Un devin de microstructures pour importer ou normaliser des ressources lexicales

Mathieu Mangeot, Valérie Bellynck

Un devin de microstructures pour importer ou normaliser des ressources lexicales

Mathieu Mangeot, Valérie Bellynck

Laboratoire LIG, équipe GETALP
Bâtiment IMAG CS 40700
38058 GRENOBLE CEDEX 9, FRANCE
{mathieu.mangeot,valerie.bellynck}@imag.fr

Résumé

Dans cet article, nous présentation un outil pour annoter une ressource non structurée ou semi-structurée, de façon à automatiser l'import de ses données dans un environnement générique facilitant et uniformisant son exploitation. Cet outil s'appuie sur le calcul d'une signature, caractérisant la ressource par des informations comptables sur ses éléments, et sur des heuristiques exploitant les statistiques de la signature pour repérer les entrées et la microstructure de chaque entrée dans le cas d'une ressource lexicale à importer dans la plateforme de bases lexicales Jibiki. Le résultat produit est géré dans iPolex, un entrepôt dédié aux traitements de fichiers de ressources lexicales permettant d'y ajouter des méta-données. La plateforme Jibiki accède à iPoLex pour importer directement des ressources lexicales bien formées et suffisamment renseignées. Le processus d'import ainsi constitué est utilisé pour créer et peupler des bases lexicales dans Jibiki. Les ressources ainsi importées sont ensuite consultables et éditables en ligne.

A micro-structure guesser to import or normalize lexical resources

In this article, we present a tool to annotate an unstructured or semi-structured resource, in order to automate the import of its data in a generic environment that facilitates and standardizes its use. This tool is based on the computation of a signature, characterizing the resource with accounting information about its elements, and on heuristics using signature statistics to identify the entries and microstructure of each entry in the case of a lexical resource to be imported into the Jibiki lexical database platform. The product result is managed in iPolex, a lexical data warehouse dedicated to processing lexical resource files to add metadata. The Jibiki platform accesses iPoLex to directly import lexical resources. The import process thus constituted is used to create and populate lexical bases in Jibiki. The imported resources are then available for consultation and editing online.

Mots-clés : iPolex, signature lexicale, jibiki, ressource lexicale

Keywords : iPolex, lexical signature, jibiki, lexical resource

1. Introduction

À l'heure actuelle, la plupart des ressources lexicales que nous manipulons sont au format XML, ce qui est une grande avancée par rapport aux débuts de l'électronisation des ressources où chacune avait son propre format. Par contre, force est de constater que, contrairement aux corpus, les efforts de standardisation ont été vains (Enguehard & Mangeot, 2013). Il en résulte que chaque ressource définit sa propre microstructure et son propre jeu de balises pour la décrire. Cette situation freine l'exploitation de ressources existantes aussi bien pour la consultation et l'édition que pour leur réutilisation dans de nouvelles ressources. Nos travaux se situent précisément dans ce domaine de la manipulation de ressources lexicales hétérogènes.

Même si les structures sont différentes, la plupart des types d'information se retrouvent d'une ressource à l'autre : le mot-vedette, la prononciation, la catégorie grammaticale, les exemples, les traductions, etc. La plupart du temps, il est possible de les repérer manuellement même si l'on ne maîtrise pas la langue représentée. Dans cet article, nous montrons qu'il est possible de repérer ces types d'information automatiquement puis de proposer ensuite à l'utilisateur de confirmer les hypothèses calculées, et ainsi d'améliorer grandement la manipulation de ressources lexicales.

Pour le prouver, nous proposons un outil qui analyse la structure d'une ressource lexicale et en construit une signature lexicale représentant les informations calculées sur cette structure. La signature lexicale est ensuite utilisée pour calculer des heuristiques permettant de formuler des hypothèses sur les différents types d'information présents dans la ressource analysée. Il est possible également d'utiliser cette signature lexicale pour détecter un certain nombre d'erreurs potentielles dans la structure analysée. Les hypothèses précédemment calculées servent ensuite à proposer à l'utilisateur un ensemble de pointeurs permettant d'identifier les informations repérées dans la structure. Celui-ci les validera ou non en fonction de son analyse manuelle. Les pointeurs sont ensuite utilisés pour importer automatiquement la ressource analysée dans la plateforme Jibiki permettant de la consulter et de l'éditer en ligne.

2. Signature dictionnairique

La signature d'une ressource lexicale au format XML est une présentation condensée de sa structure XML et de son contenu. Elle permet de visualiser le document et de repérer en un coup d'œil d'éventuelles erreurs. Ce concept, inventé pour nos besoins propres, peut être utilisé pour tout type de document XML. Le but est de caractériser toute ressource lexicale d'un point de vue structurel. Il s'agit donc d'une description synthétique présentant un ensemble d'indicateurs quantitatifs pour chaque élément de texte repérable par le même mécanisme. Pour chaque mécanisme, les valeurs calculées sont principalement le nombre de tels éléments et le nombre moyen de caractères des éléments de textes ainsi repérés.

Par exemple, dans le cas de documents XML où les éléments de texte sont encadrées par des balises, nous produisons une présentation indentée : chaque ligne présente un élément XML et est décalée autant qu'elle est imbriquée.

La signature est créée par un analyseur XML lors de l'analyse d'un document. Du fait de la grande taille des dictionnaires, nous avons opté pour un parseur événementiel SAX[1] (Simple Api for XML) plutôt que DOM[2] (Document Object Model).

Pour chaque élément XML rencontré, un certain nombre d'informations est stocké en mémoire : son nom, sa position dans l'arbre XML et le nombre de fois qu'il apparaît dans le document à cette position.

Chaque attribut est aussi analysé : on stocke son nom et pour sa valeur, on compte le nombre de valeurs différentes jusqu'à un maximum de 1000[3], le nombre de valeurs classées par ordre alphabétique selon le tri Unicode et le nombre total de valeurs.

Pour le texte contenu dans l'élément, on stocke également le nombre moyen de caractères et de mots (chaînes de caractères séparées par des espaces) ainsi que le nombre de valeurs différentes jusqu'à un maximum de 1000, le nombre de valeurs classées par ordre alphabétique selon le tri Unicode et le nombre total de valeurs.

Les informations ainsi collectées lors de l'analyse du document XML sont ensuite affichées de manière condensée sous forme d'arbre XML où chaque élément n'apparaît qu'une fois et seule la balise ouvrante est affichée, ce qui permet un gain de place. Les éléments fils sont indentés par rapport à leur élément père. Si le nombre de valeurs différentes pour un attribut ou pour du texte est inférieur à 50, toutes les valeurs sont affichées à la suite (voir Figure).

Figure 1 : signature du dictionnaire DiLAF zarma-français

1 https://fr.wikipedia.org/wiki/Simple_API_for_XML

2 https://fr.wikipedia.org/wiki/Document_Object_Model

3 pour éviter de saturer la mémoire et de ralentir les calculs

L'exemple de la figure se lit de la manière suivante :

- l'élément *dilaf* est l'élément racine de la ressource. Il n'apparaît qu'une seule fois dans toute la ressource à cette place dans l'arbre XML (logique pour un élément racine…). Il contient deux attributs. L'attribut *src* n'apparaît qu'une fois (là aussi c'est logique puisque l'élément n'apparaît qu'une fois !) et sa valeur est *dje*. L'attribut *trg* n'apparaît aussi qu'une seule fois (*idem*) et sa valeur est *fra*.

- l'élément *article* est le premier élément fils de l'élément *dilaf* rencontré dans l'arbre XML car il est indenté de deux espaces vers la droite. Il apparaît 6 914 fois dans toute la ressource à cette place dans l'arbre XML. Il n'a pas d'attribut. Son contenu textuel est d'en moyenne 1 caractère. Il contient 2 valeurs différentes qui sont classées par ordre alphabétique. Il contient en tout 3 valeurs. Les valeurs différentes sont une virgule « , » qui apparaît une fois et un point « . » qui apparaît deux fois.

- l'élément *saniize* est le premier élément fils de l'élément *article* rencontré dans l'arbre XML. Il apparaît également 6 914 fois dans toute la ressource à cette place dans l'arbre XML. Il a un attribut *lamba* qui a 10 valeurs différentes et dont 2080 valeurs sont classées par ordre alphabétique sur un total de 3 338. La liste des valeurs différentes est ensuite affichée avec le nombre de valeurs pour chaque : « 1 » apparaît 383 fois, « 10 » apparaît une fois, « 2 » apparaît 337 fois, etc. Son contenu textuel est d'en moyenne 6,2 caractères et 1 mot. Il a 704 valeurs différentes et 6 865 valeurs sont classées par ordre alphabétique sur 6 914. À noter ici que le nombre total de valeurs est supérieur à 1000. Le nombre de valeurs différentes n'est donc pas significatif.

- l'élément *ciiyaŋ* est le deuxième fils de l'élément *article* rencontré dans l'arbre XML. Il apparaît également 6 914 fois dans toute la ressource à cette place dans l'arbre XML. Il n'a pas d'attribut. Son contenu textuel est d'en moyenne 6,8 caractères et 1 mot. Il a 759 valeurs différentes et 6 507 valeurs sont classées par ordre alphabétique sur 6 914. Note : la figure affiche 6 917 au lieu de 6 914 mais cette erreur a été corrigée depuis.

- l'élément *kanandi* est le troisième fils de l'élément *article* rencontré dans l'arbre XML. Il apparaît également 6 914 fois dans toute la ressource à cette place dans l'arbre XML. Il n'a pas d'attribut. Son contenu textuel est d'en moyenne 3,9 caractères et 1 mot. Il a 25 valeurs différentes et 5 070 valeurs sont classées par ordre alphabétique sur 6 884. La liste des valeurs différentes est ensuite affichée avec le nombre de valeurs pour chaque : « alteeb. » apparaît 6 fois, « bsif. » apparaît 1 fois, « ceey » apparaît 5 fois, etc. À noter que cette liste de valeurs est coupée par la copie d'écran.

Les autres éléments se décrivent selon le même principe.

3. Calcul des heuristiques

Dans cette partie, nous montrons comment est utilisée la signature dans le cas des bases lexicales, et donc la signature dictionnairique pour deviner la structure de ces bases.

Pour identifier chaque type d'information (article, mot-vedette, prononciation, catégorie grammaticale, exemples, etc.), nous avons élaboré une fonction de calcul d'heuristiques spécifique qui utilise la signature dictionnairique. Les heuristiques sont fondées sur un travail d'analyse de plus d'une centaine de ressources lexicales mené depuis les débuts de nos recherches dans le domaine (Corréard & Mangeot, 1999). À chauqe heuristique est associé un score. Les valeurs de score sont fixées au départ arbitrairement en fonction de l'importance supposée de l'heuristique. Les valeurs sont éventuellement modifiées en fonction des résultats obtenus sur quelques analyses afin d'obtenir un meilleur résultat. Le résultat de la fonction est un pointeur XPath[4] qui identifie précisément le type d'information dans la structure analysée. Nous avons appelé l'ensemble de ces pointeurs « CDM » pour Common Dictionary Markup (Mangeot, 2002). Nous détaillons ci-dessous les principales fonctions que nous avons élaborées.

3.1. Le volume

Notre définition du volume est l'élément XML parent de celui de l'article. Dans la plupart des cas, c'est également l'élément racine du document mais pas toujours. Nous avons relevé 2 % des ressources où la racine du document n'est pas l'élément père des articles. Il s'agit des ressources respectant la partie informative du standard Lexical Markup Framework[5] où l'on trouvera le résultat suivant :

```
/LexicalResource/Lexicon
```

La fonction de calcul du volume reprend le résultat de celle de calcul de l'article et supprime un niveau dans le pointeur XML. Pour la signature dictionnairique de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf
```

3.2. L'article

Pour chaque élément dans la signature, la fonction calcule un score. L'élément dont le score est le plus élevé est sélectionné pour le résultat.

Le calcul des scores se fait récursivement en partant des éléments fils du volume. S'il y a plusieurs éléments frères, l'élément dont la fréquence est la plus élevée est sélectionné.

---

4 https://www.w3.org/TR/xpath/all/

5 https://fr.wikipedia.org/wiki/Lexical_Markup_Framework

Le score de départ est de 0,1. Il est de 1 si le nom de l'élément contient « entry ».

Ensuite, le score final est calculé en divisant le score par le niveau de l'élément dans l'arbre XML. Plus l'élément est profond dans l'arbre, plus son score sera faible.

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article
```

3.3. L'identifiant de l'article

Cette fonction calcule un score uniquement pour les attributs de l'élément article.

Le score de départ est de 0. Il est augmenté de 0,3 si le nom de l'attribut contient « id ».

Le score est ensuite augmenté de 0,5 si le nombre de valeurs différentes est égal au nombre de valeurs totales, ce qui démontre une valeur unique pour chaque article.

Pour la signature lexicale de la figure , la fonction ne renvoie rien car l'article n'a pas d'identifiant.

3.4. Le mot-vedette

Les éléments sélectionnés doivent contenir du texte.

Le score de départ de cette fonction est de 0.

Il est augmenté de 0,4 s'il y a autant de mots-vedette que d'articles et de 0,3 si leur nombre est proche à 1 % près.

Il est augmenté de 0,4 si le nom de l'élément contient les mots « headword » ou « vedette ».

Il est augmenté de 0,3 s'il y a en moyenne peu de mots (de 1 à 3).

Il est augmenté de 0,3 s'il y a beaucoup de valeurs différentes (proche du nombre total de valeurs).

Il est augmenté de 0,1 * le rang dans sa fratrie. Le mot-vedette est souvent le premier de sa fratrie.

Ensuite, le score final est calculé en divisant le score par le niveau de l'élément dans l'arbre XML. Plus l'élément est profond dans l'arbre, plus son score sera faible.

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/sanniize/text()
```

Il est à noter que cette fonction et la plupart des suivantes sont optimisées pour une valeur placée dans le contenu textuel d'un élément et non dans un attribut. Il est envisageable de prendre en compte ce dernier cas, mais sa fréquence est assez faible et concerne essentiellement les ressources au format LMF (voir la partie 3.1).

3.5. Le numéro d'homographe

Cette fonction calcule un score uniquement pour les attributs de l'élément du mot-vedette.

Le score de départ est de 0. Il est augmenté de 0,1 si le nom de l'attribut contient « hn » pour « homograph number »

Le score est ensuite augmenté de 5 divisé par le nombre de valeurs différentes. Il y a peu de valeurs différentes.

Le score est ensuite augmenté de 0,3 si la valeur de l'attribut est un entier. Les numéros d'homographes sont la plupart du temps des entiers.

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/sanniize/lamba
```

3.6. La prononciation

Les éléments sélectionnés doivent contenir du texte.

Le score de départ de cette fonction est de 0.

Il est augmenté de 0,2 si le nombre de valeurs est proche de celui du mot-vedette.

Il est augmenté de 0,7 si le nom de l'élément contient le mot «pron».

Il est augmenté de 0,2 s'il y a en moyenne peu de mots (de 1 à 2).

Il est augmenté de 0,3 s'il y a beaucoup de valeurs différentes (proche du nombre total de valeurs).

Ensuite, le score final est calculé en divisant le score par le niveau de l'élément dans l'arbre XML.

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/ciiyaŋ/text()
```

3.7. La catégorie grammaticale

Les éléments sélectionnés doivent contenir du texte et avoir un nombre de valeurs proche ou supérieur à celui du mot-vedette.

Le score de départ de cette fonction est de 0.

Il est augmenté de 0,45 si le nom de l'élément contient les mots «pos», « cat » ou « gram ».

Il est augmenté de 0,2 s'il y a en moyenne peu de mots (de 1 à 3).

Il est augmenté de 0,3 s'il y a peu de valeurs différentes (une liste fermée de valeurs).

Ensuite, le score final est calculé en divisant le score par le niveau de l'élément dans l'arbre XML.

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/kanandi/text()
```

3.8. Le sens de mot

Les éléments sélectionnés ne doivent pas contenir de texte.

Le score de départ de cette fonction est de 0.

Il est augmenté de 0,3 s'il y a au moins 1,5 fois plus de sens que d'articles.

Il est augmenté de 0,75 si le nom de l'élément contient le mot «sens».

Il est augmenté de 0,3 si l'élément contient un attribut avec peu de valeurs différentes, ce qui pourrait correspondre à un numéro de sens.

Ensuite, le score final est calculé en divisant le score par le niveau de l'élément dans l'arbre XML.

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/maana/silmaŋ
```

Le bloc de sens est ensuite calculé comme le volume à partir de l'article en supprimant le dernier élément du pointeur Xpath. Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/maana
```

Il est à noter que dans ce cas précis, le calcul du sens de mot est incorrect mais celle du bloc de sens correcte. L'élément représentant le sens est le même que le bloc de sens.

3.9. La définition

Les éléments sélectionnés doivent contenir du texte et le nombre moyen de mots doit être supérieur à 2.

Le score de départ de cette fonction est de 0.

Il est augmenté de 0,3 si le nombre de valeurs est proche de celui du mot-vedette ou supérieur.

Il est augmenté de 0,001 * le nombre moyen de caractères (s'il y a beaucoup de caractères).

Il est augmenté de 0,5 si le nom de l'élément contient les mot «definition» ou « définition » et de 0,3 s'il contient les mots « def » ou « déf ».

Il est augmenté de 0,3 s'il y a beaucoup de valeurs différentes (proche du nombre total de valeurs).

Il est augmenté de 0,05 / le rang dans la fratrie (la définition vient tôt dans la fratrie et souvent avant l'exemple).

Il est augmenté de 0,1 / le niveau de l'élément dans l'arbre XML (il vient tôt dans l'arbre XML)

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/maana/bareyaŋ
```

3.10. L'exemple

Les éléments sélectionnés doivent contenir du texte et le nombre moyen de mots doit être supérieur à 2.

Le score de départ de cette fonction est de 0.

Il est augmenté de 0,3 si le nombre de valeurs est proche de celui du mot-vedette ou supérieur.

Il est augmenté de 0,01 * le nombre moyen de mots (s'il y a beaucoup de mots).

Il est augmenté de 0,001 * le nombre moyen de caractères (s'il y a beaucoup de caractères).

Il est augmenté de 0,5 si le nom de l'élément contient les mot «example» ou «exemple» et de 0,2 s'il contient le mot « ex ».

Il est augmenté de 0,3 s'il y a beaucoup de valeurs différentes (proche du nombre total de valeurs).

Il est augmenté de 0,01 * le rang dans la fratrie (l'exemple vient tard dans la fratrie, après la définition).

Il est augmenté de 0,01 * le niveau de l'élément dans l'arbre XML (il vient tard dans l'arbre XML)

Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

```
/dilaf/article/maana/silmaŋ/version
```

Le bloc d'exemples est ensuite calculé comme le volume à partir de l'article en supprimant le dernier élément du pointeur Xpath. Pour la signature lexicale de la figure , la fonction renvoie le pointeur Xpath suivant :

`/dilaf/article/maana/silmaŋ`

4. Détection d'erreurs

La signature lexicale permet également de détecter un certain nombre d'erreurs potentielles. Nous en détaillons 4 types principaux ci-dessous illustrés à l'aide de l'exemple de la figure .

4.1. Erreurs de structuration

Il arrive que des éléments ou des blocs d'éléments soient mal placés dans la structure d'un article. L'affichage des éléments de manière indentée avec la fréquence permet de repérer ces erreurs plus facilement.

Dans l'exemple de la figure , on trouve un élément *complémént* au même niveau que les éléments *article*. Outre le fait qu'il y a une erreur d'orthographe (il faut écrire *complément*), cet élément devrait se situer à l'intérieur de l'article avec les autres éléments *complément* qui apparaissent 113 fois.

4.2. Listes fermées de valeurs

Lorsqu'il y a un maximum de 50 valeurs différentes, les valeurs sont toutes affichées avec pour chacune sa fréquence. Cela permet de contrôler toutes les listes fermées de valeurs comme par exemple la catégorie grammaticale. On peut supposer qu'une valeur avec une fréquence très faible et qui est très proche d'une autre valeur est une erreur potentielle. Par exemple, une catégorie grammaticale comme le « nom masculin » sera la plupart du temps représentée par « n.m. » et quelquefois par « n. m. » (avec un espace en plus). Il sera possible ensuite de rectifier ces erreurs à l'aide d'un autre programme.

Dans l'exemple de la figure , on trouve une valeur *bsif.* qui apparaît une fois et une valeur *dsif.* qui apparait 4 fois. On peut supposer que *bsif. e*st une erreur et devrait être remplacé par *dsif.* Il s'agira de demander confirmation à un lexicographe spécialiste de la langue en question.

4.3. Compte incorrect

Certains éléments doivent apparaître au moins autant de fois que d'autres. Dans un dictionnaire, on trouve nécessairement au moins autant de mots-vedettes que d'articles. De même, s'il y a une catégorie grammaticale ou une prononciation, il y en a généralement une par article.

Dans l'exemple de la figure , on trouve le même nombre d'éléments *article, sanniize, ciiyaŋ, kanandi, bareyaŋ* (6 914). Par contre, pour l'élément feeriji, on voit qu'il en manque 6 914 - 6 253 = 661.

Pour l'élément *soŋay,* on voit qu'il n'y en a qu'un seul. On peut s'interroger sur son utilité.

Pour les éléments *kanandi* et *bareyaŋ,* on constate également qu'il y a un certain nombre d'éléments vides (sans texte). Pour *kanandi*, il y en a 6 914 – 6 884 = 30 et pour *bareyaŋ*, il y en a 6 914 – 6 864 = 50.

4.4. Texte mal placé

Certains éléments servent uniquement à structurer le document et ne contiennent donc pas de texte. Il arrive que du texte apparaisse malgré tout dans un élément qui ne devrait pas en contenir. La signature nous renseigne sur ce point.

Dans l'exemple de la figure , c'est visiblement le cas de l'élément *article* qui ne devrait pas contenir de texte. On constate qu'il contient une virgule « , » et deux points « . ». Ce texte est donc à supprimer.

5. Utilisation dans les plateformes iPoLex et Jibiki

5.1. Description du script d'analyse

Le script d'analyse produit la signature lexicale et calcule les différentes hypothèses pour chaque type d'information lexicale décrites dans la section 3. Il affiche également le tableau de chaque élément XML rencontré avec sa fréquence, trié par ordre alphabétique. Enfin, il calcule un squelette d'article vide avec tous les éléments rencontrés lors de l'analyse. Ces informations serviront lors de l'import de la ressource dans la plateforme Jibiki (voir section 6).

Programmé en perl, le script utilise pour l'analyse XML le module perl XML::Parser basé sur le parseur XML SAX expat de James Clark. Le script a une taille de 24 Ko et comporte 884 lignes de code.

Ce script a été testé sur une machine équipée d'un processeur Intel Xeon E5 4 cœurs 2,4 Ghz avec 32 Go de RAM avec le système d'exploitation Linux Debian 3.16 Jessie.

L'analyse prend 15 secondes pour le dictionnaire DiLAF bilingue zarma-français de taille 3,3 Mo contenant 6 914 articles. C'est le dictionnaire qui a généré la signature de la figure 1.

L'analyse prend 6 minutes et 35 secondes pour un dictionnaire bilingue anglais-français de taille 17 Mo contenant 39 809 articles.

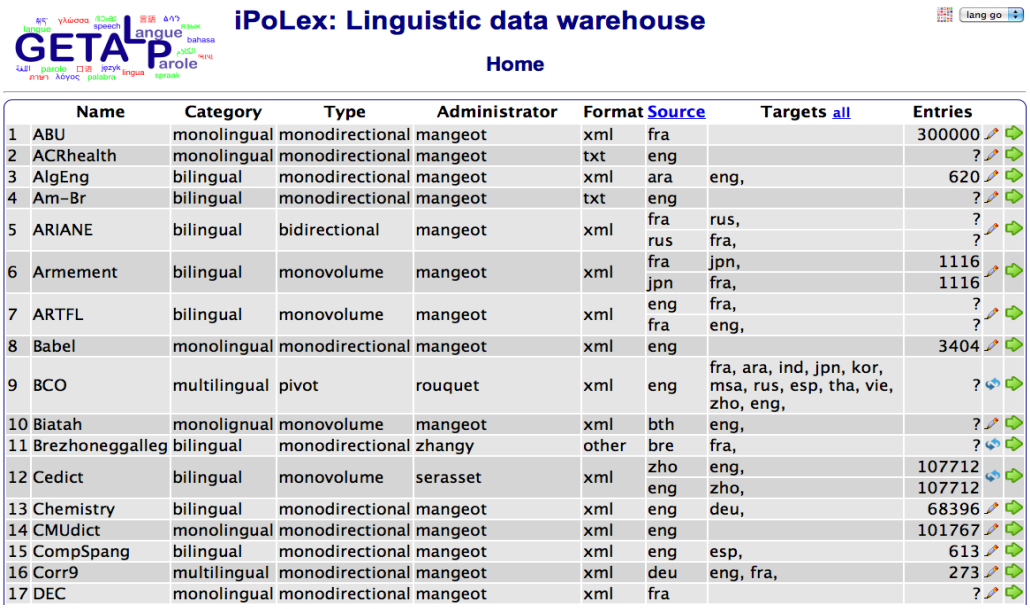
Ce script est intégré au code source de la plateforme iPolex et disponible sur le compte Github de Mathieu Mangeot[6].

5.2. Description de la plateforme iPolex

iPoLex est un « entrepôt lexical à services ». Il est apparu au cours des projets utilisant Jibiki comme une nécessité. Il s'agit en effet de fournir une infrastructure pérenne et visible pour les données lexicales récoltées dans le cadre d'un projet.

Pour clarifier cette idée, on peut utiliser une métaphore : il s'agit de réaliser le pendant, pour les données lexicales, de la fonctionnalité d'une forge qui assure le partage et la construction collaborative de logiciels. L'image de l'entrepôt serait peut-être plus adaptée si l'on réduisait la description de l'outil désiré à cette seule fonctionnalité.

Le service Web iPoLex a pour objectif de stocker toutes les ressources lexicales dans un endroit unique, accessible via le Web, avec accès restreint aux membres du laboratoire. L'accès à l'outil est réservé aux membres du laboratoire. L'authentification se fait automatiquement via l'annuaire LDAP du laboratoire. L'outil, programmé en PHP, se veut simple et rustique. Il n'utilise pas de base de données. Sa page d'accueil présente la liste des ressources disponibles (voir Figure 2). Il est possible de les trier selon leur nom, leur langue source ou leurs langues cibles.



	Name	Category	Type	Administrator	Format	Source	Targets all	Entries
1	ABU	monolingual	monodirectional	mangeot	xml	fra		300000
2	ACRhealth	monolingual	monodirectional	mangeot	txt	eng		?
3	AlgEng	bilingual	monodirectional	mangeot	xml	ara	eng,	620
4	Am-Br	bilingual	monodirectional	mangeot	txt	eng		?
5	ARIANE	bilingual	bidirectional	mangeot	xml	fra	rus,	?
						rus	fra,	?
6	Armement	bilingual	monovolume	mangeot	xml	fra	jpn,	1116
						jpn	fra,	1116
7	ARTFL	bilingual	monovolume	mangeot	xml	eng	fra,	?
						fra	eng,	?
8	Babel	monolingual	monodirectional	mangeot	xml	eng		3404
9	BCO	multilingual	pivot	rouquet	xml	eng	fra, ara, ind, jpn, kor, msa, rus, esp, tha, vie, zho, eng,	?
10	Biatah	monolignual	monovolume	mangeot	xml	bth	eng,	?
11	Brezhoneggalleg	bilingual	monodirectional	zhangy	other	bre	fra,	?
12	Cedict	bilingual	monovolume	serasset	xml	zho	eng,	107712
						eng	zho,	107712
13	Chemistry	bilingual	monodirectional	mangeot	xml	eng	deu,	68396
14	CMUdict	monolingual	monodirectional	mangeot	xml	eng		101767
15	CompSpang	bilingual	monodirectional	mangeot	xml	eng	esp,	613
16	Corr9	multilingual	monodirectional	mangeot	xml	deu	eng, fra,	273
17	DEC	monolingual	monodirectional	mangeot	xml	fra		?

Figure 2 : Page d'accueil du site iPolex : liste des ressources disponibles

Pour chaque ressource, on affiche :

- son nom,
- sa catégorie (monolingue, bilingue, multilingue),
- son type (monovolume, monodirectionnel, bidirectionnel, pivot, etc.),
- le login de l'administrateur de la ressource,
- son format,
- sa langue source,
- ses langues cibles
- et le nombre d'entrées.

On peut ensuite accéder aux métadonnées de la ressource ainsi qu'aux données brutes en cliquant sur les boutons correspondants : crayon ou flèches bleues en cercle pour les métadonnées et flèche verte pour les données brutes. Seuls les administrateurs peuvent modifier les métadonnées d'une ressource existante. Si l'on est administrateur d'une ressource, le bouton affiché pour accéder aux métadonnées sera alors le crayon et sinon les flèches bleues en cercle.

Chaque ressource est décrite par au moins deux fichiers de métadonnées. Le premier fichier décrit la ressource et sa macrostructure. On retrouve les informations affichées sur la page d'accueil de iPolex. La figure 3 montre les métadonnées du dictionnaire DiLAF.

6 https://github.com/mangeot/ipolex/blob/master/pl/dictionary_analysis.pl

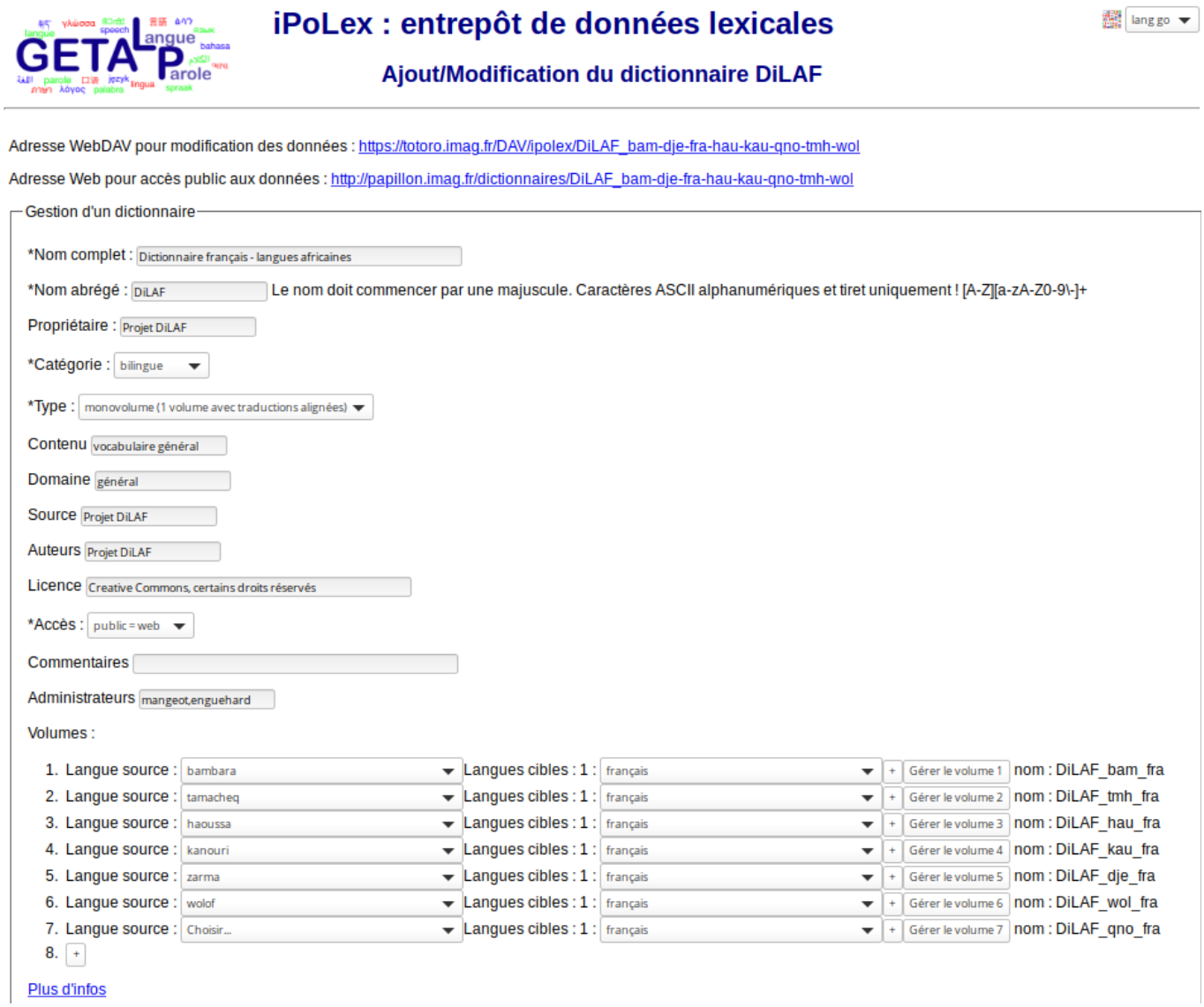


Figure 3 : Description du dictionnaire DiLAF dans la plateforme iPolex

Ensuite, chaque volume qui compose cette ressource est à son tour décrit dans un fichier de métadonnées du volume. Par exemple, un dictionnaire bilingue bidirectionnel sera composé de deux volumes : un volume langue 1→ langue 2 et un volume langue 2 → langue 1 et chaque volume sera décrit par un fichier de métadonnées. Les informations décrites dans ces fichiers sont : le nombre d'entrées, le format du fichier (principalement XML), l'encodage du fichier (principalement UTF-8), les pointeurs CDM XPath décrivant la microstructure et un article squelette vide.

La figure 4 montre les métadonnées du volume DiLAF zarma→français.

Figure 4 : Description du volume DiLAF zarma → français dans la plateforme iPolex

Ces fichiers de métadonnées peuvent être ensuite utilisés directement pour importer la ressource sur la plate-forme Jibiki.

Le code source de la plateforme iPolex est disponible sur le compte Github de Mathieu Mangeot[7]. La plateforme est également disponible prête à l'emploi sous forme d'image Docker sur le compte docker de Mathieu Mangeot[8].

7 https://github.com/mangeot/ipolex/

5.3. Intégration du script d'analyse dans la plateforme iPolex

L'outil iPolex permet également d'importer une nouvelle ressource. C'est lors de ce processus que le script d'analyse d'une ressource lexicale sera utilisé.

Lors de l'import d'une ressource, un répertoire est créé sur le serveur. Ensuite, lors de la saisie des métadonnées, des fichiers correspondants sont générés au format XML et stockés dans le répertoire de la ressource.

La première étape consiste à saisir les métadonnées de la ressource à importer. Ces données sont saisies grâce au formulaire HTML de la figure 3. Lors de l'enregistrement du formulaire, un fichier de métadonnées du dictionnaire au format XML est généré et stocké sur le serveur.

Ensuite pour chaque volume constituant la ressource, il faut d'abord téléverser le fichier de données sur le serveur. Ensuite, le script d'analyse des ressources lexicales décrit précédemment sera utilisé pour calculer le nombre d'articles, les pointeurs CDM décrivant les informations lexicales présentes dans la microstructure ainsi qu'un article squelette vide.

Ces informations sont ensuite affichées dans un formulaire HTML (voir figure 4) pour que l'utilisateur qui importe la ressource puisse rectifier les pointeurs CDM qui sont incorrects (dont les heuristiques de calcul n'ont pas fonctionné). Lors de l'enregistrement du formulaire, un fichier de métadonnées du volume au format XML est généré et stocké sur le serveur.

L'outil trang[9] est également utilisé pour générer un schéma XML décrivant la structure XML du volume. Ce schéma sera ensuite utilisé par la plateforme jibiki pour paramétrer automatiquement l'éditeur d'articles en ligne.

5.4. Intégration dans la plateforme Jibiki

Jibiki (Mangeot & Thévenin, 2004 ; Zhang et al., 2014) est une plate-forme générique en ligne pour manipuler des ressources lexicales avec gestion d'utilisateurs et groupes, consultation de ressources hétérogènes et édition générique d'articles de dictionnaires. C'est un site Web communautaire développé au départ pour le projet Papillon (Sérasset & Mangeot, 2001). La plate-forme est programmée entièrement en Java, basée sur l'environnement Enhydra[10]. Toutes les données sont stockées au format XML dans une base de données (Postgres).

Ce site Web propose principalement deux services : une interface unifiée permettant d'accéder simultanément à de nombreuses ressources hétérogènes (monolingues, dictionnaires bilingues. bases multilingues, etc.) et une interface d'édition spécifique pour contribuer directement aux dictionnaires disponibles sur la plate-forme.

L'import d'une ressource dans la plate-forme se fait automatiquement à partir des fichiers de méta-données générés par l'outil iPolex présenté précédemment. Il suffit d'indiquer à l'interface d'import l'URL des méta-données décrivant le dictionnaire et à partir de là, toutes les autres informations (méta-données des volumes, article squelette, schémas XML, etc.) ainsi que les données sont téléchargées automatiquement puis ajoutées dans la base de données.

L'éditeur est fondé sur un modèle d'interface HTML instancié avec l'article que l'on veut éditer. Le modèle peut être généré automatiquement depuis une description de la structure de l'entrée à l'aide d'un schéma XML. Il peut être modifié ensuite pour améliorer le rendu à l'écran. La seule information nécessaire à l'édition d'un article de dictionnaire est donc le schéma XML représentant la structure de cette entrée.

Plusieurs projets de construction de ressources lexicales ont utilisé ou utilisent toujours cette plate-forme avec succès. C'est le cas par exemple du projet GDEF de dictionnaire bilingue estonien-français (Mangeot & Chalvin, 2006) ou du dictionnaire bilingue japonais-français jibiki.fr[11] (Mangeot, 2016). Le code source de la plateforme Jibiki est disponible sur le compte Github de Mathieu Mangeot[12]. La plateforme est également disponible prête à l'emploi sous forme d'image Docker sur le compte docker de Mathieu Mangeot[13].

6. Conclusion et perspectives

Dans cet article, nous avons montré une série d'outils permettant de mettre en place un processus complètement automatisé d'import d'une ressource lexicale au format XML sur une plateforme de consultation et d'édition en ligne. Le script d'analyse qui génère la signature lexicale et calcule les heuristiques permettant de décrire la microstructure de la ressource à l'aide de pointeurs pour chaque type d'information lexicale est la partie centrale de l'automatisation de ce processus.

Le processus d'import a été testé avec succès sur de nombreuses ressources (environ une centaine à ce jour).

Concernant le script d'analyse d'une ressource lexicale, nous avons envisagé quelques perspectives afin d'améliorer son efficacité.

---

8 https://hub.docker.com/r/mangeot/ipolex

9 http://www.thaiopensource.com/download/

10 http://enhydra.ow2.org/about.html

11 http://jibiki.fr/

12 https://github.com/mangeot/jibiki

13 https://hub.docker.com/r/mangeot/jibiki

L’utilisation d’un devineur de langues permettrait d’automatiser complètement le processus d’import. Toutefois plusieurs difficultés se présentent : les dictionnaires bilingues ou multilingues comportent des segments de texte dans différentes langues, certaines informations textuelles ne sont pas spécifiques à une langue (prononciation, transcriptions, etc.), les segments de texte sont très courts et enfin, seules les langues les mieux dotées sont devinées par ces outils.

Nous souhaitons tester le calcul d’heuristiques avec un réseau de neurones pour comparer les résultats avec nos heuristiques.

Nous aimerions tester également l’utilisation de la signature sur des documents qui ne sont pas des ressources lexicales.

Références

Corréard, Marie-Hélène ; Mangeot, Mathieu (1999) XML- A Solution For LDBs, EDs and MRDs? Proc. of COMPLEX 1999, Pécs, Hungary, 16-19 June 1999, 6 p.

Enguehard, Chantal ; Mangeot, Mathieu (2013) *LMF for a selection of African Languages*. Chapter 7, book "LMF: Lexical Markup Framework, theory and practice", Ed. Gil Francopoulo, Hermès science, Paris, France, 17 p.

Mangeot, Mathieu (2001). *Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue.* Thèse de nouveau doctorat, Spécialité Informatique, Université Joseph Fourier Grenoble I, jeudi 27 septembre 2001, 280 p.

Mangeot, Mathieu (2002) *An XML Markup Language Framework for Lexical Databases Environments: the Dictionary Markup Language.* Proc. of International Standards of Terminology and Language Resources Management, LREC 2002 workshop, Las Palmas, Islas Canarias, Spain, 28 May 2002, pp. 37-44.

Mangeot, Mathieu (2016) *Collaborative Construction of a Good Quality, Broad Coverage and Copyright Free Japanese-French Dictionary.* International Journal of Lexicography, Oxford University Press (OUP), 2016, 31 (1), pp.78-112.

Mangeot, Mathieu ; Enguehard, Chantal (2013) *Des dictionnaires éditoriaux aux représentations XML standardisées. Chapitre 8, livre "Ressources Lexicales : contenu, construction, utilisation, évaluation",* Eds. Nuria Gala & Michael Zock, Linguisticae Investigationes Supplementa, John Benjamins Publishing, Amsterdam, Pays-Bas, 24 p.

Mangeot, Mathieu ; Thevenin, David (2004) *Online Generic Editing of Heterogeneous Dictionary Entries in Papillon Project.* Proc. of COLING 2004, ISSCO, Université de Genève, Switzerland, 23-27 August 2004, vol 2/2, pp 1029-1035.

Zhang, Ying ; Mangeot, Mathieu ; Bellynck, Valérie ; Boitet, Christian (2014) *Jibiki-LINKS: a tool between traditional dictionaries and lexical networks for modelling lexical resources.* Proceedings of the 4th Workshop on Cognitive Aspects of the Lexicon (CogALex) 2014 (Eds. Michael Zock, Reinhard Rapp, Chu-Ren Huang), Dublin, Ireland, 23 August 2014, 12 p.