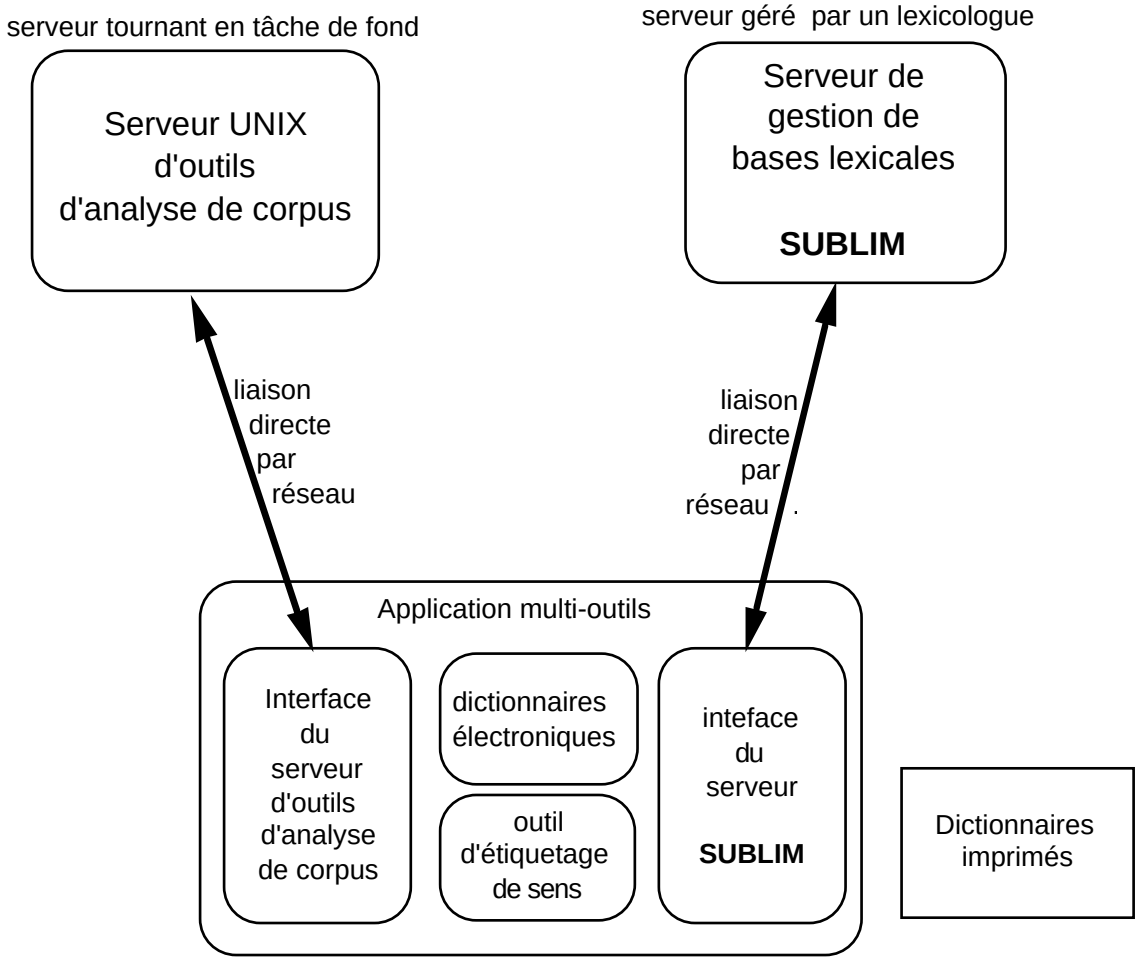
La première solution qui nous vient à l'esprit est d'intégrer le noyau de gestion de bases lexicales SUBLIM dans chaque poste de travail. Les lexicographes travailleront indépendamment en ayant chacun à sa disposition les outils de SUBLIM (défauteurs et vérificateurs de cohérence). L'indexation se fait directement sans nécessité de passer par un format intermédiaire.

Des problèmes peuvent par contre surgir lors de la fusion des données. En effet, si le travail de lexicographie est effectué par plusieurs personnes indépendamment les unes des autres, les choix pris lors de l'indexage peuvent diverger fortement.

De plus, si la structure linguistique doit être changée en cours d'indexage, il faudra la changer sur chaque poste de travail.

La solution reste cependant intéressante dans le cas où le lexicographe travaille seul sur un projet.

1.2. Installer SUBLIM sur un serveur accessible par interface réseau



1.2.1. Description générale de la solution

Le système de gestion de bases lexicales SUBLIM est installé sur un serveur. Les postes de travail des lexicographes sont construits de façon à être modulables et aisément paramétrables. Ils seront reliés au serveur du système SUBLIM par l'intermédiaire d'une interface réseau. Une solution technique évidente dans cette architecture est l'utilisation d'un client web supportant les applets Java comme poste du lexicographe.

1..2.2. Le serveur SUBLIM

Le noyau SUBLIM étant installé sur un serveur, Il faudra donc mettre en oeuvre une boîte à outils pour les différentes structures de données déclarées et créées en SUBLIM pour qu'elles puissent être accessibles et modifiables à distance.

Le lexicologue disposera de son interface qui lui permettra de spécifier et construire les types de structures utilisés par les lexicologues pour l'indexation ainsi que de les modifier par la suite. Il pourra aussi définir le type de vues de la base qu'il permettra aux lexicographes. Il aura à sa disposition un véritable outil de création d'interfaces pour les lexicographes.

Cette interface pour lexicographes devra être définie pour une base précise et permettre des visualisations différentes pour la même vue ainsi qu'une gestion globale des différentes vues associées. Elle sera incluse dans le poste de travail grâce à une architecture mettant en œuvre des objets collaborant du type de Open Doc ou des Java Beans.

1.2.3. Les postes de travail

Sur les postes de travail, on doit pouvoir avoir accès au serveur SUBLIM mais aussi à un serveur d'analyse de corpus sur UNIX ou à d'autres serveurs. Il faut aussi avoir la possibilité d'accéder à des dictionnaires électroniques distants ou sur le poste de travail, des dictionnaires sur papier, des outils d'étiquetage de sens, etc...

L'architecture du poste de travail du lexicographe doit donc être ouverte et permettre la communication entre applications, l'intégration d'outils extérieurs et la gestion de versions différentes d'un même outil. Elle doit simplifier le travail et minimiser les développements.

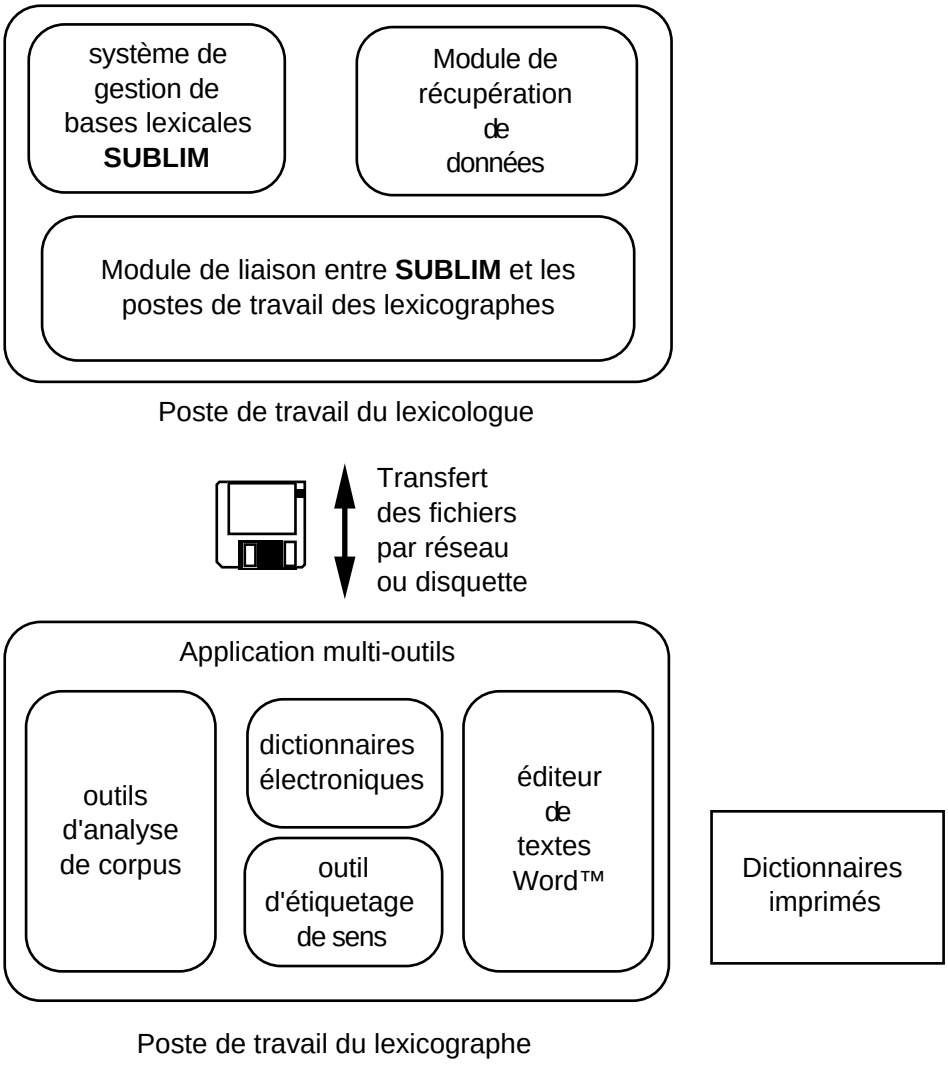
Il faut pour cela mettre en place un protocole d'intégration d'objets dans le poste de travail et organiser une communication directe entre les modules. Ceux-ci resteront cependant indépendants.

Pour que l'outil puisse servir véritablement de poste de travail pour lexicographe, il faudra intégrer rapidement plusieurs modules.

La réalisation d'un poste de travail ainsi défini représente un effort considérable. Au GETA, plusieurs personnes travaillent à sa réalisation en en développant chacun une partie. Les résultats de la recherche en matière d'outils pour lexicographes nous permettrait d'intégrer rapidement les travaux de M. Hai Doan Nguyen sur un système de récupération de données à partir de dictionnaires existants ainsi que ceux de M. Jean Gaschler qui étudie les résultats d'analyse de corpus.

Pour une tâche particulière, il suffira d'ajouter un module spécifique ou d'adapter les outils déjà disponibles sur le poste de travail.

1.3. Un serveur SUBLIM et des postes de travail indépendants



1.3.1. Description générale de la solution

Cette solution répond aux besoins du projet UNL pour lequel il va falloir indexer un très grand nombre de termes le plus rapidement possible. Une équipe de lexicologues sera chargée de collecter les données existantes, les envoyer pour être révisées à une équipe de lexicographes et les rassembler.

Les contraintes de temps et de coût nous obligent à utiliser des logiciels existants et répandus. Il nous faut donc mettre en place un outil de liaison entre le poste d'indexage simple et la base lexicale aux formats logiques plus élaborés.

1.3.2. Le poste du lexicologue

Le poste du lexicologue sera donc construit autour du système de gestion de bases lexicales SUBLIM. Le lexicologue pourra récupérer des données de dictionnaires existants, préparer ces données pour les envoyer en révision/complétion aux lexicographes et les rassembler une fois l'indexage fini pour les intégrer dans la base sous SUBLIM.

Il aura besoin de spécifier la structure linguistique de sa base lexicale et sa correspondance dans l'éditeur du lexicologue. Il devra pouvoir aussi gérer différentes versions en même temps.

1.3.3. Les postes d'indexation

Le lexicographe éditera les données reçues du lexicologue pour les compléter. Pour cela, il lui faut des outils d'aide à l'indexage sur son poste de travail. Il pourra aussi utiliser d'autres outils de son choix. Une fois son travail fini, il devra renvoyer le tout au lexicologue.

1.4. Conclusion

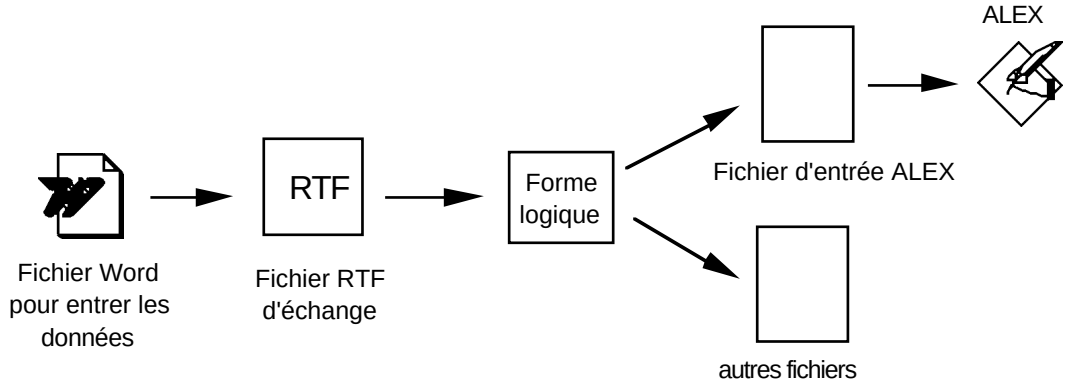
Ces trois solutions sont complémentaires. On pourrait imaginer voir fonctionner un serveur avec une interface Java et qui puisse aussi préparer des fichiers dans le format d'un éditeur de textes.

L'inconvénient majeur de la deuxième solution est qu'elle requiert d'importants moyens informatiques qui sont d'ailleurs rarement disponibles sur les plateformes d'indexage. Nous avons pu le constater, par exemple, lors du projet d'indexation du dictionnaire FeM.

Pour répondre aux besoins très forts à court terme du projet UNL dans les temps qui nous sont impartis et tenant compte de cette contrainte, nous avons choisi d'implémenter la dernière solution. Nous essayerons dans la mesure du possible de garder le maximum de fonctionnalités d'un outil générique qui sera réalisé en « mode dégradé ».

La conception et la réalisation de la deuxième solution demande plus de moyens mais le manque constaté dans l'état de l'art d'un tel outil fait qu'il semble indispensable de travailler ultérieurement à son élaboration.

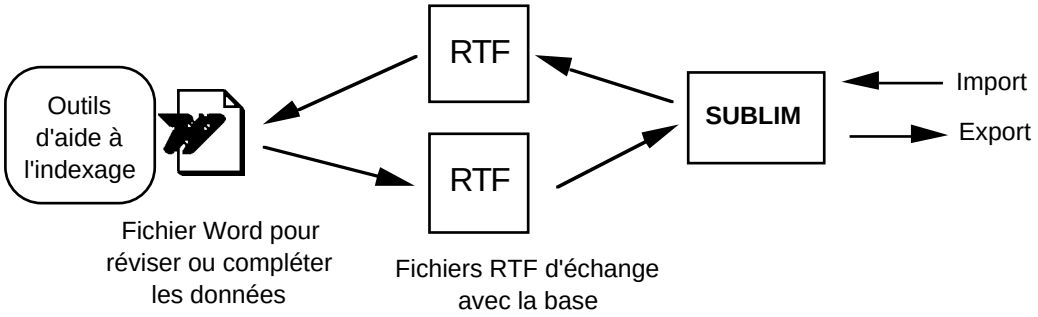
2. Description des spécifications



Indexation du dictionnaire FeM

L'indexation du dictionnaire FeM a été réalisée sur le logiciel Microsoft Word™ par des personnes sans compétences particulières en informatique ni en lexicologie. Les fichiers Word™ étaient ensuite révisés par des linguistes puis transformés en forme logique LISP qui est ensuite utilisée par d'autres applications (ALEX, base 4D, autres dictionnaires, etc).

Microsoft Word™ est un logiciel très répandu. Il peut fonctionner aussi bien sur Macintosh que sur PC. Le coût de l'équipement et de la main d'œuvre a donc pu être diminué.



Indexation des dictionnaires UNL

Lors de l'indexation des dictionnaires UNL, nous reprendrons ce scénario pour réduire les coûts en l'améliorant pour faciliter le travail.

Les informations de référence seront aussi des objets LISP éventuellement traitables dans une base SUBLIM. Nous importerons des données de dictionnaires existants pour les transformer en objets LISP. À partir de ces objets, nous générerons des fichiers Word™ pour les envoyer à la révision/complétion et les récupérer ensuite. Lorsque les objets LISP seront complétés, nous pourrons produire des fichiers de différents formats pour les outils dont nous disposons (4D, Alex, ARIANE-G5, etc).

Pour améliorer l'indexage, nous proposerons des outils d'aide sur le poste de travail du lexicographe. Ces outils devront être génériques pour ne pas perdre de vue l'objectif d'un poste de travail aisément paramétrable. Nous utiliserons ce poste en « mode dégradé ».

Partie 4 : Construction générique d'interfaces Word pour la lexicographie distribuée

1. Les besoins du projet UNL

1.1. Description du langage UNL

UNL sert de langage pivot pour la traduction et à ce titre, il doit être capable de représenter exactement toute l'information exprimée dans n'importe quelle langue.

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 convertisseurs et des générateurs.

Le vocabulaire UNL consiste en :

- UWs (Universal Word) qui représentent le sens d'un mot. 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 qui représentent une relation entre UWs

- étiquettes d'attributs qui expriment d'avantage de définition ou des informations additionnelles d'UWs qui apparaissent dans les phrases.

Les UWs dérivés du mot anglais « book » sont décrits comme suit :

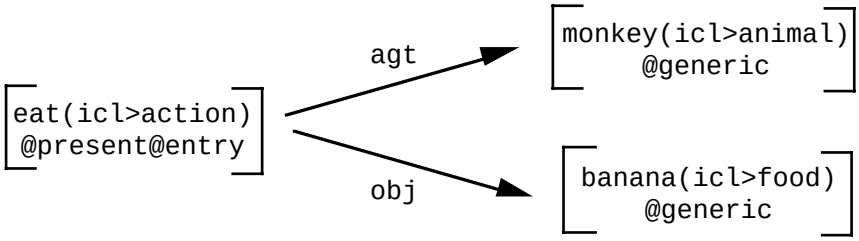
```
book                        : garde tout sens possible

book(icl>publications)      : sens limité par un UW hyper ordonné
                            = livre en tant que publication

book(=accounts)             : sens limité par un UW synonyme
                            = livre de comptes

book(obj>room)              : restriction par une relation distinctive
                            = réserver (une chambre)
```

Les étiquettes de relations explicitent les relations entre les sens. Elles doivent être telles que le sens composé puisse être représenté par une combinaison de sens et une relation.

Les étiquettes d'attributs se distinguent par le symbole @ au début. Elles expriment des relations logiques, des informations de temps, d'aspect, des intentions.

Exemple d'expression logique UNL :

Monkey eats bananas.



Pour représenter du texte au format UNL, différentes étiquettes indiquent le début et la fin des documents ([D][/D]), paragraphes ([P][/P]), phrases ([S][/S]), et expressions UNL ([UNL][/UNL]). Les expressions sont composées d'un ensemble de mots UNL, les UW ([W][/W]), et de relations binaires ([R][/R]). Voici la même expression au format physique telle qu'on la trouvera sur le réseau sous forme de page HTML ou de fichier texte :

Monkey eats bananas.

```
[S]
[W]
eat(icl>action).@present.@entry:00
monkey(icl>animal).@generic:01
banana(icl>food).@generic:02
[/W]
[R]
00agt01
00obj02
[/R]
[/S]
```

1.2. Les dictionnaires UNL à construire

Dans le projet UNL, le GETA est chargé de développer le générateur et l'analyseur du français. Il faut donc dans un premier temps réunir rapidement un grand nombre de données et construire un dictionnaire UNL-français qui servira à traduire les structures UNL. Pour des raisons de compatibilité évoquées plus haut, le dictionnaire aura la forme du dictionnaire UNL-anglais dont voici un extrait :

```
[he]{} he "he" (3SG,HPRON,PRON,SUBJ) <E,0,1>;
[him]{} him "he" (3SG,HPRON,OBJ,PRON) <E,0,1>;
[his]{} his "he" (3SG,HPRON,POSS,POSSN,PRON)  <E,0,1>;
[how]{} how "how" (ADV,QADV,STM) <E,0,1>;
[investigat]{} investigate "investigate"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VI) <E,0,1>;
[investigat]{} investigate "investigate(agt>police)"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VI);
[investigat]{} investigate "investigate(icl>examination)"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VI);
[investigat]{} investigate "investigate(icl>research)"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VI);
[investigat]{} investigate "investigate(icl>search)"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VI);
[investigat]{} investigate "investigate(icl>survey)"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VI);
[investigation]{} investigation "investigation" (C,N,PLS);
[is]{}is "exist" (V,BE,3SG);
```

```
[me]{} me "I" (1SG,HPRON,OBJ,PRON) <E,0,1>;
[mine]{} mine "I" (1SG,HPRON,OBJ,POSSN,PRON) <E,0,1>;
[my]{} my "I" (1SG,DET,HPRON,POSS,PRON) <E,0,1>;
[myself]{} myself "I" (1SG,HPRON,PRON,SELF) <E,0,1>;
[persuad]{} persuade "persuade"
(3SGES,BAE,EDED,ENED,INGING,V,VCTI,VDO,VDON,VDOT,VIO,VOC);
[persuad]{} persuade "persuade(icl>inaugurate-in_service-retire)"
(3SGES,BAE,EDED,ENED,INGING,V,VCTI,VDO,VDON,VDOT,VIO,VOC);
[persuad]{} persuade "persuade(icl>persuade)"
(3SGES,BAE,EDED,ENED,INGING,V,VCTI,VDO,VDON,VDOT,VIO,VOC);
[persuasion]{} persuasion "persuade" (C,N,NRQSG,SG,U);
[promis]{} promise "promise"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VDOT,VDOTI,VI,VIO,VOTH)
<E,0,1>;
[promis]{} promise "promise(icl>expect)"
(3SGES,BAE,EDED,ENED,INGING,V,VDO,VDON,VDOTI);
[promise]{} promise "promise" (C,N,NRQSG,PLS,S5,SG);
[promise]{} promise "promise(=good_sign)" (N,U)
[promise]{} promise "promise(icl>statement)"
(C,N,NWAR,PLPAS,PLS,SG,SGPAS,THATCB);
[promised]{} promised "promised" (ADJ,BA,NOCS,PREN)
[the]{} the "the" (ART,DET);
[to]{} to "to"  (PREP);
[was]{} was "exist" (V,BE,1SG,3SG,ED);
[were]{} were "exist" (V,BE,1PL,2PL,3PL,2SG,PL,ED);
[what]{} what "what" (DET,QDET);
[what]{} what "what" (DT?,OBJ,PRON,QPRON,SUBJ) <E,0,1>;
[what]{} what "what" (INTJ);
[when]{} when "when" (ADV,QADV,STM) <E,0,1>;
[where]{} where "where" (ADV,QADV,STM) <E,0,1>;
[why]{} why "why" (ADV,QADV,STM) <E,0,1>;
[you]{} you "you" (2PL,2SG,HPRON,OBJ,PRON,SUBJ) <E,0,1>;
[your]{} your "you" (2PL,2SG,DET,HPRON,POSS,PRON) <E,0,1>;
[yours]{} yours "you" (2PL,2SG,HPRON,POSSN,PRON) <E,0,1>;
[yourself]{} yourself "you" (2SG,HPRON,PRON,SELF) <E,0,1>;
[yourselves]{} yourselves "you" (2PL,HPRON,PRON,SELF) <E,0,1>;
```

On dispose déjà d'un certain nombre de dictionnaires contenant des lexiques français-anglais et anglais-français. Nous nous en servirons pour préparer, dans la base lexicale, les entrées UNL. Il nous faudra ensuite les générer en RTF pour révision/complétion. Pour cela, nous avons défini une liste de styles Microsoft Word™ selon nos besoins.

Il nous faut préciser :

- les lemmes français

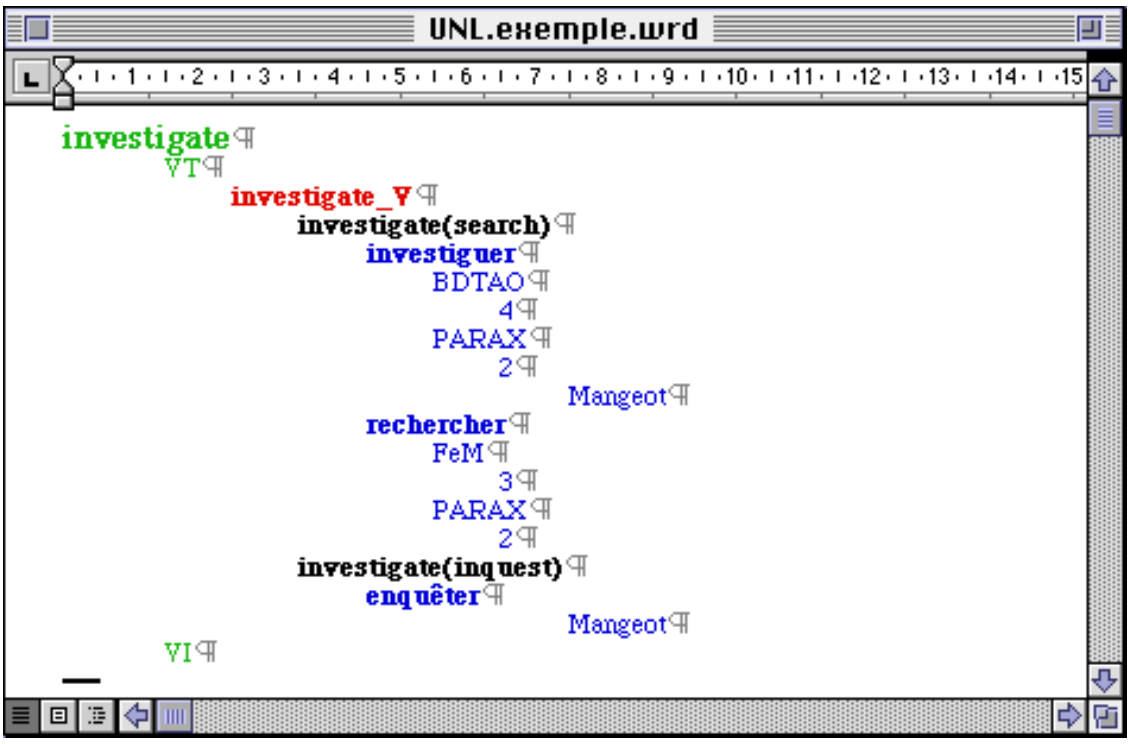
- les propriétés de chaque lemme (arguments, valeur, traits sémantiques, ...)

- les mots reliés (fonctions lexicales, dérivés)

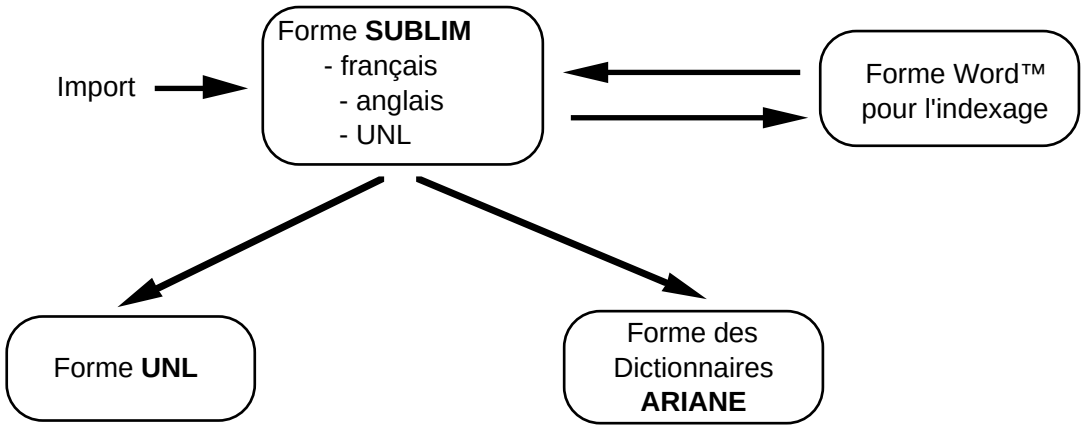
Voici un aperçu de la liste des styles Microsoft Word™ que l'on utilisera pour l'indexage :

style word	contenu	commentaire
lemme_a	investigate	lemme anglais
cat_a	VT	catégorie grammaticale
lemme_geta	investigate_V	appellation interne
UW	« investigate(search) »	mot UNL
lemme_f	investiguer	lemme français
provenance_f	BDTAO	dictionnaire ou le lemme figure déjà
num_sem_f	4	nombre de divisions sémantiques
provenance_f	PARAX	
num_sem_f	2	
manuel_f	« nom du lexicographe »	si le lemme est indexé manuellement
lemme_f	rechercher	autre lemme français
provenance_f	FeM	
num_sem_f	3	
provenance_f	PARAX	
num_sem_f	2	
UW	« investigate(inquest) »	autre mot UNL
lemme_f	enquêter	autre lemme français
manuel_f	« nom du lexicographe »	
cat_a	VI	autre catégorie

Voici le même exemple indexé en fichier Word™ :



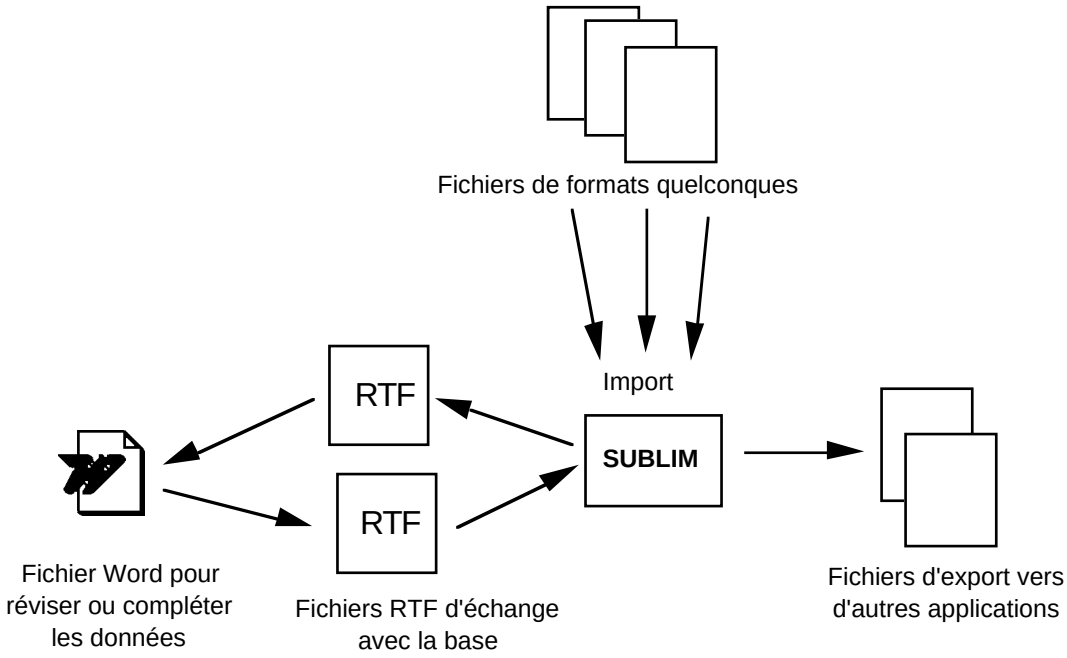
La base lexicale sera définie dans SUBLIM comme un dictionnaire trilingue. (anglais, français, UNL). À partir de cette base, nous générerons la forme UNL et la forme nécessaire aux applications de traduction en ARIANE-G5.



Conversion des données entre les différentes formes utilisées

2. Principe Général

2.1. Introduction



Une session d'indexage se décompose en trois parties : la préparation des données, les entrées que l'on va envoyer à l'indexage, l'indexation à proprement parler exécutée par plusieurs lexicographes sur leur poste de travail et la récupération des données une fois indexées.

Nous avons testé l'application et les macros avec les styles et les fichiers de données du dictionnaire FeT. Ce dictionnaire est une extension du dictionnaire FeM (français, anglais, malais) décrit dans l'état de l'art. Il reprend les données du FeM et y ajoute des équivalents en thai.

La définition des styles et des paramètres du dictionnaire UNL n'étant pas encore fixée, nous avons préféré tester notre application avec des données complètes.

Nous pourrons ainsi vérifier la généricité de notre implémentation lors de l'adaptation de l'application aux dictionnaires UNL.

2.2. Une session d'indexage

2.2.1. Préparation des données

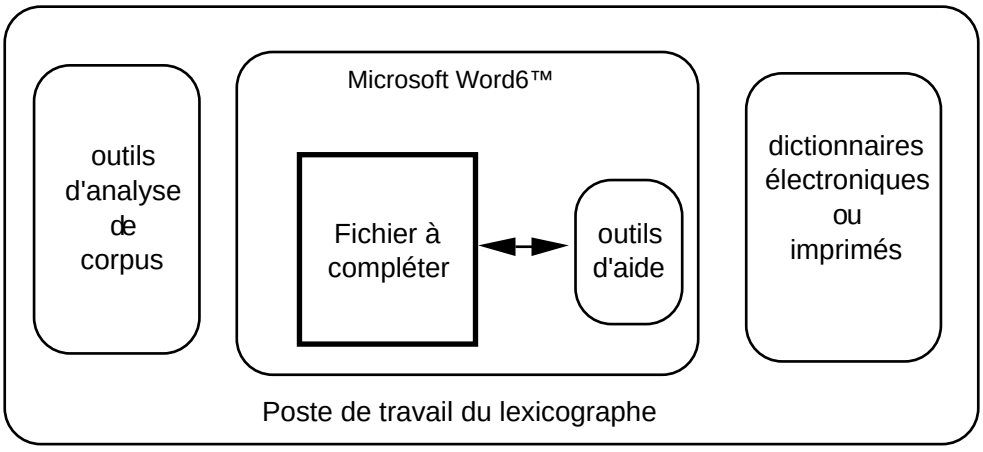
Au départ, on essaye de rassembler le maximum de données déjà disponibles même si leur format n'est pas le même que celui que l'on va utiliser. Il est très rare que l'on construise un dictionnaire ex nihilo, sans références.

Voici un exemple d'objets LISP incomplets que l'on va envoyer à l'indexation. Il contient l'entrée « s'abaisser »

```
(FET_ENTRY
(:ENTRY "abaisser")
(:FRENCH_PRON "<french_pron>")
(:FRENCH_CAT "<french_cat>")
(:ENGLISH_EQU "<english_equ>")
(:MALAY_EQU "<malay_equ>")
(:THAI_EQU "<thai_equ>"))
```

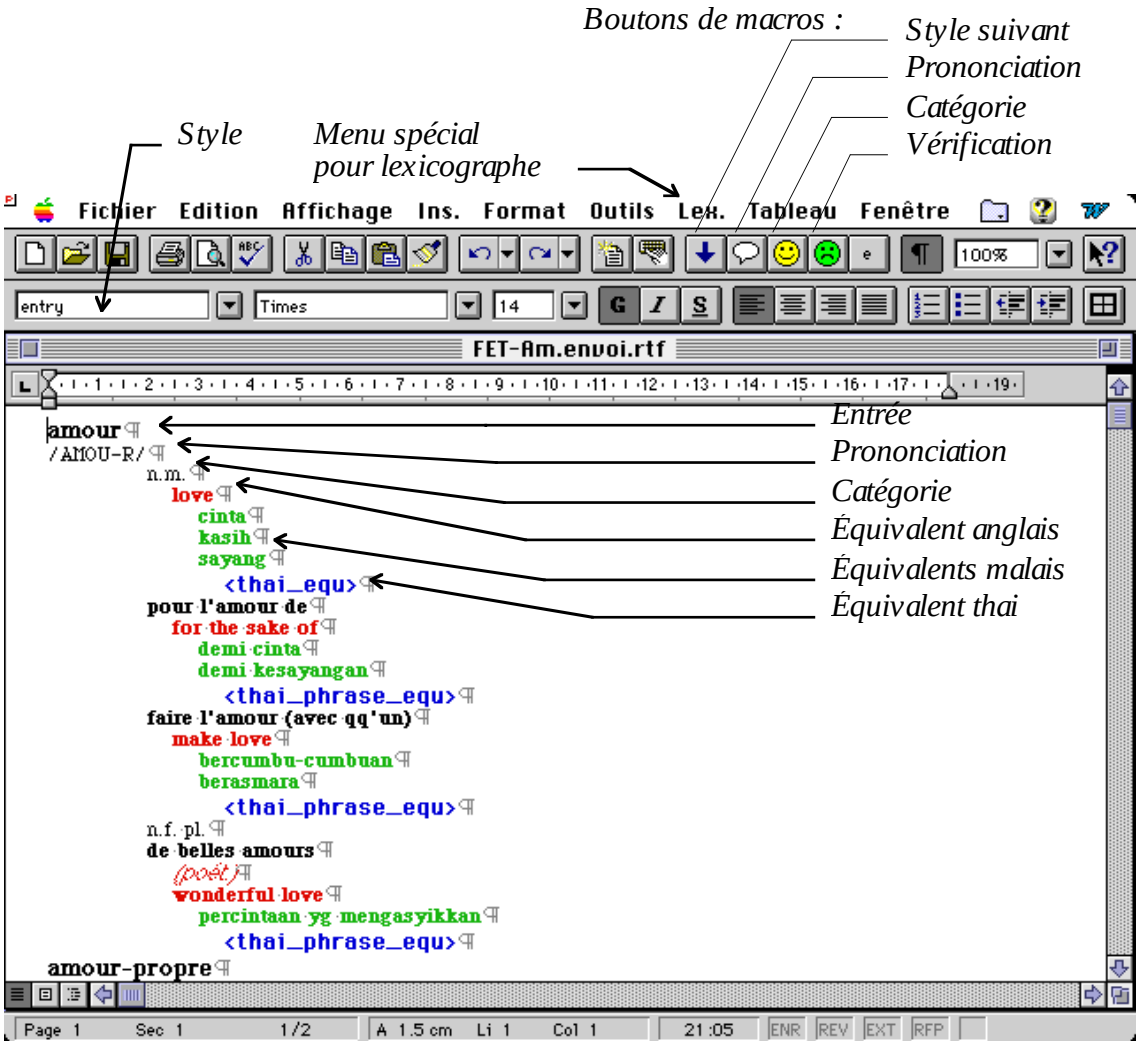
À partir de ces objets, le lexicologue génère des fichiers RTF. Il les envoie ensuite par disquette ou réseau à chaque lexicographe pour la révision/complétion.

2.2.2. Indexation



Le lexicographe organise son poste de travail comme il le souhaite. Il peut utiliser des outils complémentaires d'analyse de corpus ou d'étiquetage de sens. Il doit cependant être équipé du logiciel Microsoft Word 6™ pour pouvoir éditer les fichiers en format RTF mais surtout utiliser les macros créées pour l'aide à l'indexage et la liste des styles définis par le lexicologue pour assurer la correspondance entre un champ de donnée et un style.

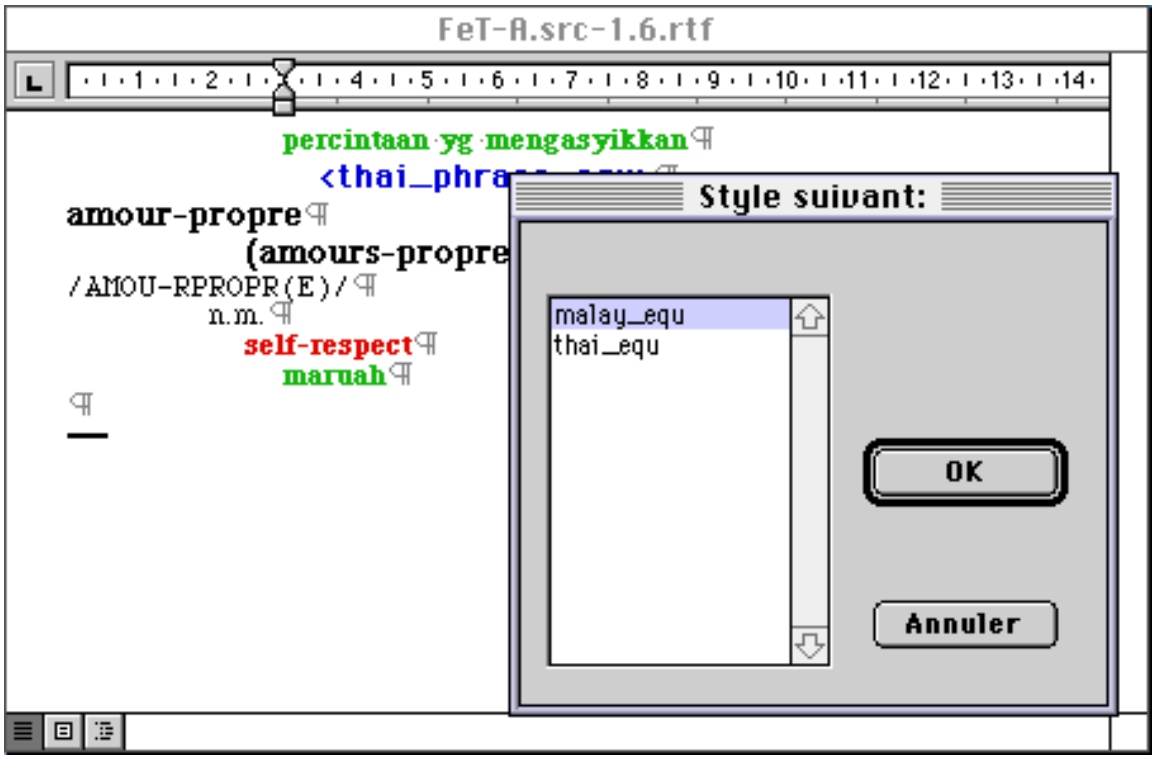
Chaque lexicographe travaille de son côté sur son poste de travail. Une fois les fichiers reçus, il peut éditer le fichier RTF et le compléter.



Indexation de l'entrée « amour »

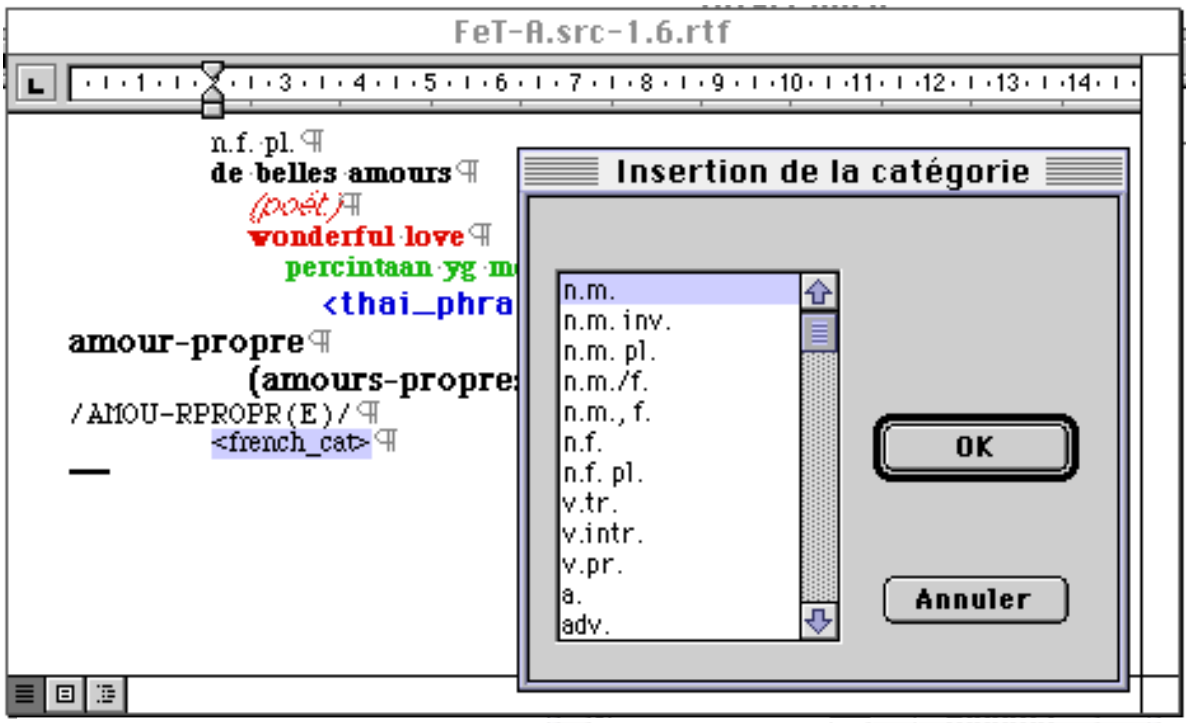
Pendant l'indexage, il peut consulter ses propres données : dictionnaires papier, autres dictionnaires électroniques ou un exemple de fichier RTF complété. Il est libre d'utiliser d'autres outils d'analyse de corpus ou d'étiquetage de sens qui peuvent l'aider à indexer ses termes.

Les macros sont présentes pour l'aider à indexer. Lorsque le lexicographe a fini de remplir un champ, il appelle la macro **style suivant** 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 de style automatiquement.



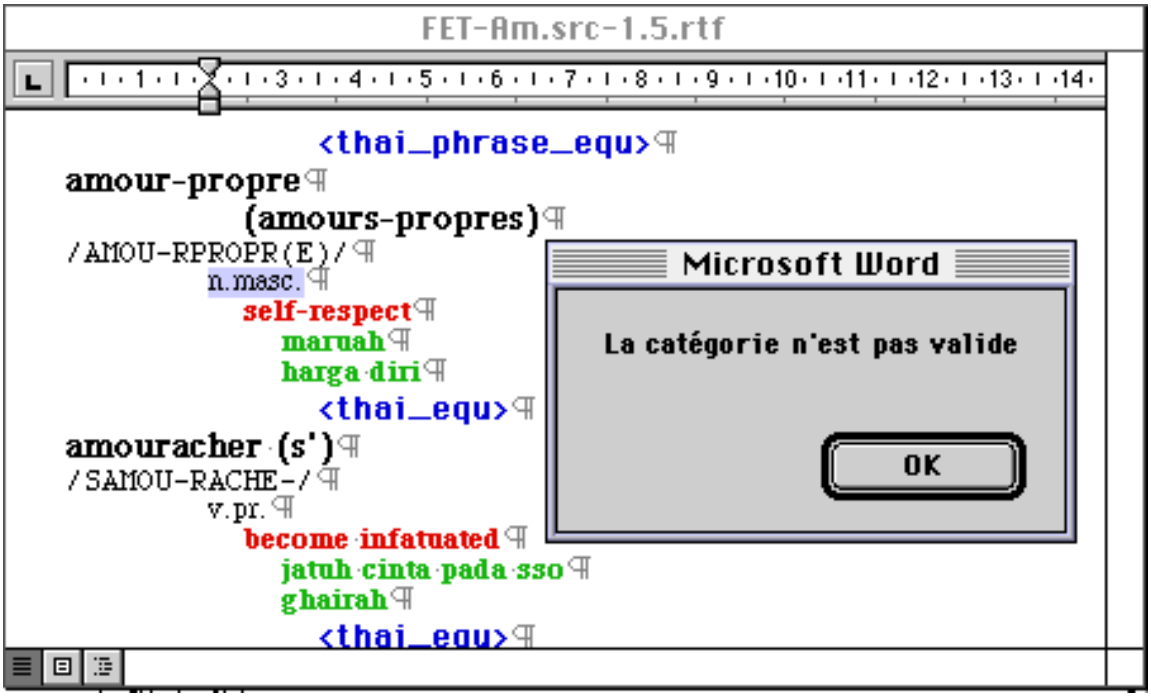
Fenêtre de la macro **style suivant**

Grâce à la macro **liste valeurs**, 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. Lorsqu'il doit indiquer une catégorie grammaticale, le lexicographe appelle la macro **liste valeurs** qui affiche automatiquement la liste des catégories autorisées. Il en sélectionne une et la macro l'insère.



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 si le champ n'est pas correctement rempli.



Message d'erreur suite à la vérification d'une catégorie

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 à ce champ. Cette macro n'a pas encore été implémentée.

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

2.2.3. Synthèse

Lorsque le lexicologue reçoit les fichiers de données RTF, celui-ci les reconvertit en une forme logique. Voici le résultat de l'indexation de l'entrée « abaisser »

abaisser
/ABE-SE-/
v.tr.
(baisser)
to lower
to pull down
to push down
merendahkan
menurunkan
tarik ke bawah
tolak ke bawah (dgn penyungkit, pengumpil, dsb)
∑Ì"„ÀÈ≈¥≈ß
∑Ì"„ÀÈµÌË"≈ß
¥÷ß≈ß
º≈—°≈ß
(fig.)
(humilier)
to humiliate
memperkecilkan
∑Ì"„ÀÈ‰¥ÈÕ"
∑Ì"„ÀÈµÌË"µÈÕ

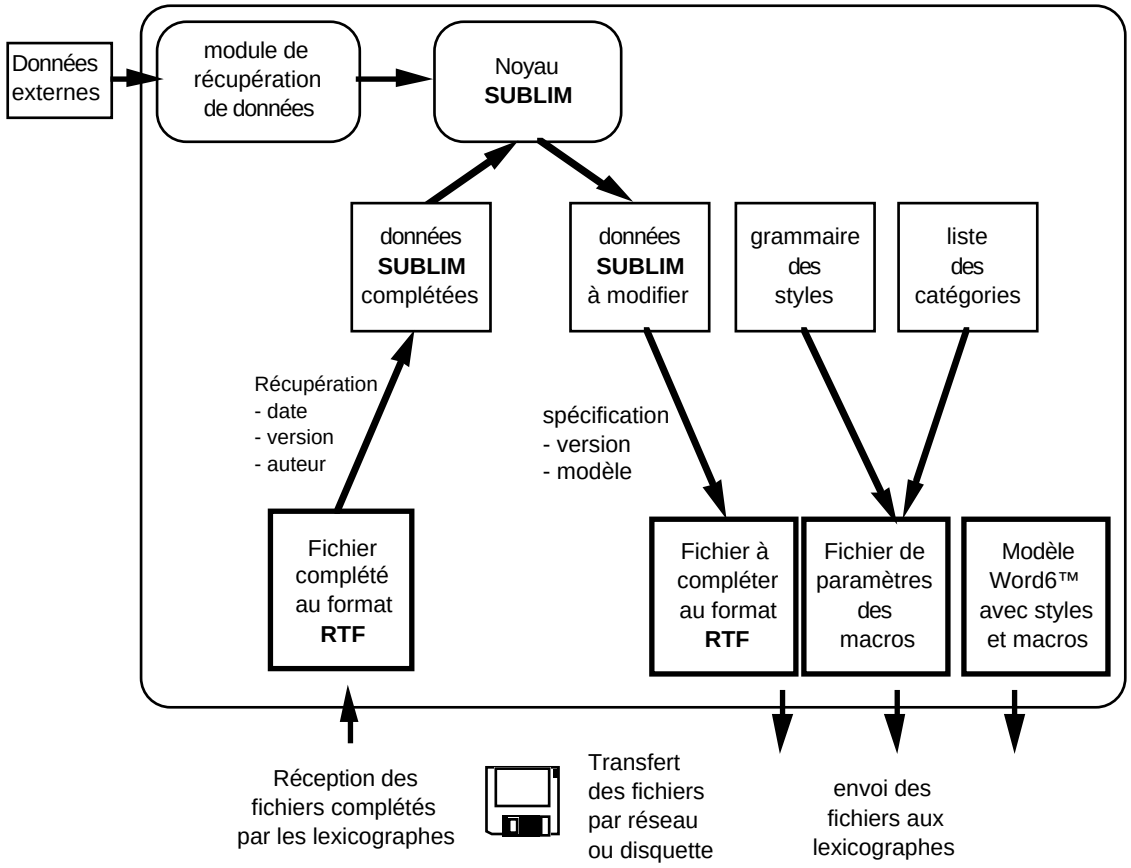
La forme logique de l'entrée « abaisser » est la suivante :

```
(:AUTHOR "Mathieu Mangeot")
(:VERSION "FET-A.src-1.2")
(:DATE "mai 29, 1997")
(FET_ENTRY
(:ENTRY "abaisser")
(:FRENCH_PRON "/abe-se-/")
(:FRENCH_CAT "v.tr.")
(:FRENCH_GLOSS "(baisser)")
(:ENGLISH_EQU "to lower")
(:ENGLISH_EQU "to pull down")
(:ENGLISH_EQU "to push down")
(:MALAY_EQU "merendahkan")
(:MALAY_EQU "menurunkan")
(:MALAY_EQU "tarik ke bawah")
(:MALAY_EQU "tolak ke bawah (dgn penyungkit, pengumpil, dsb)")
(:THAI_EQU "∑Ì“„ÀÈ≈¥≈ß")
(:THAI_EQU "∑Ì“„ÀÈµÌË“≈ß")
(:THAI_EQU "¥÷ß≈ß")
(:THAI_EQU "°≈—°≈ß")
(:FRENCH_LABEL "(fig.)")
(:FRENCH_GLOSS "(humilier)")
(:ENGLISH_EQU "to humiliate")
(:MALAY_EQU "memperkecilkan")
(:THAI_EQU "∑Ì“„ÀÈ‰¥ÈÕ“¬")
(:THAI_EQU "∑Ì“„ÀÈµÌË“µÈÕ¬"))
```

Lors de l'extraction des données des fichiers, le lexicologue peut vérifier si les styles utilisés dans le fichier appartiennent bien à la liste des styles définie au préalable. Dans le cas contraire, l'application génère un message d'erreur.

Si le cas l'exige, il peut appliquer un changement de version automatique au fichier. Si des données manquent, cela sera lui signalé et il pourra prendre les mesures nécessaires soit pour compléter lui même les données, soit pour les retourner à l'indexage.

3. Le poste du lexicologue



Poste de travail du lexicologue

Le poste est articulé autour du noyau de gestion de bases lexicales SUBLIM. Le lexicologue dispose d'un certain nombre d'outils qui lui permettent de :

- décrire la structure de ses entrées utilisée pour l'indexage;
- générer les fichiers Word™ contenant des données incomplètes;
- les récupérer une fois complétés par les lexicographes;
- faire des conversions de version si nécessaire;
- générer des fichiers dans d'autres formats comme UNL ou ARIANE-G5.

3.1. Description des entrées

En premier lieu, le lexicologue doit spécifier la grammaire de sa structure. L'indexation à l'aide du logiciel Word™ nous oblige à utiliser une structure de champs. Une grammaire des champs du dictionnaire FeT se trouve en annexe A. Cette grammaire servira à calculer la liste des champs (nœuds terminaux). On trouvera en annexe B un exemple de liste produite pour le FeT.

La liste des suivants possibles ainsi que la liste des valeurs pour tous les champs ayant une liste fermée de valeurs sont elles aussi générée à partir de la grammaire. Elle seront utilisées par les macros **style suivant**., **vérification**, et **vérification générale.**

Nous utiliserons aussi la grammaire pour vérifier la structure dans son ensemble et pour construire une structure plus abstraite qu'une simple suite de champs lors de l'analyse suivie de la liste des champs du dictionnaire FeT en annexe B.

Il doit définir à l'aide de la commande style de Word™ dans un fichier la liste des styles que le lexicographe utilisera pour l'indexage. Chaque style doit correspondre à un champs défini plus haut. De plus, le nom du style doit être le même que le nom du champs de la structure. Pour une présentation plus agréable, il est nécessaire de définir des styles bien différents les uns des autres. Pour cela, le lexicologue peut utiliser des polices, des tailles et des couleurs de caractères ainsi que des indentations différentes.

Le fichier est ensuite convertit au format RTF et sa partie en-tête récupérée. Elle sera utilisée lors de la génération. Si un champ n'a pas de correspondance avec un style Word™ défini dans cet en-tête, il ne pourra donc pas être affiché dans les fichiers envoyés aux lexicographes par la suite.

3.2. Génération des fichiers

3.2.1. Le modèle Word™

Lors d'une session avec le logiciel Microsoft Word 6™, le lexicographe utilise un modèle de documents. Les modèles contiennent la configuration de la barre des menus et de la barre d'outils, une liste des styles utilisés ainsi que les macros programmées en Wordbasic. Nous avons donc construit un modèle de documents qui contient les macros d'aide à l'indexation.

Comme les macros, une fois écrites, sont compilées et enregistrées dans le modèle, il nous a paru plus simple de programmer des macros génériques et de les paramétrer que de générer des macros en Wordbasic automatiquement à partir de leur définition par le lexicologue. De plus, si l'on veut écrire une macro pour répondre à des besoins plus spécifiques d'un certain travail d'indexation, il est assez aisé de le faire.

Pour gérer les différentes versions, les macros vont lire dans un fichier de paramètres. Elles vont le chercher en premier lieu dans le dossier associé au fichier RTF mais si elles ne le trouvent pas, elles vont prendre un fichier par défaut situé dans un des dossiers de l'application Microsoft Word™. Cela permet d'associer des paramètres différents pour chaque fichier RTF envoyé à l'indexation.

Nous avons modifié la configuration de la barre des menus de notre modèle pour y insérer un menu spécial appelé « **Lex.** ». Il contient la liste des macros d'aide à l'indexation. Nous avons aussi ajouté un bouton pour chaque macro dans la barre d'outils ainsi qu'un équivalent clavier.

Le modèle Word™ ainsi défini se comporte comme un fichier classique. Il est lisible aussi bien par des Macintosh que par des PC. Le lexicologue le garde à sa disposition. Il l'envoie ensuite sans le modifier à chaque lexicographe lors de sa première session d'indexage. Par la suite, il suffira de changer les fichiers de paramètres des macros pour prendre en compte les modifications des structures de champs définies par le lexicologue.

3.2.2. Le fichier RTF

Le poste génère, à partir d'objets LISP incomplets la plupart du temps, un fichier en format Rich Text Format (RTF) lisible par le logiciel Microsoft Word™. Pour cela, il utilise l'en-tête du fichier dont le lexicologue s'est servi pour définir les styles au préalable. Si un champ d'un objet LISP n'a pas un style équivalent dans cet en-tête, un message d'erreur est généré.

Pour plus de commodité, le lexicologue peut, s'il le souhaite spécifier un numéro de version pour ce fichier généré ainsi que le nom du modèle de documents à utiliser pour l'indexage des entrées contenues dans le fichier. Ces informations sont insérées dans l'en-tête du fichier RTF.

3.2.3. Le fichier de paramètres

Les macros d'aide à l'indexation du modèle de documents utilisent un fichier de paramètres. Le poste génère ce fichier à partir de la grammaire des champs définie antérieurement. Il utilise l'algorithme du calcul des suivants défini dans [2]. Ce qui lui permet de calculer pour chaque champ la liste des champs suivants autorisés par la grammaire. Cette fonction est en cours d'implémentation.

À la suite de la liste des suivants de chaque champ sont insérées les listes de valeurs des champs qui ont une liste fermée. Ces fichiers sont écrits en format texte et placés dans le même dossier que le fichier RTF.

3.3. récupération des fichiers

Lorsque le lexicologue reçoit les fichiers RTF complétés par le lexicographe, il doit récupérer les données et les convertir en objets LISP. La liste des champs produite à partir de la grammaire nous permet de vérifier que chaque champ récupéré appartient à la liste des champs comme celle définie en annexe B.

On récupère en même temps des informations nécessaires à la gestion du travail : la version du fichier, le nom de l'auteur et la date de dernière modification. Ces informations sont contenues dans l'en-tête du fichier.

3.4. Conversion des fichiers

Pour la gestion des différentes versions, nous pouvons indiquer le numéro de version des fichiers envoyés à l'indexage et le récupérer en même temps que les données.

Nous avons aussi défini et implémenté une fonction qui permet de passer d'une version à une autre. Il faut pour cela spécifier la correspondance entre les champs des deux versions. Un exemple de fichier est défini dans l'annexe D.

Conclusion

Nos outils ont été conçus pour répondre à la demande du projet UNL. Nous les avons finalement testé 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 se sont adaptés sans modifications de notre part. Cela confirme que l'aspect générique de notre poste de travail a bien été respecté.

Nous pouvons aussi assurer une gestion aisée des différentes versions grâce au numéro de version placé dans l'en-tête des fichiers envoyés à l'indexage, à l'utilisation de fichiers de paramètres différents pour chaque fichier d'indexage et à la fonction qui permet de changer de version.

Au premier abord, les outils d'aide à l'indexage semblent prometteurs. Ils permettent manifestement de gagner du temps et d'éviter des erreurs. Nous allons tout de suite pouvoir tester et améliorer l'ensemble de nos outils dans le cadre du projet UNL puisque l'indexation des dictionnaires devrait débuter le mois prochain.

Cette version dégradée d'un poste de travail plus complet ne doit cependant pas nous faire perdre de vue l'objectif décrit dans la deuxième des solutions proposées dans la partie 3. L'idée de reprendre cette solution et de la développer dans la perspective d'un travail futur nous paraît très prometteuse.

Bibliographie

[1] **J. Aarts & T. V. D. Heuvel (1985)** *Computational Tools for the Syntactic Analysis of Corpora.* Linguistics, **23**/pp. 303-335.
[2] **A. Aho, R. Sethi, J. Ullman (1986)** *COMPILATEURS Principes, techniques et outils* ed. Intereditions, Paris, 875 p.
[3] **B. T. S. 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.
[4] **D. Bachut (1984)** *ATLAS, manuel d'utilisation.* GETA, rapport interne, 37 p.
[5] **D. Bachut & N. Verastegui (1984)** *Software tools for the environment of a computer aided translation system.* Proc. COLING-84, Stanford, GETA, 4 p.
[6] **B. Boguraev et al. (1989)** *Computational lexicography for natural language processing.* B. Boguraev & T. Briscoe ed., Longman, Londres & New York, 310 p.
[7] **C. Boitet (1982)** *Le point sur ARIANE-78 debut 82 (DSE 1).* GETA-CHAMPOLLION, CAP SOGETI FRANCE, 252 p.
[8] **A. Buseman et al. (1996)** *The Linguist's Shoebox.* Summer Institute of Linguistics, 111 p.
[9] **R. J. Byrd et al. (1987)** *Tools and Methods for Computational Lexicology.* Journal of Computational Linguistics, **13**/3-4, pp. 219-240.
[10] **K. W. Church (1994)** *Unix (TM) for Poets.* Proc. ELSNET, European Summer School, Utrecht, Pays Bas, 53 p.
[11] **H. Cunningham, R. J. Gaizauskas & Y. Wilks (1996)** *GATE: A General Architecture for Text Engineering.* ILASH & DCS, University of Sheffield, UK, Décembre 95, 53 p.
[12] **H. Cunningham et al. (1997)** *Software Infrastructure for Natural Language Processing.* DCS, University of Sheffield, 10 février 1997, 9 p.
[13] **D. Curbow & E. Dykstra-Erickson (1995)** *The OpenDoc User Experience.* 22, juin 1995, pp. 83-97.
[14] **J. Gaschler & M. Lafourcade (1994)** *Manipulating Human-Oriented Dictionaries with Very Simple Tools.* Proc. COLING'94, Kyoto, Japon, vol. 1/2, pp. 283-286.
[15] **U. 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, Budapest, pp. 161-174.
[16] **M. Lafourcade (1996)** *Structured Lexical data: how to make them widely avalaible, useful and reasonable protected? - a practical example with a trilingual dictionary.* Proc. COLING-96, Copenhagen, Denmark, Vol 2/2, pp. 1106-1110.
[17] **M. Lafourcade (1996)** *Serveurs de dictionnaires - Etude de cas avec l'outil ALEX et le projet de dictionnaire françasi-anglais-malais.* Proc. Séminaire LEXIQUE, Grenoble, 13 et 14 novembre 1996, CLIPS-IMAG, Pôles langage naturel et parole du GDR-PRC CHM., vol. 1/1, pp 185-192.
[18] **G. Pérennou et al. (1992)** *Le Projet BDLEX de base de données lexicales du français écrit et parlé.* IRIT, UMR CNRS 5505 Groupe IHM-PT Université Paul Sabatier de Toulouse, 1992, 21 p.
[19] **G. Sérasset (1994)** *SUBLIM: un systeme universel de bases lexicales multilingues et NADIA: sa specialisation aux bases lexicales interlingues par acceptions.* Thèse de Doctorat, Spécialité Informatique, Université Joseph Fourier GRENOBLE 1, 194 p.

[20] **G. Sérasset (1996)** *Un Editeur pour le DEC du francais contemporain.* Proc. Séminaire Lexique, Grenoble, CLIPS IMAG, pp. 131-138.
[21] **UNL (1997)** *DeConverter Specification.* UNL center, Institute of Advanced Studies, The United Nations University, 1er Avril 1997, 25 p.
[22] **UNL (1996)** *Universal Networking Language.* UNL center, Institute of Advanced Studies, The United Nations University, 1996, 74 p.
[23] **M. Yaguello (1986)** *Alice au pays du langage: pour comprendre la linguistique.* 135 p.
[24] **R. Zajac, M. Casper & N. Sharples (1997)** *An Open Distributed Architecture for Reuse and Integration of Heterogeneous NLP Components.* Proc. ANLP'97, 7 p.

Annexe A
Grammaire des styles du dictionnaire FeT

```
START      -->  entry french_pron VAR_F
VAR_F      -->  cross_ref_marker cross_ref_entry START |
                french_masc_plus_form french_masc_plus_pron
                  french_cat GLOSES |
                french_fem_form french_fem_pron french_cat GLOSES |
                french_cat GLOSES
GLOSES     -->  english_equ E_EQU |
                english_gloss E_GLOSS |
                french_gloss F_GLOSS |
                french_pron VAR_F
E_EQU      -->  english_equ E_EQU |
                malay_equ M_EQU
E_GLOSS    -->  english_gloss E_GLOSS |
                english_equ E_EQU
F_GLOSS    -->  french_gloss F_GLOSS |
                french_label F_LABEL |
                english_equ E_EQU
M_EQU      -->  malay_equ M_EQU |
                thaï_equ T_EQU
T_EQU      -->  START |
                thaï_equ T_EQU |
                french_pron VAR_F |
                french_gloss F_GLOSS |
                french_label F_LABEL |
                french_sentence english_sentence malay_sentence
                  thaï_sentence FIN |
                thaï_equ
F_LABEL    -->  french_gloss F_GLOSS |
                english_equ E_EQU |
                english_phrase E_PHRASE
FIN        -->  START |
                french_gloss F_GLOSS |
                french_label F_LABEL |
                french_pron VAR_F |
                thaï_sentence
E_PHRASE   -->  english_phrase E_PHRASE |
                malay_phrase M_PHRASE
M_PHRASE   -->  malay_phrase M_PHRASE |
                thaï_phrase T_PHRASE
T_PHRASE   -->  START |
                french_pron VAR_F |
                french_gloss F_GLOSS |
                french_sentence english_sentence malay_sentence
                  thaï_sentence FIN |
                thaï_phrase T_PHRASE |
                thaï_phrase

french_cat (one-of « n.m.  » « n.m. inv. » « n.m. pl. » « n.m./f.  »
                « n.m., f.  » « n.f.  » « n.f. pl.  » « v.tr.  »
                « v.intr.  » « v.pr.  » « a.  » « adv.  » « prép.  »)
```

```
0entry          string :encoding
                         :roman-mac
english_equ     string :encoding
                         :roman-mac
malay_equ       string :encoding
                          :roman-mac
thai_equ        string :encoding
                          :thai-mac
```

Annexe B
Liste des styles du dictionnaire FeT

```
(:entry

:french_entry_number
:french_cat
:french_label
:french_gloss
:french_phrase
:french_sentence

:french_entry_variant

:french_subentry
:french_subentry_pron

:french_fem_form
:french_fem_plur_form
:french_fem_plur_pron
:french_fem_pron

:french_masc_plur_form
:french_masc_plur_pron
:french_masc_plur_pron_variant

:french_plur_form
:french_plur_form_pron
:french_plur_form_variant
:french_plur_form_variant_pron

:french_pron
:french_pron_variant

:french_rection

:cross_ref_entry
:cross_ref_marker

:commentaire

:english_equ
:english_phrase
:english_sentence
:english_label
:english_gloss

:malay_equ
:malay_phrase
:malay_sentence
:malay_label
:malay_gloss

:thai_equ
```

```
    :thai_equ_pron
    :thai_phrase
    :thai_phrase_pron
    :thai_sentence
    :thai_sentence_pron
    :thai_label
    :thai_gloss

    :comm_KIM
    :comm_GF
    :comm_PR
    ))
```

Annexe C
Fichier de paramètres des macros du modèle FeT

Il se compose de deux parties. Après chaque style précédé d'une étoile (*), on définit le nombre de styles suivants autorisés puis leur nom à la suite. Après chaque style précédé d'un dièse (#), on trouve le nombre des vaeurs possibles que peut prendre ce champ puis leur liste.

```
*entry
2
french_pron
french_plur_form
*french_pron
2
french_cat
french_fem_form
*french_plur_form
1
french_pron
*french_cat
2
english_equ
french_phrase
*french_fem_form
1
french_pron
*english_equ
1
malay_equ
*malay_equ
2
malay_equ
thai_equ
*french_phrase
2
english_label
english_phrase
*english_label
1
english_phrase
*english_phrase
1
malay_phrase
*malay_phrase
1
thai_phrase
*thai_phrase
1
entry
*thai_equ
4
```

```
entry
french_cat
french_phrase
french_sentence
*french_sentence
1
english_sentence
*english_sentence
1
malay_sentence
*malay_sentence
1
thai_sentence
*thai_sentence
1
entry
#french_cat
13
n.m.
n.m. inv.
n.m. pl.
n.m./f.
n.m., f.
n.f.
n.f. pl.
v.tr.
v.intr.
v.pr.
a.
```
adv.
prép.

Annexe D
Fichier de conversion de versions

Ce fichier de conversion permet de convertir des objets LISP résultat de l'indexation du dictionnaire FeM en objets LISP qui seront envoyés pour complétion pour construire le dictionnaire FeT.

Le champ :# est remplacé par le nom du champ lors de la conversion,

Le champ := est remplcaé par le champ d'origine,

Le mot nil n'est pas remplacé.

```
(:version                              nil)
(:author                               nil)
(:date                                 nil)
(:entry                                :entree)
(:french_entry_number                  :numero_entree)
(:french_cat                           :cat_francais)
(:french_label                         :etiq_francais)
(:french_gloss                         :glose_francais)
(:french_phrase                        :groupe_francais)
(:french_sentence                      :phrase_francais)
(:french_entry_variant                 :variante_francais)
(:french_subentry                      :sousentree_francais)
(:french_subentry_pron                 :pron_sousentree_francais)
(:french_fem_form                      :forme_fem_francais)
(:french_fem_plur_form                 :forme_fem_plur_francais)
(:french_fem_plur_pron                 :pron_fem_plur_francais)
(:french_fem_pron                      :pron_fem_francais)
(:french_masc_plur_form                :forme_masc_plur_francais)
(:french_masc_plur_pron                :pron_masc_plur_francais)
(:french_masc_plur_pron_variant        :var_pron_masc_plur_francais)
(:french_plur_form                     :plur_forme_francais)
(:french_plur_form_pron                :pron_forme_plur_francais)
(:french_plur_form_variant             :var_plur_forme_francais)
(:french_plur_form_variant_pron        :var_pron_plur_forme_francais)
(:french_pron                          :pron_francais)
(:french_pron_variant                  :variante_pron_francais)
(:french_rection                       :rection_francais)
(:cross_ref_entry                      :entree_ref_croisee)
(:cross_ref_marker                     :marqueur_ref_croisee)
(:commentaire                          nil
(:english_equ                          :equ_anglais)
(:english_phrase                       :groupe_anglais)
(:english_sentence                     :phrase_anglais)
(:english_label                        :etiquette_anglais)
(:english_gloss                        :glose_anglais)
(:malay_equ                            :equ_malais)
(:malay_phrase                         :groupe_malais)
(:malay_sentence                       :phrase_malais)
```

```
(:malay_label                          :etiq_malais)
(:malay_gloss                          :glose_malais)
(#                                     :equ_thai)
(#                                     :pron_thai_equ)
(#                                     :groupe_thai)
(#                                     :pron_groupe_thai)
(#                                     :phrase_thai)
(#                                     :pron_phrase_thai)
(#                                     :etiq_thai)
(#                                     :glose_thai)
(:comm_KIM                             nil)
(:comm_GF                              nil)
(:comm_PR                              nil)
```