Réseau Sémantique à partir d'un dictionnaire

Bawden Rachel, Brogniez Caroline et Parslow Nicholas

1 Introduction

2 Objectifs

3 La Conception du Graphe

3.1 Reseau Sémantique

L'ambiguité inhérent des langues humaines est un grand défi pour le traitement automatique de langue naturelle. En linguistique, même s'il y a beaucoup d'avances dans la désambiguisation syntaxique, l'ambiguité sémantique reste une tâche extrèment difficile. Un supercomputer Watson de IBM a réussi a gagné a Jeopardy avec toute sa puissance mais sans eviter de faire des fautes simples. En plus, la répresentation sémantique logique n'est toujours pas au point de résoudre des tâches qui ont besoin des liens moins concrète ou plus flous, telles que

- la désambiguïsation lexicale (Word Sense Disambiguation)
- la recherche d'information (par exemple sur la toile ou dans une bibliothèque)
- la catégorisation de documents
- le résumé automatique de texte
- la traduction automatique

L'idée le plus populaire pour résoudre ses difficultés est l'utilisation d'un réseau sémantique qui donne l'avantage de pouvoir répresenter tous les connections taxonomiques ensembles en une seule entité. Cette simplication n'est pas sans coût, en particulier la répresentation des liens logiques telles que la négation et la disjonction et en plus connaissances non-taxonomique sont particulièrement difficile à capter.

L'idée de base pour un reseau semantique est de construire un graphe avec les concepts pour noeuds et les liens entre les concepts comme arrêts. Il exist deux approches générales à la construction. Premièrement, à partir d'un dictionnaire, et deuxièment, à partir d'un corpus. Un corpus a l'avantage de contenir les vraies utilisations des mots non-biasé par les selections des auteurs, et aussi une grande quantité d'utilisations, mais au même temps exige un algorithme non-supervisé pour la désambiguisation et un autre algorithme pour la conversion de collocation en lien sémantique. Word2Vec [7] de google réprésente le point actuel de cet approche. Par contre, le dictionnaire, même si plus petit, donne plus d'information sur les mots rares et possède déjà des liens sémantiques explicites entre les mots.

3.2 Structure du reseau

L'utilisation d'un dictionnaire dans les tâches semantiques computationnelles existe depuis les années 60 [6]. La construction d'un réseau sémantique à partir d'un dictionnaire (par exemple en français [8], [9]) est possible grâce à la structure des dictionnaires qui permet de faire ressortir des liens sémantiques. La structure en sens, sous-sens, définitions, et exemples, parmi d'autres informations, mais aussi la structure interne des définitions contient une régularité qu'il est important d'exploiter au maximum. Nous tenons donc à conserver le plus possible cette structure dans la transformation de dictionnaire en graphe. Un graphe est défini formellement comme un ensemble de sommets et un ensemble d'arcs qui relient une paire de sommets.

$$G =< S, A >$$

Pour représenter un dictionnaire par un graphe, nous considérons que les sommets peuvent être les mots individuels du dictionnaire ou même les niveaux intermédiaires de la structure tels que 'exemple', 'définition', 'synonyme', 'antonyme' etc. Les arcs sont alors les liens qui lient les différents éléments d'une entrée de dictionnaire et permettraient de trouver un lien entre un lexème donné et la manière dont il est décrit dans son entrée du dictionnaire.

Le graphe d'un reseau sémantique est connu d'exhiber les propriétés 'Small World' [12] - terme qui existe depuis l'experience fameux de Travers et Milgram [10]. Watts et Strogatz [11] ont défini les propriétés d'un graphe 'Small World' ainsi : Si un graphe possède N noeuds, et $d_{min}(i, j)$ est la distance minimum entre noeuds i et j, le 'Characteristic Path Length' L se défini par :

$$L = \frac{1}{N(N-1)} \sum_{i \neq j} d_{min}(i, j)$$

et si pour un noeud i, l'ensemble de ses voisins immédiats est $\Gamma(i)$ et le nombre d'arrêts entre ses voisins immédiats est $E(\Gamma(i))$ alors le 'Clustering Co-efficient' C se défini par :

$$C = \frac{1}{N} \sum_{i=1}^{N} \frac{E(\Gamma(i))}{\binom{|\Gamma(i)|}{2}}$$

Autrement dit, L est la moyenne des distances minimales entre chaque couple de noeuds dans le graphe, et C est la moyenne du ratio des liens entre les voisins d'un noeud et le nombre maximum de liens possible.

Un graphe 'Small World' est défini par les propriétés suivants :

$$L \sim L_{rand} \sim \frac{log(N)}{log(k)}$$

et

$$C >> C_{rand} \sim \frac{2k}{N}$$

où k est le dégée moyenne des noeuds du graphe et L_{rand} et C_{rand} sont respectivement le Characteristic Path Length et Clustering Co-efficient pour un graphe aléatoire.

Par conséquent, un graphe 'Small World' n'est pas dense, mais a des distances entre les noeuds très petites. Une autre conséquence démontré par Barabási et Albert [13] est que la distribution des dégrés des noeuds suivre une loi de puissance :

$$p(k) \propto k^{-\alpha}$$

où $p(k)$ est la probabilité d'un noeud quelconque d'avoir un dégré k, et α s'approche à l'unité ([12])

3.3 Marche Aléatoire

3.4 Remarques sur la terminologie

Nous appelons 'mot' toute unité minimale du lexique. Un mot peut être soitfléchi, soit non-fléchi et par défaut nous faisons référence aux mots non-fléchis sous leur forme de dictionnaire. Par principe, nous restreignons le réseau aux lemmes, mais il est possible qu'il y apparaît des formes fléchies en cas de non-identification du lemme.

Par 'entrée' de dictionnaire nous faisons référence à un groupe d'informations (catégories syntaxiques, sens, définitions, exemples etc.) associées à un mot donné. Par conséquent, le terme 'entrée' peut aussi être utilisé pour dénoter le mot lui-même, et par extension les informations contenues pour ce mot donné.

La relation sémantique de synonymie est définie entre deux termes de la même catégorie de discours qui ont le même sens et qui peuvent donc être substitués l'un pour l'autre sans modifier le sens de la phrase. Cette définition pose évidemment des problèmes, surtout à cause du fait qu'il est toujours possible de trouver une différence de sens ou d'usage entre deux mots malgré le fait qu'ils soient habituellement classés en synonymes. Il est parfois souhaitable de parler de proche-synonymes au lieu de synonymes tout court. Néanmois, nous préférons utiliser le terme 'synonyme' pour parler de ces cas, sans postuler de théorie sur les frontières de la synonymie. Par la suite, la synonymie sera définie en termes de relations attestées dans des ressources externes et nous nous reportons à ces références pour établir si deux mots sont en relation de synonymie ou pas.

De même pour les relations d'antonymie, d'hyperonymie et d'hyponymie. L'antonymie est définie comme la relation entre deux mots à sens opposé. L'hyperonymie entre un mot dont l'extension contient l'extension d'un autre mot (par exemple, 'véhicule' est l'hypernym de 'voiture'). L'hyponymie est la relation inverse d'hyperonymie, entre un mot dont l'extension est incluse dans l'extension d'un autre (pour reprendre le même exemple, 'voiture' est un hyponyme de 'véhicule').

[AUTRES DEFINITIIONS...]

4 La structure des dictionnaires utilisés

Les deux ressources qui seront utilisées sont le Wiktionnaire français en format XML du CLLE-ERSS dans le cadre du projet WiktionaryX [2] et le Littré , qui est aussi disponible

en format XML [3].

Les informations contenues dans les deux dictionnaires sont similaires. Chaque dictionnaire est organisé en entrées, et chaque entrée contient plusieurs définitions, des exemples, des synonymes et des informations grammaticales. Le wiktionnaire contient en plus d'autres relations sémantiques telles que l'antonymie, l'hyperonymie et l'hyponymie.

En termes de style, les deux dictionnaires sont très différents.

Le Wiktionnaire est une ressource libre qui a comme objectif de décrire tous les mots dans toutes les langues. C'est une ressource participative qui peut donc être éditée par tout le monde. Les informations sont très structurée et destinée à un format informatique, malgré le fait que la ressource soit ouvert à l'édition par tout le monde. Il contient un très grand nombre termes, anciens comme nouveaux et aussi des formes fléchies.

Le Littré est inspiré du dictionnaire d'Emile Littré du dix-neuvième siècle et contient de nombreuses exemples et de citations littéraires. Le dictionnaire, n'étant pas destiné au départ à un format en-ligne, est moins structuré en termes des informations contenues que le Wiktionnaire et contient aussi moins de termes en général. Les informations contenues sont destinées à un approfondissent des connaissances, ce qui est reflété parfois dans la manière très précise de décrire une entrée, jusqu'à sa catégorie syntaxique. Par exemple, l'entrée 'F' a l'en-tête suivante (contenant sa catégorie syntaxique) :

```
<entree terme="F">
<entete>
<prononciation> èf, ou, suivant la manière moderne d'épeler, fe </prononciation>
<nature> s. f. quand on prononce cette lettre èf, une petite f, et s. m. quand on la prononce fe, un f majuscule. </nature>
</entete>
```

Ceci pose des difficultés pour notre traitement automatique des données, qui doivent être prise en compte lors du pré-traitement.

Des extraits des deux dictionnaires se trouvent dans l'annexe. ([REF])

4.1 Statistiques

NOMBRE d'ENTREES NOMBRE DE CATS SYNT DIFF NOMBRE DE MOT-FORMES DIFF

5 Pré-traitement

Dans le but de pouvoir procéder à un traitement homogène des deux dictionnaires, une normalisation des deux formats était nécessaire. Ce pré-traitement a permis à la fois de supprimer certaines informations qui n'étaient pas utilisée par la suite et de convertir les balises et leur contenu en un format plus convenable pour notre traitement. Nous réunissons

les deux structures dans un seul type de fichier XML avec les balises suivantes :

- entry : un mot du dictionnaire (lexème)
- pos : la catégorie syntaxique (il peut y en avoir plusieurs par lexème)
- sense : le niveau hiérarchique représentant plusieurs sens d'un même mot
- subsense : le niveau hiérarchique représentant plusieurs sous-sens d'un même sens global
- def-text : le texte d'une défintion
- semantic : l'emploi sémantique d'un mot (figuratif, absolu etc.)
- register : le registre du mot (familier, vulgaire, soutenu etc.)
- domain : le domaine sémantique du mot (géographie, biologie, culinaire etc.)
- refs : des références vers d'autres entrées lexicales
- examples : des phrases exemples contenant le mot de l'entrée lexicale
- synonyms : synonymes de du mot de l'entrée
- antonyms : antonymes de du mot de l'entrée
- hyperonyms : hyperymes de du mot de l'entrée
- hyponyms : hyponymes de du mot de l'entrée

La DTD des fichiers XML normalisés se trouve dans l'appendice E, avec les scripts de normalisation C et D. Figure 1 illustre l'hiérarchie des balises, qui correspond aussi à la structure théorique du réseau.

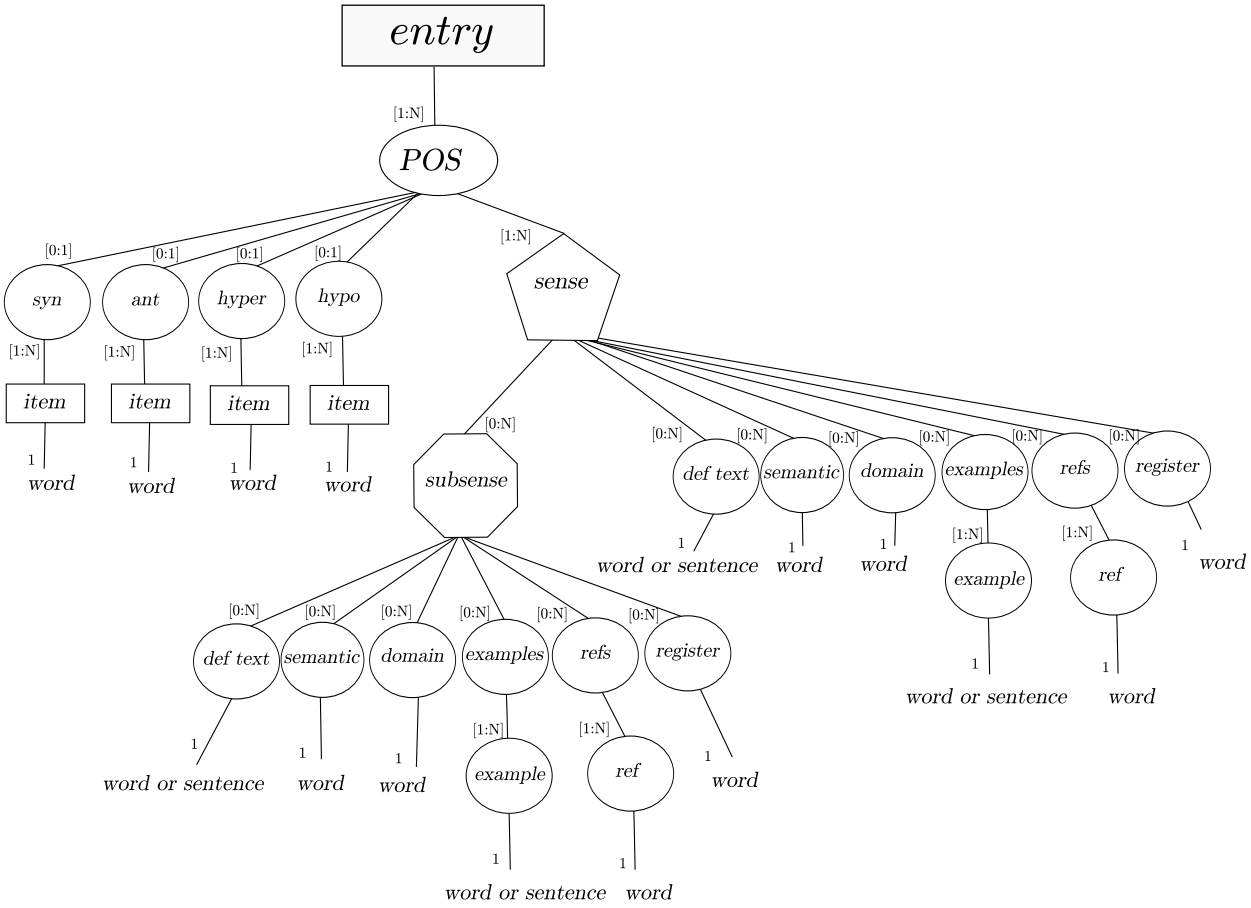

FIGURE 1 – L'hiérarchie des entrées définie pour les dictionnaires en XML

De légères différences existent entre les structures des deux dictionnaires. Tandis que la plupart des informations du Wiktionnaire sont localisées au niveau du 'sense', la plupart des informations du Littré sont localisées au niveau du 'subsense', ce qui ne pose aucun problème pour la construction des deux réseaux.

En plus de la restructuration des fichiers XML, les dictionnaires ont été nettoyés pour convenir à nos besoins. Certaines balises contiennent trop d'informations ou des informations non-pertinentes. Par exemple, les balises des commentaires pour le Littré ne contiennent pas de contenu exploitable et les citations des textes anciens des mots vieillis qui risquent d'ajouter du bruit dans le réseau résultant. Les entrées sont ainsi réduites au schéma ci-dessus (Figure 1). Le Littré en particulier nécessite une étape importante de nettoyage, puisque des erreurs existent dans le balisage des données et un manque d'homogénéité dans la structuration des entrées risque de perturber la manière dont laquelle les relations sont établies entre les éléments qu'elles contiennent. Les catégories syntaxiques utilisées ne sont pas d'une liste énumérable et contiennent des descriptions plutôt littéraires. Cette étape de normalisation consistait aussi à traduire ces catégories syntaxiques en une liste plus formelle et identifiable.

Les définitions, exemples et autres informations dans les entrées étaient ensuite taggés et lemmatisés en utilisant l'outil MElt ([4]). Cette étape est importante pour pouvoir relier les mots fléchis des définitions avec leurs lemmes tels qu'ils apparaissent dans les entrées et pour savoir de quelle catégorie syntaxique il s'agit. Le script XXX qui sert à tagger et lemmatiser les documents se trouvent dans l'annexe [F].

6 La représentation en graphe

6.1 La première version du graphe

Les fichiers XML sont facilement transférables en représentation en graphe, puisqu'ils continennent une hiérarchie d'entrées. En principe chaque niveau de l'hiérarchie est représenté par un nœud différent avec des arcs qui lie chaque nœud à ses fils dans l'hiérarchie, comme dans Figure 1. Cette représentation a l'avantage de préserver la structure hiérarchique d'origine.

En plus des arêtes descendantes qui existent dans le schéma Figure 1, nous établissons des arêtes montantes, afin de retrouver facilement la relation entre un mot qui apparaît dans une entrée et le lexème de l'entrée. Les poids sur ces arêtes ne sont pas les mêmes que les arêtes descendantes afin de retrouver une différence dans ces relations (Voir [REF] la pondération des relations pour plus de détails).

6.1.1 Simplification de la structure

En pratique, il est possible de surpasser d'un grand nombre de nœuds intermédiaires dans l'hiérarchie en attribuant une arête directe entre une paire de mots dont le poids serait l'addition de toutes les valeurs des liens qui constituent le chemin entre les deux mots. Le choix des sommets est un compromis entre mettre le plus d'informations possible dans le

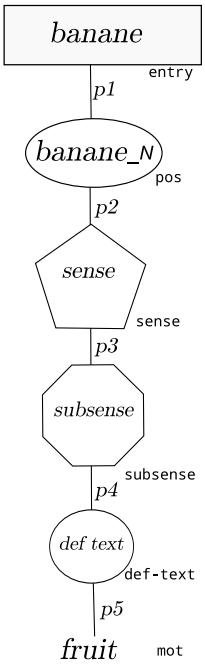

FIGURE 2

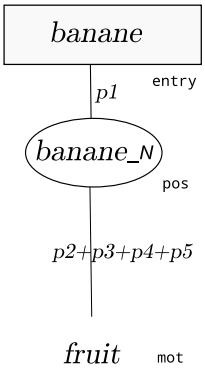

FIGURE 3

graphe et veiller à la non-explosion de la taille du graphe. Ceci n'est que possible parce que dans les tâches effectuées (détaillés dans la partie XXX), nous n'aurons jamais besoin d'extraire ces nœuds intermédiaires, même si nous souhaitons tenir compte de leur présence.

Ainsi, l'entrée dans Figure 2 est simplifiée en l'entrée dans Figure 3 :

Notons que nous préservons deux niveaux d'hiérarchie pour le mot d'entrée : un contenant le mot seul et l'autre contenant aussi sa catégorie syntaxique. Ceci est un moyen d'assurer à ce que le mot soit trouvable dans le réseau sans devoir spécifier une catégorie syntaxique particulière, tout en permettant de faire la distinction entre plusieurs catégories syntaxiques pour un mot donné. Chaque mot taggé à l'intérieur d'une entrée est ainsi lié à une version non-taggée qui représente le niveau 'entry' de ce mot, même s'il n'existe pas comme entrée dans le dictionnaire de départ.

Le résultat est donc un graphe qui contient deux sortes de nœuds : des mots non-taggés (correspondant au niveau entry' et des mots non-taggés (correspondant au niveau 'pos'), avec des relations pondérées selon le lien établi entre les deux mots dans le dictionnaire.

[IMAGE plein d'arêtes].

6.2 Graphe minimaliste

7 La pondération des relations

Pour injecter plus de sophistication dans le réseau, la distance entre deux mots ne se limite pas au nombre d'arêtes dans le chemin. Les arêtes sont pondérées selon le type de nœud sortant et entrant afin de distinguer entre les relations différentes qui peuvent exister à

différents endroits de l'entrée. Contrairement aux réseaux sémantiques plus sophistiqués tels que WordNet [1], qui encodent des relations typées spécifiques, notre réseau se base sur un principe plus simple de pondération, avec des valeurs réelles assignées par rapport à l'emplacement des mots dans l'entrée du dictionnaire.

Les poids des arêtes font partie des paramètres du réseau et sont conservés dans un fichier de configuration externe (weight.config), les valeurs étant choisies pour optimiser le réseau pour une tâche particulière. L'optimisation sera discutée dans la partie [XXX]. Il existe un total de 42 liens différent [CHECK] avec un poids différent entre 0 et 1, un lien de 0 étant un lien non-existant dans le graphe.

Ci-dessus un extrait de ce fichier, qui se trouve dans l'annexe [REF] :

```
entry2pos = 0.01
pos2entry = 0.01
pos2sense = 0.01
sense2pos = 0.01
pos2syn = 0.01
syn2pos = 0.01
...
sense2subsense = 0.001
subsense2sense = 0.001
subsense2deftext = 0.00001
deftext2subsense = 0.00001
subsense2semantic = 0
```

En plus de ces paramètres basés uniquement sur la structure des entrées, d'autres paramètres sont utilisés pour faire ressortir des informations lexicales en plus de ces relations déjà trouvées. Ces paramètres se trouvent dans un deuxième fichier de configuration, qui peut être modifié de la même façon que le fichier de poids. La différence est que l'utilisateur peut spécifier le type d'opération à effectuer sur le poids d'origine. Chaque paramètre est écrit sur une seule ligne avec l'étape de la création du graphe à laquelle le paramètre sera implémenté, ainsi que d'autres options utilisées pour spécifier comment le paramètre peut être utilisé.

Etant donné un mot x et un mot y, où x est le mot d'entrée et y le mot de sa définition, le poids d'origine serait le poids spécifié entre un mot d'entrée et sa définition (c'est-à-dire entry2pos + pos2sense + sense2subsense + subsense2deftext). Ensuite ce poids peut être transformé de plusieurs façons :

7.1 En fonction des propriétés catégorielles

Pendant l'étape de préparation du graphe, avant la création de la matrice, les poids sont traffiqués en fonction des propriétés catégorielles pour donner plus ou moins de poids à certaines catégories. Les paramètres sont de la forme suivante :

POS_de_y, POS_de_x, fichiers_contextes, opération_math, valeur

7.1.1 Modification uniquement en fonction de la catégorie syntaxique du mot y.

Ces paramètres servent à privilégier certaines catégories de mot par rapport à d'autres, en supposant que certaines catégories sont plus utiles pour établir des relations sémantiques. Ceci se fait en fonction des relations que nous souhaitons faire ressortir. Par exemple, il peut être plus pertinent de prendre en compte les adjectifs si ce sont des descriptions qui doivent ressortir et de donner plus d'importance aux verbes si ce sont des actions qu'il faut caractériser (une plus grande importance se traduisant par un plus petit poids). Un paramètre est spécifié pour chaque catégorie syntaxique afin d'établir des modifications relatives entre catégories. Par exemple, pour la catégorie 'adjectif' : 'prepare : Adj, None, None, +, 0.6' signifie que 0.6 est ajouté à tout noeud qui va vers un mot de catégorie 'Adj'.

7.1.2 Modification en fonction de la catégorie syntaxique du mot x, du mot y et d'un fichier de contextes spécifiques.

Jusqu'à présent le poids des relations a été fondé uniquement sur l'emplacement des mots y dans l'entrée et de leurs catégories syntaxiques. Mais il est aussi important d'analyser la structure des informations à l'intérieur des définitions, puisque les définitions font preuve d'une régularité qui permet de donner plus de poids à certains mots. Sans faire une analyse syntaxique complète des définitions, nous visons à cibler certains mots qui apparaissent dans un certain contexte afin de renforcer la relation entre le mot de l'entrée et ces mots cibles. La forme du paramètre dans le fichier de configuration est la même que pour les paramètres précédents, sauf que la deuxième option lqPOS_de_x' est spécifiée, ainsi que le chemin vers le fichier de contextes légitimisants.

Nous jugeons nécessaire de spécifier la catégorie syntaxique du mot x ainsi que la catégorie du mot y parce que la forme de la définition dépend en grande partie de la catégorie syntaxique du mot de l'entrée et la relation entre un mot x de catégorie 'nom' et un mot y de catégorie 'nom'n'est pas la même relation que celle entre un mot x de catégorie 'nom'et un mot y de catégorie 'verbe'.

Par exemple, pour un mot x de catégorie 'adverbe', il pourrait être pertinent de renforcer le lien entre x et y, si y est un adjectif qui apparaît dans le contexte 'd'une manière mot_y'.

Syntaxe
Nous définissons une syntaxe particulière pour représenter les contextes légitimisants. En général ce sont une séquence de mots précédant et/ou suivant le mot y en question, qui signale que ce mot doit être mis en valeur.

— Les contextes précédents et suivants sont des séquences de mots. Ces séquences

peuvent contenir soit des mots-formes, soit des lemmes, ce qui permet plus de flexibilité dans la rédaction des règles et de ne pas devoir expliciter toutes les combinaisons de formes fléchies possibles.
— # représente la position du mot y, ce qui permet de spécifier le contexte précédent et le contexte suivant du mot.
— Les symboles ˆ et $ représentent respectivement le début et la fin de la phrase
— Un chiffre indique le nombre maximum de mots qui peuvent apparaître à une position donnée, sans devoir spécifier la forme ou le lemme de ces mots. Notez que ces mots ne peuvent pas être un signe de ponctuation ou de la même catégorie syntaxique que le mot y. Ceci fait en sorte que ce soit le premier mot de la catégorie recherchée qui est renforcée par ce contexte.

Par exemple, dans l'entrée du mot 'langoureusement', la définition 'd'une manière langoureuse' et le contexte légitimisant 'de une manière #' permettrait de renforcer la relation entre ' langoureusement' et 'langoureux', ce dernier apparaissant dans la position spécifiée par #. Dans cet exemple, il n'y a pas de contexte suivant spécifié (puisqu'aucun mot ne suit le #) et ceci est le cas dans la plupart des contextes, où le contexte précédent est le plus important. Pour simplifier les contextes, nous permettons de ne pas exprimer le # et dans ce cas, la séquence de mots est considérée comme la séquence précédente du mot y. Pour reprendre l'exemple précédent, le contexte 'de une manière #' peut également être exprimé 'de une manière'. Notons que si un contexte suivant est explicité, le # est nécessaire.

Ci-dessous quelques exemples de contextes pour une paire de catégories pour les mots x et y :

POS de x	POS de y	Contexte	Description
Adj	Adj	. # .	Un adjectif entre deux points
Nom	Nom	ˆ # ,	Un nom au début de la définition, suivi d'une virgule
Nom	Nom	synonyme de	Un nom qui suit la séquence 'synonyme de'
Adj	Verbe	en parlant de une personne qui 3	Un verbe qui suit la séquence 'en parlant de une personne qui'avec un maximum de 3 mots intervenants qui ne sont pas de catégorie PONCT ou V
Adj	Nom	qui a des rapports avec 2	Un nom qui suit la séquence 'qui a des rapports avec'avec un maximum de 2 mots intervenants qui ne sont pas de catégorie PONCT ou N

Table 1 – Quelques exemples de contextes légitimisants

Les contextes ont été choisis en analysant les entrées de dictionnaire pour chaque type de catégorie syntaxique et en remarquant les régularités. Quelques contextes sont applicables pour un nombre de catégories différentes, par exemple 'synonyme de', 'hyperonyme de', et d'autres sont plus spécifiques aux contextes.

L'implémentation en Python

La correspondance entre les contextes d'un mot à l'intérieur d'une définition et les contextes légitimisant se fait par des expressions régulières, qui sont générées à partir des contextes

des fichiers et utilisé pour matcher les contextes précédents et suivant.

Chaque contexte est un tuple contenant une expression régulière pour le contexte précédent et une expression régulière pour le contexte suivant.

Les séquences sont transformées de la façon suivante :
- Toute la ligne est divisée en deux sur le symbole #. S'il n'existe pas de symbole #, toute la séquence est le contexte précédent est le contexte suivant est null.
- Pour chaque contexte (précédent et suivant), l'expression régulière est générée :
  - tout caractère qui est un caractère spécial pour les expressions régulières doit être échappé (sauf êt $ qui retiennent leur propriété de caractères spéciaux)
  - tout mot est converti en une entité taggée et lemmatisée tel que le mot peut être considéré comme le mot-forme ou le lemme du mot. Par exemple 'mot'est transformé en ((mot/[^ /]+?/[^ /]+?)—([^ /]+?/[^ /]+?/mot))

[EXAMPLE]

7.2 Modification en fonction de l'emplacement du mot dans une définition

Ce paramètre permet de diminuer le poids des premiers mots d'une définition, en supposant que ce sont les termes les plus importants pour représenter l'entrée. La valeur à rentrer est le nombre de poids au début de la définition qui recevront une valeur diminuée. Le principe est de multiplier chaque arête par une valeur qui commence très petit et augmente jusqu'à la valeur maximale de 2. Cette valeur maximale de 2 sera utilisée pour tous les mots qui suivent ces premiers mots mis en avant. L'accroissement de la valeur multiplicatrice suit une pente avant d'arriver à cette valeur maximale.

Par exemple, si le nombre de mots (n) mis en avant et de cinq, les cinq premiers mots seront multipliés par un plus petit nombre que les mots suivants. Le premier mot sera multiplé par la plus petite valeur, le deuxième mot par la deuxième plus petite valeur et ainsi de suite, jusqu'à la position n+1, où la multiplication reste constante à 2.

[IMAGE]

7.3 Modification en fonction de la catégorie syntaxique recherchée

Une autre option est de donner plus d'importance (en forme d'une valeur diminuée) aux mots qui sont de la même catégorie qu'un mot recherché dans le réseau. Ceci peut être important pour la recherche de synonymes, où nous souhaitons privilégier les relations entre mots de la même catégorie. Cette étape n'est pas implémentée dans la création de la matrice, mais plutôt pendant la recherche du graphe (Voir Parcours en largeur [REF]).

8 L'implementation du graphe

8.1 Choix de langage et de bibliothèques

Nous implémentons le graphe en python, pour lequel il existe de nombreuses bibliothèques (Scipy, NumPy) efficaces qui permettent de manipuler un grand nombre de données numériques. Nous utilisons cElementTree pour traiter les fichiers XML et les SparseMatrices de NumPy pour représenter le graphe. Le réseau est alors une matrice NxN où N est le nombre de nœuds différents dans le graphe. Etant donné le grand nombre d'entrées dans les dictionnaires, il y a un avantage clair d'utiliser les matrices sparses, qui permettent de stocker plus efficacement un graphe qui contient de nombreux sommets et peu d'arêtes.

8.2 La création du graphe

Comme mentionné précedemment, le graphe lui-même est implementé en forme de matrice.

9 Parcours

10 Synonymes

10.1 Programme

10.2 Evaluation

11 Désambiguation

11.1 Programme

La deuxièmme tâche que nous nous sommes donnée est la désambiguisation des mots en contexte. Cela consiste à calculer la distance entre, d'une part, une phrase entrée par l'utilisateur et, d'autre part, chaque définition du mot à disambiguïser. Le programme va alors calculer un vector pour chaque couple (phrase / définition) et la définition renvoyée sera celle ayant la distance la plus petite.

Dans un premier temps on tagge puis lemmatise la phrase et les définitions pour qu'ils soient en adéquation avec nos données.

Ensuite, on peut calculer la distance entre la phrase et une définition, pour se faire, il y a deux étapes : la création des veteurs et le calcul de la distance euclidéenne entre ces vecteurs.

11.2 création des vecteurs

Pour chaque mot (de la phrase ou des définitions), nous récupérons ses k plus proches voisins. Puis nous mémorisons dans un dictionnaire chaque voisin (clef) ainsi que son vecteur dans la matrice (valeur). Ce dictionnaire représente le vecteur de notre mot.

Pour effectuer le calcul du vecteur de la phrase (ou des dictionnaires) nous comparons chaque vecteur-mot au vecteur-phrase. si l'un de ses voisin n'est pas encore voisin de la phrase, on l'ajoute. Sinon on prend le vecteur-voisin le plus petit entre celui du mot et celui déjà présent dans la phrase.

11.3 calcul de la distance euclidienne

Une fois que nous possédons le vecteur-phrase ainsi que les vecteurs-définition, nous calculons la distance euclidienne entre la phrase et chaque définition. Il ne reste plus qu'à récupérer la plus petite distance euclidiene pour connaître la définition prédite.

11.4 Evaluation : RomansEval

L'évaluation de ce programme a été faite grâce au corpus RomansEval. Il a été mis au point dans l'optique d'évaluer, lors d'une compétition, des systèmes de désambiguisation de mots sur des textes littéraires.

Ce corpus est constitué d'environ 45000 phrases. Certains mots présents dans ce corpus sont ambigus. Ces mots sont regroupés dans trois fichiers HTML (un fichier par catégorie syntaxique : Adj, N, V) qui contiennet en plus leurs différents définitions. Et pour finir, ils sont aussi répertoriés dans un fichier de référence avec diverses informations supplémentaires telles que :

1. la catégorie syntaxique du mot
2. son lemme
3. le numero de la phrase dans laquelle il apparait
4. l'occurrence présente dans la phrase
5. les différents votes des personnes qui ont fait ce corpus
6. la définition qui eu le nombre de voix le plus élevé

Pour réaliser notre évaluation, nous avons mis au point un script qui, pour chaque fichier HTML, le lit et récupère les informations pertinentes dans un dictionnaires python. Ensuite le fichier de référence est lu ligne par ligne ce qui permet de connaître chaque occurrence, son contexte et les définitions liées à ce lemme.

Le corpus étant beaucoup trop important, nous n'avons pas pu nous servir de toutes les données qu'il contient. Pour chaque lemme nous avons analysé les 20 premières occurrences pour avoir un score d'exactitude le plus précis possible.

Le score obtenu est un peu différent selon la catégorie observée :

1. pour les adjectifs, le score d'exactitude est de 20
2. pour les noms, le score d'exactitude est de 15
3. pour les verbes, le score d'exactitude est de 25

Nous avons conscience que ces score sont mauvais mais celà peut en partie s'expliquer en étudiant de plus près les définitions. En effet, si l'on regarde par exemple le lemme FRAIS. Il possède, dans le RomansEval, huit définitions différentes. Et parmi ces huit définitions certaines sont très proches et la différence entre elles est difficile à percevoir, même pour un être humain. Par exemple : 'Qui est légèrement froid ou qui procure une sensation de froid léger.' et 'Qui est empreint de froideur, dépourvu de cordialité.' sont deux défintions qui utilisent plusieurs mots identiques ou très similaires. Dans ce cas-là l'être humain comprend bien que la première définition désigne plutôt une sensation physique tandis que la deuxième désigne un sentimentn resenti.

12 Le Bras Droit du Cruciverbiste (i.e. La Résolution de Mots Croisés)

La troisième tâche est la résolution d'indices de mots croisés. Le concept est simple et ressemblent aux tâches précédentes dans son exploitation du réseau sémantique. Tandis que le chercheur de synonymes calcule la distance entre des mots individuels et le désambiguïseur trouve la distance entre deux séquences de mots, le resolveur de mots-croisés cherche la distance entre une séquence de mots (l'indice) et un mot (le mot cible). Comme précédemment, le noyau du programme est donc une recherche de la matrice de relations pour trouver un certain nombre de mots candidats. Par contre, des étapes supplémentaires sont nécessaires pour faire en sorte que les mot cibles adhèrent à certaines contraintes imposées par le jeu.

Le principe des mots croisés est de remplir une grille de lettres en fonction d'indices, de longueur de mots cibles et de caractères qui ont déjà été devinés dans la grille. Notre application est un outil support qui permet de fournir des mots candidats au cruciverbiste [1]. A part ces deux contraintes, il est important que les mots candidats correspondent à la catégorie syntaxique indiquée par l'indice. Notons que pour l'instant l'application ne détecte que les mots des catégories 'nom', 'adj', 'adv' et 'verbe', puisque les autres catégories ne sont pas présentes dans le réseau. Par exemple, pour une indice 'escroqués', les mots candidats doivent être des participes passés qui sont conjugués au masculin pluriel, et pour une indice 'a l'air chouette', les mots candidats doivent être des verbes conjugués au troisème personne singulier du présent indicatif [2].

Le processus suivi est donc séquentiel et peut être représenté par le schéma suivant :

1. Un amateur de mots croisés
2. Les bonnes réponses sont 'eus' et 'ulule' respectivement

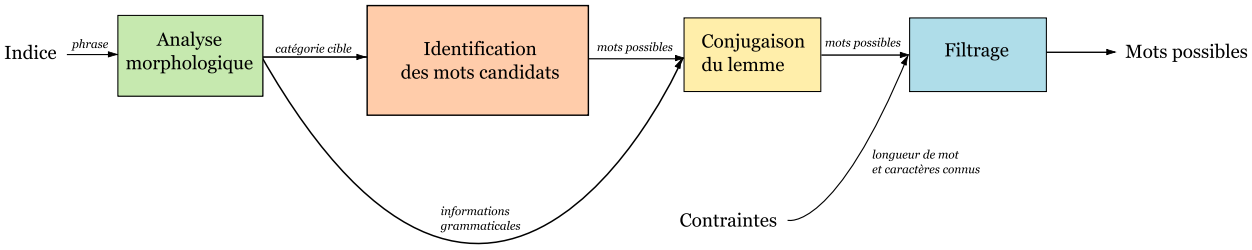

FIGURE 4 – Le schéma du traitement pour la résolution des indices de mots croisés

12.1 Les étapes de traitement

12.1.1 Etape 1 : Identification de la catégorie syntaxique du mot cible

Les indices sont de petites séquences de mots qui sont très rarement des phrases complètes et bien formées. Elles correspondent souvent à un syntagme particulier et contiennent parfois des ambiguïtés de catégorie syntaxique dont il faut tenir compte. Il est aussi possible à cette étape d'identifier quelques marques de flexion qui peuvent indiquer la forme du mot cible. Le réseau sémantique ne contient a priori ques des lemmes et donc il est important de garder ces indices flexionnelles pour re-fléchir les mots candidats à l'étape 3 [REF].

Un étiqueteur syntaxique automatique tel que MElt[4] est peu adapté pour cette identification puisqu'il se base sur les probabilités pour renvoyer la séquence de tags la plus probable pour l'indice. Ceci signifie que le tag prédit peut ne pas correspondre à un des tags possibles pour un mot donné ou les ambiguïtés peuvent ne pas être trouvées. De plus, de tels étiqueteurs sont dépendants de leurs données d'entraînement, et les indices des mots croisés sont d'une forme très différente des données du FTB sur lequel MElt a été entrainé. Nous choissisons donc de faire une analyse point à point des mots de l'indice afin de renvoyer toutes les possibilités catégorielles pour chaque mot. Le Lefff (Lexique des Formes Fléchies du Français)[5] est utilisé pour retrouver toutes les catégories syntaxiques et marques flexionnelles de chaque mot. Cette méthode renvoie évidemment plus de catégories syntaxiques que souhaitées en termes de la séquence de tags pour une indice et crée donc de l'ambiguïté artificielle. Cependant, c'est un moyen d'augmenter le rappel des résultats, même si la précision en souffre. Il est important pour nous de renvoyer le plus de mots candidats possibles et une erreur d'étiquetage syntaxique à cette étape peut significativement nuire les chances de trouver les bons candidats par la suite.

La catégorie syntaxique du mot cible peut être identifiée à partir de la séquence d'ensembles de catégories grâce à la régularité des formes des indices. Si une indice contient un seul mot, les catégories cibles sont celles du mot de l'indice. Sinon, la catégorie dépend en grande partie des premiers mots de l'indice. Par exemple, si une indice commence par un nom, ou une séquence 'det nom' ou une séquence 'det adj nom', la catégorie cible, étant donné qu'il n'y a pas d'ambiguïté catégorielle serait, serait un nom. Dans certains cas, il est aussi possible d'identifier quel mot de l'indice indique la conjugaison du mot cible. Cette identification se fait à l'aide d'une mini-grammaire de seulement 18 règles, qui sont reproduites ci-dessous :

```
det nc ->2, nc
det adj nc ->3, nc
adj nc ->2, nc
```

```
nc ->1, nc
adj prep ->1, adj
adv v ->2, v
advm v ->2, v
advneg v ->2, v
advneg advneg v ->3, v
v ->1, v
clr v ->2, v
prel v ->2, adj
cln v ->2, n
prep nc -> ?, adj
ce v det nc ->4, nc
adj coo adj ->1, adj
nc coo nc ->1, adj
pri v ->2, adj
```

Chaque ligne représente une règle différente. A gauche de la flèche est la séquence de mots qui indique, au début d'une indice, une certaine catégorie syntaxique pour le mot cible. A droite de la flèche est l'indice (à partir de 1) du mot de la partie gauche dont les informations grammaticales seront utilisées pour fléchir les mots cibles (qui seront des lemmes), suivi de la catégorie syntaxique du mot cible. Dans le cas où aucun mot de l'indice en particulier donne des indices flexionnelles, ' ?' indique que les indices flexionnelles sont inconnus. Par exemple, une indice telle que 'avec soin' correspond à la règle 'prep nc -> ?, adj', où la catégorie cible serait 'adv', mais où aucun mot entre la préposition et le nom fournit d'indice flexionnelles.

Le jeu d'étiquettes utilisées est celle du Lefff[3], qui est utilisée pour retrouver les formes fléchies. Une même indice peut correspondre à plusieurs règles différentes si un mot donné est ambigu pour la catégorie syntaxique.

Les catégories syntaxiques sont retenues pour le parcours du réseau et les informations grammaticales pour retrouver les formes fléchies correspondant aux lemmes trouvés dans le réseau.

Remarque : Une particularité des indices est que parfois elles peuvent contenir plusieurs parties, séparées d'une virgule et qui représentent plusieurs sous-indices à l'intérieur de l'indice. Par exemple, une indice peut avoir la forme 'habillée, fringuée' et dans ce cas, il est souhaitable de séparer cette indice en 'habillée' et 'fringuée' afin d'avoir la meilleure chance de trouver la catégorie voulue. Idéalement, ces deux parties devraient aussi correspondre en termes de catégorie syntaxique, ce qui signifie qu'il faudrait en théorie ne garder que les catégories cibles qui correspondent aux deux parties de l'indice. Par contre, nous optons pour le choix de garder toutes les catégories potentielles à partir des deux parties de l'indice pour nous assurer de la couverture maximale.

3. Ce jeu a été réduit pour ne contenir que les noms, adjectifs, adverbes et verbes et les catégories nécessaire pour la mini-grammaire

12.1.2 Etape 2 : Recherche du graphe

Une fois que la catégorie syntaxique de la solution est ciblée (ou au moins réduite à une liste de catégories possibles), et les mots de l'indice sont lemmatisés, il est possible d'utiliser le réseau sémantique pour trouver les mots les plus proches à l'indice. Nous testons les deux stratégies de recherche du graphe : l'algorithme de parcours en largeur décrit dans la partie [REF] pour renvoyer les k plus proches voisins de chaque lemme, et la matrice complète détaillé dans la section [REF]. Le choix des plus proches candidats va dépendre de l'algorithme utilisé.

Une indice peut contenir plusieurs lemmes et avec le réseau sémantique, il est facile de renvoyer les voisins (avec leurs distances) d'un mot individuel. Pour le parcours en largeur il s'agit de parcourir le graphe jusquà'ce que les k plus proches voisins sont trouvés. Dans la matrice complète il est possible d'atteindre la liste complète des sommets du réseau avec leurs distances du mot recherché.

Les voisins d'un lemme donné peuvent être vu comme un vecteur où les paramètres sont les voisins et les valeurs sont leurs distances du lemme. Par exemple une indice 'lemme1 lemme2 lemme3' aurait trois vecteurs différents pour représenter chacun des lemmes, chacun avec un certain nombre de paramètres spécifiés (en fonction de l'algorithme de recherche utilisé), qui ne seront pas forcément les mêmes pour chaque vecteur (si seulement les k plus proches sont trouvés) :

vec_lemme1 : <a : 1, b : 2, c : 3>
vec_lemme2 : <a : 1, b : 5, d : 4>
vec_lemme3 : <a : 4, c : 1, e : 1>

Le vecteur global doit être représentatif des trois lemmes et des paramètres que contiennent leurs vecteurs. Si la matrice complète est utilisée, les vecteurs des lemmes contiendraient les mêmes voisins et un moyennage des distances est possible pour chaque voisin. Par contre, un moyennage simple des valeurs n'est pas possible pour le parcours en largeur, puisque les voisins non-représentés auraient une valeur infinie pour la distance, ce qui fausserait les distances moyennées. Le choix dans ce cas-là est donc d'ajouter les vecteurs pour en faire un vecteur global, et dans le cas où un paramètre apparaît dans plusieurs vecteurs, comme ci-dessus avec 'a', 'b' et 'c', la distance minimale est choisie. Les trois vecteurs ci-dessus aurait alors comme vecteur global :

vec_indice : <a : 1, b : 2, c : 1, d : 4, e : 1>

Remarques sur les formes recherchées : les lemmes des mots de l'indice sont rechechés dans le réseau, suffixés de leurs catégories syntaxiques possibles, ce qui signfie qu'un mot qui est syntaxiquement ambigu sera recherché plusieurs fois. Si le lemme ne se trouve pas dans le Lefff, au lieu d'abandonner la recherche de ce mot, nous utilisons le mot-forme par défaut, en espérant que ce mot appartient au réseau (particulièrement important pour les noms propres par exemple). [4]. Les catégories syntaxiques cibles sont celles obtenues à partir de la mini-

4. Même si le réseau ne devraient pas contenir de formes fléchies, il est possible qu'il en contient à cause des erreurs de tagging, et donc nous exploitons cette faiblesse ici pour augmenter la chance de retrouver le plus de mots de l'indice possibles.

grammaire est en cas de non-identification de catégorie, des candidats de toute catégorie seront rechechés.

12.1.3 Etape 3 : Conjugaison des lemmes obtenus

Les résultats de la recherche du graphe sont (pour la plupart) des lemmes, qui sont à modifier pour correspondre à la forme du mot recherché. En utilisant l'ensemble de catégories et d'informations grammaticales obtenues lors de l'étape 1 [REF], cette étape cherche les formes fléchies correspondantes à chaque lemme et vérifie lesquelles correspondent aux informations grammaticales recherchées pour les mots candidats.

La ressource utilisée est encore une fois le Lefff, avec pour chaque lemme qui y apparaît, ces catégories syntaxiques possibles et les formes fléchies associées. En sachant les catégories syntaxiques possibles pour le mot cible (obtenues à l'étape 1), les formes fléchies du lemme sont triées en fonction de leur correspondance aux informations grammaticales aussi obtenues à l'étape 1. Si un trait grammatical (ex : 'nombre') est exprimé pour les flexions voulues, la forme fléchie sera retenue seulement si elle a la même valeur de trait (pourvu que ce trait soit exprimé).

Par exemple, si l'indice est 'mangée', les informations grammaticales retenues sont que la catégorie est une participe passée et que le mot recherché doit être féminin singulier. Les voisins du lemme 'manger' seront recherchés dans le réseau et renvoyés :

consommer, alimenter, boire etc.

Il existe un grand nombre de formes fléchies de ces verbes, mais dont un petit nombre correspondent aux contraintes grammaticales 'féminin singulier'. Les formes suivantes seront alors rejetées :

consommâtes, consommèrent, consommé, consomme, ..., alimentez, alimentées, etc.

et les formes suivantes retenues :

consommée, alimentée, bue

12.1.4 Etape 4 : Filtrage selon les contraintes

La dernière étape filtre les candidates conjugués afin de ne renvoyer que les candidats qui se conforment aux contraintes de longueur de mot et de caractères déjà devinés. Il n'est pas possible de faire cette étape avant la conjugaison, puisque la conjugaison des lemmes peut avoir pour effet un changement dans la longueur du mot et des caractères qui y apparaîssent.

Souvent dans les outils d'aide de résolution de mots fléchés, cette étape est la seule incluse dans le programme, et donc un grand nombre de mots correspondent aux contraintes de longueur et de caractères. Dans notre programme, cette étape sert à enlever la grande

quantité de bruit dans les résultats produits directement à partir du réseau, et permet de raffiner en grande partie les résultats. L'avantage d'utiliser le réseau et qu'il y a plus de chance d'avoir des résultats qui correspondent à l'indice donnée et non pas des mots candidats indépendant de l'indice, ce qui permet un traitement plus sophistiqué à cette étape.

12.2 L'interface graphique

L'interface graphique, implementée en python en utilisant les bibliothèques Tkinter [REF] et PMW [REF] sert à visualiser les résultats renvoyés aux différentes étapes du processus. Elle est mi-chemin entre un outil pour un joueur de mots croisés, qui ne s'intéresse que aux mots qui se conforment aux contraintes données et un outil plus pédagogue pour montrer les étapes de traitement.

1. L'indice qui sert à deviner le mot recherché
2. Le nombre de lettres du mot recherché. En cliquant sur 'Préparer les cases', cette contrainte sera enregistrée et les cases générées en 3.
3. Les cases pour entrer les caractères déjà devinés. Cette contrainte sera prise en compte dans le filtrage des mots candidats
4. Le log qui permet de tracer le suivi du programme. Ici seront affichés des avertissements dans le cas où un mot n'est pas reconnu, la/les catégorie(s) syntaxique(s) du mot recherché et le nombre de résultats renvoyés à chaque tour.
5. Les plus proches voisins de l'indice. Au départ les vingt mots les plus proches seront retournés.

6. Les formes fléchies des lemmes renvoyés en 5., qui devraient correspondre aux contraintes flexionnelles imposées par les mots de l'indice, si elles sont reconnues
7. Les candidats filtrés selon les contraintes de longueur de mot et de caractères déjà devinés.
8. Si l'utilisateur veut continuer en regardant les vingt prochains plus proches mots, il peut cliquer sur ce bouton pour renvoyer plus de résultats. Ceci peut être utile dans le cas où peu de résultats fléchis ou filtrés sont renvoyés.

12.3 Evaluation

Le système est difficilement évaluable contre d'autre systèmes de la même nature, parce qu'il n'existe pas de système d'évaluation standard et de scores comparatifs. Il existe beaucoup de systèmes qui ne se basent pas sur la similarité sémantique, mais qui renvoient plutôt tous les mots qui correspndent aux contraintes de longueur et de caractères déjà devinés, et ces systèmes ne fournissent pas un bon moyen de comparer l'approche par réseau sémantique, qui fournit moins de formes globalement.

Notre évaluation est donc une évaluation simple de la capacité du réseau de trouver la solution d'une indice, et qui compare le taux de réussite pour différents niveaux de difficulté et pour différentes catégories syntaxiques. L'évaluation se fait sur un petit corpus d'indices pris d'un site internet de mots fléchés [REF http ://www.notretemps.com/jeux/jeux-en-ligne/mots-fleches-gratuits], qui classe les grilles par niveaux. Le corpus se trouve dans l'annexe [REF - put in annexe]. Il est vrai que l'emplacement des mots dans la grille les uns par rapport aux autres contribue au niveau global de la grille, mais nous estimons que le niveau est aussi reflété dans la difficulté des indices individuelles. Le corpus contient deux niveaux différents (1 et 2) et pour chaque niveau vingt noms communs, vingt noms propres, vingt adjectifs et vingt verbes. Les indices étaient prises aussi objectivement que possible, avec soin de ne pas répéter le même mot cible entre deux indices, ou de prendre exclusivement des indices qui contiennent une seule tournure. Autant de solutions de chaque catégorie ont été sélectionnées afin de comparer le taux de réussite pour chaque catégorie, mais le constat a été que le niveau contient plus de noms communs et moins de nom propres, ce qui influence aussi la difficulté des indices.

A part le but principal de trier les mots candidats par rapport à la similarité sémantique, les choix de traitement sont clairement orientés vers une optimisation du rappel (par exemple, le choix de faire une analyse morphologique point à point). Le système d'évaluation se juge donc sur le rappel des résultats. Pour chaque indice, nous cherchons d'abord si la solution apparaît dans le réseau et si oui, après combien de voisins les plus proches le mot est trouvé. Pour éviter de parcourir tout le réseau en cas de non-découverte, nous limitons le maximum de voisins renvoyés à 200. Au délà de 200, le mot est considéré non-trouvé. Une analyse détaillée se trouve ci-dessous :

			Solution trouvée Après					
Level	POS	#	40	80	120	160	200	Non-trouvée
1	Adj							
1	NC							
1	NP							
1	Adv							
1	V							
2	Adj							
2	NC							
2	NP							
2	Adv							
2	V							

TABLE 2 – Résultats de la recherche pour chaque niveau et chaque catégorie syntaxique

12.4 Améliorations possibles

13 Conclusion

A Parametres des poids

B Parametres supplémentaires pour la ponderation des aretes

Text of Appendix B is Here

C Script de normalisation du Littre

D Script de normalisation du Wiktionnaire

Text of Appendix B is Here

E DTD des fichiers XML produits

Text of Appendix B is Here

Level x	POS	Nombre	POS identifié	Solution dans le réseau	Solution dans le Lefff	Pas de voisins
1	Adj					
1	NC					
1	NP					
1	Adv					
1	Verbe					
2	Adj					
2	NC					
2	NP					
2	Adv					
2	Verbe					

TABLE 3 – Analyse du processus

F Script utilise pour l'etiquetage syntaxique et la lemmatisation

G Script pour la creation de la matrice

Text of Appendix B is Here

Références

[1] Princeton University "About WordNet." WordNet. Princeton University. 2010. http ://-wordnet.princeton.edu

[2] WiktionaryX. http ://redac.univ-tlse2.fr/lexiques/wiktionaryx.html. Franck Sajous. CLLE-ERSS.

[3] Littré, Émile. Dictionnaire de la langue française, Supplément. Paris, L. Hachette, 1878. Electronic version created by François Gannaz. http ://www.littre.org

[4] Pascal Denis and Benoît Sagot (2012). Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging. In Language Resources and Evaluation 46 :4, pp. 721-736, DOI 10.1007/s10579-012-9193-0

[5] Sagot (2010). The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French. In Proceedings of the 7th international conference on Language Resources and Evaluation (LREC 2010), Istanbul, Turkey

[6] Olney J., Revard C. et Ziff P. (1968) Toward the development of computational aids for obtaining a formal semantic description of English. System Development Corporation, Santa Monica

[7] Mikolov T., Chen K., Corrado G. et Dean J. (2013) Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.

[8] Gaume B., Hathout N. et Muller P. (2004) Word Sense Disambiguation using a dictionary for sense similarity measure. In Proceedings of the 20th international conference on Computational Linguistics.

[9] Muller P., Hathout N. et Gaume B. (2006) Synonym extraction using a semantic distance on a dictionary. In Proceding of the First Workshop on Graph Based Methods for Natural Language Processing.

[10] Travers J. et Milgram S. (1969) An Experimental Study of the Small World Problem. In Sociometry, Vol 32, No. 4, pp. 425-443

[11] Watts D. et Strogatz S. (1998) Collective dynamics of 'small-world' networks In Nature Vol 393 (6684), pp. 440-442

[12] Véronis J. (2004) HyperLex : lexical cartography for information retrieval. In Computer Speech and Language 18(3) pp 223-252.

[13] Barabási A. et Albert R. (1999) Emergence of scaling in random networks. In Science 286 (5439) pp. 509-512