

LEXIKA VEEIKV

Que faire d'un lexique et comment en tirer toute sa substance ?

Manuel de l'utilisateur

Version du 29 novembre 2017

Table des matières

1	Présentation générale	2
1.1	À qui s'adresse ce logiciel ?	2
1.2	Compatibilité	2
1.3	Mise en œuvre technique	2
1.4	Installation	2
1.4.1	De l'interpréteur	2
1.4.2	Du module	2
2	Principe de fonctionnement	3
2.1	Définitions	3
2.2	Représentations	3
2.3	Illustration	4
3	Mise en œuvre	7
3.1	Généralités	7
3.2	Algorithme général	7
3.3	Remarques	7
3.4	Fichiers d'intérêt	8
3.5	Configuration	8
3.6	Illustration	9
3.7	Remarques	9
3.8	Exemple	10
4	Tutoriel – sans interface graphique	13
4.1	Configuration du dictionnaire	13
4.2	Exécution du programme	13
5	Tutoriel – avec interface graphique	14
6	Aide-mémoire	15
6.1	Commandes	15
6.2	Fichier de configuration	15
7	Remarques pour les développeurs et pour la réunion	16
8	Améliorations et bogues	18
9	Annexes	19
9.1	Dépôt du logiciel	19

Chapitre 1

Présentation générale

Lexika, du grec *λεξικά*, signifiant « dictionnaires, lexiques », est un logiciel qui permet de générer des dictionnaires multilingues à partir de fichiers sources plus ou moins structurés comme ceux au format MDF (de Toolbox). Les dictionnaires ainsi créés passent par un format pivot XML, dont la structure est paramétrable, pour notamment produire un fichier PDF directement publiable, ou un site internet à l'aide d'un fichier XSL.

1.1 À qui s'adresse ce logiciel ?

Ce logiciel a été développé principalement pour des linguistes, notamment les légendaires linguistes de terrain qui ne sont pas toujours très à l'aise avec l'outil informatique. Les ingénieurs qui travaillent avec eux seront aussi d'une précieuse aide lorsqu'il faudra correctement configurer un dictionnaire non totalement compatible avec des formats imaginés par d'autres personnes.

1.2 Compatibilité

Systèmes d'exploitation compatibles : Linux et en théorie Windows et Mac (tests à faire).

Configuration minimale : aucune, si l'ordinateur semble dater de la dernière décennie...

1.3 Mise en œuvre technique

Ce logiciel a été développé en Python 3 et son interface graphique en Qt 5.

Il utilise les bibliothèques Python suivantes pour fonctionner :

- cchardet ;
- lxml ;
- regex.

1.4 Installation

1.4.1 De l'interpréteur

Linux

Veuillez vérifier que l'interpréteur Python 3 est installé (notamment grâce au gestionnaire de paquets de votre distribution).

Windows

Veuillez installer l'[interpréteur Python 3](#) (attention à l'architecture : 64 bits pour les ordinateurs actuels).

Mac

Veuillez vérifier que l'interpréteur Python 3 est installé (notamment grâce au gestionnaire de paquets [Homebrew](#)).

1.4.2 Du module

Vous pouvez à présent exécuter la commande suivante dans le terminal :

- la racine (où se trouve le fichier setup.py), par utilisation de l'archive locale téléchargée :

```
pip3 install .
```

- n'importe où, par utilisation du dépôt Pypi automatiquement (connexion internet requise) :

```
pip3 install lexika
```

Chapitre 2

Principe de fonctionnement

2.1 Définitions

À partir de la lecture des données du fichier source qui comporte les données linguistiques (éventuellement aussi des métadonnées) associées à des balises, le logiciel va tenter à partir desdites balises de réorganiser hiérarchiquement les données linguistiques selon la configuration choisie. En effet, en abstrayant le dictionnaire final que vous pouvez lire, au niveau structurel, nous pouvons voir que chaque élément dudit dictionnaire est hiérarchiquement l'enfant ou le parent d'un autre (au sens inclusif : le parent englobe l'enfant qui y est inclus).

Ces éléments seront par la suite nommés entités linguistiques. Chaque entité linguistique contient obligatoirement au moins une caractéristique qui représente la donnée linguistique elle-même et éventuellement d'autres qui représentent des paramètres informatifs sur l'entité, non toujours directement visibles. Le choix de considérer un élément comme entité ou comme caractéristique d'entité n'est pas absolu et dépend notamment de la complexité des éléments (une entité peut posséder des caractéristiques tandis qu'une caractéristique n'est qu'une information simple).

Par convention, les **balises** seront écrites en rouge, les **entités linguistiques** en bleu, les **caractéristiques** en vert et les **abstractions** (voir plus loin) en violet. Les diverses *valeurs* des caractéristiques seront en italique.

Par exemple, pour reprendre des entités classiques et courantes, nous pouvons dire qu'une **entrée** de dictionnaire est un enfant du **dictionnaire** lui-même (donc que les **entrées** sont des sœurs), qu'une **définition** est un enfant d'une **entrée**, qu'un **exemple** est un enfant d'une **définition**. Pour donner des exemples plus labiles et plus difficiles, nous pouvons aussi dire qu'une **sous-entrée** est un enfant d'une **entrée** et le parent d'un **sens**, ou bien que le **sens** est l'enfant d'une **entrée** et le parent d'une **sous-entrée**, selon ce qui semble le plus approprié à la langue étudiée et au linguiste.

En ce qui concerne les caractéristiques, c'est plus inconstant : si nous nous plaçons selon la représentation β (voir partie suivante) et que nous prenons par exemple la **définition** d'une **entrée**, la **définition** stricto sensu est une caractéristique, la **langue** dans laquelle elle est écrite en est une autre. Un autre exemple : une **note** accompagnant une **définition** peut être discursive, grammaticale, etc., ainsi l'entité **note** peut avoir comme caractéristique son **type** (ainsi que sa **langue**).

Il pourrait en théorie être possible de combiner caractéristiques et entités pour former des entités très précises comme une **définition française** ou une **note grammaticale française**, mais cela crée une forte redondance en brisant les points communs de toutes les **définitions** d'une part et de toutes les **langues** et **types** d'autre part.

Il est donc important de bien concevoir ce qui relève de l'entité ou de la caractéristique. C'est avant tout une question de choix selon l'objectif voulu et les parties suivantes illustreront bien les différences entre ces deux notions et les différentes représentations.

Le nom de l'entité et de la caractéristique est cependant primordial et est l'une des bases de l'algorithme, qui consiste en la recherche paramétrée de l'entité parente de chaque entité linguistique analysée : c'est le moteur clef de voûte du logiciel. Il tente de gérer la plupart des cas difficiles induits généralement par des choix structurels anciens ou par des choix de conception réalisés par des linguistes qui ont en tête un dictionnaire fini et lisible par des humains (ce qui peut créer des incohérences vis-à-vis de la structure informatisée).

2.2 Représentations

Il peut être possible de représenter les mêmes informations selon différentes représentations :

- représentation α : les informations sont des entités et les méta-informations sont des caractéristiques ;
- représentation β : les informations conteneurs sont des entités et les informations non-conteneurs sont des caractéristiques ;
- représentation γ : les informations conteneurs sont des entités et les informations non-conteneurs sont des entités nommées *caractéristiques* qui contiennent de manière développée les informations sous forme de caractéristiques.

Repr.	Entité (élément XML)	Caractéristique (attribut XML)	Avantages	Inconvénients
α	Informations importantes	Informations auxiliaires ou méta-informations	Lisible, concis, ouvert (notamment au texte enrichi)	–
β	Informations conte-neurs	Informations non conteneurs	Concis	Moyennement lisible
γ	Toutes les informa-tions	Nom et valeur des caractéristiques développées	norme LMF	Verbeux, redondant

2.3 Illustration

Pour illustrer (figure 2.1), prenons une entrée simplifiée d'un mot français (à gauche) et une représentation structurée β pertinente (à droite). Portons notre attention sur le style décrit précédemment, et notons que les parties entre parenthèses sont facultatives (selon le dictionnaire en cours de création).

	• Entrée :
	– Vedette : <i>Complétude</i>
	– Classe grammaticale : <i>nom</i>
	– Genre grammatical : <i>féminin</i>
	– Définition :
	* Définition : <i>Caractère de ce qui est complet, achevé.</i>
	* (Langue : <i>français</i>)
Complétude, <i>nom féminin</i> Caractère de ce qui est complet, achevé.	

FIG. 2.1: Représentation β

Notons que nous pouvions aisément préciser la **langue** de la **définition** (non nécessaire pour un dictionnaire unilingue), étant donné que la **définition** était la seule caractéristique dans sa propre entité (de même nom), mais ajouter les **langues** de la **classe grammaticale** et du **genre grammatical** pose un problème de redondance, soulevé précédemment (figure 2.2).

	• Entrée :
	– Vedette : <i>Complétude</i>
	– Langue de la vedette : <i>français</i>
	– Classe grammaticale : <i>nom</i>
	– Langue de la classe grammaticale : <i>français</i>
	– Genre grammatical : <i>féminin</i>
	– Langue du genre grammatical : <i>français</i>
	– Définition :
	* Définition : <i>Caractère de ce qui est complet, achevé.</i>
	* Langue : <i>français</i>
Complétude, <i>nom féminin</i> Caractère de ce qui est complet, achevé.	

FIG. 2.2: Représentation β avec redondance

En transformant des caractéristiques en entités par regroupement thématique (figure 2.3), nous voyons mieux les points communs entre toutes les **langues**, qui peuvent à présent être traitées de la même manière.

Remarque : c'était une illustration et il serait de bon aloi de représenter les classes et genres grammaticaux par des codes issus d'une liste fermée (faite par le linguiste) qui seraient ensuite traduits selon les besoins lors d'une étape ultérieure...

Complétude, <i>nom féminin</i> Caractère de ce qui est complet, achevé.	• Entrée :	
	– Vedette :	
	* Lexème :	<i>Complétude</i>
	* Langue :	<i>français</i>
	– Classe grammaticale :	
	* Type :	<i>nom</i>
	* Langue :	<i>français</i>
	– Genre grammatical :	
	* Type :	<i>féminin</i>
	* Langue :	<i>français</i>
	– Définition :	
	* Définition :	<i>Caractère de ce qui est complet, achevé.</i>
	* Langue :	<i>français</i>

FIG. 2.3: Représentation β sans redondance

Notons qu'une autre représentation, α , est possible, séparant les entités et les caractéristiques respectivement en informations et méta-informations (figure 2.4).

Complétude, <i>nom féminin</i> Caractère de ce qui est complet, achevé.	• Entrée :	
	– Vedette : <i>Complétude</i>	
	* Langue :	<i>français</i>
	– Classe grammaticale : <i>nom</i>	
	* Langue :	<i>français</i>
	– Genre grammatical : <i>féminin</i>	
	* Langue :	<i>français</i>
	– Définition : <i>Caractère de ce qui est complet, achevé.</i>	
	* Langue :	<i>français</i>

FIG. 2.4: Représentation α

Malgré tout, la représentation actuellement utilisée par défaut pour des raisons de rétrocompatibilité est la γ (figure 2.5).

Complétude, *nom féminin*
 Caractère de ce qui est complet, achevé.

- Entrée :
 - Vedette :
 - * caractéristique :
 - attribut : *lexème*
 - valeur : *Complétude*
 - * caractéristique :
 - attribut : *langue*
 - valeur : *français*
 - Classe grammaticale :
 - * caractéristique :
 - attribut : *classe grammaticale*
 - valeur : *nom*
 - * caractéristique :
 - attribut : *langue*
 - valeur : *français*
 - Genre grammatical :
 - * caractéristique :
 - attribut : *genre grammatical*
 - valeur : *nom*
 - * caractéristique :
 - attribut : *langue*
 - valeur : *français*
 - Définition :
 - * caractéristique :
 - attribut : *définition*
 - valeur : *Caractère de ce qui est complet, achevé.*
 - * caractéristique :
 - attribut : *langue*
 - valeur : *français*

FIG. 2.5: Représentation γ

Chapitre 3

Mise en œuvre

3.1 Généralités

Le format choisi pour représenter les informations du dictionnaire est XML. Notons que les différentes représentations décrites précédemment sont possibles,

De manière générale, il y a une bonne correspondance entre les terminologies Lexika et XML :

- éléments XML = entités Lexika ;
- attributs XML = caractéristiques Lexika.

3.2 Algorithme général

Voici l'algorithme général simplifié, sans détail de programmation et de gestion des erreurs (la version complète se trouve dans la documentation technique autogénérée du module Lexika) :

1. création des entités primordiales techniques et issues des informations du fichier de configuration ;
2. lecture du fichier source, ligne par ligne :
 - (a) décomposition de la ligne selon une expression rationnelle paramétrable en différents éléments, dont la **balise**, la donnée linguistique et les éventuelles métadonnées ;
 - (b) vérification de la présence de la **balise** dans le format d'entrée et si elle doit être traitée (sinon, passage à la ligne suivante) ;
 - (c) lecture des paramètres techniques d'entrée associés à la **balise**, notamment l'**abstraction**, les paramètres éventuels et les mots clefs techniques ;
 - (d) application des règles du format de sortie associées à l'**abstraction**, notamment l'**entité linguistique** (son nom et ses **caractéristiques**), ses **parents potentiels** (dans quelle **entité** elle se place) et d'autres mots clefs techniques comme ceux de structure (formation d'identifiant, etc.) :
 - i. si l'**entité parente** est trouvée, l'**entité en cours de traitement** s'y place ;
 - ii. sinon, création de l'**entité parente** ou entreposage de l'**entité en cours** en attendant que son **parent** soit rencontré (paramétrage)
3. création du fichier XML paramétré par conversion de la structure objet interne (objets et attributs Python (représentant les **entités** et les **caractéristiques** Lexika) vers éléments et attributs XML) ;
4. si voulu, création du fichier HTML par transformation XSL ;
5. si voulu, création du fichier \LaTeX par transformation XSL.

3.3 Remarques

La première version de l'architecture s'attendait assez naïvement à toujours trouver l'entité parente avant son entité enfant, autrement dit à toujours suivre un ordre de création des entités du plus général au plus spécifique (d'abord une entrée, puis un sens, puis une définition et des exemples, puis des notes diverses pour chacune de ces dernières entités, etc.). Cette *règle* est tout de même bien suivie mais comporte quelques exceptions, notamment parce que les logiciels en amont (comme Toolbox) ont différentes utilisations ou conceptions, ce qui implique quelques écarts qu'il faut pouvoir gérer à notre niveau en limitant autant que faire se peut les cas particuliers.

Plusieurs moyens étaient à disposition, notamment :

- la création d'un entrepôt qui garderait en mémoire des entités orphelines en attendant que leur parent soit créé normalement ultérieurement (méthode rétrospective : l'entrepôt est contrôlé régulièrement pour voir si dans le passé des orphelins sont en recherche), cette méthode peut devenir complexe si de nombreux orphelins apparentés et dans le désordre sont tous en attente ;
- la création *ex nihilo* du parent manquant avant que le réel ne le remplace adéquatement ultérieurement (méthode prospective : le parent est créé en avance sans les informations primordiales en attendant un futur changement par le vrai parent), cette méthode peut devenir complexe si les parents peuvent avoir des frères (listes) et demande donc des contrôles sur les informations.

Les deux méthodes ont chacun leurs avantages et sont ainsi toutes les deux accessibles.

De plus, l'on pourrait se demander pourquoi des **abstractions** existent puisque l'on pourrait directement lier une **balise** à une **entité linguistique**, or c'est la présence de certains formats qui nécessitent la création d'**entités linguistiques intermédiaires ex nihilo** qui a motivé ce choix de disjoindre ce qui relève de l'entrée et ce qui relève de la sortie : une **balise** d'entrée peut appeler une **abstraction** qui va en appeler une **autre** pour finalement créer plusieurs **entités linguistiques** de sortie, selon le format de sortie sans pour autant changer celui d'entrée.

3.4 Fichiers d'intérêt

Outre le fichier `base.py` qui sert de point d'entrée au logiciel, un fichier contient toute la configuration nécessaire pour paramétrer la création d'un nouveau dictionnaire (et qui est donc la cible principale de l'interface graphique) : `personnalisation/informations_linguistiques.py`. Ce dernier contient notamment les formats d'entrée et de sortie, donc d'une part le lien entre les **balises** et les **abstractions** et d'autre part le lien entre **celles-là** et les **entités linguistiques**, ainsi que toutes les expressions régulières utiles et d'autres variables d'intérêt (comme les différentes langues, etc.).

Il n'est point besoin d'être exhaustif dans la description du format puisque ce fichier surdéfinit par ajout le fichier `configuration/informations_linguistiques.py` qui contient les informations de base des formats classiques et qui de ce fait devrait être suffisant dans les cas simples (dictionnaires peu complexes utilisant les balises par défaut).

3.5 Configuration

```

1 BALISE:
2     abstraction: ABSTRACTION
3     caractéristiques:
4         CARACTÉRISTIQUE: VALEUR
5
6 ABSTRACTION:
7     entité:
8         nom: ENTITÉ
9         attribut: ATTRIBUT
10    parent:
11        nom: ENTITÉ
12        conditions: CONDITIONS
13    préabstraction:
14        nom: ABSTRACTION
15        conditions: CONDITIONS

```

Explication des variables (en capitales) :

- **balise** : identifiant de la balise dans le fichier source ;
- **abstraction** : identifiants de l'abstraction à appeler pour le traitement des données associées à la balise (dans le bloc balise) et de l'abstraction à appeler conditionnellement au préalable (dans le bloc préabstraction) ;
- **caractéristique** & **valeur** : caractéristiques associées à l'entité finale ;
- **entité** : identifiants de l'entité à créer (dans le bloc entité) et du parent dans lequel placer l'entité à créer (dans le bloc parent) ;
- **conditions** : conditions diverses pour choisir le parent adéquat (dans le bloc parent) et pour créer la préabstraction (dans le bloc préabstraction) :
 - *impérieuse* : appelle la préabstraction quel que soit l'état du parent (notamment si un parent potentiel est déjà présent), à utiliser pour systématiquement créer une entité ;
 - égalité de caractéristiques ou si vide...
- **attribut** : nom de remplacement de l'entité sous forme d'attribut XML dans le format γ ;

3.6 Illustration

Extrait du code Python représentant la partie qui concerne la définition dans les formats d'entrée et de sortie :

```
1 de:
2   abstraction: définition
3   caractéristiques:
4     langue: cible 1

1 définition:
2   entité:
3     nom: représentation de texte
4     attribut: forme écrite
5   parent:
6     abstraction: bloc définition
7     nom: définition
8     conditions: impérieuse
9 bloc définition:
10  entité:
11    nom: définition
12    structure:
13    identifiant:
14      constante: []
15  parent:
16    abstraction: sens
17    nom: sens
18 sens:
19  entité:
20    nom: sens
21    structure:
22    identifiant:
23      constante: []
24  parent:
25    nom: entrée lexicale
26 entrée:
27  entité:
28    nom: entrée lexicale
29    structure:
30    identifiant:
31      constante: []
32  parent:
33    abstraction: dictionnaire
34    nom: dictionnaire
```

3.7 Remarques

Si nous prenons par exemple la norme LMF, les entités suivantes peuvent poser quelques difficultés de configuration (bien que la norme soit déjà bien configurée, ces explications peuvent aider à la création de formats maison ou dépassant ladite norme) :

- Entité **lemme** : la balise **lx**, bien que signalant instinctivement la nouvelle entrée, ne renseigne que le **lemme** de la nouvelle entrée et crée donc au préalable son parent **entrée lexicale**.
- Entité **exemple** : la balise **xv** crée un **exemple** tandis que les balises **xe** et **xn** créent des **traductions d'exemples**, puisque même si dans le XML LMF les trois entités sont au même niveau dans le parent **contexte**, c'est bien l'**exemple** qui va impérieusement créer un nouveau bloc **contexte** tandis que les **traductions** se placeront dans le dernier **contexte** créé.
- Entité **définition** : la balise **de** (au même titre que **dn**, etc.) crée une **représentation de texte** qui se place dans le parent **définition** (respectivement par les abstractions **définition** et **bloc définition**), ce niveau supplémentaire (relativement au cas où la définition serait la valeur directe de l'entité **définition**, sans **représentation de texte**), optionnel selon LMF et prévu notamment pour des définitions dans plusieurs langues, est présent pour deux raisons : mieux structurer ce qui relève de la définition elle-même et permettre la présence de texte enrichi (renvois et style pour les emphases sémantiques) sans rentrer en conflit avec d'autres entités au même niveau (comme les **déclarations**).

- Entité **sens** : la balise **sn** renseigne un **numéro de sens** et crée une entité sans valeur qui servira de conteneur à d'autres entités (comme les **définitions** et les **exemples**).
- Entité **relation sémantique** : les balises **sy**, **an**, **cf** ainsi que les couples de balises **lf X/lv Y** renseignent tous sur une nouvelle **relation sémantique**, et ce, par différentes voies.

3.8 Exemple

Prenons à présent un exemple plus concret, tiré du dictionnaire japhug de Guillaume Jacques. À partir d'une source MDF (à gauche), nous créons un fichier XML (à droite) selon différentes configurations.

Analyse simplifiée par le logiciel avec une configuration maison, étape par étape (figures 3.1, 3.2 et 3.3), pour une représentation β :

- (création de l'entité parente primordiale **diçtionnaire** de **langue japhug**) ;
- lecture de la balise **lx** et de l'information **aj**, correspondant à une **vedette** d'**entrée**, créée comme enfant du **diçtionnaire japhug** ;
- lecture de la balise **ps** et de l'information **pro**, correspondant à une **classe grammaticale** de l'**entrée aj** ;
- lecture de la balise **ge** et de l'information **moi**, correspondant à une **glose** de **définition** (avec une **langue français**), créée comme enfant de l'**entrée aj** ;
- lecture de la balise **gn** et de l'information **我**, correspondant à une **glose** de **définition** (avec une **langue chinois**), créée comme enfant de l'**entrée aj** ;
- lecture de la balise **cf** et de l'information **azo**, correspondant à une **cible** de **relation sémantique** (avec un **type renvoi**), créée comme enfant de l'**entrée aj** ;

<pre> \lx aj \ps pro \ge moi \gn 我 \cf azo </pre>	<pre> 1 <Entrée> 2 <Vedette>aj</Vedette> 3 <ClasseGrammaticale>pro</ClasseGrammaticale> 4 <Définition language="fra">moi</Définition> 5 <Définition language="cmn">我</Définition> 6 <RelationSémantique type="renvoi">azo</RelationSémantique> 7 </Entrée> </pre>
---	--

FIG. 3.1: Représentation XML- α

<pre> \lx aj \ps pro \ge moi \gn 我 \cf azo </pre>	<pre> 1 <Entrée vedette="aj" classe_grammaticale="pro"> 2 <Définition glose="moi" language="fra" /> 3 <Définition glose="我" language="cmn" /> 4 <RelationSémantique cible="azo" type="renvoi" /> 5 </Entrée> </pre>
---	--

FIG. 3.2: Représentation XML- β

<pre> \lx aj \ps pro \ge moi \gn 我 \cf azo </pre>	<pre> 1 <Entrée> 2 <caractéristique attribut="vedette" valeur="aj" /> 3 <caractéristique attribut="classe grammaticale" valeur="pro" /> 4 <Définition> 5 <caractéristique attribut="glose" valeur="moi" /> 6 <caractéristique attribut="langue" valeur="fra" /> 7 </Définition> 8 <Définition> 9 <caractéristique attribut="glose" valeur="我" /> 10 <caractéristique attribut="langue" valeur="cmn" /> 11 </Définition> 12 <RelationSémantique> 13 <caractéristique attribut="cible" valeur="azo" /> 14 <caractéristique attribut="type" valeur="renvoi" /> 15 </RelationSémantique> 16 </Entrée> </pre>
---	---

FIG. 3.3: Représentation XML- γ

Analyse simplifiée par le logiciel avec une configuration LMF, étape par étape (figures 3.4, 3.5 et 3.6), pour une représentation β :

1. (création de l'entité parente primordiale **dictionnaire** de **langue japhug**) ;
2. lecture de la balise **lx** et de l'information *aj*, correspondant à une **lexème** de **lemme**, créée comme enfant d'**entrée lexicale**, elle-même créée comme enfant du **dictionnaire japhug** ;
3. lecture de la balise **ps** et de l'information *pro*, correspondant à une **partie du discours** de l'**entrée lexicale aj** ;
4. lecture de la balise **ge** et de l'information *moi*, correspondant à une **glose** de **définition** (avec une **langue français**), créée comme enfant du **sens général**, lui-même créé comme enfant de l'**entrée aj** ;
5. lecture de la balise **ge** et de l'information *我*, correspondant à une **glose** de **définition** (avec une **langue chinois**), créée comme enfant du **sens général** ;
6. lecture de la balise **cf** et de l'information *azo*, correspondant à une **cible** de **relation sémantique** (avec un **type renvoi**), créée comme enfant de l'**sens général** ;

<pre> 1 2 3 4 5 6 7 8 9 10 11 \lx aj \ps pro \ge moi \gn 我 \cf azo </pre>	<pre> 1 <EntréeLexicale> 2 <PartieDuDiscours>pro</PartieDuDiscours> 3 <Lemme> 4 <Lexème>aj</Lexème> 5 </Lemme> 6 <Sens sens="général"> 7 <Définition langue="fra">moi</Définition> 8 <Définition langue="cmn">我</Définition> 9 <RelationSémantique type="renvoi">azo</RelationSémantique> 10 </Sens> 11 </EntréeLexicale> </pre>
---	---

FIG. 3.4: Représentation XML-LMF- α

<pre> 1 2 3 4 5 6 7 8 \lx aj \ps pro \ge moi \gn 我 \cf azo </pre>	<pre> 1 <EntréeLexicale partie_du_discours="pro"> 2 <Lemme lexème="aj" /> 3 <Sens sens="général"> 4 <Définition glose="moi" langue="fra" /> 5 <Définition glose="我" langue="cmn" /> 6 <RelationSémantique cible="azo" type="renvoi" /> 7 </Sens> 8 </EntréeLexicale> </pre>
---	---

FIG. 3.5: Représentation XML-LMF- β

```

\lx aj
\ps pro
\ge moi
\gn 我
\cf azo

```

```

1      <EntréeLexicale>
2          <Lemme>
3              <caractéristique attribut="lexème" valeur="aj" />
4          </Lemme>
5          <caractéristique attribut="partie du discours" valeur="pro" />
6          <Sens>
7              <caractéristique attribut="sens" valeur="général" />
8              <Définition>
9                  <caractéristique attribut="glose" valeur="moi" />
10                 <caractéristique attribut="langue" valeur="fra" />
11             </Définition>
12             <Définition>
13                 <caractéristique attribut="glose" valeur="我" />
14                 <caractéristique attribut="langue" valeur="cmn" />
15             </Définition>
16             <RelationSémantique>
17                 <caractéristique attribut="cible" valeur="azo" />
18                 <caractéristique attribut="type" valeur="renvoi" />
19             </RelationSémantique>
20         </Sens>
21     </EntréeLexicale>

```

FIG. 3.6: Représentation XML-LMF-γ

Ces deux analyses, correspondant à des configurations, des représentations et des formats différents, ne donneront pas les mêmes fichiers XML en sortie, chacun d'eux aura ses qualités et inconvénients selon l'objectif voulu, mais les deux pourront donner *in fine* des dictionnaires similaires voire identiques (selon les post-traitements, notamment la transformation XSL).

Chapitre 4

Tutoriel – sans interface graphique

Tout d'abord, il faut s'assurer que l'interpréteur Python 3 est installé, et faire particulièrement attention si Python 2 est installé. Voyons maintenant un exemple avec le dictionnaire japhug de Guillaume Jacques.

4.1 Configuration du dictionnaire

La configuration se trouve dans le fichier `configuration.yml` (au format YAML qui permet une modification plus aisée pour des profanes) situé dans le dossier `japhug` :

Vous ne devriez jamais avoir à modifier un autre fichier puisque celui-ci contient toutes les informations nécessaires au bon déroulement du programme.

4.2 Exécution du programme

```
python3 base.py japhug/configuration.yml
```

Chapitre 5

Tutoriel – avec interface graphique

Chapitre 6

Aide-mémoire

6.1 Commandes

Si vous travaillez avec plusieurs dictionnaires, il est préférable de donner comme premier argument le chemin d'accès au fichier de configuration décrit précédemment :

```
python3 base.py configuration.yml
```

Sinon, s'il est configuré dans le fichier `base.py`, vous pouvez vous contenter de cette commande :

```
python3 base.py
```

6.2 Fichier de configuration

Le fichier de configuration, seul fichier nécessaire à modifier vous-même dans un cas classique, est un fichier au format YAML, commode pour modifier de manière lisible les informations. Il est fortement structuré en arborescence et contient notamment ces informations essentielles :

- le chemin source du lexique source (absolu ou relatif au chemin du fichier de configuration) ;
- le format d'entrée (MDF, etc.) ;
- le format de sortie (direct, LMF, personnalisé...) ;
- l'identifiant du dictionnaire (le nom de la langue, par exemple) ;
- des informations relatives au dictionnaire lui-même :
 - le nom du dictionnaire ;
 - les auteurs ;
 - des commentaires...
- des informations relatives au format XML :
 - le chemin cible du fichier XML ;
 - la langue du format ;
 - le format (avec listes des éléments frères ou non) ;
- des informations relatives au format HTML :
 - le chemin cible du fichier HTML ;
 - le chemin cible du gabarit XSL ;
 - la langue du format ;
- des informations relatives au format Latex :
 - le chemin cible du fichier \LaTeX ;
 - le chemin cible du gabarit XSL ;
 - la langue du format ;
 - le format (avec listes des éléments frères ou non) ;
- l'ordre lexicographique (une liste pouvant contenir des listes) ;
- des informations sur les langues utilisées dans le dictionnaire, sous forme de liste ordonnée :
 - leur identifiant (l'identifiant ISO 639-3, par exemple) ;
 - leur statut (cible ou source)

Chapitre 7

Remarques pour les développeurs et pour la réunion

L'architecture suppose qu'une ligne *B* placée sous une ligne *A* créera une entité *B* qui sera soit un enfant de l'entité *A*, soit un frère de *A* ou soit un élément non lié à *A*, et en aucun cas un parent de *A*, or certaines entités, comme les notes ou les langues d'étymon, peuvent être placées avant leur parent structurel (note avant la définition, etc.), or cela crée un problème double :

1. comment savoir si cette entité parente se trouve avant ou après (cas d'une note placée entre 2 définitions) ?
2. qu'en faire si le parent n'existe pas encore ?

Une solution au second problème consiste à entreposer l'entité en attendant que son parent soit analysé et créé (solution simple).

La solution au premier problème est plus complexe : si l'entité ne sait pas si son parent est placé avant et donc déjà créé ou s'il est placé après et donc non encore créé, il faut d'autres indices : si l'entité ne peut pas exister en plusieurs exemplaires, le problème devient plus simple, mais c'est irréaliste (il peut y avoir plusieurs notes pour une définition, voire plusieurs du même type) ; si l'entité a des paramètres qui doivent être identiques à ceux du parent (la langue, par exemple), le problème devient aussi plus simple, mais la multiplicité des cas possibles, l'absence de paramètres inclus, rend la tâche plus difficile pour tout le monde ; sinon, l'autre possibilité consiste à indiquer comme paramètre de configuration de balise où est censé se trouver le parent (avant par défaut), en ce cas il faut prévoir un moyen de vérifier que le parent adéquat a été créé après (compteur d'entités ou numéro de ligne) et bien gérer le polymorphisme d'entité (si une entité placée avant peut être dans l'absolu l'enfant de plusieurs types de parents différents).

Exemples

```
\sn 3
\he partic
\we syntagme possessif sujet
\lt le X de Y existe
\de Y a X. Traduit le verbe "avoir"
\hn hence
\wn possessive phrase as subject
\ll Y's X exists
\dn Y has X. Translates the verb "have"
```

Combien d'entités importantes au sein du sens (sn) ? La définition (de/dn) est-elle l'entité clef et donc la note syntaxique (we/wn) est-elle une caractéristique de la définition ? La traduction littérale (lt/l) est-elle une entité sœur de la définition ou une caractéristique de cette dernière ? Si on considère que l'étiquette sémantique (he/hn) est une entité enfant du sens (ou des caractéristiques du sens), pourquoi la mettre en plusieurs langues et surtout pourquoi ne pas mettre he et hn à la suite ?

```
\el P0c
\et *Raka
\eg vine, creeper
```

Quelle est l'entité clef de ce bloc ? La langue (el) ou l'étymon (et) ? Laquelle est primordiale et peut exister seule ? La glose étymologique est l'enfant/la caractéristique de la langue ou de l'étymon ?

```
\rf th-179
\xv n-eh a kēy laklak aē en
\xe la chanson sur laquelle ils sont en train de danser
\xn the song on which they are dancing
```

La référence (rf) est vraisemblablement une caractéristique de l'exemple (xv), et les traductions d'exemple (xe/xn) sont des entités enfants de l'exemple.

```
\lf Syn.
\lv mey a
\lv mey

<RelationSémantique>
  <caractéristique attribut="type" valeur="Syn."/>
  <caractéristique attribut="cible" valeur="mey a"/>
</RelationSémantique>
<RelationSémantique>
  <caractéristique attribut="type" valeur="Syn."/>
  <caractéristique attribut="cible" valeur="mey"/>
</RelationSémantique>
```

La factorisation de la cible de relation sémantique (lv) pour un même type de relation sémantique (lf) révèle le même problème : quelle est l'entité importante, la cible ou le type ?

Chapitre 8

Améliorations et bogues

- Améliorer l'interface graphique.
- Générer la documentation automatique.
- Permettre la création d'un lexique inverse.
- Traduire davantage dans des grandes langues (notamment le présent fichier).
- sens imbriqués ?

Chapitre 9

Annexes

9.1 Dépôt du logiciel

<https://bitbucket.org/BenjaminGalliot/lexika>