

Université de Montréal  
IUP GMI d'Avignon

Prise en compte d'informations  
sémantiques dans un système de traduction  
phrase-based.

Par  
Anne Laure Pelurson

Département d'informatique et de recherche opérationnelle  
Faculté des arts et des sciences

Recherche appliquée en linguistique informatique

2006

# Remerciements

Je remercie tout d'abord Philippe Langlais, mon tuteur, de m'avoir accueilli au sein du RALI, et ainsi m'offrir la possibilité de travailler dans un laboratoire de linguistique informatique de renommée mondiale. Il a toujours été disponible, malgré de nombreuses occupations, pour m'aider dans mon travail par ses conseils.

Je remercie également Elliott Macklovitch et les membres de RALI pour le climat professionnel et agréable, et plus particulièrement Fabrizio Gotti qui a toujours répondu à mes questions.

Enfin je remercie ma famille et mes amis de France et de Montréal qui m'ont permis de passer une année inoubliable à Montréal.

# Résumé

La traduction automatique est actuellement en plein essor dû aux besoins de communication ouverts par les nouvelles technologies. Notre travail s'intéresse essentiellement aux systèmes probabilistes (Statistical Machine Translation en anglais). La SMT repose essentiellement sur l'apprentissage de paramètres de différents modèles à partir d'une grande quantité de textes bilingues (corpus d'entraînement). Ces modèles ne prennent en compte que les mots eux-mêmes sans jamais s'intéresser à leur sens.

L'information sémantique permet de connaître une information permettant de trouver le sens d'un mot. Cette information permet dans de nombreux cas de lever des ambiguïtés et ainsi améliorer les traitements de la langue naturelle. Elle peut être retrouvée de différentes façons plus ou moins efficaces selon les cas. La plus connue reste la méthode de désambiguïsation qui consiste à déterminer directement pour un mot son sens dans le texte.

Nous allons donc dans notre étude essayer d'ajouter de l'information sémantique dans le traitement de traduction automatique pour améliorer les résultats. Nous avons procédé en deux phases : une première partie essaye une amélioration avec un traitement avant la construction du traducteur et une deuxième partie fait un traitement pendant la construction du traducteur.

Mots clés : traduction automatique, information sémantique, Wordnet.

# Abstract

Nowadays Machine Translation is getting more and more important with the rapid development of new technologies. The work here is only dealing with Statistical Machine Translation. The principle of SMT is to learn different model parameters from an important quantity of bilingual texts (training corpus). This model only deals with the word in itself, and never with its sense.

Semantic information gives the necessary information allowing finding the correct sense of an ambiguous word in a given context. In many cases, this information allows to remove ambiguity and so improve Natural Language Processing. It can be found in different more or less effective ways according to case. The best known way is the disambiguation method that can be defined as the task of detecting directly the correct sense of an ambiguous word.

In the present study we will try to add semantic information in the processing of automatic translation to improve results. We will proceed in two phases: in a first part, we will make a treatment before the construction of the translator and in a second part, we will make a treatment during the construction.

Keywords : machine translation, semantic information, Wordnet.

# SOMMAIRE

I. Introduction .....	1
A. Historique.....	1
B. L'importance applicative.....	2
C. Notre travail .....	3
II. Traduction automatique.....	5
A. Le bitexte.....	6
B. Modèle de langue (LM) .....	8
C. Modèle de traduction .....	10
1. Modèle de traduction basé sur les mots .....	10
2. Modèle de traduction basé sur les segments de phrases (PBM ou modèle phrase-based) .....	11
D. Décodage.....	12
E. Tunning .....	13
F. Evaluation .....	13
III. Désambiguïisation .....	16
A. Les ressources .....	16
3. WordNet.....	17
4. Sencor.....	20
B. Etiqueteur morphosyntaxique (POS).....	23
1. BRILL .....	24
2. Tree Taggers.....	25
3. Evaluation.....	25
C. Algorithme de désambiguïisation .....	28
4. Sens le plus fréquent .....	29
5. SenseLearner .....	30
6. Naïve Bayes.....	32

7. Evaluation.....	33
<b>IV. Enrichissement du corpus d'apprentissage.....</b>	<b>35</b>
A. Création des paraphrases.....	35
B. Evaluation .....	40
1. Test sur les modèles de langue.....	40
2. Test sur la traduction.....	46
<b>V. Ajout de score.....</b>	<b>56</b>
A. Calcul du nouveau score .....	56
B. Evaluation .....	57
1. Désambiguïsation sur le segment anglais du PBM réduit.....	57
2. Désambiguïsation sur le segment anglais du PBM .....	59
3. Désambiguïsation sur le corpus original .....	60
<b>VI. Conclusion.....</b>	<b>61</b>
A. Bilan de l'étude .....	61
B. Bilan personnel .....	62

## I. Introduction

La traduction automatique (T.A.), connu sous le terme de "*Machine Translation*" (M.T.) dans la littérature anglophone, désigne au sens strict le fait de traduire entièrement un texte grâce à un ou plusieurs programmes informatiques, sans qu'aucun traducteur humain n'ait à intervenir dans le processus. La mondialisation du commerce a eu des effets considérables sur l'essor de l'industrie de la langue, et plus particulièrement en traduction où la demande ne cesse de croître à cause du besoin de communiquer rapidement dans toutes les langues.

### A. Historique<sup>1</sup>

La plupart des grands projets de traduction automatique sont nés entre 1958 et 1966 des besoins de traduction du russe engendrés par la guerre froide. Il existe deux générations de systèmes de traduction :

- La première génération de programmes est ce qu'on appelle des systèmes directs. Ils se basent sur des équivalences de termes, traduisent mot à mot à partir de la consultation d'un dictionnaire et ne font aucune analyse. Ce sont des systèmes bilingues (ils traitent une seule paire de langues) et unidirectionnels. Ces systèmes sont limités, mais peuvent s'avérer utiles dans certains cas d'application restreinte (avec un vocabulaire limité).

- La deuxième génération de programmes de traduction regroupe les systèmes de traduction automatique et les systèmes de transfert. Ces programmes de deuxième génération ont un principe plus complexe que celui des systèmes directs. Les systèmes de transfert sont basés sur trois modules : l'analyse du texte en langue source, le transfert, et la génération dans la langue cible. Actuellement, les systèmes à architecture basée sur le transfert sont les plus couramment utilisés. Ils permettent plus facilement d'intégrer une nouvelle langue que les systèmes directs, pour lesquels ajouter une langue revient à créer un nouveau système.

Systran (utilisé entre autres par la Foreign Technology Division de l'armée de l'air américaine) est le système de transfert le plus connu du grand public (c'est le système auquel donne accès *voilà*<sup>2</sup>). Il est basé principalement sur la consultation de grands dictionnaires bilingues à grande couverture et mis au point à grands renforts des ressources humaines.

La recherche en traduction automatique a été freinée vers 1966, suite à la sortie du rapport ALPAC de la National Science Foundation qui concluait à l'impossibilité d'une traduction automatique de qualité.

Cependant le Canada, en raison de sa politique bilingue a connu une activité faste en traduction automatique. METEO est un système de traduction automatique parmi les premiers systèmes de deuxième génération, il est entré en exploitation le 24 mai 1977 à Montréal. C'est un système très spécifique qui traduit toutes les prévisions météorologiques destinées au

---

<sup>1</sup> Extraits pris du site du *Centre Pluridisciplinaire de Sémio linguistique Textuelle*, Université Toulouse-Le Mirail <http://www.univ-tlse2.fr/gril/>

<sup>2</sup> <http://trans.voila.fr/>

grand public, émises par le service d'environnement atmosphérique du Canada. Une mise à jour de ce système est encore utilisée quotidiennement à cette tâche.

Le milieu des années 1980 a été le témoin de l'essor des mémoires de traduction. Une mémoire de traduction est une base de données contenant un grand nombre de traductions existantes. L'idée des mémoires est de fournir automatiquement à un traducteur des traductions déjà faites à des phrases présentes dans la base. Les mémoires sont de plus souvent capables de proposer des suggestions pour des phrases proches de celles disponibles dans la base. Il existe plusieurs systèmes commerciaux (TransSearch<sup>3</sup>, EUROLANG<sup>4</sup>...) exploitant cette idée qui est très populaire chez les traducteurs professionnels.

Jusqu'à la fin des années quatre-vingt le cadre dominant a été l'approche basée sur les règles linguistiques, mais depuis 1990 ce cadre a été rompu par l'entrée en scène de méthodes et de stratégies nouvelles. Un projet important de IBM a donné naissance à un prototype de traduction CANDIDE<sup>5</sup> (Berger et. al., 1994) introduisant l'approche probabiliste au sein de la communauté linguistique. L'équipe d'IBM a publié les résultats de ses expériences avec un système de traduction purement statistique. La caractéristique principale de cette approche est d'analyser une grande quantité de textes parallèles qui sont les traductions l'un de l'autre (*bitextes*) et d'inférer à partir de ces textes les paramètres d'un modèle probabiliste de manière automatique.

### B. L'importance applicative.

Avec le développement des outils de communication et surtout internet, les usagers veulent de plus en plus avoir accès à l'information rapidement et ce quelle qu'en soit la langue. De plus nous voyons d'après le résultat de l'étude réalisée par Funredes<sup>6</sup> se trouvant dans le tableau 1.1, que les langues sur le web se diversifient et l'anglais perd peu à peu son monopole. En effet l'anglais est passé à une présence de 75% à 45%, au profit des autres langues qui ont doublé leur présence. L'accès aux nouvelles technologies a permis une multiplication de l'information mais aussi de leur source. L'accès à toute l'information ne peut se faire alors que si l'utilisateur a une connaissance de toutes ces langues. C'est dans des cas comme celui-ci, que l'utilité de la traduction automatique se fait ressentir et prend toute sa valeur. Elle nous permet de comprendre sans aucune aide extérieure, certes pas dans le détail, l'information contenue dans un texte d'une langue qui n'est pas maîtrisée. Un logiciel de traduction automatique permet donc d'avoir une autonomie accrue non négligeable dans l'accès à l'information et la communication.

---

<sup>3</sup> <http://www.tsrali.com/>. TransSearch, système conçu au sein de Laboratoire RALI à l'université de Montréal.

<sup>4</sup> the European Languages Centre, <http://www.eurolang.com>

<sup>5</sup> <http://www.itu.int/mls/briefingpaper/wipo/french/annexeI-fr.html>

<sup>6</sup> <http://funredes.org/lc/francais/medidas/detalle.htm>

	% ABSOLUE PAGES WEB PAR LANGUE							
	anglais	espagnol	français	italien	portugais	roumain	allemand	autres
<b>09-98</b>	75,00%	2,53%	2,81%	1,50%	0,82%	0,15%	3,75%	13,44%
<b>08-00</b>	60,00%	5,05%	4,40%	2,76%	2,37%	0,22%	3,00%	22,20%
<b>01-août</b>	55,00%	5,20%	4,34%	2,71%	2,44%	0,18%	6,29%	25,45%
<b>02-févr</b>	50,00%	5,80%	4,80%	3,26%	2,81%	0,17%	7,21%	25,97 %
<b>03-févr</b>	49,00%	5,31%	4,32%	2,59%	2,23%	0,11%	6,80%	29,65 %
<b>04-mai</b>	46,30%	4,72%	4,93%	2,85%	1,86%	0,14%	7,12%	32,09 %
<b>05-mars</b>	<b>45,00%</b>	<b>4,60%</b>	<b>4,95%</b>	<b>3,05%</b>	<b>1,87%</b>	<b>0,17%</b>	<b>6,94%</b>	<b>33,43 %</b>

Tab 1.1 : La répartition des pages web par langues (Funredes)

### C. Notre travail

Un logiciel de traduction est donc une application informatique qui permet d'obtenir de façon automatique une traduction de tout type de textes d'une langue source vers une autre langue cible. Le texte traduit doit être autant que possible correct dans la langue cible et restituer l'information contenue dans le texte d'origine. La traduction automatique est encore aujourd'hui très imparfaite et loin d'être aussi bonne qu'une traduction humaine, l'améliorer est un long travail. En effet la traduction est un système lié principalement à la linguistique et donc contraint aux difficultés s'y rapportant :

- Ambiguïtés grammaticales : un même mot peut être de deux catégories grammaticales différentes.
- Ambiguïtés sémantiques : selon le contexte, la traduction peut être différente.

Lors de notre étude réalisée au laboratoire de Recherche appliquée en linguistique informatique (RALI), nous avons essayé d'améliorer la traduction en utilisant de l'information sémantique.

Nous allons dans une première partie nous intéresser aux corpus utilisés pour construire notre traducteur automatique. Nous utilisons ces corpus pour construire des modèles. Ces modèles nous permettent d'avoir les caractéristiques d'une langue et d'avoir les possibilités de traduction d'une langue à l'autre. La qualité de ces modèles est directement liée à la taille et diversité de ces corpus. Ainsi plus ils sont grands plus nos modèles seront meilleurs. Nous allons plus particulièrement nous intéresser au modèle de traduction qui permet d'avoir les différentes traductions connues. Plus le corpus d'apprentissage de modèles est grand plus nous aurons de possibilités de traductions différentes alors plus de possibilité de contenir la bonne traduction. Nous allons donc enrichir le corpus d'entraînement par des paraphrases qui permettent de garder le même sens général en ajoutant des mots différents. Pour l'ajout de ces paraphrases nous aurons bien sûr besoin d'informations sémantiques qui nous permettront de garder le même sens entre phrases et paraphrases.

Dans une seconde partie, nous allons directement travailler sur les modèles de traduction. Comme il sera explicité dans la section II, le modèle de traduction contient un ou plusieurs scores de probabilité pour la traduction d'un mot ou groupe de mots d'une langue

## Chapitre 1 – Introduction

alpha dans une langue beta. Nous allons ajouter un nouveau score qui permettra de prendre en compte de nouvelles informations. Nos nouveaux scores prendront en compte un certain nombre d'informations sémantiques pour leur calcul pour ainsi mettre en évidence des mots ou groupe de mots ayant une forte ressemblance.

La base de notre étude est l'information sémantique. Cette information n'est bien sûr pas simple à trouver. Nous allons pour cela utiliser la désambiguïsation qui nous permet de récolter assez d'informations. Mais nous allons tout d'abord expliquer le fonctionnement et la construction d'un traducteur automatique.

## II. Traduction automatique

Il existe aujourd'hui deux grands types de méthodes, la méthode empirique et la méthode statistique. Ces deux méthodes peuvent être utilisées individuellement ou bien être combinées ensemble.

La première méthode est fondée sur une base de connaissance de style dictionnaire et d'une multitude de règles. Nous avons donc une base de données avec pour les mots de la langue source et son mot correspondant dans la langue cible qui permet de faire une première traduction grossière. Ensuite une série de règles est appliquée pour corriger les fautes syntaxiques. Un exemple de règle peut être la négation "ne pas" qui entoure le verbe en français. Cette méthode est assez longue à mettre en place car il faut créer le "dictionnaire" et les règles, ce qui demande beaucoup de travail. Cette approche reste quand même la plus utilisée pour la plupart des systèmes de traduction disponible sur le web dont babelfish<sup>7</sup>.

La deuxième est la traduction statistique ou probabiliste qui est basée essentiellement sur les probabilités et que nous allons utiliser dans notre étude. La traduction statistique (ou traduction probabiliste) est une approche proposée dans les années 90 qui se base sur le modèle du canal bruité de Shannon. L'idée est de voir le problème de la traduction comme un problème de communication : la phrase source que nous voulons traduire peut être vue comme un message qui serait arrivé erroné et que nous souhaitons corriger. Nous voulons retrouver le sens du message, c'est-à-dire sa traduction dans notre langue cible. Dans notre cas, nous traduisons des phrases de l'anglais vers le français, nous avons donc la maximisation de la probabilité  $\operatorname{argmax}_{f \in \mathcal{F}} P(f|e)$  (Brown et al., 1993) équivalent à  $\operatorname{argmax}_{f \in \mathcal{F}} P(e|f) \cdot P(f)$ , où  $f$  est la traduction recherchée,  $\mathcal{F}$  est l'ensemble des phrases françaises possibles et  $e$  est la phrase anglaise reçue. Cette équation peut être décomposée en trois composantes :

1. le modèle de langue  $P(f)$  qui nous donne la probabilité qu'une phrase soit du français correct.

2. le modèle de traduction  $P(e|f)$  qui nous donne la probabilité qu'une phrase soit la traduction anglaise d'une phrase française donnée. Notons que la probabilité donnée par le modèle de traduction est la probabilité d'une phrase anglaise étant donnée une phrase française. Bien que nous traduisons de l'anglais vers le français, la phrase source du modèle de traduction est donc en français et sa phrase cible en anglais.

3. le décodeur (représenté par la fonction  $\operatorname{argmax}$  dans l'équation) qui trouve la meilleure traduction, c'est-à-dire la phrase qui maximise le produit des probabilités données par les deux modèles.

Nous avons besoin de ressources pour construire le modèle de langue et le modèle de traduction. Le premier sera construit à partir de corpus unilingues, nous n'aurons donc pas

---

<sup>7</sup> <http://www.babelfish.org/>

beaucoup de difficulté à en trouver. Le deuxième, en revanche, à besoin de *bitextes*, dont nous expliquerons les caractéristiques plus loin, pour sa construction.

Nous décrivons le modèle de langue et le modèle de traduction que nous allons utiliser ainsi que le décodeur. Nous avons utilisé pour ce faire beaucoup de ressources déjà existantes car le but de notre recherche n'est pas de développer ces outils mais de faire en sorte d'améliorer ces outils. Ainsi nous avons pu concentrer nos efforts sur l'amélioration de la traduction.

### A. Le bitexte

Nous avons besoin de bitexte pour la création de nos modèles. Le concept de bitexte est dû à (Melby, 1981) qui pense à sauvegarder des textes sources et leurs traductions à des fins pédagogiques. Mais le terme "bitexte" a quand à lui été inventé par (Harris, 1988a, 1988b), professeur de traduction.

Un bitexte est constitué de deux documents parallèles où les alignements, relations de traduction, sont explicitement marqués habituellement au niveau de la phrase. De plus deux documents sont parallèles s'ils véhiculent le même contenu dans le même ordre. Il existe quelques corpus parallèles comme :

- les débats parlementaires canadiens (français, anglais, inuktitut)
- les débats parlementaires de Hong-Kong (anglais, chinois)
- les débats parlementaires européens (français, italien, espagnol, portugais, anglais, allemand, hollandais, danois, suédois, grecque, finnois)
- la bible (nouveau testament ~140 K tokens en Grec ; bible ~30 K types), Coran, charte des droits de l'homme (petit), Harry Potter,
- etc.

Mais ces corpus restent encore peu nombreux et pas forcément adaptés à l'entraînement de modèles de traduction. Il reste donc internet comme base de données de ces corpus. D'après une étude de (Ma & Liberman, 1997), 1 site sur 1000 est bi- ou multi-lingues alors que 1 site sur 10 dans le domaine ".de" (allemand) l'est. Il existe donc une réelle base de corpus parallèles sur internet, mais il reste encore de nombreuses faiblesses dans cette base. En effet nous ne pouvons garantir la bonne traduction des pages, surtout lorsque celles-ci font partie de sites non officiels. De plus les mises à jour des sites peuvent être incomplètes en modifiant, par exemple, que la page d'une langue et pas l'autre. Les deux pages ne seront plus parallèles.

Les corpus parallèles peuvent être transformés en bitexte en alignant les phrases des deux textes. Certains aligneurs sont disponibles : (Gale & Church, 1993) ou (Moore, 2002). Ainsi nous pouvons facilement et efficacement produire les bitextes à partir de corpus parallèles. Une fois tous ces traitements effectués nous avons toutes les ressources nécessaires pour construire nos modèles de traductions ce qui est l'étape suivante. Nous aurons donc au

final deux fichiers textes de langues différentes alignés de telle façon que la i-ième phrase de premier corresponde à la i-ième phrase du second.

**Corpus parallèle :**

Monsieur le Président, nous suivons depuis quelques semaines le déroulement des élections à Madagascar. Malgré un premier tour de scrutin qui n'a pas donné de résultat concluant, le candidat de l'opposition s'est déclaré vainqueur et a depuis enjoint ses partisans à la grève générale et aux manifestations. Aux dernières nouvelles, il y a maintenant deux gouvernements parallèles et deux capitales.	Mr. Speaker, over the past few weeks, we have been following the general election in Madagascar. In spite of an inconclusive first ballot, the opposition candidate declared himself the winner and then enjoined his supporters to go on a general strike and to organize protests. According to the latest news, there are now two parallel governments and two capitals.
---	---

Fig 2.1 : Corpus parallèle, "Le bitexte et ses applications"<sup>8</sup>

Un corpus parallèle est présenté sur la figure 2.1. Il contient un paragraphe en français et en anglais, les 2 paragraphes ont bien sûr le même contenu.

**Bitexte :**

Monsieur le Président, nous suivons depuis quelques semaines le déroulement des élections à Madagascar.	Mr. Speaker, over the past few weeks, we have been following the general election in Madagascar.
Malgré un premier tour de scrutin qui n'a pas donné de résultat concluant, le candidat de l'opposition s'est déclaré vainqueur et a depuis enjoint ses partisans à la grève générale et aux manifestations.	In spite of an inconclusive first ballot, the opposition candidate declared himself the winner and then enjoined his supporters to go on a general strike and to organize protests.
Aux dernières nouvelles, il y a maintenant deux gouvernements parallèles et deux ca-	According to the latest news, there are now two parallel governments and two capitals.
Débat	Artificial intelligence
L'intelligence artificielle	A Debat
Depuis 35 ans, les spécialistes d'intelligence artificielle cherchent à construire des machines pensantes.	Attempts to produce thinking machines have met during the past 35 years with a curious mix of progress and failure.
Leurs avancées et leurs succès alternent curieusement.	
	Two further points are important.
Les symboles et les programmes sont des notions purement abstraites.	First, symbols and programs are purely abstract notions.

2.2 : Bitexte, "Le bitexte et ses applications"<sup>8</sup>

Fig

<sup>8</sup> <http://www.iro.umontreal.ca/~felipe/Papers/slides-tana-2005.pdf>

<sup>8</sup> <http://www.iro.umontreal.ca/~felipe/Papers/slides-tana-2005.pdf>

Sur la figure 2.2, nous avons le bitexte construit à partir du corpus parallèle de la figure 1.1. Les paragraphes ont été découpés et alignés l'un par rapport à l'autre en essayant de garder le plus possible le même sens pour les phrases alignées.

### B. Modèle de langue (LM)

Un modèle de langue est un modèle qui spécifie une distribution  $P(s)$  sur les chaînes  $s$  de la langue modélisée:  $\sum_s P(s) = 1$

Nous pouvons ajouter de l'information si on considère que  $s$  est une suite de  $N$  mots,  $s = w_1 \dots w_N$ , alors  $P(s) = \prod_{i=1}^N P(w_i | \underbrace{w_1 \dots w_{i-1}}_h)$ , où  $h$  est appelée l'historique.

Un modèle de langue probabiliste peut être présenté comme une fonction donnant la probabilité d'observer un mot étant donné ceux déjà observés. Cette approche a déjà montré son utilité dans plusieurs applications dont la reconnaissance de la parole et de caractères ou encore la correction de fautes d'orthographe.

L'estimation des distributions  $P(w|h)$  où  $w$  est un mot et  $h$  l'historique (l'ensemble des mots déjà vus) est un problème complexe (trop de paramètres à estimer) que nous pouvons simplifier par une approximation markovienne d'ordre  $n-1$ ,  $n-1$  étant le nombre de mots dans l'historique :

$$P(s) = \prod_{i=1}^N P(w_i | w_{i-n+1}^{i-1})$$

La probabilité d'un mot est conditionnée "seulement" par les  $n-1$  derniers mots dans l'historique de  $w$ . Cette simplification est appelée un modèle  $n$ -gramme. Dans le cas du trigramme nous avons donc la formule :

$$P(e) \approx P(w_1)P(w_2 | w_1)P(w_3 | w_1 w_2)P(w_4 | w_2 w_3) \dots P(w_N | w_{N-2} w_{N-1})$$

Les probabilités sont apprises à partir d'un corpus d'apprentissage. Il existe plusieurs méthodes pour les calculer, que ce soit de la plus simple avec la fréquence relative qui compte simplement le nombre de fois où apparaît le mot ou le groupe de mots ou des plus complexes avec l'application d'un lissage sur la fréquence relative.

Nous allons dans notre recherche utiliser la ressource SRILM<sup>9</sup> (A. Stolcke, 2002) pour créer notre modèle de langue. SRILM est une collection de bibliothèques c++, de programmes exécutables et de scripts. Il permet la production et l'expérimentation de modèles de langue statistiques pour la reconnaissance de la parole et d'autres applications. Il est disponible gratuitement pour les projets non commerciaux. Cet ensemble d'outils permet la création et l'évaluation de différents types de modèles de langue basés sur les comptes  $N$ -gram. Le modèle de langue peut avoir plusieurs applications dans les technologies du langage naturel et dans d'autres domaines.

---

<sup>9</sup> <http://www.speech.sri.com/projects/srilm/>

Nous allons pour notre calcul de probabilités utiliser l'approche Kneser-Ney (Kneser et Ney 1995), celui-ci obtenant des performances tout à fait acceptables (Goodman, 1995). Le lissage de Kneser-Ney repose sur le constat que les modèles d'ordre inférieur sont ceux sur lesquels l'attention doit être vraiment portée, étant donné que nous pouvons habituellement se fier à la qualité de l'estimateur MLE d'ordre  $n$ . Cette méthode propose donc de pondérer l'importance des modèles d'ordre inférieur de manière plus précise, en utilisant le nombre de contextes différents dans lesquels l'événement intervient. Si nous voulons estimer la probabilité du bigramme New York par exemple,  $P(\text{York}|\text{New}) = \alpha P(\text{York}|\text{New}) + \beta P(\text{York})$ , l'importance du modèle du premier ordre estimant la probabilité de York devrait être faible, étant donné que York apparaît rarement dans un autre contexte (il est raisonnable de croire que le modèle bigramme se tirera d'affaire seul).

Nous utilisons pour créer le modèle la commande Ngram-count de SRILM qui comporte un grand nombre d'options pour contrôler les divers paramètres du LM :

- ordre du N-gram
- le type d'algorithme utilisé
- une option peut prédéfinir le vocabulaire pour limiter ou augmenter le jeu de mots des données d'entraînement.
- un mot inconnu peut être ignoré ou traité comme un élément spécial "mot inconnu".

Pour notre modèle nous utilisons la commande suivante : Ngram-count -ukndiscount -order 3, c'est à dire l'algorithme original de Kneser-Ney avec un N-gram d'ordre 2. Nous nous retrouvons alors avec un fichier de compte de mots (figure 2.3).

```
\data\  
ngram 1=60  
ngram 2=80  
ngram 3=1  
\1-grams:  
...  
-1.822684      a      -0.04180291  
...  
-2.594761      with  -0.02626889  
\2-grams:  
...  
-1.317018      a new  
...  
-1.015988      which facilitates  
-1.015988      with the  
\3-grams:  
-0.2908108     the charter is  
\end\
```

Fig 2.3 : Exemple d'un modèle de langage d'ordre 2

### C. Modèle de traduction

Nous avons aussi besoin d'un modèle de traduction pour faire de la traduction automatique. Celui-ci est construit à partir des bitextes créés précédemment. Contrairement à un modèle de langue, le modèle de traduction permet d'avoir la probabilité d'un mot ou groupe de mots sachant un mot ou groupe de mots. Ce modèle va permettre de modéliser une traduction c'est à dire qu'il va nous donner une probabilité qu'un mot ou groupe de mots d'une langue alpha corresponde à un mot ou groupe de mots d'une langue beta. Notre modèle sera constitué alors de deux segments (un ou plusieurs mots) de phrases avec un ou plusieurs scores, ces scores correspondent à diverses probabilités calculées. Ces modèles peuvent donc être basés sur des mots ou des segments de phrases. Ainsi nous allons expliquer les deux modèles avec leurs avantages et leurs inconvénients.

#### 1. Modèle de traduction basé sur les mots

Soit  $E$  l'ensemble des phrases anglaises et  $F$  l'ensemble des phrases françaises. Soit  $e = (e_1 \dots e_n)$  une phrase de  $E$  que nous voulons traduire et  $f = (f_1 \dots f_m)$  une phrase de  $F$  que nous considérons comme traduction potentielle de  $e$ . Nous avons dit précédemment que pour traduire de l'anglais vers le français, le modèle de traduction doit nous donner la probabilité  $P(e|f)$ . Les modèles de traduction IBM sont tous basés sur la notion d'alignement de mots. Un alignement entre la source  $f$  et la cible  $e$  est représenté par un vecteur de positions  $a = (a_1 \dots a_n)$  avec  $\forall i \in [1, n], a_i \in [0, m]$ . Chaque  $a_i$  est tel que  $f_i$  est en relation de traduction avec  $e_{a_i}$ .  $a_i = 0$  si  $f_i$  n'est en relation de traduction avec aucun mot de  $e$ . Un tel alignement est représenté à la figure 2.4. Les flèches  $y$  indiquent les alignements d'un mot de la source vers un mot de la cible.

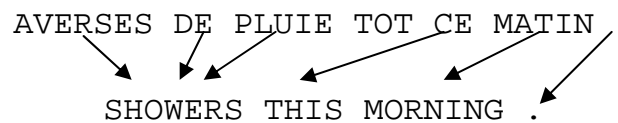


Fig 2.4 : Alignement entre une phrase française et une phrase anglaise.

Avec cette représentation des alignements, chaque mot de la source est aligné avec au plus un mot de la cible. Par contre, plusieurs mots de la source peuvent être alignés avec le même mot de la cible. Par exemple, nous voyons dans la figure 2.4 que les 3 mots "averses de pluie" sont alignés à "showers". Remarquons qu'il ne serait pas possible d'aligner correctement les phrases dans l'autre direction (anglais vers français) car "showers" ne pourrait être aligné qu'à un seul mot. Il s'agit d'une limitation connue des modèles de type IBM. Une fois l'alignement fait, le calcul des probabilités est fait sur les mots mis en relation.

## 2. Modèle de traduction basé sur les segments de phrases (PBM ou modèle phrase-based)

Ce modèle de traduction est basé sur les séquences de mots (Phrase-Based Model ou PBM). Dans un modèle PBM, les phrases sont partitionnées en séquence de mots qui sont les paramètres du modèle. Le principal avantage de ces modèles est qu'ils sont capables de capturer des traductions d'expressions entières récurrentes dans le bitexte. Un autre avantage de ces modèles est que le réordonnement local des mots d'une langue à l'autre est pris en compte de manière passive par un tel modèle. De plus ils sont d'une grande simplicité, souple et sont tolérants aux langues difficiles à segmenter en mots, comme le chinois. Le fichier de la figure 2.5, représentant le modèle, est obtenu à partir de l'outil Giza++, (Och & Ney, 2000). Nous pouvons voir que pour chaque segment de phrase en français correspond un segment de phrase en anglais et une probabilité. La première ligne de ce fichier indique par exemple que  $P(\text{ne pas reformer} \mid \text{not reconstitute}) = 1$ .

```
ne pas reformer ||| not reconstitute ||| 1
japonais et américains ||| Japanese and American ||| 1
Une recriminalisation ||| Re-criminalization violates ||| 1
serait moins inquiétant ||| would be less disturbing ||| 0.5
serait moins inquiétante ||| would be less disturbing ||| 0.5
nous aurons tous noté avec intérêt ||| we all noted with interest ||| 1
accepté cet amendement pour répondre ||| accepted this amendment to respond ||| 1
n'ont pas toutes ||| there was not unanimity of feeling throughout the country with
regard to all ||| 1
augmenter plus vite que le taux ||| grow faster than the rate ||| 1
dommages faits par ||| damage that is being done by ||| 1
on ne peut pas demander ||| they cannot seek ||| 1
une erreur d'interprétation . ||| a misinterpretation . ||| 1
Monsieur le Président, ce gouvernement est symbole ||| Mr. Speaker, this Government
is synonymous ||| 1
a été présenté aux ||| was presented to ||| 0.285714
```

Fig 2.5 : Exemple modèle de traduction de segments de phrases

Mais cette méthode comporte également des points faibles. Elle ne permet aucune généralisation comme nous pouvons le voir sur l'exemple suivant :

```
passer un sapin ||| pulling a fast one ||| 1
nous passer un sapin ||| pull a fast one on us ||| 1
passer un sapin . ||| pull a fast one on us . ||| 0.5
nous passer un sapin . ||| pull a fast one on us . ||| 0.5
passer un sapin. ||| pull a fast one ||| 1
passer un sapin . ||| play a fast one ||| 1
```

Fig 2.6 : Exemple des faiblesses des modèles de traduction PBM

Ainsi nous avons plusieurs expressions qui pourraient être regroupées et ne le sont pas. En effet nous retrouvons dans le modèle de la figure 2.6, 6 fois les mots "passer un sapin"

avec trois traductions identiques ou proches. Regrouper ces traductions permettrait de leur donner plus de poids car vues plus souvent elles sont donc plus susceptibles d'être la bonne traduction. Ces traductions étant proches pourraient être regroupées. La conséquence de ce problème est que les fichiers créés ont une taille assez grande mais aussi que les estimés sont bruités. Nous devons donc manipuler un gros volume de données, de l'ordre de 1Gig pour 1,7 M de paires de phrases ce qui est lourd à charger en mémoire.

Nous allons tout de même utiliser ce type de modèle pour notre étude. Nous pensons qu'elle nous apporte plus d'avantages que le simple modèle de mots. Nous utilisons pour créer ces modèles une ressource disponible gratuitement qui est Giza++<sup>10</sup>, (Och & Ney, 2000). Ce programme va nous permettre de créer automatiquement notre modèle. Les modèles sont appris sur un bitexte d'entraînement constitué bien sûr de la langue source et de la langue cible. Nous faisons ensuite une restriction du modèle à partir du fichier test pour ne prendre que les segments apparaissant dans le fichier test pour avoir un modèle moins gros. En effet comme nous avons vu précédemment, les modèles peuvent être volumineux, donc longs à charger et les segments n'apparaissant pas dans le fichier à traiter ne servent à rien donc nous les enlevons. Ainsi nous nous retrouverons avec un modèle de plus petite taille et gagnerons du temps dans l'exécution.

## D. Décodage

Nous décrivons dans cette section l'étape de décodage. Dans la traduction automatique probabiliste, le but d'un décodeur est de chercher la phrase française  $f = (f_1 \dots f_m)$  la plus probable étant donnée une phrase anglaise  $e = (e_1 \dots e_1)$  et des modèles (modèle de traduction et modèle de langue) où  $m$  et  $f_i, i \in [1, m]$  sont des inconnus.

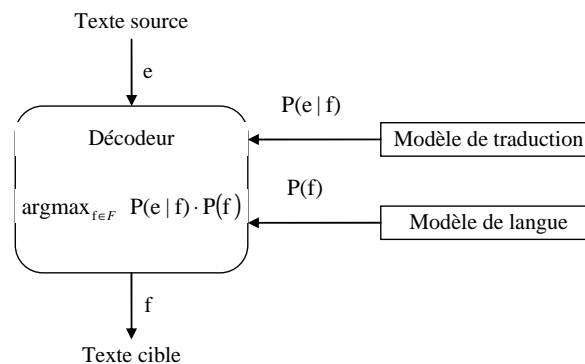


Fig 2.7 : L'architecture de la traduction probabiliste (Nießen et al. 1998)

Nous rappelons que pour effectuer notre traduction il nous faut résoudre  $\text{argmax}_{f \in F} P(e|f) \cdot P(f)$ . Nous avons créé notre modèle de langue ( $P(f)$ ) qui nous permet d'avoir les caractéristiques de la langue cible et notre modèle de traduction ( $P(e|f)$ ) qui nous

<sup>10</sup> <http://www.fjoch.com/GIZA++.html>

permet d'avoir la probabilité d'une traduction d'un mot ou groupe de mots suivant le type de modèle utilisé. Le décodeur devra donc maximiser le produit pour trouver la meilleure traduction.

Il existe plusieurs ressources, Pharaoh (Koehn, 2004), SIS (Sentovich, 1993), permettant de faire cette partie. Nous avons choisi d'utiliser Pharaoh qui est le décodeur le plus utilisé actuellement, y compris au RALI, et il permet d'avoir des performances état de l'art.

Pharaoh<sup>11</sup> est un décodeur utilisant les modèles de traductions automatiques statistiques phrase-based. Cette ressource a été développée en partie par PhD thesis (Koehn, 2003) à l'Université de Californie du Sud. Elle est d'ailleurs disponible gratuitement sur le site web de l'Institut des Sciences de l'Information de l'Université de Californie du Sud. Elle est l'implémentation d'une des meilleures méthodes courantes pour la traduction automatique statistique.

Pharaoh prend en entrée un modèle de langue et un modèle de traduction dans les formats présentés en figure 2.3 et 2.5 ainsi qu'un texte source. Il donne en sortie le texte traduit dans la langue cible.

### E. Tunning

Le décodage nous permet d'insérer un poids à chacun des modèles pour permettre l'utilisation astucieuse de ceux-ci et ainsi générer la meilleure traduction possible. Le tuning consiste alors à tester toutes les combinaisons de poids permettant d'obtenir les meilleures traductions possibles avec nos modèles. Cette étape se fait sur un fichier tune qui est disjoint du corpus d'entraînement et du fichier test. Une fois que les meilleurs poids sont trouvés, ils seront utilisés pour toutes les traductions. Le décodeur pharaoh possède des poids par défaut dans le cas où l'étape du tuning n'est pas faite. Pour faire le tuning nous utilisons le script tune-stripped-down du RALI qui prend en entrée le modèle de traduction, le modèle de langue, le fichier tune source (en langue source) et le fichier tune de référence (en langue cible). Il nous donne en sortie un fichier avec les meilleurs poids à utiliser avec notre traducteur. Nous allons utiliser l'étape du tuning que dans la deuxième partie de notre travail que vous retrouvez dans la section V. Les poids par défaut de pharaoh sont utilisés pour les expériences de la première partie, section IV.

### F. Evaluation

Nous allons utiliser pour mesurer nos traductions le score BLEU. BLEU (BiLingual Evaluation Understudy) est une méthode pour évaluer une traduction automatique présentée par (Papineni et al, 2002). L'idée de BLEU est de comparer les phrases de traduction et de référence en se basant sur les séquences n-gram (calcul pondéré sur les unigrammes,

---

<sup>11</sup> <http://www.isi.edu/licensed-sw/pharaoh/>

## Chapitre 2 - Traduction automatique

bigrammes, trigrammes et quadrigrammes). Une traduction est d'autant meilleure qu'elle partage un grand nombre de n-grams avec une ou plusieurs traductions de référence. BLEU donne un score entre 0 et 1. Plus le score est élevé, meilleure est la traduction. (Papineni et al, 2002) ont montré que BLEU est cohérente avec l'évaluation humaine.

Texte anglais original

1. add to this movement from team to team the loss of a child this past fall .
2. the co-operative agreements in such areas as agriculture , forestry , mining and economic development are extremely important to the citizens of new brunswick .
3. the high speed and aggressive edge of short track have been attracting a worldwide following .
4. collège militaire royal de saint-jean
5. however i will comment on the canadian economy .
6. we are not talking about submissions by witnesses but about his own members who came out in favour of applying the gst to health care and medication .
7. with the interest rate assumptions that we have made in the budget and if the interest rates continue where they are right now , we will come in at a lower cost of interest than we had in our budget .

*Fig 2.8 : Texte à traduire par Pharaoh*

Texte français original

1. on peut ajouter à ces changements nombreux d' équipe , la perte d' un enfant l'automne dernier .
2. ces ententes de coopération dans des domaines comme l' agriculture , la foresterie , l' exploitation minière et le développement économique , sont extrêmement importantes pour les habitants du nouveau-brunswick .
3. la vitesse et la vive concurrence du patinage sur courte piste soulève l' intérêt partout dans le monde .
4. le collège militaire royal de saint-jean
5. mais je vais en faire sur l' économie canadienne qui , elle , est forte .
6. il n' était pas question des représentations des témoins , hier , mais de ses propres députés qui se sont prononcés en faveur de l' application de la tps sur les soins de santé et les médicaments .
7. si la tendance actuelle des taux d' intérêt se maintient , le loyer de l' argent sera moins élevé que ce que nous avons prévu dans notre budget .

*Fig 2.9 : Texte de référence français*

## Chapitre 2 - Traduction automatique

Texte traduit automatiquement de l'anglais vers le français

1. ajoutez à cela acheminement de l' équipe de équipe la perte d' un enfant cet automne passé .
2. les accords de coopération dans ces domaines comme l' agriculture , forestières , minières et du développement économique est extrêmement important pour les nouveaux citoyens du nouveau-brunswick .
3. le haute vitesse et agressive de pointe courte hippodrome ont été investissements un entraînés suivantes .
4. collège militaire royal de saint-jean
5. mais je vais parler de l' économie canadienne .
6. nous ne parlons pas de intances de témoins , mais sur ses propres députés qui se sont manifestés en faveur d' appliquer la tps aux soins de santé et prescrits .
7. avec le taux d' intérêt hypothèses que nous avons fait dans le budget et si les taux d' intérêt continuera dès maintenant , nous allons venir à un coût inférieur de intérêt que nous avons dans notre budget . où ils sont en

*Fig 2.10 : Texte traduit par Pharaoh (texte original Fig 2.8)*

Nous pouvons voir sur l'exemple présenté dans les figures 2.8, 2.9, 2.10, que la traduction automatique est loin d'être parfaite. Notre but dans cette étude est donc d'arriver à l'améliorer en y ajoutant de l'information sémantique. Nous devons pour cela avoir un moyen de récupérer cette information. Nous allons donc voir maintenant les outils le permettant.

### III. Désambiguïsation

Un problème fondamental en traitement automatique du langage est la désambiguïsation du sens, problème plus communément appelé WSD en anglais pour Word Sense Disambiguation. Dans la vie, nous manipulons avec une grande aisance le sens des mots ou expressions, aisance qui nous fait penser qu'aucune action spéciale n'est faite pour cette compréhension. En effet une simple phrase comme "dépose le livre sur la table !" peut nous paraître tout à fait anodine et simple de compréhension. Pourtant lorsque nous tentons de traiter automatiquement une langue, cette facilité se transforme tout de suite en problème. En effet nous voyons bien que lors d'un traitement mot à mot de la phrase nous arrivons à une ambiguïté pour le mot "livre" qui pourrait être interprété comme "la livre" la monnaie ou "le livre" le bouquin ou même comme le verbe "livrer". Contrairement à un être humain, l'ordinateur ne pourra choisir instinctivement tel ou tel sens. Le problème peut être restreint ou même évité si nous travaillons dans un espace fermé, c'est-à-dire un espace avec un vocabulaire fermé et non ambigu ou avec peu d'ambiguïté. Mais cet espace réduit énormément l'utilisation et n'est que peu probable dans le monde qui nous entoure. Aujourd'hui nous voulons que les systèmes puissent être utilisés dans des domaines différents et variés.

Pourtant même des phrases peuvent hors contexte nous paraître, à nous être humain, aussi ambiguës. En effet la phrase "La belle ferme le voile" ne nous permet pas sans contexte de définir si belle est le nom en parlant d'une femme ou l'adjectif belle pour jolie. Nous pouvons donc montrer le problème rencontré par l'ordinateur pour des mots plus anodins mais l'interrogation reste la même, quel sens choisir ?

La désambiguïsation a donc pour but de nous permettre de lever le maximum d'ambiguïté liée aux mots. Les applications telles que la traduction automatique, l'acquisition de connaissances et autres qui requièrent une connaissance pourraient être améliorées par la désambiguïsation. Cette tâche peut être réalisée de façon passive, étiqueteurs syntaxiques, ou de façon active, désambiguïseurs. Dans tous les cas, nous avons besoin pour implémenter ces méthodes de ressources. Ces ressources doivent nous permettre de retrouver de l'information sémantique.

#### A. Les ressources

Nous avons besoin pour la désambiguïsation de corpus contenant de l'information sémantique et syntaxique. En effet pour la plupart des méthodes supervisées que nous utilisons dans notre étude, nous allons avoir besoin d'un maximum de mots annotés avec leur sens et leur étiquette syntaxique. Nous nous sommes intéressés à WordNet (Fellbaum, 1998) qui nous permet d'avoir une base de données lexicale et va donc nous permettre d'avoir de bonnes connaissances lexicales. De plus cette ressource est disponible gratuitement et est facile d'utilisation ce qui a confirmé notre choix. Nous nous sommes aussi intéressés à

SemCor qui n'est autre qu'un ensemble de corpus annotés par leur sens et leur étiquette morphosyntaxique.

### 3. WordNet

Le laboratoire des sciences cognitives de l'Université de Princeton a développé WordNet<sup>12</sup> (Fellbaum, 1998) qui est une sorte de base de données lexicale. Elle répertorie, classe et relie suivant diverses relations les sens des mots de la langue anglaise. Le système se présente sous la forme d'une base de données électronique qui peut être téléchargée. WordNet est distribué avec une licence spéciale très libérale, permettant de l'utiliser commercialement ou à des fins de recherche, et il est utilisé localement grâce à des interfaces disponibles dans différents langages comme C++ ou java.

WordNet jouit d'une bonne popularité au sein de la communauté scientifique, et joue également un rôle important dans plusieurs projets commerciaux. Sa richesse et sa précision en font un outil de choix, susceptible d'être utilisé par une multitude de techniques et de théories diverses. Son utilisation permet d'avoir une importante base de connaissances à priori du langage et du monde.

L'information est organisée par synset qui regroupe plusieurs mots autour d'un sens ou d'un usage particulier. Chaque synset représente un sens différent, sens qui est décrit par une définition. Souvent le sens est également illustré par un exemple. Le mot polysémique pourrait être dans l'exemple remplacé par l'un ou l'autre des mots du même synset sans aucune altération de la signification de la phrase. Nous pouvons voir le synset de la catégorie nom du mot "car" sur la figure 3.1.

- 1 **car**, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"*
- 2 **car**, railcar, railway car, railroad car (a wheeled vehicle adapted to the rails of railroad) *"three cars had jumped the rails"*
- 3 cable car, **car** (a conveyance for passengers or freight on a cable railway) *"they took a cable car to the top of the mountain"*
- 4 **car**, gondola (the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant)
- 5 **car**, elevator car (where passengers ride up and down) *"the car was on the top floor"*

Fig 3.1 : exemple du synset de la catégorie nom du mot "car" de WordNet 2.1.

De plus les synsets sont séparés en catégories lexicales : nom, verbe, adjectif et adverbe. Nous avons donc pour chaque mot au plus 4 synsets qui représentent chacune des catégories. WordNet est une base de données de grande ampleur : la version la plus récente (2.1) répertorie plus de 155 327 mots de classes ouvertes (pour lesquelles l'ajout d'éléments lexicaux est possible) ainsi que plus de 117 597 synsets et regroupe 207 016 sens différents (voir tableau 3.1).

---

<sup>12</sup> <http://wordnet.princeton.edu/>

POS	Unique Strings	Synsets	Total Word-Sense Pairs
Noun	117 097	81 426	145 104
Verb	11 488	13 650	24 890
Adjective	22 141	18 877	31 302
Adverb	4 601	3 644	5 720
Totals	155 327	117 597	207 016

Tab 3.1: Détails données de WordNet (WordNet 2.1 Reference Manual)

Les mots dans WordNet sont représentés par leur forme canonique (de base) ou encore appelé lemme : singulier pour les noms (book, table); infinitif court pour les verbes (be, read); degré positif pour les adjectifs (good, lovely). Les mots composés, faisant référence à un même concept, sont encodés par une succession de mots individuels, reliés par souligné (underscore en anglais) (fontain\_pen, take\_for\_granted). En moyenne, il y a 1,3 sens par lemme dans WordNet

Comme dans un dictionnaire traditionnel, chaque mot a une liste de synsets associés qui représentent tous les sens possibles de ce mot. Mais les synsets peuvent également nous permettre d'avoir accès à des relations de plus haut niveau que le mot et le sens comme l'hyponymie. Ces relations sont organisées sous forme d'ontologie, qui n'est autre qu'un système de catégories permettant de classifier les éléments d'un univers. Les différentes catégories sont les relations sémantiques avec lesquelles nous pouvons regrouper de manière cohérente les caractéristiques linguistiques d'un mot (mot, sens, concept).

La relation sémantique qui constitue la base du regroupement des mots définira le type d'ontologie. WordNet répertorie ainsi une grande variété de relations sémantiques permettant d'organiser le sens des mots, et donc par extension les mots eux-mêmes, en des systèmes de catégories que nous pouvons consulter facilement. Nous pourrions ainsi interroger le système sur les *hyponymes* d'un mot particulier. À partir par exemple du sens le plus commun du nom "car", issu du premier sens du mot "car" dans WordNet montré sur la figure 3.2, la relation d'*hyponymie* définit un arbre de concepts de plus en plus généraux. Dans cet exemple, il est clair que le dernier concept, "entity, something", est le plus général, le plus abstrait. Il pourrait ainsi être le super-concept d'une multitude de concepts plus spécialisés.

- 1. car, auto, automobile, machine, motorcar
  - => motor vehicle, automotive vehicle
  - => vehicle
  - => conveyance, transport
  - => instrumentality, instrumentation
  - => artifact, artefact
  - => object, physical object
  - => entity, something

Fig 3.2 : Arbre de concept du premier sens "car" dans WordNet

- Nouns
  - synonyms
  - hypernyms
  - hyponyms
  - coordinate terms
  - holonym
  - meronym
- Verbs
  - synonyms
  - hypernym
  - coordinate terms
- Adjectives
  - synonyms and related nouns
  - antonyms
- Adverbs
  - synonyms and root adjectives
  - antonyms

Fig 3.3 : Relations sémantiques de WordNet

WordNet offre une grande palette d'ontologies représentant des relations sémantiques plus ou moins spécialisées et restrictives (voir la figure 3.3). Nous pouvons également interroger le système sur la relation d'*hyperonymie* ou d'*hyponymie* ou encore de *méronymes* (constitué de). Nous pouvons aussi consulter le système sur la relation inverse, l'*holonymie* ou encore pour les relations de *synonymie* ou d'*antonymie*. Chaque catégorie lexicale a bien sûr ces propres relations.

1. car, auto, automobile, machine, motorcar
  - HAS PART: accelerator, accelerator pedal, gas pedal, gas, throttle, gun
  - HAS PART: air bag
  - HAS PART: auto accessory
  - HAS PART: automobile engine
  - HAS PART: automobile horn, car horn, motor horn, horn
  - (...)

Fig 3.4 : Méronymes associés au sens "car, auto..." du mot "car"

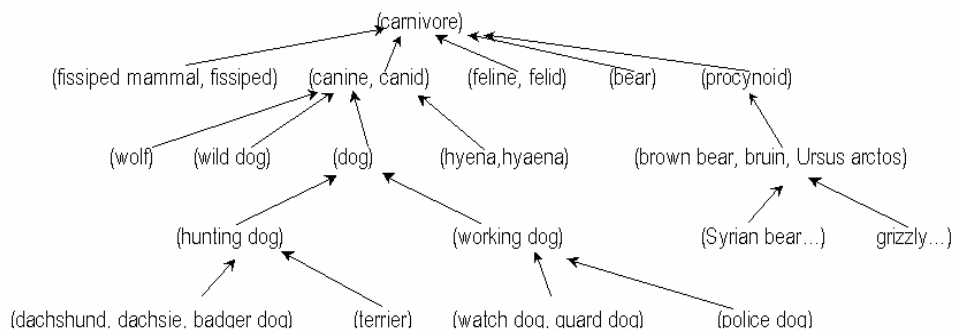


Fig 3.5: Sous hiérarchie de WordNet correspondant au concept "dog" (RFIEC<sup>13</sup>)

<sup>13</sup> <http://www.irit.fr/RFIEC/wordnet/wordnet.htm>

WordNet est un système d'une grande ampleur comme nous avons pu le voir précédemment. Son statut de projet "en développement" implique toutefois que certaines de ses composantes sont incomplètes. À chaque nouvelle version, le lexique est enrichi de nouveaux mots, et des relations sémantiques sont ajoutées, modifiées, ou encore rendues désuètes. Si on examine par exemple l'ontologie générée par la relation d'hyponymie, il est vrai qu'elle est la plus complète dans le cas des noms. Les noms sont ainsi classés en un système de catégories complet et précis comprenant plusieurs niveaux d'imbrication, nous retrouvons notamment certaines sections de cette ontologie où la profondeur dépasse dix niveaux. Nous avons en revanche un système de classification beaucoup moins élaboré pour les verbes, qui sont organisés en un système hiérarchique beaucoup plus "plat", c'est à dire avec moins de niveaux d'imbrication, où on passe très rapidement d'un concept spécialisé à un concept très général. À ce jour, il n'y a aucune catégorisation hiérarchique définie pour relier les adjectifs et les adverbes. Ce déséquilibre potentiellement problématique se retrouve à l'intérieur même des super-catégories, où il est beaucoup plus apparent dans la branche nominale : certains mots sont ainsi liés à une grande chaîne de concepts finement graduée, tandis que d'autres sont très proches des concepts les plus généraux.

Un deuxième problème peut être mis à jour lors de certaines utilisations de WordNet, en effet il utilise les mots dans leur forme canonique ce qui peut créer une perte d'information. Il permet de faire la lemmatisation mais ne refait aucunement la phase inverse. Ainsi lors de remplacement de mots conjugués dans une phrase, le remplacement va se faire par un mot non conjugué ce qui peut générer des erreurs.

#### 4. Semcor

Le corpus SemCor<sup>14</sup> (Miller et al., 1993) est un extrait du corpus de Brown de 352 documents différents contenant 778 587 mots, répartis en 37 176 phrases (la plus longue a 251 mots). Les différents textes sont répartis dans trois répertoires, brown1, brown2, brownv. Nous pouvons voir les caractéristiques des textes contenus dans chacun de ces dossiers dans le tableau 3.2.

	nb de fichiers	nb de phrases	nb de mots	nb mots max dans une phrase	nb de mots désambiguïsés
brown1	103	11 182	229 358	134	103 292
brown2	83	8 956	184 930	251	82 421
brownv	166	17 038	364 299	219	40 850
Semcor	352	37 176	778 587	251	226 563

Tab 3.2: Détails des dossiers de Semcor

Chaque mot du Semcor est annoté par une étiquette morphosyntaxique qui nous permet de connaître sa catégorie lexicale. Nous pouvons donc faire une analyse du type de

<sup>14</sup> <http://www.cs.unt.edu/~rada/downloads.html>

mots contenus dans les corpus. Chaque mot est également étiqueté manuellement à l'aide des sens fournis dans WordNet. Seuls les noms, les verbes et les adjectifs sont annotés dans les textes de dossier brown1 et brown2 et seuls les verbes le sont dans le dossier brownv ce qui explique le peu de mots désambiguïsés dans celui-ci. Selon le manuel fourni avec le corpus, le taux d'erreur de l'étiquetage du sens serait de 13 % (SemCor, 1995). En fait, il est toujours difficile d'évaluer la qualité d'un étiquetage, le taux d'erreur n'est donné qu'à titre indicatif pour se faire sa propre idée sur la pertinence des résultats des applications utilisant ce corpus. Nous pouvons aussi voir le détail de la distribution des étiquettes dans chaque dossier dans le tableau 3.3. Il nous permet de connaître les étiquettes présentes et en quelle quantité.

	Brown	Brown1	Brown2	Brownv
nom	22,89%	22,35%	22,07%	23,65%
verbe	16,17%	16,27%	16,58%	15,91%
adjectif	8,35%	8,95%	8,89%	7,69%
adverbe	6,15%	6,61%	6,63%	5,62%
préposition	12,03%	11,63%	12,14%	12,23%
déterminant	10,19%	10,17%	9,83%	10,40%
pronom	6,17%	6,28%	6,54%	5,91%
conjonction de coordination	3,29%	3,31%	3,32%	3,25%
ponctuation	13,54%	13,71%	13,41%	13,50%
nombre	1,11%	0,60%	0,49%	1,76%
divers	0,09%	0,10%	0,09%	0,09%

Tab 3.3: Détails des étiquettes POS des dossiers de Semcor

Semcor est présenté dans un format SGML, les données textuelles étant segmentées en fichiers, paragraphes, phrases, mots et signes de ponctuation (figure 3.6). Une étiquette sémantique attachée à un mot indique le sens de WordNet approprié au contexte où il apparaît. Seulement les noms, les verbes, les adjectifs et les adverbes sont étiquetés. Les noms propres sont annotés par quatre types d'étiquettes : person, location, group ou other. Il faudra donc la plupart du temps une mise en forme des données pour pouvoir les utiliser plus facilement.

## Chapitre 3 - Désambiguïsation

```
<contextfile concordance=brown>
<context filename=br-a01 paras=yes>
<p pnum=1>
<s snum=1>
<wf cmd=ignore pos=DT>The</wf>
<wf cmd=done rdf=group pos=NNP lemma=group wnsn=1 lexs=1:03:00::
pn=group>Fulton_County_Grand_Jury</wf>
<wf cmd=done pos=VB lemma=say wnsn=1 lexs=2:32:00::>said</wf>
<wf cmd=done pos=NN lemma=friday wnsn=1 lexs=1:28:00::>Friday</wf>
<wf cmd=ignore pos=DT>an</wf>
<wf cmd=done pos=NN lemma=investigation wnsn=1 lexs=1:09:00::>investigation</wf>
...
<punc>.</punc>
</s>
</p>
<p pnum=2>
<s snum=2>
<wf cmd=ignore pos=DT>The</wf>
...
</context>
</contextfile>
```

Fig 3.6 : Extrait du corpus *Semcor*

Nous allons découper et coller ces différents textes pour créer un corpus dit d'apprentissage qui nous permettra par la suite d'entraîner des méthodes et un corpus dit de test qui nous permettra d'évaluer ces méthodes, ces deux corpus sont disjoints

Notre texte d'apprentissage sera donc composé de 34 177 phrases contenant 707 961 mots. Nous avons donc une moyenne de 20,71 mots par phrase, avec la phrase la plus longue contenant 251 mots. Nous avons dans le corpus 193 474 mots annotés d'un sens WordNet. Notre fichier de test quand à lui comprend 3 000 phrases pour 70 626 mots avec une moyenne de 23,54 mots par phrases. Nous avons dans le fichier test 33 089 mots annotés d'un sens WordNet, donc autant de mots qui devront être désambiguïsés. La distribution des différentes catégories, détaillées dans le tableau 3.4, restent à peu près les mêmes pour le fichier test et train.

	train	test
nom	22,76%	24,22%
verbe	16,28%	15,16%
adjectif	8,22%	9,62%
adverbe	6,12%	6,47%
préposition	12,01%	12,23%
déterminant	10,12%	10,96%
pronom	6,33%	4,60%
conjonction de coordination	3,29%	3,28%
ponctuation	13,64%	12,59%
nombre	1,15%	0,80%
divers	0,10%	0,06%

Tab 3.4: Détails des étiquettes POS des corpus

## B. Etiqueteur morphosyntaxique (POS)

Cet outil consiste à donner une étiquette morphosyntaxique à un mot. Cette étiquette représente la catégorie du mot, ce sera par exemple NN pour nom, PP pour préposition. Ces étiquettes seront plus ou moins précises selon l'étiqueteur. En effet certains pourront contenir une vingtaine d'étiquettes alors que d'autres n'en n'auront qu'une dizaine. Nous retrouvons tout de même une similarité entre les étiqueteurs les plus utilisés ce qui permet une meilleure comparaison. Les étiqueteurs sont utilisés dans plusieurs applications liées au langage comme la recherche d'informations. Ils permettent d'avoir une information syntaxique qui peut être utile dans le traitement d'une phrase.

Dans notre cas, un tel étiqueteur peut être considéré comme une aide à la désambiguïisation, en effet il nous permet d'avoir une information supplémentaire dans le choix d'un sens. Par exemple, le mot livre peut être un nom commun dans "Paul pose le livre sur la table", un verbe dans "Paul livre la pizza", donc si nous pouvons savoir la catégorie du mot nous pouvons savoir son sens. Cette information n'est bien sûr utile pour la désambiguïisation que s'il n'y a qu'un sens par étiquette POS.

Plusieurs méthodes sont proposées pour annoter automatiquement les mots par des étiquettes morphosyntaxiques. Plusieurs outils sont fondés sur des systèmes basés sur des règles (Greene and Rubin, 1971), (Brill, 1992). D'autres implémentent des méthodes probabilistes (Bahl and Mercer, 1976), (Schmid, 1994), (Church, 1988), (Cutting et al., 1992), (DeRose, 1988), (Kempe, 1993). Les réseaux de neurones ont aussi été testés dans l'étiquetage POS (Frederici and Pirrelli, 1994).

Dans notre étude nous nous sommes intéressés à deux étiqueteurs POS utilisant des méthodes différentes puisque Brill (Brill, 1992) est basé sur des règles et Tree Tagger

(Schmid, 1994) sur des probabilités. Nous avons donc évalué ces deux ressources pour pouvoir choisir celle que nous utiliserons dans notre étude.

### 1. BRILL

L'étiqueteur de Brill<sup>15</sup> permet un étiquetage lié à des règles syntaxiques. Une règle syntaxique est tout simplement une règle qui permet de structurer la syntaxe d'une phrase. Par exemple, nous savons qu'un verbe ne peut pas succéder à un déterminant, nous pouvons donc créer une règle qui dirait que l'étiquette "Verbe" ne peut pas se trouver derrière l'étiquette "déterminant". Mais la méthode n'utilise pas seulement la technique des règles. En effet dans un premier temps elle fait un premier étiquetage du texte qui permet d'avoir une première ébauche. L'étiquetage est fait par exemple à partir d'un modèle de Markov qui part du principe suivant :

$$P(\text{tag}) = \prod_{i=1}^N P(\text{tag}_i | \text{tag}_{i-n+1}^{i-1}).$$

Pour choisir une étiquette nous cherchons donc l'étiquette la plus probable d'apparaître à la position  $i$  en ayant un historique donc en sachant ce qui s'est passé auparavant. Cette méthode très simpliste et pas toujours efficace est améliorée par les règles syntaxiques. Ces règles permettent de prendre en compte l'information syntaxique, par exemple, un verbe ne suivra jamais un déterminant. Nous pouvons construire ces règles manuellement ou automatiquement à partir d'un corpus d'apprentissage. Ainsi l'application de ces règles sur le premier texte étiqueté nous permettra de corriger les erreurs faites lors de la première phase. Nous retrouvons un exemple d'étiquetage sur la figure 3.7.

```
tabling/VBG of/IN documents/NNS
house/NN of/IN commons/NN
thursday/NN ./, april/NNP 17/CD ./, 1986/CD
the/DT house/NN met/VBD at/IN 11/CD a.m./NN
statements/NNS by/IN ministers/NNS
routine/JJ proceedings/NNS
petitions/NNS
government/NN response/NN
canadian/NN charter/NN of/IN rights/NNS and/CC freedoms/NNS
fourth/JJ anniversary/NN of/IN proclamation/NN
mr./CD speaker/NN ./, our/PRP$ government/NN has/VBZ demonstrated/VBN its/PRP$
support/NN for/IN these/DT important/JJ principles/NNS by/IN taking/VBG steps/NNS
to/TO enforce/VB the/DT provisions/NNS of/IN the/DT charter/NN more/RBR
vigourously/RB ./.
```

Fig 3.7 : Exemple de texte étiqueté par Brill

---

<sup>15</sup> <http://www.cs.jhu.edu/~brill/>

## 2. Tree Taggers

Tree Tagger<sup>16</sup> a beaucoup de points communs avec les étiqueteurs conventionnels n-gram (Church, 1988), (Kempe, 1993). Le choix se fait à partir des probabilités, plus exactement en maximisant la probabilité que l'étiquette tag apparaisse à la position  $i$  sachant un historique. Nous utilisons pour faire cela la formule suivante :

$$\operatorname{argmax}_{f \in F} P(w_1 w_2 \dots w_n, t_1 t_2 \dots t_n) = P(t_n | t_{n-2} t_{n-1}) \cdot P(w_n | t_n) \cdot P(w_1 w_2 \dots w_{n-1}, t_1 t_2 \dots t_{n-1})$$

La spécificité de cette méthode se trouve dans le calcul de la probabilité de transition  $P(t_n | t_{n-2} t_{n-1})$ , qui n'est autre que la probabilité d'une étiquette par rapport aux étiquettes précédentes. En effet dans cette méthode, elle est calculée à partir d'un arbre de décision binaire créé pour tous les choix possibles alors qu'habituellement elle est calculée à partir de simples probabilités. Nous retrouvons sur la figure 3.8 un exemple de texte étiqueté par Tree Tagger.

tabling	VVG	table	
of	IN	of	
documents		NNS	document
house	NN	house	
of	IN	of	
commons		NN	commons
thursday	NN	Thursday	
,		,	
april	NN	April	
17	CD	@card@	
,		,	
1986	CD	@card@	
the	DT	the	
house	NN	house	
met	VVD	meet	
at	IN	at	
11	CD	@card@	
a.m.	NN	a.m.	

Fig 3.8 : Exemple de texte étiqueté par Tree Tagger

## 3. Evaluation

Nous avons évalué les étiqueteurs sur un corpus annoté issu du Hansard, débats parlementaires canadiens. Ce corpus est composé de 4 528 phrases, de 107 165 mots. Nous avons donc 23,7 mots en moyenne par phrase, 202 mots pour la plus grande. Ces mots ont été étiquetés manuellement par le RALI à l'aide d'un jeu d'étiquettes riches (183 étiquettes comme AdjQ-Compar pour adjectif qualificatif de comparaison, Verb-aux-PAST-plur-p1 pour verbe auxiliaire à la première personne du passé). Nous avons du faire un pré traitement sur le corpus pour unifier les noms des étiquettes. En effet les étiquettes utilisées sur ce corpus

<sup>16</sup> <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html>

sont des étiquettes françaises alors que nos deux étiqueteurs, Brill et Tree Tagger utilisent des étiquettes anglaises. Nous avons donc transformé les étiquettes en faisant une correspondance manuelle entre les deux types d'étiquettes puis un changement automatique pour avoir la version finale du fichier test de référence. Nous avons donc testé nos deux étiqueteurs sur le fichier test non étiqueté pour pouvoir ensuite comparer les fichiers créés avec le fichier original. La répartition des catégories lexicales du fichier test est détaillée dans le tableau 3.5. Nous pouvons remarquer que les noms, les verbes et les prépositions sont les catégories les plus présentes dans le texte.

Catégories	Nb %
N	25,44
V	15,93
Pre	15,51
Det	12,67
Pos	9,95
Adj	7,01
Pro	4,68
Adv	4,47
Cc	2,65
Cd	1,26
Divers	0,43

Légende :

- N : Nom ou Nom propres
- V : Verbe
- Det : Déterminant
- Pre : Préposition ou conjonction de subordination
- Pos : Ponctuation ou Symbole
- Adj : Adjectif
- Pro : Pronom
- Cc : Conjonction de coordination
- Cd : Nombre

Tab 3.5 : Détails des catégories du corpus de test

Les étiqueteurs donnent une étiquette à tous les mots du texte, nous prenons donc comme critère d'évaluation le pourcentage de mots bien étiquetés c'est à dire avec la même étiquette que celle du fichier original pour le mot.

	Brill	Tree-taggers
Corpus en minuscule	86.5143	73.6444
Corpus normal	86.3528	82.5643

Tab 3.6 : Evaluation des étiqueteurs

Nous avons pu remarquer en regardant les types d'erreurs que souvent dès qu'un nom a une majuscule au milieu de la phrase, les étiqueteurs ont tendance à l'étiqueter comme un nom propre. Nous avons donc testé les méthodes sur le corpus en minuscule. Nous pouvons faire un premier constat en disant que l'étiqueteur Brill obtient des résultats supérieurs à Tree-taggers. L'écart de l'erreur entre les deux est supérieur pour le corpus en minuscule que le corpus normal.

Nous avons ensuite regardé les sources d'erreurs les plus importantes pour les étiqueteurs sur le corpus normal. Nous avons au vue des résultats faits une analyse plus poussée des erreurs. Les résultats présentés dans le tableau 3.6 ne nous permettent pas de faire une analyse très minutieuse des erreurs car nous ne savons pas où celles-ci se font. Nous

### Chapitre 3 - Désambiguïsation

allons faire une évaluation un peu plus détaillée en étudiant sur quelles étiquettes se font les principales erreurs.

Erreur sur		remplacé par	
Det	10,52 %	PP	73,42 %
Adj	15,35 %	N	73,85 %
N	44,33 %	NP	81,01 %
Adv	7,99 %	In	24,49 %
		Adj	39,73 %
		N	24,32 %
V	11,34 %	V	78,66 %
divers	10,47 %		

Légende :

- Det : Déterminant
- PP : Pronom possessif
- Adj : Adjectif
- N : Nom
- NP : Nom propre
- Adv : Adverbe
- In : préposition ou conjonction de subordination
- V : Verbe

Tab 3.7 : Détails des erreurs de l'étiqueteur Brill

Nous pouvons voir dans le tableau 3.7 où se trouvent les principales erreurs d'étiquetages pour Brill. Nous voyons que 44,33% donc plus du tiers se produisent sur les noms et que ces erreurs sont dues à 81,01% à un remplacement par un nom propre ce qui nous confirme bien que l'étiqueteur a tendance à étiqueter à outrance un nom en nom propre juste parce qu'il commence par une majuscule. Nous pouvons aussi constater que l'autre erreur la plus fréquente se fait sur les adjectifs qui sont généralement confondus avec les noms.

Erreur sur		remplacé par	
Det	10,69	PP	60,72
		IN	12,12
Adj	12,33	N	72,03
		V	8,80
		DET	7,61
N	37,76	NP	72,84
Adv	6,76	IN	21,61
		Adj	35,07
		N	26,04
V	10,24	V	63,38
Aligne	18,59		
divers	3,64		

Légende :

- Det : Déterminant
- PP : Pronom possessif
- Adj : Adjectif
- N : Nom
- NP : Nom propre
- Adv : Adverbe
- In : préposition ou conjonction de subordination
- V : Verbe

Tab 3.8 : Détails des erreurs de l'étiqueteur Tree Tagger

Nous pouvons voir que les erreurs de Tree Tagger sont sensiblement les mêmes que pour Brill. Les principales se retrouvent au niveau des noms avec encore une fois le problème des mots avec une majuscule qui se retrouvent de suite catalogués en noms propres. Ensuite nous avons aussi les adjectifs qui sont étiquetés en tant que nom. Nous pouvons constater une nouvelle catégorie d'erreurs nommée aligne. En effet nous avons rencontré un problème avec l'étiqueteur Tree Taggers lors de l'évaluation de celui-ci. En regardant de plus près les

erreurs, nous nous sommes aperçus qu'il segmentait certains mots qui ne le sont pas dans le texte original, ce qui produit un décalage pour l'évaluation. Nous avons tout d'abord mis en place un premier traitement automatique pour palier à ce décalage mais il nous reste encore beaucoup d'erreurs, 18,59%, puisque le tableau 2.8 est établi à partir de cette nouvelle évaluation. Nous avons donc passé le texte manuellement pour pouvoir avoir des résultats réels. Nous avons après recalculé le pourcentage d'erreur de l'étiqueteur pour pouvoir faire une analyse juste et précise. Nous avons ainsi pu calculer que sur les 18,59% d'erreurs trouvées, il n'y avait que 13,66% de vraies mauvaises erreurs et 86,64% de bons étiquetages ce qui transforme le taux de 18,59% d'erreurs d'aligne en 2,53% d'erreurs diverses. Nous avons alors recalculé le vrai score de Tree Taggers à partir de ces nouveaux résultats.

	Brill	Tree-taggers
Corpus normal	86.3528 %	86.7028 %

Tab 3.9 : Deuxième évaluation des étiqueteurs

Nous pouvons donc constater qu'en fait la différence entre les erreurs des deux étiqueteurs est vraiment minime. De plus le type des erreurs est assez semblable pour les deux. Il reste le problème du réalignement des mots de l'étiqueteur Tree Tagger qui pourrait poser problème suivant l'application que nous voulons faire de l'étiquetage. Nous avons pour notre part choisi d'utiliser Tree Tagger, en effet les résultats étant équivalents nous avons choisi de le prendre car il était déjà installé dans les ressources du RALI. De plus il nous permet d'avoir accès au lemme du mot ce qui pourrait nous être utile plus tard. Mais l'étiquetage morphosyntaxique ne représente qu'une aide à la désambiguïsation mais ne tient aucunement le rôle de désambiguïseur que nous allons étudier ensuite.

### C. Algorithme de désambiguïsation

La tâche d'un désambiguïseur de sens des mots consiste à trouver automatiquement le sens le plus approprié à un mot polysémique dans son contexte. Aujourd'hui il existe plusieurs formes d'algorithmes permettant d'effectuer cette tâche. Ces méthodes peuvent être divisées en trois grandes classes :

- la désambiguïsation basée sur la connaissance qui essaye d'extraire de manière automatique de ces ressources l'information nécessaire à la désambiguïsation. Ainsi le système calcule la similarité entre les informations pouvant être prises dans le contexte où se trouvent le mot et les informations contenues dans la base.
- la désambiguïsation supervisée qui utilise des corpus d'entraînement annotés, comportant des étiquettes de sens, pour désambiguïser les nouvelles occurrences des mots polysémiques, en faisant appel à des hypothèses de type théorie de l'information, Naïve Bayes ou modèles Markov cachés.

- la désambiguïsation non supervisée qui essaye de dériver les informations nécessaires à la désambiguïsation à partir des corpus non-annotés, par des méthodes de classification des sens ou clustering.

Il existe aujourd'hui des compétitions SENSEVAL<sup>17</sup> (Senseval-1 1998, Senseval-2 2001, Senseval-3 2004) qui ont pour but l'évaluation des systèmes de désambiguïsation lexicale. Elles permettent donc de faire le tour des méthodes existantes et de les comparer. Dans la récente évaluation Senseval 3, les meilleures approches pour la désambiguïsation de tous les mots sont basées sur l'information tirée du corpus annoté. Le système développé par (Decadt et al., 2004) utilise deux classifieurs *memory-based* en cascade. Une séparation "word expert" est apprise pour chaque mot ambigu, utilisant une concaténation du corpus anglais du sens cible, incluant SemCor, jeu de données de SENSEVAL, et un corpus construit à partir d'exemples de WordNet. La performance de ce système sur SENSEVAL-3 Anglais pour un jeu de données de tous les mots est évaluée à 65,2%. Un autre système intéressant est le premier développé par (Yuret, 2004) qui combine deux modèles statistiques Naïve Bayes, un modèle Naïve Bayes permet de calculer le bon sens à partir des probabilités des sens et mots. Le premier est basé sur les collocations voisines, c'est à dire qu'il prend directement les mots à coté du mot cible pour le calcul des probabilités. L'autre est basé sur les mots autour du mot cible, il choisit les mots les plus représentatifs dans la phrase pour le calcul. Le modèle statistique est construit à partir de SemCor et WordNet, pour un total de désambiguïsation de 64,1%. Une version différente de SenseLearner (Mihalcea and Faruque, 2004), utilisant trois des modèles sémantiques décrits ci-dessous, combiné avec la généralisation sémantique basée sur les dépendances syntaxiques, obtient un score de 64,6%.

Nous allons dans notre cas étudier trois méthodes différentes tant par leur complexité que par leur implémentation. Ainsi nous allons en premier évaluer une méthode de base, qui sera la plus simple et deux autres méthodes plus complexes.

#### 4. Sens le plus fréquent

Pour cette évaluation il nous faut une méthode de base (ou *baseline*) nous permettant de comparer les autres méthodes. Cette méthode est la plus simple et la plus rapide à implémenter. Notre approche baseline est le sens le plus fréquent. Cette méthode est très facile à implémenter et ne nécessite qu'"un" corpus annoté de sens. Les corpus facilement accessibles sont ceux de Semcor qui contiennent des mots désambiguïsés par le sens WordNet mais qui ne sont pas disponibles en grande quantité. Nous allons donc utiliser notre corpus d'apprentissage issu de Semcor. Cette approche bien que simpliste permet d'avoir quand même des résultats assez probants car certains mots sont toujours utilisés dans le même sens malgré leurs polymorphismes.

L'idée de base est donc de prendre à partir d'un corpus annoté le sens le plus fréquent de chaque mot à désambiguïser. Nous construisons ainsi un lexique qui va permettre de désambiguïser à partir de notre corpus d'entraînement pris dans Semcor. Pour la

---

<sup>17</sup> <http://www.senseval.org/>

désambiguïsation d'un texte nous n'avons qu'à prendre le sens se trouvant dans le lexique construit précédemment pour chaque mot. Lorsque le mot ne se trouve pas dans le lexique nous ne faisons aucune désambiguïsation. Cette méthode ne demande que peu de temps à mettre en place et à tester, mais pour qu'elle soit efficace il lui faut un assez gros corpus pour pouvoir connaître le maximum de mots, les mots inconnus n'étant pas traités. Nous allons donc nous intéresser à des méthodes plus complexes mettant en place des stratégies plus évoluées.

### 5. SenseLearner

SenseLearner (Rada Mihalcea and Andras Csomai, 2005) est une méthode permettant de résoudre l'ambiguïté sémantique de tous les mots d'un texte. L'algorithme peut être vu comme un algorithme de désambiguïsation du sens d'un mot minimal où il utilise un petit jeu de données pour l'entraînement et généralise les concepts appris sur les données d'entraînement pour désambiguïser les mots dans le jeu de données de test. En conséquence, l'algorithme apprend un modèle général pour chaque catégorie de mots. Les catégories de mots sont les catégories lexicales qui sont verbe, nom et adjectif.

Le but est donc ici d'utiliser le plus petit corpus possible, et dans le même temps de faire un algorithme assez général pour être capable de faire la désambiguïsation de la plupart des mots contenus dans un texte, et aussi assez efficace pour qu'un texte puisse être annoté en temps réel. Le corpus d'entraînement est basé sur Semcor.

L'entrée de l'algorithme de désambiguïsation est constituée d'un texte annoté avec des étiquettes morphosyntaxiques (POS) et avec des lemmes. Ce travail sera fait par Tree Taggers qui nous permet d'avoir l'étiquette POS et le lemme d'un mot (II.B Etiqueteur morphosyntaxique). La sortie est un texte avec le sens du mot annoté pour tous les mots de classes ouvertes. L'algorithme commence par un prétraitement où le texte est lemmatisé dans le cas où il ne l'est pas déjà. Les collocations sont identifiées en utilisant une approche de fenêtres coulissantes, où une collocation est définie par une séquence de mots dont la forme du concept composé est définie dans WordNet (Miller, 1995). Les entités nommées (noms propres, acronymes, etc.) sont aussi identifiées à cette étape.

Ensuite, le modèle sémantique est appris pour toutes les catégories (nom, verbe et adjectif) de mots prédéfinis qui sont définies comme groupes de mots partageant une syntaxe commune ou une propriété sémantique. Pour l'instant, en utilisant le mécanisme d'apprentissage de SenseLearner, un modèle de noms peut être défini et entraîné sur une partie des noms du corpus test. Nous pouvons avoir le même raisonnement avec les verbes. Une fois défini et entraîné, le modèle est utilisé pour annoter les mots ambigus dans le corpus de test avec son sens correspondant. Les différents modèles, qui sont implémentés dans SenseLearner, sont détaillés plus bas.

Notons que les modèles sémantiques sont applicables seulement sur les mots qui sont couverts par une catégorie lexicale dans le modèle et les mots apparus au moins une fois

dans le corpus d'entraînement. Les mots non couverts par ces modèles (à peu près 10-15% des mots du corpus de test) sont annotés avec le sens le plus fréquent de WordNet.

Différents modèles sémantiques peuvent être définis et entraînés pour la désambiguïsation des différentes catégories de mots. Bien que plus générale que les modèles qui ont été construits individuellement pour chaque mot du corpus de test (Decadt et al., 2004), l'application de ces modèles construits comme une partie de SenseLearner est limitée à certains mots vus préalablement dans le corpus d'entraînement, la couverture totale de ces modèles n'est donc pas de 100% mais est aidée par le sens le plus fréquent de WordNet.

A partir d'un corpus annoté constitué de tous les fichiers annotés de SemCor, des jeux de données d'entraînement sont construits pour chaque modèle. Il y a sept modèles fournis avec la distribution courante de SenseLearner, implémentant les caractéristiques suivantes :

- ModelNN1 : modèle conceptuel qui dépend du premier nom, verbe ou adjectif précédant le mot cible, et de leur correspondance tags part-of-speech.
- ModelNNColl : modèle de collocations qui implémente les caractéristiques de la meilleure collocation basée sur le premier mot à gauche et le premier mot à droite du nom cible.
- ModelVB1 : un modèle conceptuel qui dépend du premier mot précédent et du premier mot suivant le verbe cible, et de leurs correspondances tags part-of-speech.
- ModelVBColl : modèle de collocations qui implémente les caractéristiques de la meilleure collocation basée sur le premier mot à gauche et le premier mot à droite du verbe cible.
- ModelJJ1 : un modèle conceptuel qui dépend du premier nom après l'adjectif cible.
- ModelJJ2 : un modèle conceptuel qui dépend du premier mot précédent et du premier mot suivant l'adjectif cible et de leurs correspondances tags part-of-speech.
- ModelJJColl : modèle de collocations qui implémente les caractéristiques de la meilleure collocation basée sur le premier mot à gauche et le premier mot à droite de l'adjectif cible.

De nouveaux modèles peuvent être définis facilement et entraînés en suivant la même méthodologie d'apprentissage de SenseLearner. En fait, la distribution courante de SenseLearner inclus un *template* (ou schéma ou patron) pour le sous-programme requérant de définir de nouveau modèle sémantique, qui peuvent être facilement adapté pour de nouvelles catégories de mots.

Dans l'étape d'entraînement, un vecteur de caractéristiques est construit pour chaque mot annoté d'un sens couvert par un modèle sémantique. Les caractéristiques sont spécifiques au modèle et le vecteur de caractéristiques est ajouté au jeu d'entraînement qui se rapporte au modèle correspondant. L'étiquette de chacun des vecteurs de caractéristiques est constituée du mot cible et du sens correspondant, représenté par `word#sens`.

Pour annoter le nouveau texte, des vecteurs similaires sont créés pour tous les mots contenus dans le texte. Comme pour l'étape d'entraînement, les vecteurs de caractéristiques

sont créés et stockés séparément pour chaque modèle sémantique. Ces vecteurs sont ensuite utilisés pour la validation de la prédiction du sens.

Après, la prédiction du sens du mot est faite pour tous les exemples tests, avec un processus d'apprentissage séparé fait pour chaque modèle sémantique. Pour l'apprentissage, Timbl Memory basé sur un algorithme d'apprentissage (Daelemans et al., 2001) est utilisé, il a été auparavant trouvé pour la tâche de désambiguïisation de sens des mots (Hoste et al., 2002), (Mihalcea, 2002).

Après l'étape d'apprentissage, chaque vecteur du jeu de données test est étiqueté par la prédiction du mot et du sens. Si plusieurs modèles sont utilisés simultanément pour un cas donné, et que tous les modèles sont d'accord sur l'étiquette assignée, alors la prédiction peut être faite. Si le mot prédit par l'algorithme d'apprentissage coïncide avec le mot dans le vecteur de caractéristiques de test, alors la prédiction du sens est utilisée pour annoter le texte test. Au contraire, si la prédiction du mot est différente du mot, aucune annotation n'est produite et le mot est mis de côté pour être annoté plus tard.

Cette méthode reste quand même assez lente. De plus nous avons constaté que lorsque les corpus contenaient un nombre excessifs de mots, SenseLearner avait du mal à gérer la désambiguïisation, l'exécution devient alors très lente. Nous devons donc pour avoir une désambiguïisation en temps réel couper le corpus en plusieurs sous-corpus. Nous avons ensuite implémenté une méthode pour comparer à celle-ci et ainsi avoir une bonne évaluation pour notre choix final.

## 6. Naïve Bayes

Cette méthode est uniquement basée sur les probabilités. En effet nous allons apprendre un modèle à partir d'un corpus d'apprentissage dont les mots sont annotés par leur sens. Nous allons dans ce modèle trouver la probabilité d'un sens donné. Nous pourrions ainsi calculer la probabilité qu'un sens donné corresponde à un mot donné et selon un historique. Comme nous avons besoin d'un corpus pour faire un entraînement, cette méthode de désambiguïisation fait partie des méthodes supervisées.

Nous allons entrer plus dans les détails du calcul de la probabilité. En effet nous devons savoir quelle sera la probabilité d'un sens, pour la calculer nous allons essentiellement nous baser sur la probabilité suivante :

$$P(\text{Sens} | \mathcal{W}_n \mathcal{W}_{n-1} \mathcal{W}_{n-2}) = P(\mathcal{W}_n \mathcal{W}_{n-1} \mathcal{W}_{n-2} | \text{Sens}) \cdot P(\text{Sens})$$

Avec  $\mathcal{W}_n$  le mot à désambiguïiser,  $\mathcal{W}_{n-1} \mathcal{W}_{n-2}$  l'historique, *Sens* le sens possible du mot.

Nous prenons une hypothèse d'indépendance ce qui nous permet d'obtenir

$$P(\text{Sens} | \mathcal{W}_n \mathcal{W}_{n-1} \mathcal{W}_{n-2}) = P(\mathcal{W}_n | \text{Sens}) \cdot P(\mathcal{W}_{n-1} | \text{Sens}) \cdot P(\mathcal{W}_{n-2} | \text{Sens}) \cdot P(\text{Sens}).$$

En prenant le logarithme de la probabilité nous avons :

$$\log(P(\text{Sens} | \mathcal{W}_n \mathcal{W}_{n-1} \mathcal{W}_{n-2})) = \log(P(\text{Sens})) + \sum_{i=n-2}^{i=n} \log(P(\mathcal{W}_i | \text{Sens})).$$

Pour éviter d'avoir le logarithme de la probabilité infinie dans le cas où  $P(W_i|Sens) = 0$ , nous mettons un lissage sur  $P(W_i)$ . Ainsi nous trouvons la formule suivante :

$$\log(P(Sens|W_n W_{n-1} W_{n-2})) = \log(P(Sens)) + \sum_{i=n-2}^{i=n} \log(\lambda \cdot P(W_i|Sens) + (1-\lambda) \cdot P(W_i)).$$

En partant de cette probabilité nous cherchons le sens le plus probable. Nous allons donc maximiser ce résultat pour le trouver :  $\arg \max_{\forall Sens} (\log(P(Sens|W_n W_{n-1} W_{n-2})))$

Nous devons donc pour utiliser cette méthode construire un modèle de langue nous permettant de calculer les probabilités dont nous avons besoin. Nous utilisons pour cela le corpus train créé à partir de SemCor constitué de 193 474 mots désambiguïsés par le sens WordNet. Nous fabriquons ainsi notre modèle de langue grâce à une implémentation que nous avons fait. Ensuite nous pouvons désambiguïser un texte en trouvant pour chaque mot du texte le sens permettant de maximiser la probabilité. Cette méthode a un gros point faible qui est sa lenteur, en fait nous devons calculer un très grand nombre de probabilité ce qui le ralentit.

## 7. Evaluation

Nous avons évalué les trois désambiguïseurs sur le corpus test créé à partir de Semcor, ce corpus est disjoint de corpus Semcor d'entraînement. Le corpus est constitué de 70 626 mots, dont 33 089 mots assignés par leur sens Wordnet. Notre fichier de référence aura pour chaque mot les mots synonymes de son sens WordNet.

Nous utilisons pour la comparaison deux critères, le rappel et la précision. Le but étant bien sûr de maximiser les deux scores. La *précision* représente une mesure de l'efficacité du système par rapport au nombre de cas traités. Pourtant, elle n'est pas suffisante pour caractériser le comportement global du système parce qu'une précision de 100 % n'indique pas toujours un fonctionnement parfait. Par exemple, un système qui ne traite que seulement 2 cas sur un total de 10, même pour une précision de 100% (2 réponses correctes pour 2 cas traités), ne représente pas un système satisfaisant.

En revanche, le *rappel* tient compte de cet aspect, en indiquant pour l'exemple considéré une performance de 20 % de traitements corrects par rapport au nombre total des cas à traiter.

Les formules que nous avons utilisées pour le calcul des deux métriques sont :

$$prec = 100 \cdot \frac{nb. \text{ de réponses correctes}}{nb. \text{ de cas traités}}$$

$$rapp = 100 \cdot \frac{nb. \text{ de réponses correctes}}{nb. \text{ de cas à traiter}}.$$

	Baseline	Naïve bayes	SenseLearner
précision	51.5948	57.0942	91.5633
rappel	56.8044	66.9039	88.2468

*Tab 3.10: Evaluation des désambiguïseurs*

Les résultats sont rassemblés dans le tableau 3.10. Nous pouvons voir que SenseLearner a le meilleur résultat pour la précision et le rappel. Il est sans conteste le meilleur des désambiguïseurs testés dans notre étude. Cette méthode permet de très bien désambiguïser les mots et d'en traiter le maximum possible. Nous retrouvons la méthode Naïve Bayes en deuxième position. Ces résultats sont toutefois fort décevants car beaucoup moins bons que ceux de SenseLearner et assez proches de ceux de la Baseline. Nous retrouvons enfin en dernier, comme nous aurions pu le prédire, la méthode Baseline, étant la plus simple son résultat est tout à fait normal et nous permet de garantir l'efficacité des autres méthodes.

Nous avons donc décidé de prendre SenseLearner comme désambiguïseur vu ces bons résultats lors du test. Il nous permet d'avoir la meilleure désambiguïsation sans à avoir à choisir entre le nombre de mots à désambiguïser et l'efficacité de la désambiguïsation, le rappel et la précision obtenant un bon score. Nous avons donc tous les outils nécessaires à la collecte d'information sémantique, nous devons à présent les intégrer dans notre traduction.

## IV. Enrichissement du corpus d'apprentissage

La première approche que nous avons mise en place pour améliorer sur les performances est basée sur l'enrichissement du corpus d'apprentissage. En effet nous avons pu voir dans la section II que la traduction est basée sur un modèle de langue et un modèle de traduction. Nous allons tenter d'enrichir le corpus servant à la construction de ces modèles. Les enrichir va leur permettre d'avoir plus d'informations, de contenir plus de mots, plus de sens, ce qui, nous pensons, permettra d'avoir de meilleurs modèles. Nous proposons pour l'enrichissement d'introduire des paraphrases dans le corpus, nous gardons donc le même sens en intégrant un nouveau vocabulaire. Nous allons travailler sur le modèle de traduction donc sur les bitextes. Pour rappel, le modèle de traduction permet d'avoir la traduction d'un segment de mots. En enrichissant ce corpus, nous espérons donc enrichir les choix de traduction et ainsi l'améliorer. Nous devons donc savoir comment enrichir nos bitextes par des paraphrases, c'est l'objet de la section suivante.

### A. Création des paraphrases

Nous avons pour enrichir nos corpus pensé à construire des paraphrases, en effet en générant des paraphrases, nous pouvons garder le même sens de la phrase tout en utilisant des mots différents. Nous allons donc insérer un nouveau vocabulaire dans nos corpus d'entraînements. Nous allons utiliser la ressource WordNet pour faire cette tâche. Elle nous permet d'avoir le synonyme des mots des catégories lexicales nom, verbe, adverbe et adjectif. Malheureusement ils ne sont fournis que pour les mots anglais, donc nous ne pourrions ajouter les paraphrases que dans le texte en anglais. Nous devons tout de même préserver l'alignement phrastique des deux corpus, mais comme nous ajoutons des paraphrases, elles gardent le même sens donc nous pouvons prendre la même phrase que pour l'alignement original. Nous nous limiterons aussi à cause de cette contrainte à la traduction de l'anglais vers le français. Nous rappelons que dans le système que nous utilisons, le modèle de traduction est un modèle "inversé", c'est à dire qui donne la probabilité d'une séquence anglaise étant donnée une séquence de langue française.

Le plus gros travail ici sera donc de construire nos paraphrases. Nous appelons paraphrase, une phrase dont au moins un mot varie par rapport à la phrase originale. Plusieurs problèmes peuvent apparaître lors de la mise en place de la méthode. Nous devons dans un premier temps éviter le plus possible un glissement de sens. En effet, il est possible de modifier le sens d'un mot lors de son remplacement par un synonyme. Par exemple, dans la phrase "il ferme le verrou", le mot "ferme" peut être remplacé par le synonyme "exploitation agricole", si nous prenons ce synonyme et non le synonyme du verbe "ferme", nous allons donc nous retrouver avec la phrase "il exploitation agricole le verrou".

Pour fabriquer nos paraphrases nous allons utiliser un principe assez simple pour une première approche à laquelle nous rajouterons des filtres qui nous permettront d'affiner la

méthode. Nous allons dans un premier temps prendre tous les mots du vocabulaire et rechercher leurs synonymes dans la base de données WordNet. Nous produisons un fichier contenant sur chaque ligne un mot suivi de tous ses synonymes et de leur probabilité qui est par défaut 1. Nous ajoutons le mot original dans la liste des synonymes pour une raison d'équitabilité des cas qui sera présentée dans le paragraphe suivant. Nous nous sommes restreints à la relation de symétrie pour ne pas fabriquer trop de bruit dans le corpus en y infiltrant des mots trop éloignés du mot original. Nous appellerons ce fichier le tableau des synonymes par la suite.

```
arrivals arrivals 1 reaching 1 arriver 1 comer 1
arrive arrive 1 get 1 come 1 make_it 1 get_in 1 go_far 1
arrived arrived 1 get 1 come 1 make_it 1 get_in 1 go_far 1
arrives arrives 1 get 1 come 1 make_it 1 get_in 1 go_far 1
arriving arriving 1 get 1 come 1 make_it 1 get_in 1 go_far 1
arrogance arrogance 1 haughtiness 1 hauteur 1 high-handedness 1 lordliness 1
arrogant arrogant 1 chesty 1 self-important 1
arrogantly arrogantly 1
arrows arrows 1 pointer 1
arrest arrest 1
arsenal arsenal 1 armory 1 armoury 1 armory 1 armoury 1 armory 1 armoury 1
```

*Fig 4.1 Tableau des synonymes*

Nous allons ensuite construire nos paraphrases. Nous allons pour cette construction utiliser une fonction score qui prend en entrée le tableau des synonymes et la phrase initiale et produit en sortie des paraphrases différentes. Cette fonction peut être formulée de la manière suivante :

$$\hat{e}_1^n = \arg \max_{\bar{e}_1^n} (\text{score}(\bar{e}_1^n, e_1^n))$$

Où  $\text{score}(\bar{e}_1^n, e_1^n)$  est une fonction qui quantifie la qualité de la paraphrase  $\bar{e}_1^n$  de la phrase  $e_1^n$ . Nous allons pour calculer cette qualité utiliser le modèle de langue qui nous permet d'obtenir un score de probabilité d'existence d'un mot. Notre modèle de langue utilisera l'algorithme Kneser Ney non modifié avec la prise en compte des mots inconnus d'ordre 2 et 3. Pour prendre en compte l'ensemble de la phrase nous allons utiliser un algorithme de Viterbi (Viterbi, 1967) qui nous permet de calculer rapidement la qualité de plusieurs mots consécutifs. Pour rappel Viterbi est un algorithme permettant de trouver la séquence d'état la plus probable, les états étant ici les mots et les probabilités calculées par le modèle de langue. Les mots susceptibles d'être mis dans la phrase à la place d'un autre sont choisis dans la liste des synonymes du mot se trouvant dans le tableau des synonymes. C'est donc pour cette raison que nous avons ajouté dans les synonymes le mot original, car il peut ne pas être modifié et ce cas doit donc apparaître. Nous devons aussi créer une paraphrase en plus de celle que nous devons construire. En effet la phrase originale peut se retrouver en paraphrase nous devons donc prendre la paraphrase la plus probable après celle-ci à la place.

Nous avons dans un premier temps implémenté une méthode permettant de faire cette tâche. Notre implémentation s'est relevée trop lente pour notre application. Nous nous sommes donc tournés vers une méthode de SRLIM qui nous permet de faire la même chose en un temps plus convenable. Cette méthode est la fonction Disambig de SRILM. Disambig traduit un flot de mots d'un vocabulaire V1 à partir de la correspondance d'un flot de mots d'un vocabulaire V2, selon des probabilités, ce qui correspond exactement à nos besoins. Les ambiguïtés dans le tableau des synonymes sont résolues en trouvant les séquences V2 ayant la plus grande probabilité de donner la séquence V1. Cette probabilité est calculée par la probabilité conditionnelle  $P(V1|V2)$ , mais aussi avec le modèle de langue pour la séquence sur la séquence V2. Nous lui passons donc en entrée le texte original, notre modèle de langue qui lui permettra de calculer les probabilités et bien sûr notre tableau de synonymes. Nous retrouvons en sortie un fichier avec les paraphrases créées (fig 4.2).

```
NBEST_0 -3.49459 <s> tabling of documents
NBEST_1 -8.43534 <s> tabling of papers
NBEST_2 -12.0357 <s> defer of documents
NBEST_3 -12.2952 <s> postpone of documents
```

*Fig 4.2.a : paraphrases de "tabling of documents"*

```
NBEST_0 -3.11769 <s> house of commons
NBEST_1 -9.50941 <s> house of park
NBEST_2 -9.50941 <s> house of green
NBEST_3 -11.2918 <s> firm of commons
```

*Fig 4.2.b : paraphrases de "house of commons"*

```
NBEST_0 -6.72951 <s> thursday , april 17 , 1986
NBEST_1 -107.17 <s> Thursday , april 17 , 1986
NBEST_2 -107.17 <s> Th , april 17 , 1986
NBEST_3 -108.877 <s> thursday , april seventeen , 1986
```

*Fig 4.2.c : paraphrases de "thursday , april 17 , 1986"*

```
NBEST_0 -4.5093 <s> the house met at 11 a.m.
NBEST_1 -11.3385 <s> the family met at 11 a.m.
NBEST_2 -11.7104 <s> the home met at 11 a.m.
NBEST_3 -12.1968 <s> the firm met at 11 a.m.
```

*Fig 4.2.d : paraphrases de "the house met at 11 a.m."*

*Fig 4.2 : 4 exemples de 4 paraphrases créées par Disambig*

Une fois que nous avons le fichier des paraphrases il nous faut ensuite ajouter les paraphrases dans le corpus original et réaligner nos bitextes. Nous devons supprimer la paraphrase supplémentaire qui sera soit l'identique de la phrase originale soit la plus mauvaise et ajouter autant de phrases dans le corpus français que de paraphrases. Nous pouvons noter qu'il n'y aura pas tout le temps le nombre voulu de paraphrases, en effet dans certain cas nous

## Chapitre 4 - Enrichissement du corpus d'apprentissage

ne pourrons construire autant de paraphrases que nécessaire à cause du nombre limité de synonymes.

### **government response**

government answer

government reply

government reaction

### **canadian charter of rights and freedoms**

canadian charter of rights and exemption

canadian take of rights and freedoms

canadian hire of rights and freedoms

### **fourth anniversary of proclamation**

fourth anniversary of announcement

fourth anniversary of declaration

quarter anniversary of proclamation

### **mr. speaker , our government has demonstrated its support for these important principles by taking steps to enforce the provisions of the charter more vigourously .**

mr. speaker , our government has demonstrated its support for these important principles by taking steps to implement the provisions of the charter more vigourously .

mr. speaker , our government has demonstrated its support for these important principles by taking steps to apply the provisions of the charter more vigourously .

*Fig 4.3 : Extrait du corpus anglais enrichi (phrase original en gras)*

*réponse du gouvernement*

*réponse du gouvernement*

*réponse du gouvernement*

*réponse du gouvernement*

*la charte canadienne des droits et libertés*

*la charte canadienne des droits et libertés*

*la charte canadienne des droits et libertés*

*la charte canadienne des droits et libertés*

*quatrième anniversaire de la proclamation*

*quatrième anniversaire de la proclamation*

*quatrième anniversaire de la proclamation*

*quatrième anniversaire de la proclamation*

*monsieur le président , notre gouvernement a prouvé son adhésion à ces importants principes en prenant des mesures pour appliquer plus systématiquement les préceptes de la charte .*

*monsieur le président , notre gouvernement a prouvé son adhésion à ces importants principes en prenant des mesures pour appliquer plus systématiquement les préceptes de la charte .*

*monsieur le président , notre gouvernement a prouvé son adhésion à ces importants principes en prenant des mesures pour appliquer plus systématiquement les préceptes de la charte .*

*Fig 4.4 : Extrait du corpus français aligné au corpus anglais enrichi (fig 4.3)*

Nous nous retrouvons alors avec deux corpus, un dans la langue cible et un dans la langue source, alignés, la *i*-ème phrase du corpus en langue cible correspondant à la *i*-ème phrase du corpus de la langue source, avec l'enrichissement du corpus en anglais par des paraphrases.

Nous pouvons voir sur la figure 4.3 un exemple d'un corpus enrichi. Nous avons enrichi ce corpus de trois paraphrases construites à partir d'un modèle trigramme. Nous pouvons voir sur la figure 4.4 le corpus français aligné au corpus anglais enrichi.

Dans notre méthode initiale nous avons ajouté tous les synonymes possibles pour un mot. Des erreurs de synonymes peuvent alors facilement se glisser dans la liste et ainsi entraîner la construction de mauvaises paraphrases. Nous pensons que limiter ces synonymes permettrait de limiter l'insertion d'un mauvais synonyme dans les paraphrases générées. Les paraphrases auraient une majorité de bons synonymes pour leur construction. Nous allons utiliser pour créer ces filtres de l'information sémantique car nous avons besoin de connaître le sens des mots pour pouvoir faire une sélection. Nous avons vu dans la section III deux manières d'obtenir l'information. Nous allons mettre en place deux filtres différents :

- un filtre utilisant l'étiquetage POS
- un filtre utilisant le WSD

Dans un premier temps nous avons ajouté comme critère la prise en compte de la catégorie lexicale du mot. En effet comme nous avons pu le voir, les mots sont rangés par catégories lexicales dans WordNet. Jusqu'à présent nous prenions tous les sens de toutes les catégories lexicales. Or nous savons aussi que des mots ont des sens différents en fonction de leur catégorie lexicale, par exemple "voile" n'a pas le même sens si ce mot est un verbe ou un nom. Pour permettre d'avoir l'accès à cette information nous nous sommes tournés vers un étiqueteur syntaxique (voir section II.B). Nous avons appliqué Tree Tagger à notre corpus. Nous utilisons cette étiquette pour filtrer les sens possibles des mots. Nous réduisons ainsi la table des synonymes et espérons ainsi réduire le nombre de cas énoncés. Nous allons donc avoir un corpus annoté par des étiquettes qui vont nous permettre de filtrer la recherche dans WordNet. Cette fois nous n'allons donc prendre que les synonymes correspondant à la catégorie syntaxique du mot au lieu de prendre tous les synonymes ainsi nous pourrions limiter les glissements de sens des mots. De plus en filtrant les synonymes, nous allons avoir moins de choix à tester donc la méthode peut gagner en rapidité lors de la création des paraphrases. Cette méthode n'a pas pour but de faire une désambiguïsation, elle n'a pour but que de restreindre l'espace des synonymes pour améliorer la probabilité de garder le même sens. Nous nous sommes donc tournés dans un second temps vers un vrai désambiguïseur qui permettra alors de faire une désambiguïsation totale.

Nous avons voulu insérer un désambiguïseur dans la construction de nos paraphrases. Nous avons évalué plusieurs ressources (III.C Algorithme de désambiguïsation) pour choisir SenseLearner qui nous paraît le meilleur. SenseLearner nous permet d'avoir les mots annotés par leur sens WordNet. En effet SenseLearner nous permet d'avoir de suite les synonymes du bon sens et ainsi avoir un minimum de glissement de sens, mais il reste toujours les erreurs

possibles de SenseLearner. Le nombre de synonymes est donc considérablement réduit ce qui par la même occasion réduit le temps pour la construction des paraphrases puisque nous avons moins de cas à traiter. La désambiguïsation permet donc de mettre un filtre non négligeable dans le choix de nos paraphrases ce qui permettra sûrement de réduire les erreurs faites lors du choix du synonyme.

Nous allons ensuite lors de nos tests pouvoir évaluer les répercussions de ces différents filtres. En effet nous avons essayé de faire une ressource avec un maximum d'outils ce qui nous permettra de mesurer l'impact de ces différentes options sur nos traductions. Ainsi nous pourrons apporter une analyse complète et en tirer les meilleures conclusions. Nous avons fait toute une série de tests préalables sur des corpus unilingues pour avoir une idée du comportement du corpus enrichi avant de passer à la traduction et de faire des tests plus approfondis.

### B. Evaluation

Nous avons donc lors de nos expériences voulu savoir si notre hypothèse selon laquelle l'enrichissement du corpus permettrait d'améliorer les résultats pouvait être vérifiée. En effet un corpus avec des paraphrases pourrait apporter plus de caractéristiques car il contient plus de mots tout en gardant le sens général du texte puisque les paraphrases ont le même sens que la phrase originale. Dans un premier temps nous avons voulu voir les effets de cette méthode sur les modèles de langue, nous avons donc travaillé sur des corpus unilingue.

#### 1. Test sur les modèles de langue

Dans cette première partie de test nous nous sommes intéressés à évaluer les effets que pourraient engendrer l'enrichissement d'un corpus sur un modèle de langue. Pour évaluer un modèle de langue, il nous faut tout d'abord construire ce modèle à partir d'un corpus d'apprentissage monolingue. Ensuite nous allons mesurer celui-ci à partir de sa perplexité. La perplexité représente approximativement le nombre moyen de mots auquel doit faire face un modèle lors d'une prédiction. Donc plus la perplexité est petite plus le modèle de langue est efficace. Cette métrique se mesure donc à partir d'un texte de test, différent bien sûr du corpus d'entraînement, avec le modèle de langue.

Nous prenons un corpus d'entraînement issu du Hansard, qui regroupe les textes parlementaires canadiens. Il contient 500 000 phrases, comprenant 9 013 395 mots. Nous avons donc une moyenne de 18,03 de mots par phrase, la plus grande phrase ayant 79 mots. Notre fichier test est lui composé de 1 000 phrases et il est issu lui aussi du Hansard. Il est bien sûr différent du corpus d'apprentissage. Il contient 11 284 mots, avec une moyenne de 11,28 mots par phrase, 65 mots pour la phrase la plus longue.

L'enrichissement est fait sur le corpus d'apprentissage car c'est lui qui nous permet de construire le modèle. Nous nous sommes intéressés à différents facteurs pouvant influencer sur les

performances d'une telle approche. Ces facteurs ont été détaillés dans la section II.A. Plus nous étudions de cas plus nous pourrions évaluer en détails l'impact sur la traduction.

Nous avons dans un premier temps créé des corpus de tailles différentes à partir du corpus d'origine pour voir l'impact de l'enrichissement selon la taille du corpus. Nous avons pour cela pris 6 tailles de corpus différentes (1 000, 5 000, 10 000, 50 000, 100 000 et 500 000 phrases) Nous allons par la suite appliquer sur chaque corpus des constructions de paraphrases différentes.

Nous avons vu dans le paragraphe (III.A Création des paraphrases), que nous pouvions construire nos paraphrases de différentes façons. Nous avons de plus étudié différentes façons d'obtenir nos corpus enrichis. Nous allons tout d'abord jouer sur la longueur du nouveau corpus en y insérant plus ou moins de paraphrases. Nous allons ainsi construire des corpus contenant 1, 5 et 10 paraphrases. Nous avons essayé de prendre des nombres permettant d'avoir un éventail de cas ; ajouter plus de 10 paraphrases apporterait selon nous beaucoup trop de chance d'insérer des glissements de sens dans les paraphrases. Nous allons ainsi avoir des corpus contenant plus ou moins d'informations grâce au nombre différent de paraphrases ajoutées mais aussi augmenter ou diminuer les chances de probabilités d'avoir un glissement de sens. En effet la dixième paraphrase est faite avec les synonymes qui ont été jusqu'ici écartés donc qui ont plus de chance d'appartenir à un autre sens.

Nous avons ensuite voulu tester l'impact que le modèle de langue servant à la construction des paraphrases avait. Nous avons donc décidé d'utiliser un modèle bigramme et un modèle trigramme. Le modèle trigramme permet d'avoir un plus grand historique que le modèle bigramme donc apporte plus de contraintes.

Nous avons ensuite voulu voir ce que l'insertion d'une aide à la désambiguïsation pouvait apporter comme amélioration à la méthode de base. Nous avons donc utilisé la méthode s'aidant des étiquettes morphosyntaxiques pour le choix des synonymes. Nous pourrions ainsi comparer cette méthode et la méthode de base et voir si elle permet d'avoir un impact positif.

Mais tout d'abord nous allons nous intéresser à une option de SRLIM. L'option "interpolate" permet de combiner le modèle avec des modèles inférieurs. Elle peut s'employer sur la méthode Kneser-Ney. Nous voulons savoir si cette prise en compte de modèle plus petit pourrait permettre d'améliorer les modèles. Nous l'avons donc étudié de plus près. Au cours de nos expériences nous utilisons deux modèles de langue, le modèle de langue servant à la construction des paraphrases que nous appellerons modèle construction et le modèle servant à l'évaluation du corpus que nous appellerons modèle évaluation. Nous avons donc testé l'impact de l'option sur les deux modèles en l'utilisant soit pour les deux modèles, soit pour le modèle d'évaluation soit pour aucun des modèles. Nous allons ainsi nous faire une idée plus précise de cette option qui reste assez floue dans la documentation de SRLIM et ainsi décider de l'utiliser ou non pour les futurs tests.

Nous pouvons retrouver les résultats de nos nouveaux corpus créés avec les différentes options citées au-dessus. La notation permettant de distinguer chaque cas est  $x \text{ para } y \text{ mod } [+$

tag], où  $x$  est le nombre de paraphrases ajoutées,  $y$  l'ordre du modèle utilisé (2 pour bigramme et 3 pour trigramme) et la notation [+ tag] n'est ajouté que dans les cas où l'étiquetage est intervenu dans la construction des paraphrases. Les corpus originaux sont tout simplement notés original. Nous retrouvons en abscisse les différents corpus rangés en fonction de leur taille et en ordonné la perplexité.

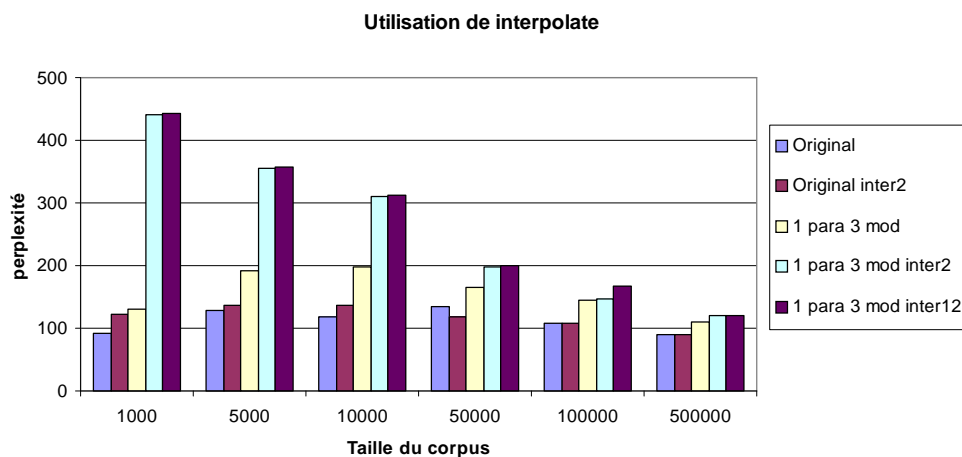


Fig 4.5 : Evaluation de l'utilisation de interpolate

Nous nous sommes donc dans un premier temps penchés sur l'utilisation de l'option interpolate. Nous avons regroupé sur le graphe 4.5 toutes les informations nécessaires à son évaluation. Nous nous sommes donc intéressés à des cas bien particuliers qui nous permettent d'avoir tout ce qui faut pour l'étudier. Nous avons donc pris 5 cas bien précis pour chaque taille de corpus :

- Original avec la non utilisation de l'option interpolate pour les deux modèles: Original
- Original avec l'utilisation de l'option interpolate pour le modèle d'évaluation: Original iner2
- Corpus enrichi avec 1 paraphrase construite avec un modèle trigramme avec l'option interpolate utilisée pour aucun des deux modèles: 1 para 3 mod
- Corpus enrichi avec 1 paraphrase construite avec un modèle trigramme avec l'option interpolate utilisée pour le modèle évaluation: 1 para 3 mod inter2
- Corpus enrichi avec 1 paraphrase construite avec un modèle trigramme avec l'option interpolate utilisée pour les deux modèles: 1 para 3 mod inter12

Nous voyons tout d'abord que pour les corpus originaux, l'option ne permet pas d'avoir de meilleurs résultats, étant qu'une seule fois meilleure, mais étant très proche tout le temps du résultat sans l'option. Il n'y a donc pas de changement assez significatif. Nous remarquons au contraire que pour les corpus enrichis, la différence est beaucoup plus marquante, et l'utilisation de l'option n'apporte qu'une augmentation de la perplexité. Cette augmentation est moins importante dans les gros corpus mais existe quand même. De plus les résultats sont plus mauvais lorsque l'option est utilisée sur les deux modèles que lorsqu'elle est uniquement utilisée sur le modèle d'évaluation. Au vue de ces résultats nous avons décidé de ne pas utiliser

l'option "interpolate" lors de la construction des modèles fait sur des corpus enrichis. Nous allons tout de même présenter tous nos résultats dans cette partie y compris ceux fait avec l'option.

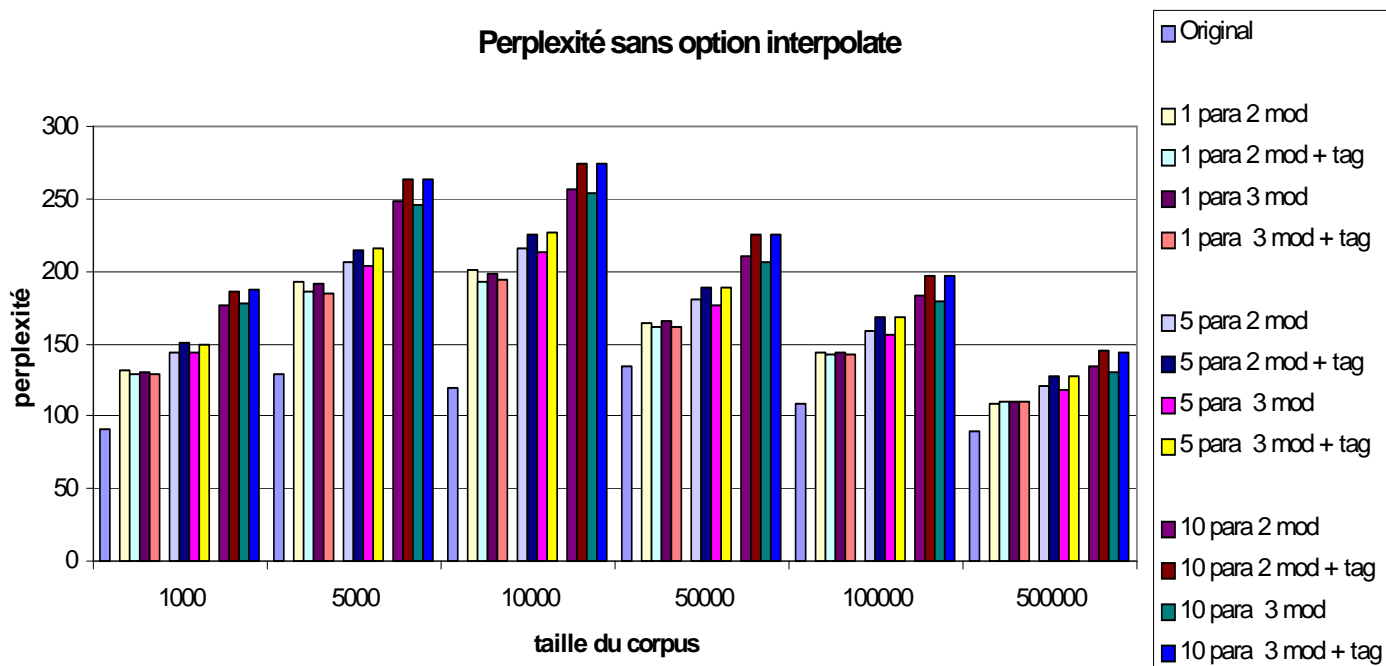


Fig 4.6 : Perplexité sans option interpolate

Nous avons représenté en figure 4.6 les résultats obtenus sans l'utilisation de l'option "interpolate". Nous avons pour chaque taille de corpus, 13 cas différents, tous ces cas ont été présentés précédemment. Nous pouvons faire le constat général que les nouveaux corpus ne permettent aucune amélioration du modèle. En effet, à chaque fois la perplexité est supérieure au modèle de base qui est construit à partir du corpus original. Cette différence diminue lors de l'utilisation de corpus de plus en plus gros. Nous pouvons ensuite faire un deuxième constat évident sur le nombre de paraphrases et la qualité du modèle. Nous pouvons voir que les corpus ayant le plus petit nombre de paraphrases sont meilleurs que ceux qui en contiennent plus. Sur la figure 4.6, les corpus ayant une paraphrase, situés dans le deuxième groupe de données, sont meilleurs que ceux contenant cinq paraphrases, situés dans le troisième groupe de données. De plus les corpus contenant dix paraphrases, situés dans le quatrième groupe de données sont plus mauvais que ceux avec une ou cinq paraphrases. Nous pouvons donc dire que plus il y a de paraphrases plus la perplexité est mauvaise. En regardant chaque groupe d'ajout de paraphrases, nous retrouvons pratiquement toujours le même ordre dans la qualité des résultats. Ainsi l'utilisation du modèle trigramme nous permet d'avoir un résultat légèrement meilleur qu'avec l'utilisation du bigramme. Nous voyons aussi que l'utilisation de l'étiquette lors de la construction des paraphrases ne permet pas tout le temps d'améliorer le score. En effet il permet de diminuer l'erreur lorsque nous ajoutons une paraphrase au corpus mais elle ne fait qu'augmenter la perplexité dans les cas d'ajout de cinq et dix paraphrases.

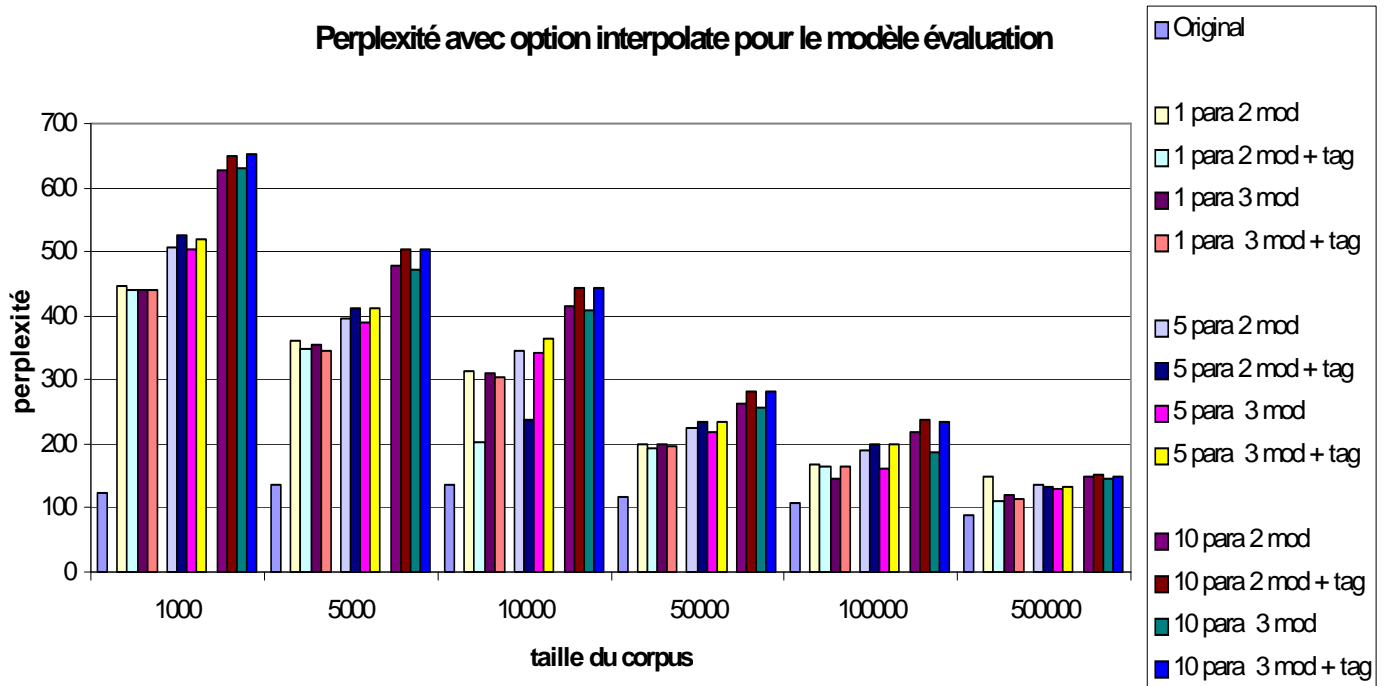


Fig 4.7 : Perplexité avec option interpolate pour le modèle évaluation

Nous voyons sur la figure 4.7 les résultats obtenus sur les corpus avec l'utilisation de l'option pour le modèle évaluation. Nous pouvons remarquer tout de même que les résultats obtenus même si leur forme générale est la même sont beaucoup plus mauvais que ceux de la figure 4.6 surtout pour les petits corpus. En effet pour le cas d'un corpus enrichi d'une paraphrase avec l'utilisation d'un modèle trigramme pour le modèle de construction nous obtenons un résultat de 441,481 alors que précédemment nous avons un résultat de 130,114. Nous pouvons dire que sur les effets des différents cas sur la perplexité nous pouvons en règle générale faire les mêmes remarques que pour la figure 4.6. En effet nous avons toujours l'augmentation de la perplexité en fonction du nombre de paraphrases ajoutées mais aussi une moins grande perplexité lors de l'utilisation d'un modèle de construction d'ordre 3 plutôt qu'un modèle d'ordre 2. Cette différence est un peu plus marquée ici avec une différence de 27 pour le cas de l'ajout de cinq paraphrases dans le corpus de 100 000 phrases. Nous avons aussi la même remarque sur les effets de l'utilisation de l'étiquetage dans l'enrichissement du corpus, en effet celui-ci ne permet d'obtenir des résultats moins mauvais que dans le cas de l'ajout d'une paraphrase, dans les autres cas son utilisation n'apporte qu'une augmentation de la perplexité donc ne fait que rendre le résultat plus mauvais.

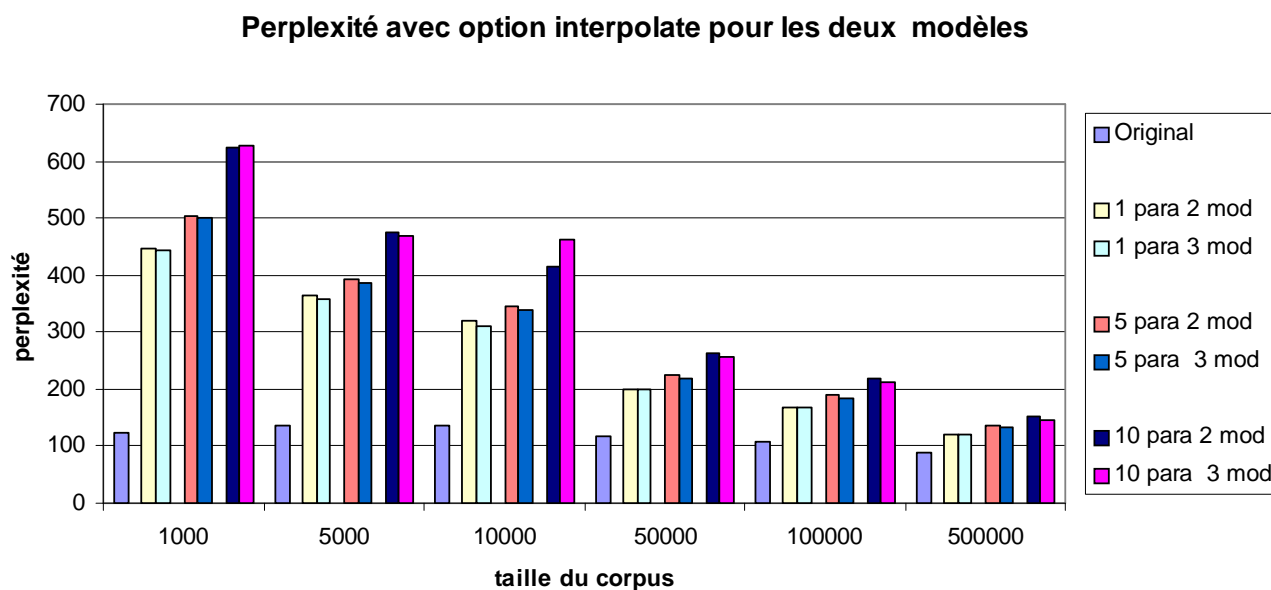


Fig 4.8 : Perplexité avec option interpolate pour les deux modèles

Nous pouvons observer sur la figure 4.8 les résultats obtenus avec l'utilisation de l'option "interpolate" dans le modèle de construction et le modèle d'évaluation. Encore une fois les mêmes remarques peuvent être faites. Les perplexités sont plus élevées que dans le cas où nous n'utilisons jamais l'option. De plus la différence avec le modèle original reste moins grande avec l'utilisation du trigramme pour le modèle de construction que le bigramme. La perplexité est toujours plus élevée lorsque nous ajoutons le plus de paraphrases.

D'après tous nos tests, nous pouvons faire le constat général que l'augmentation du corpus n'a aucun effet positif sur la perplexité bien au contraire. En effet les perplexités des nouveaux corpus créés sont plus grandes que celles de l'original donc plus mauvaises.

Nous pouvons voir que l'utilisation du modèle trigramme nous permet d'obtenir de meilleurs résultats que l'utilisation du modèle bigramme. Nous pouvons expliquer ce phénomène par le fait que le modèle trigramme nous permet d'augmenter la restriction des mots synonymes et ainsi choisir les meilleurs. Par le même fait nous pouvons restreindre les erreurs de sens possibles en remplaçant un mot.

Nous avons aussi mis à jour une correspondance entre le nombre de paraphrases ajoutées et la qualité du modèle. En effet nous avons pu constater que plus nous ajoutons de paraphrases plus le modèle est mauvais. Nous savons que chaque fois que nous ajoutons une paraphrase nous avons une probabilité, que nous noterons  $p$ , d'avoir un glissement de sens. Les glissements de sens en changeant le sens du texte original lui insèrent des erreurs et ne font que rendre le modèle plus mauvais. Si nous ajoutons donc  $n$  paraphrases nous aurons cette fois  $n \cdot p$  probabilités de commettre des glissements de sens donc de rendre le modèle plus mauvais,  $n$  étant supérieur à 0  $n \cdot p > p$  donc plus nous ajoutons de paraphrases plus le modèle a de chance de devenir plus mauvais. Nous pouvons constater de plus que l'augmentation de l'erreur n'est pas linéaire par rapport au nombre de paraphrases ajoutées. En effet l'erreur entre l'ajout de 5

et 10 paraphrases n'est pas proportionnelle à 2. Plus nous ajoutons de paraphrases plus nous pouvons commettre des erreurs de sens. En effet plus nous faisons de paraphrases plus nous avons tendance à utiliser des mots d'un autre sens car nous devons trouver obligatoirement une paraphrase non utilisée donc prendre des synonymes non utilisés, non utilisés justement jusqu'ici car probablement d'un autre sens.

L'utilisation des étiquettes morphosyntaxiques pour la construction permet dans le cas d'ajout d'une paraphrase d'obtenir des résultats légèrement meilleurs. Ce phénomène est expliqué par le fait que l'utilisation de cette donnée permet de faire une légère "désambiguïsation" en ne prenant que les synonymes correspondant à la catégorie syntaxique du mot. Malgré tout elle ne permet en aucun cas de prendre le bon sens. De plus nous sommes contraints à une répercussion des erreurs. En effet, l'étiquetage d'un texte induit des erreurs, erreurs qui ensuite peuvent encore induire des erreurs dans le choix du synonyme.

Des problèmes d'ordre général peuvent aussi être mis en lumière. Tout d'abord nous sommes confrontés à un problème de conjugaison, en effet les synonymes donnés par WordNet sont des mots lemmatisés, donc nous perdons toutes formes de conjugaisons. En effet en enlevant la conjugaison des mots nous ne pouvons que rajouter des erreurs dans le texte. Ainsi nous pouvons penser que ce problème est un des principaux que nous allons tenter de modifier par la suite. Ainsi nous devons donc mettre en place la conjugaison des noms et des verbes pour pouvoir espérer faire moins d'erreur.

Dans un constat final, nous pouvons dire que les nouveaux modèles créés à partir des corpus agrandi par les paraphrases ne sont pas meilleurs que les modèles d'origine bien au contraire. Ces résultats sont dus principalement aux erreurs ajoutés dans les nouveaux corpus. En effet nous pouvons dire que nos paraphrases peuvent dans plusieurs cas explicités au-dessus, engendrer des erreurs, comme le glissement de sens dans nos corpus et donc nos modèles. Nous pouvons en déduire que la principale cause de ces résultats est la construction des paraphrases qui n'est pas assez efficace. Nous devons donc nous pencher sur ce problème pour essayer d'avoir de meilleurs résultats.

### **2. Test sur la traduction**

Nous avons dans une seconde série d'expériences voulu savoir quel impact avait l'ajout de paraphrases dans le corpus d'entraînement pour un système de traduction. Nous utilisons pour cela un système de traduction phrase-based que nous avons expliqué dans un paragraphe précédent (I.D. Traduction). Nous avons donc entrepris de construire de nouveaux corpus, augmentés par des paraphrases, à partir desquels nous construisons nos modèles de traduction. Nous avons pris pour ces tests un corpus d'apprentissage issu du Hansard correspondant aux textes parlementaires canadiens, comprenant 1 753 443 de phrases disponibles en anglais et en français. Le texte anglais contient 31 637 628 mots avec 99 mots pour la phrase la plus grande. Le texte français contient 34 150 043 mots avec 40 mots pour la phrase la plus grande. Nous avons aussi à notre disposition deux corpus de test. C'est sur ces corpus que seront faits les traductions que nous évaluerons. Nous avons donc un corpus de test principal tiré du Hansard,

qui est bien sûr différent du corpus d'entraînement. Ce texte contient 1 000 phrases et 23 967 mots pour la partie anglaise et 23 504 mots pour la partie française. Nous avons un deuxième texte qui ne nous servira que pour un test, ce corpus est issu de EuroParl, qui regroupe les textes parlementaires européens. La partie anglaise de ce fichier contient 28 519 mots et la partie française en contient 30 178.

Comme pour les tests précédents, nous avons voulu savoir l'impact du plus grand nombre de situations possibles pour ainsi effectuer la meilleure analyse possible. Nous avons donc découpé notre corpus en corpus de tailles différentes. Ainsi nous pourrions analyser l'impact de l'enrichissement sur des corpus de tailles variées. Nous nous retrouvons donc au début de nos tests avec cinq corpus contenant 50 000 phrases, 100 000 phrases, 500 000 phrases, 1 000 000 phrases et 1 753 443 phrases.

Nos différents cas ne se limiteront pas uniquement aux différents corpus de départ. Nous allons pour chacun des corpus faire des enrichissements de corpus avec différentes caractéristiques. Nous allons dans un premier temps faire un enrichissement avec des nombres différents de paraphrases. En effet nous allons faire des ajouts de 1 ou 3 paraphrases au corpus original pour voir l'impact du nombre de nouvelles phrases ajoutées dans le corpus sur l'efficacité du modèle et ainsi l'efficacité de la traduction.

Nous allons ensuite utiliser des modèles différents pour une construction différente des paraphrases. En effet ce modèle sera basé sur les bigrammes ou sur les trigrammes. Par contre, pour ces tests nous n'avons jamais utilisé l'option "interpolate" pour le modèle de construction. En effet selon les résultats trouvés dans le paragraphe (III.B.1 Test sur les modèles de langue), nous avons jugé que son utilisation n'apportait pas d'amélioration dans la construction des paraphrases. L'option sera quand même utilisée lors de la création du modèle de langue servant à la traduction, car elle apporte une amélioration de la traduction.

Enfin nous avons voulu savoir l'impact sur la qualité des modèles des différentes aides associées à la construction des paraphrases. Nous pouvons en effet apporter de l'information supplémentaire en utilisant l'étiquetage morphosyntaxique ou en utilisant un désambiguïseur. Nous allons donc tester la méthode utilisant l'étiquetage et la méthode utilisant le désambiguïseur. Ces méthodes seront bien sûr comparées à la méthode de base.

Après la création de tous nos nouveaux corpus, nous avons utilisé les ressources habituelles du RALI pour faire la traduction. Nous avons donc créé nos deux modèles grâce à GIZA++ pour le modèle de traduction et SRILM pour le modèle de langue. Le modèle de traduction est bien sûr créé à partir du corpus enrichi. Le modèle de langue quant à lui sera construit à partir du corpus français d'origine. En effet nous ajoutons dans le corpus français que des copies de la phrase d'origine, ces répétitions n'apporteraient qu'une mauvaise modélisation de la langue. Nous faisons ensuite la traduction du texte test anglais vers le français. Cette traduction est faite par Pharaoh avec les modèles créés précédemment. Nous devons ensuite calculer le score de la traduction en le comparant avec le fichier test original de la langue cible. Nous utilisons le score bleu comme métrique d'évaluation qui est une mesure de précision n-gramme.

Nous regroupons tous les scores trouvés sur les graphes ci dessous pour une meilleure visualisation.

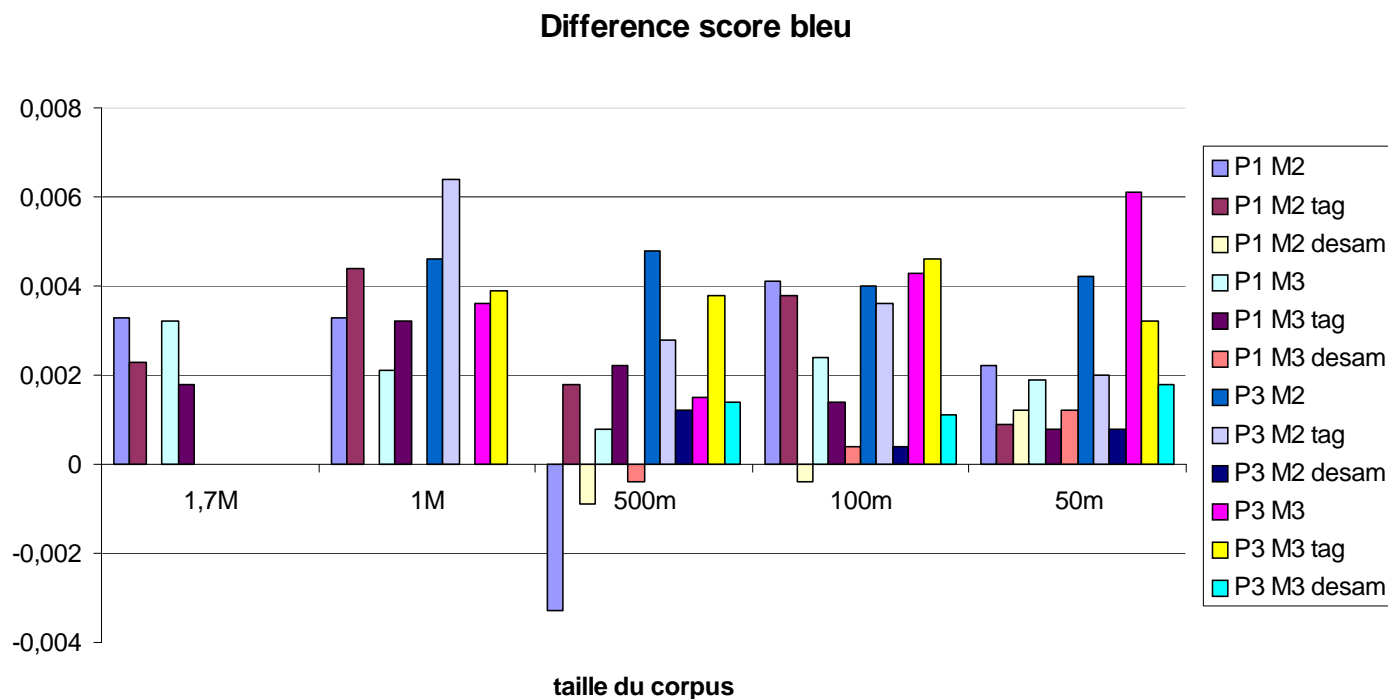


Fig 4.9 : Score bleu des traductions

Légende des graphes :

- Px : ajout de x paraphrases dans le corpus
- Mx : prise de la ou des meilleures paraphrases par un modèle d'ordre x
- Tag : étiquetage utilisé comme critère pour construire les paraphrases
- Desam : utilisation du désambigüiseur de SenseLearner pour construire les paraphrases.

Nous voyons sur le graphe de la figure 4.9 les écarts des scores BLEU de la traduction calculés grâce au corpus d'origine et des traductions calculées grâce aux nouveaux corpus d'entraînements. Le premier constat est que la majorité des scores sont supérieurs à 0 ce qui veut dire que ceux-ci sont inférieurs à celui d'origine. Nous n'avons pas pu pour le corpus de 1,7 millions de phrases ajouter 3 paraphrases car nous ne pouvons pas créer un modèle de traduction à partir d'un corpus aussi grand.

Nous pouvons voir que dans l'ensemble, l'utilisation du désambigüiseur permet d'obtenir de meilleurs résultats que la méthode s'aidant des étiquetages ou la méthode de base. Ce fait est tout à fait logique car le désambigüiseur permet de diminuer au maximum le glissement de sens d'un mot lors de son remplacement par son synonyme étant donné que les synonymes sélectionnés sont du même sens. Nous pouvons donc dire que l'utilisation de cette ressource permet d'améliorer notre construction de paraphrases. Toutefois celle-ci n'est pas sans faille puisque la désambigüisation n'est pas parfaite à 100% et engendre donc des erreurs qui sont répercutées dans nos paraphrases même si elle est aidée par le modèle de langue. Nous

voyons en revanche que l'utilisation d'une étiquette pour les paraphrases ne permet pas d'avoir de meilleurs résultats tout le temps. En effet l'amélioration ne concerne que la moitié des cas. L'étiquette permet de faire une restriction du sens en faisant une restriction des synonymes possibles par sa catégorie syntaxique pour la paraphrase. Cette restriction peut se révéler obsolète si nous avons un bon modèle de langue qui nous permet d'obtenir une restriction qui donnera au final le même résultat d'où le manque d'amélioration.

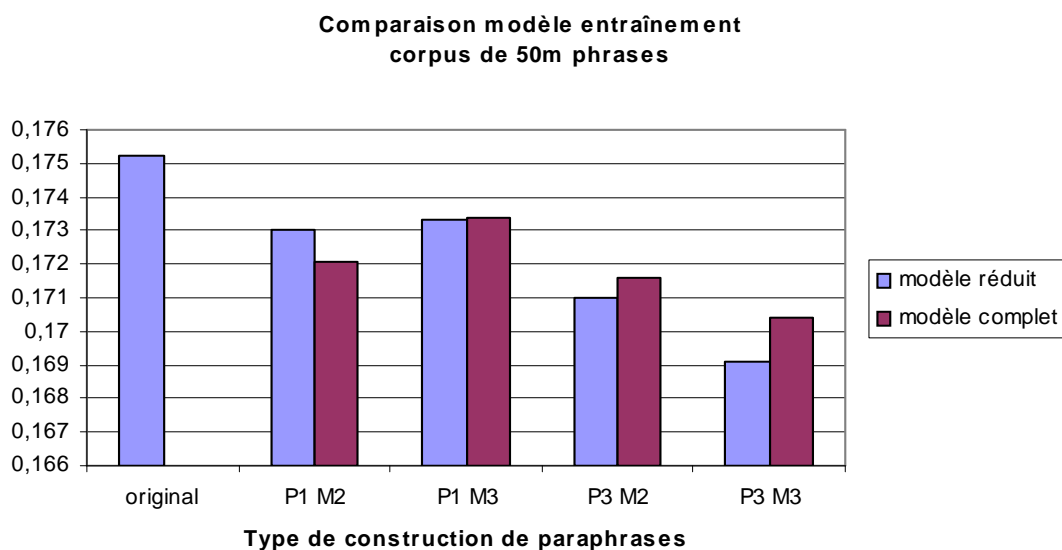
De plus nous pouvons voir que l'utilisation d'un modèle d'ordre 3 donne en général de meilleurs résultats que l'utilisation d'un modèle d'ordre 2 pour la sélection des paraphrases. En effet nous pouvons dire que le modèle d'ordre 3 permet de mieux restreindre le modèle en se basant sur plus de mots et ainsi permettre de mieux contraindre le choix du synonyme pour qu'il entre mieux dans le bon contexte donc dans le bon sens. Ainsi nous avons une meilleure contrainte pour la prise du synonyme correspondant au sens du mot original ce qui permet d'améliorer la traduction par rapport à l'utilisation du modèle utilisant les bigrammes.

La traduction est moins mauvaise aussi lors de l'ajout d'une paraphrase en comparaison à l'ajout de trois paraphrases. En ajoutant trois paraphrases c'est à dire 3 fois plus que dans le cas de l'ajout d'une paraphrase, nous avons trois fois plus de chance de commettre un glissement de sens. De plus en sachant que la probabilité de la bonne formation de la paraphrase diminue au fur et à mesure que nous en ajoutons, la probabilité du glissement de sens lui augmente. Nous avons donc une probabilité de glissement de sens beaucoup plus élevée ce qui fait que les paraphrases insèrent des erreurs dans le corpus qui se répercutent dans la traduction.

Dans un constat général, l'ajout de paraphrases dans le corpus d'entraînement pour le modèle de traduction ne permet pas d'améliorer la traduction, au contraire elle ne fait que la rendre plus mauvaise. Ce mauvais résultat est dû à plusieurs problèmes rencontrés déjà dans les tests précédents. En effet le problème majeur à régler est bien sûr la conjugaison des verbes. Les mots synonymes retournés par WordNet sont des mots lemmatisés. Nous perdons donc lors du changement du mot toute conjugaison existante et donc de l'information. Nous engendrons par la même occasion des erreurs dans notre corpus qui peuvent se répercuter dans la traduction. De plus nous pouvons être confrontés au glissement de sens. En effet nous ne pouvons dire avec certitude que le mot synonyme soit du même sens que le mot d'origine. Cette certitude est d'autant plus faible que la technique est simple. Mais même en utilisant la méthode utilisant un désambiguïseur nous ne pouvons garantir une bonne prise de sens. En effet les méthodes plus complexes utilisent des ressources extérieures qui ne sont bien sûr pas fiable à 100%. Des erreurs peuvent donc être faites par ces ressources à la base et répercutées sur les mesures suivantes. Nous sommes donc confrontés à un effet boule de neige qui à partir de petites erreurs peut en engendrer des grosses à la fin. Néanmoins les erreurs rencontrées restent quand même de faible amplitude.

Nous avons expliqué précédemment que nous utilisons un modèle de langue lors de la construction de paraphrases pour ajouter les caractéristiques du texte comme condition dans le choix des paraphrases. Nous savons aussi que plus le corpus d'apprentissage du modèle est

grand plus nous pouvons capturer de caractéristiques du modèle. Nous allons donc dans ce test essayer d'utiliser un plus grand corpus d'entraînement pour ce modèle pour améliorer notre traduction. Jusqu'à présent nous n'avons utilisé pour créer le modèle de langue que le corpus enrichi. Pourtant il est facile aujourd'hui de trouver des corpus unilingues contrairement aux corpus parallèles. De plus nous savons aussi que plus nous avons de mots dans notre corpus d'apprentissage plus nos modèles sont meilleurs. Nous allons donc cette fois construire notre modèle sur le corpus original de 1,7 millions de phrases au lieu du corpus découpé. Nous allons nous restreindre à l'évaluation des corpus de 50 000 phrases et 100 000 phrases qui sont les plus susceptibles d'être améliorés par leur grande différence de mots par rapport au corpus de 1,7 millions de phrases.

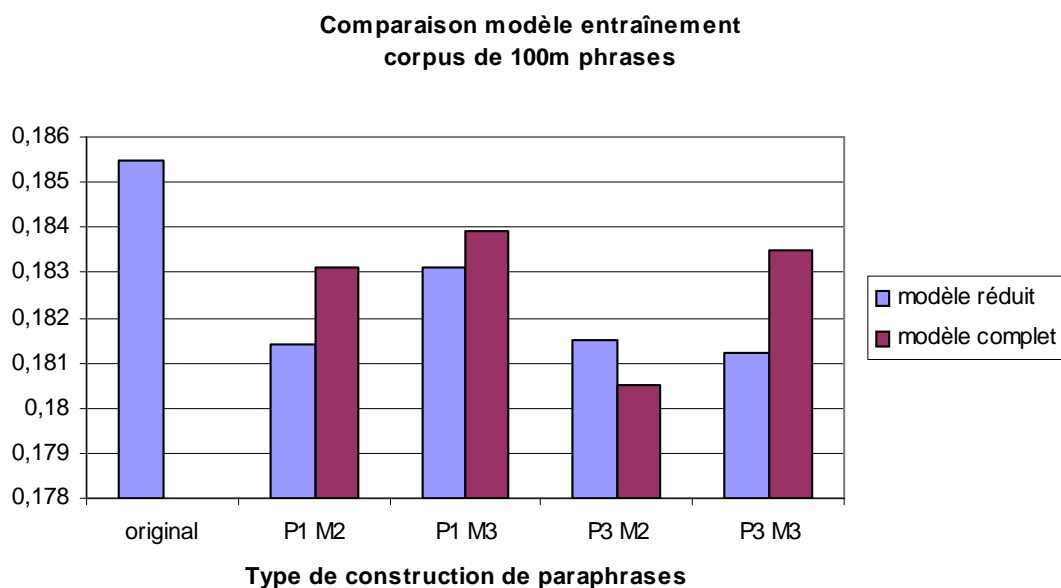


*Fig 4.10 : Traduction avec un corpus d'entraînement de 1,7M de phrases pour le modèle de langue et 50 000 phrases pour le modèle de traduction*

Nous voyons sur le graphe de la figure 4.10, les scores bleus obtenus lors de l'utilisation d'un corpus d'entraînement de 1,7 millions de phrases et d'un corpus de 50 000 phrases pour le modèle utilisé pour la construction des paraphrases. Le modèle de traduction est lui créé à partir du corpus de 50 000 phrases originales et enrichies. Nous avons gardé la même notation pour distinguer les cas :

- Px : ajout de x paraphrases dans le corpus
- Mx : prise de la ou des meilleurs paraphrases par un modèle d'ordre x

Nous pouvons constater que les résultats sont semblables pour les deux modèles construits à partir des deux corpus de différentes tailles. L'utilisation du corpus de 1,7M de phrases permet néanmoins d'obtenir de meilleurs résultats la plupart du temps mais l'écart reste tout de même minime. De plus il ne permet jamais d'être meilleur que la traduction faite à partir du modèle original.



*Fig 4.11 : Traduction avec un corpus d'entraînement de 1,7M de phrases pour le modèle de langue et 100 000 phrases pour le modèle de traduction*

Nous pouvons voir sur la figure 4.11 la comparaison des scores bleus des traductions dont les modèles de création des paraphrases ont été faits à partir de corpus différents. Comme précédemment nous ne voyons pas de différences énormes entre les deux traductions. Toutefois lorsque nous avons fait l'utilisation du corpus de 1,7 millions de phrases le score est plus souvent supérieur surtout dans le cas P3M3, cas qui était aussi largement supérieur sur la figure 4.10.

Nous pouvons constater que pour les deux types de modèles, les résultats restent assez similaires bien que c'est souvent le modèle basé sur le grand corpus qui est le meilleur. Nous expliquons bien sûr ce résultat par le fait que plus le corpus d'apprentissage est grand plus le modèle est bon. Mais à prendre un corpus trop grand nous pouvons insérer des N-grams ne correspondant plus du tout au sens original du petit corpus et ainsi ne plus assez restreindre les glissements de sens.

Dans une autre expérience nous avons voulu voir l'impact de l'enrichissement du corpus lors de traduction de texte non issu de la même source. En effet jusqu'à présent nous avons testé notre traducteur sur un corpus de test issu du Hansard tout comme le corpus d'apprentissage. En effet pour avoir un meilleur système il nous faut apprendre nos modèles sur des corpus similaires aux textes à traduire. Mais en y insérant des paraphrases nous pouvons peut être permettre une généralisation de certains mots et ainsi permettre d'avoir de meilleur résultat dans le cas d'un texte d'une source différente. Nous avons donc testé nos différents modèles sur un second corpus de test, issu de Europarl, qui est issu de la même catégorie de texte puisqu'il s'agit aussi de débat parlementaire mais de source différente puisque ce sont des textes européens. Nous avons encore une fois effectué nos tests uniquement sur les corpus de 50 000 phrases et 100 000 phrases et enrichi seulement avec 1 ou 3

paraphrases et avec un modèle de construction trigramme ou bigramme car nous voulions juste une idée de l'impact de l'agrandissement du corpus sur le type de fichier test.

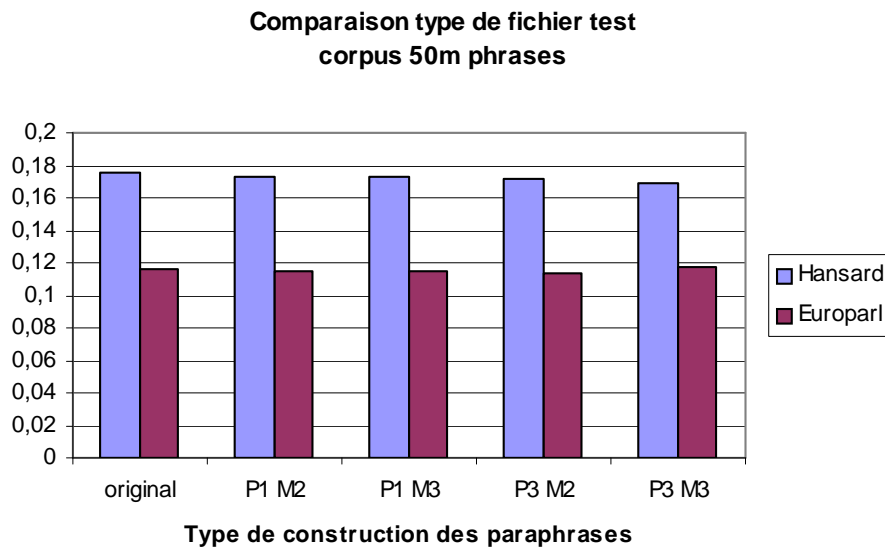


Fig 4.12 : Utilisation d'un fichier test d'origine différente avec corpus d'apprentissage de 50 000 phrases

Nous pouvons observer sur le graphe de la figure 4.12 la comparaison des scores avec un corpus d'apprentissage de 50 000 phrases. Nous pouvons voir que comme nous l'avions prévu, l'utilisation d'un corpus de test d'une origine différente donne un score plus mauvais avec l'utilisation du corpus original pour l'apprentissage de nos modèles. Nous pouvons aussi noter que nos corpus enrichis ne nous permettent pas d'améliorer ce résultat, ils obtiennent toujours des scores en dessous de l'original.

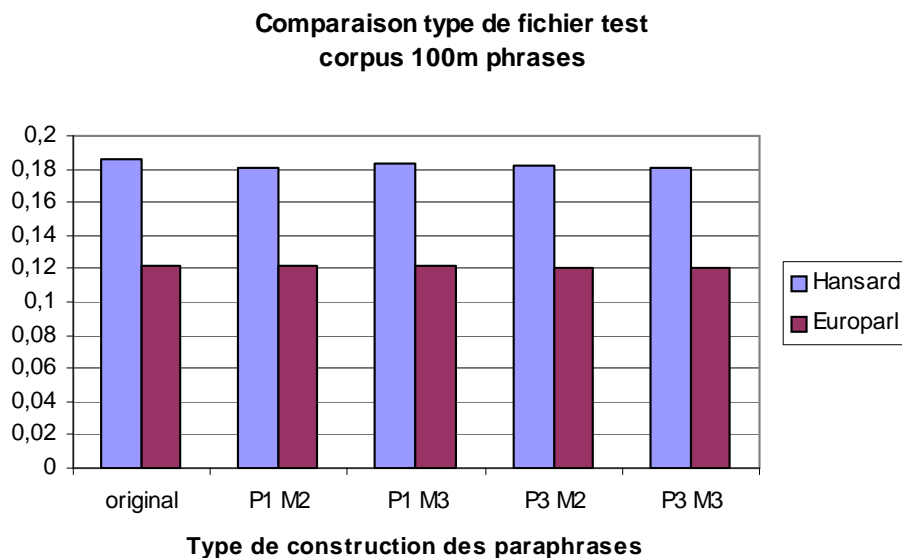


Fig 4.13 : Utilisation d'un fichier test d'origine différente avec corpus d'apprentissage de 100 000 phrases

Nous avons dans la figure 4.13 les résultats obtenus lors de l'utilisation de corpus de 100 000 phrases. Nous voyons encore une fois qu'aucune amélioration n'est apportée par l'utilisation de corpus d'entraînement enrichi.

Nous avons ici voulu savoir si l'agrandissement de notre corpus pourrait dans le cas d'un fichier test tiré d'un corpus différent de celui de l'entraînement améliorer la traduction. Nous pouvons voir que ce n'est malheureusement pas le cas. En effet les traductions des nouveaux corpus sont plus mauvaises qu'avec le corpus original. L'ajout de paraphrases ne permet donc pas d'intégrer un vocabulaire complètement différent de l'initial donc en prenant un fichier test issu d'un corpus différent nous n'arrivons pas à améliorer notre traduction. Nous pouvons encore une fois énumérer les mêmes erreurs qui bien sûr se retrouvent à chaque fois et expliquent ces mauvais résultats.

Tout au long de nos expériences, nous avons déploré le manque de conjugaisons. Nous n'avons pourtant jusqu'à maintenant fait aucun calcul du pourcentage de mots devant être conjugués et donc l'impact des erreurs. Nous avons donc regardé de plus près combien de mots, c'est à dire verbes et noms nécessitent une conjugaison. Nous voyons d'après nos résultats que nous avons très peu de noms à conjuguer puisqu'ils ne représentent à peu près que 8 % de tous les noms. En revanche nous voyons à partir du tableau 4.1 que les verbes devant être conjugués sont beaucoup plus nombreux puisqu'ils représentent plus de 60 % des verbes. Nous voyons donc bien qu'il y a un réel besoin de conjugaison dans notre cas surtout au niveau des verbes.

50m	nom	verbe
P1 M2	7,96%	65,93%
P1 M3	8,16%	66,12%
P3 M2	8,15%	59,67%
P3 M3	8,36%	59,22%

100m	nom	verbe
P1 M2	8,15%	65,57%
P1 M3	7,89%	66,08%
P3 M2	8,15%	60,28%
P3 M3	8,03%	59,74%

Tab 4.1 : Pourcentage de conjugaisons réel

Suite à ce besoin, nous avons tenté d'insérer dans notre constructeur de paragraphes un conjugeur. En effet tout au long de nos expériences nous avons déploré le manque de conjugaisons, nous allons enfin nous rendre compte de son impact. Le conjugeur utilisé est issu d'un rassemblement de deux outils existants, Tree Tagger, qui permet d'avoir les catégories syntaxiques d'un mot, et la ressource analyse, qui permet d'avoir les flexions d'un verbe. La catégorie syntaxique nous permet de savoir si le mot doit être conjugué et si oui à quel temps, ainsi nous savons si nous devons mettre le nom au pluriel ou si nous devons conjuguer le verbe au passé. Nous avons deux types de conjugaison dans notre cas, les noms et

les verbes. Dans le cas des noms, nous savons grâce à son étiquette morphosyntaxique s'il doit être mis au pluriel ou non. Dans le cas où il le doit, nous le faisons à partir de règles pré établies à partir des règles de lemmatisation de WordNet figurant dans le tableau 4.2.

singulier	pluriel
s	ses
x	xes
z	zes
y	ies
h	hes
man	men
-	s

Tab 4.2 : Règles de flexion des noms

Pour les verbes, l'étiquette morphosyntaxique nous donne le temps et la personne (troisième personne ou autre) du verbe et donc la conjugaison à lui appliquer. Nous allons ensuite chercher la flexion correspondant au temps et à la personne parmi celles retournées par la méthode analyse. Nous avons testé notre conjugueur sur les noms et verbes pris dans le corpus du Hansard. Les résultats de notre conjugueur sont les suivants :

- pour les verbes : 89,87% de bonnes conjugaisons sur un total de 151 770 verbes.
- pour les noms : 97,9% de bonnes conjugaisons sur un total de 157 009 noms.

Ces résultats ne tiennent pas compte de l'erreur de Tree Tagger faite sur la catégorie syntaxique mais seulement celle faite sur la conjugaison à faire.

Nous avons appliqué la conjugaison sur nos nouveaux corpus servant à faire nos modèles. Nous n'avons fait l'expérience que pour les deux plus petits corpus et sur la méthode de base.

Comparaison avec conjugueur corpus 50m

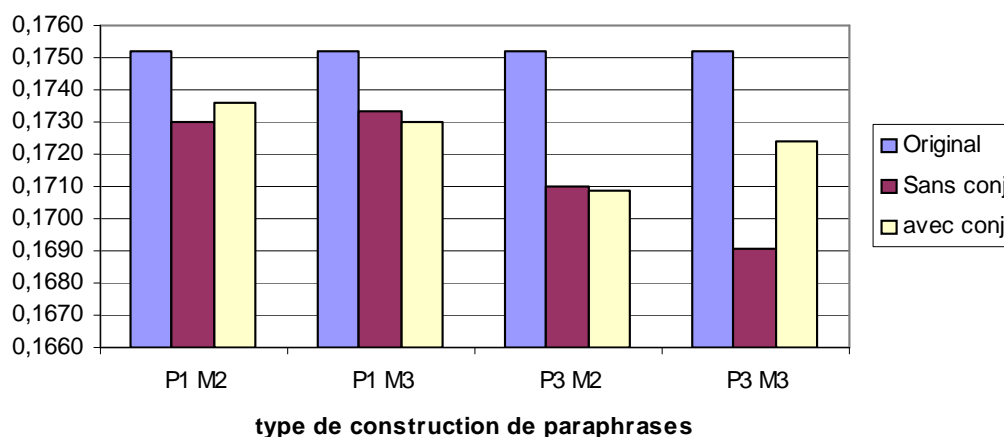


Fig 4.14 : Utilisation du conjugueur dans les corpus de 50 000 phrases enrichis

Nous pouvons voir sur la figure 4.14 que contrairement à ce que nous pensions, la conjugaison n'apporte pas un gros changement dans le score. En effet les deux résultats (avec et sans conjugaison) restent assez semblables sauf dans le cas P3M3 où elle apporte une différence non négligeable.

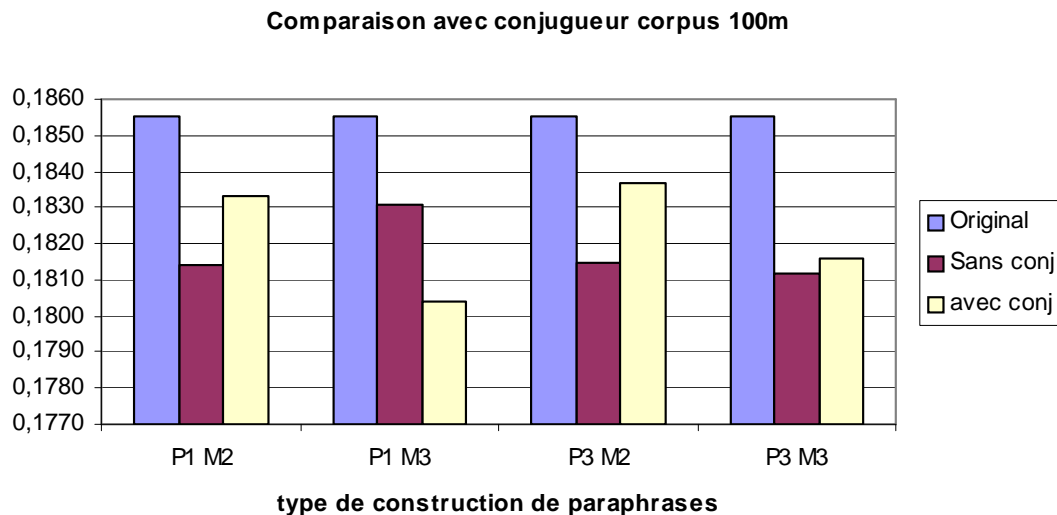


Fig 4.15 : Utilisation du conjugueur dans les corpus de 100 000 phrases enrichis

Nous pouvons encore une fois faire les mêmes remarques sur les résultats, représentés sur la figure 4.15, que précédemment. La conjugaison n'apporte pas tout le temps des meilleurs résultats, bien que dans ce cas elle permette d'obtenir deux fois des meilleurs résultats, nous avons aussi le cas P1M3 qui en obtient des moins bons

A notre grand étonnement, les résultats ne sont pas meilleurs. En effet la conjugaison permet d'apporter une légère amélioration mais celle-ci est trop minime ou trop imprévue pour être vraiment valable. Nous avons peut être apporté trop d'importance à la conjugaison, en effet bien que nous ayons vu qu'il y avait un besoin, ce besoin n'existe peut être beaucoup moins au niveau du modèle de traduction ce qui explique le peu d'amélioration. Bien que faible, nous pouvons noter que le conjugueur apporte quand même le plus souvent une amélioration donc nous pouvons toujours l'utiliser pour éviter une trop grande propagation d'erreurs.

## V. Ajout de score

### A. Calcul du nouveau score

Au vue des résultats non concluant obtenus dans notre première partie, nous nous sommes intéressés à une autre approche dans cette deuxième partie. Nous allons continuer notre tentative d'amélioration en travaillant encore sur les modèles de traduction (voir section III). Nous avons essayé dans notre première approche d'enrichir le modèle pour améliorer la traduction. Nous allons cette fois travailler directement sur le modèle de traduction ou plus exactement sur les scores du modèle. Nous rappelons tout d'abord qu'un modèle de traduction permet de calculer la probabilité  $P(x|y)$ , il est constitué du segment de mots en langue  $x$ , du segment de mots en langue  $y$  ainsi que des scores de probabilités, deux dans notre cas, permettant le calcul de la probabilité. Nous allons travailler sur ces scores, plus précisément sur l'insertion d'un nouveau score qui permet de prendre en compte de l'information sémantique. Nous allons prendre comme nouveau score le même calcul de probabilité que doit donner le modèle en modifiant les mots par les sens des mots. Nous faisons une traduction de l'anglais vers le français à partir d'un modèle "inverse", notre modèle de traduction calcul donc la probabilité  $P(e|f)$ . Nous devons donc changer les mots en anglais et en français par leur sens respectif pour calculer le nouveau score  $P(\text{sens}_e|\text{sens}_f)$ , où nous avons remplacer le segment de mots  $e$  par le segment de sens  $\text{sens}_e$  et de même pour le segment de mots  $f$ . Changer les mots en sens permettra de faire des regroupements et donner alors plus de poids à certaines informations revenant plusieurs fois mais sous des formes légèrement différentes. Nous ne disposons malheureusement que d'un désambiguïsateur anglais dans nos ressources. Nous devons alors nous contenter d'utiliser que les segments de sens anglais et garder les segments de mots français. Notre nouveau score devient alors  $P(\text{sens}_e|f)$ . Nous allons utiliser SenseLearner pour désambiguïser les mots de notre modèle. Nous avons ainsi le segment de sens anglais retrouvé à partir du code Wordnet donné par SenseLearner et le segment de mots français. Le calcul du nouveau score se fera simplement par fréquence relative. Le nouveau score est ajouté aux deux autres scores du modèle. Nous avons dans un premier temps effectué une désambiguïsation sur toutes les catégories syntaxiques des mots (SenseLearner en compte quatre : nom, verbe, adjectif, adverbe) pour voir l'impact sur la traduction. Pour voir ensuite l'impact que pourrait avoir chaque catégorie sur le score et la traduction, nous effectuons la désambiguïsation que sur une seule catégorie et donc le score ne dépend que des sens de cette catégorie.

## B. Evaluation

Nous avons utilisé pour notre évaluation les mêmes corpus que pour les évaluations précédentes. Nous avons utilisé un corpus d'apprentissage de 1 753 443 phrases disponibles en anglais et en français et un corpus de test de 1 000 phrases, tous deux issus du Hansard (voir la section IV.B pour plus d'informations). Nous avons fait 5 tailles de corpus d'apprentissage comme précédemment pour voir l'impact selon la longueur du corpus. Nous avons de plus utilisé pour ces évaluations l'étape de tuning lors de la construction de nos traducteurs pour améliorer nos traductions.

### 1. Désambiguïsation sur le segment anglais du PBM réduit

Nous avons fait la désambiguïsation sur les segments anglais du PBM. Nous rappelons que le PBM est constitué d'un segment anglais, d'un segment français et de scores. Nous allons donc pour chaque segment anglais faire une désambiguïsation qui permettra de faire les regroupements éventuels. Nous avons tout d'abord testé l'ajout d'un score avec le modèle réduit pour une plus grande rapidité et voir l'impact de ce nouveau score.

Nous avons dans un premier temps ajouté un score que nous avons calculé à partir de la désambiguïsation de toutes les catégories syntaxiques. Nous avons ensuite limité la désambiguïsation à une seule catégorie pour voir l'impact que peut avoir la prise en compte du sens de certaine catégorie. Nous retrouvons les résultats sur la figure 5.1. Nous pouvons voir sur cette figure la différence entre les scores obtenus avec le modèle original et les modèles modifiés.

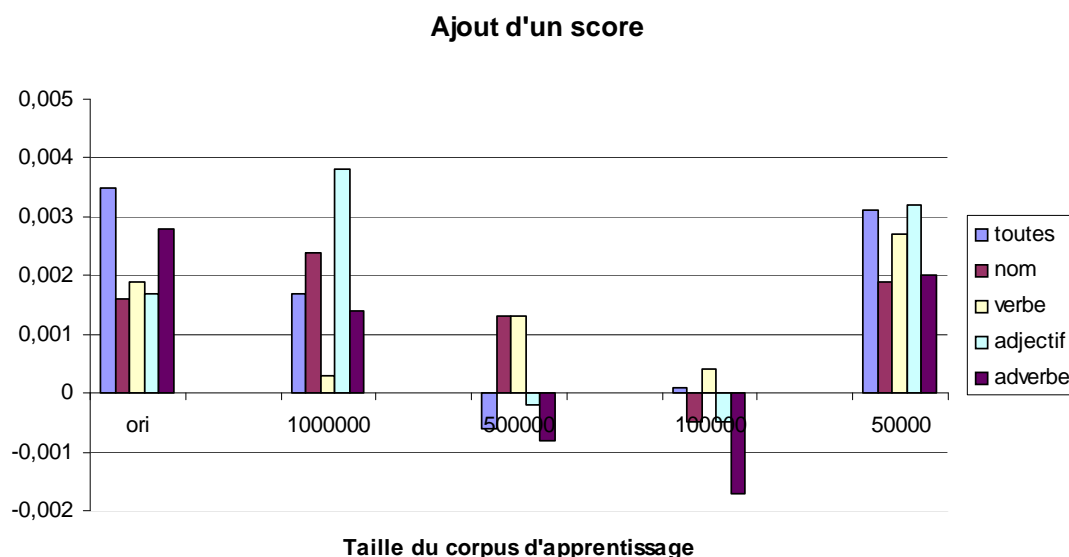


Fig 5.1 : Ajout d'un nouveau score dans le modèle réduit

Nous pouvons de suite constater que dans la majorité des cas le score est souvent plus bas donc moins bon que le score original. Les différences restent aussi très minimes surtout dans le cas des améliorations. En effet toutes les différences sont inférieures à 0.004 donc peuvent être considérées comme négligeables surtout dans les cas des meilleurs scores où la différence est inférieure à 0.002. Nous pouvons donc conclure que le nouveau score n’apporte pas de réelle modification dans la traduction.

Nous pouvons donc nous pencher sur le réel impact de ce nouveau score. Nous avons donc examiné de plus près son fondement. Nous avons évalué le pourcentage de segments qui sont regroupés et de ce fait la valeur du nouveau score. Nous retrouvons ce résultat dans le tableau 5.1.

	50000	100000	500000	1000000	1700000
toutes	2,5672%	2,7230%	2,7166%	2,6443%	2,5572%
nom	0,9743%	0,9236%	0,8224%	0,7950%	0,7604%
verbe	1,5651%	1,7697%	1,8754%	1,8375%	1,7931%
adj	0,1222%	0,0892%	0,0957%	0,0980%	0,0861%
adv	0,0000%	0,0000%	0,0000%	0,0000%	0,0000%

Tab 5.1: Segments de mots regroupés

Nous pouvons voir que les pourcentages présents dans le tableau sont très bas. Nous constatons qu’avec toutes les catégories, moins de 3% des segments peuvent être regroupés. Rappelons que la particularité de nos scores est le fait qu’ils permettent de mettre en valeur des segments se ressemblant. Plus ces regroupements seront faibles moins notre nouveau score aura d’impact car il n’apportera pas d’information supplémentaire. En effet si chaque segment est vu une fois, tous les segments auront les mêmes scores et ils ne serviront à rien.

Nous pouvons dire que nos faibles résultats sont dus à ce problème, très peu de regroupements, c’est à dire de prédominance de segments sont mis au jour par notre score. L’un des facteurs de ce problème est dû à l’utilisation de modèle réduit ce qui nous restreint considérablement le nombre de segments mais nous permet une plus grande rapidité d’exécution car la désambiguïsation est lente. Nous pouvons aussi nous poser la question du bon choix du score ou bien du bon choix de regroupement. Nous pouvons par la suite faire une évaluation sur d’autre score pour voir le résultat ou bien revoir plus profondément la partie de regroupement des segments pour laisser plus de souplesse dans celui-ci.

Nous avons tout de même malgré cette constatation voulu voir l’impact que pouvait avoir ce nouveau modèle lors d’une utilisation d’un corpus de test de source différente. Nous avons utilisé un fichier de 1 000 phrases tiré de Europarl (voir section IV.B.2 pour plus d’information). Nous allons voir l’impact sur un texte de source différente mais de nature identique, débat parlementaire. Le nouveau score pourrait peut être nous apporter une petite amélioration car il pourrait permettre une petite généralisation. Nous retrouvons les résultats de cette évaluation sur la figure 5.2.

Nous pouvons constater que le score du modèle à trois scores est en générale inférieur au score du modèle à deux scores sauf dans le cas des 100 000 phrases. Les différences restent encore ici très minimales ce qui nous permet encore de nous poser la question sur l'importance du nouveau score. En effet en voyant les résultats nous pouvons voir que le nouveau score n'apporte aucun changement significatif que ce soit négatif ou positif.

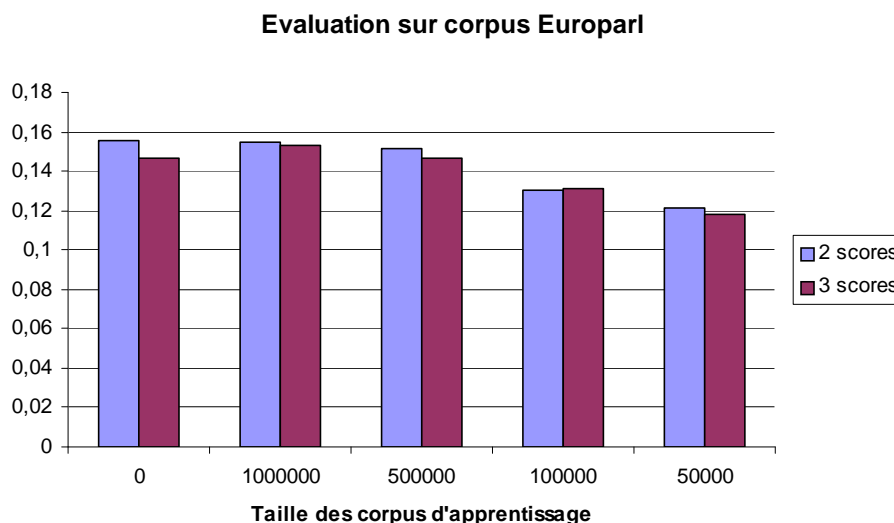


Fig 5.2 : Ajout d'un score dans le modèle évalué sur Europarl

Comme nous aurions pu nous en douter par les constatations faites précédemment pour l'évaluation principale, le modèle n'a pu apporter aucune amélioration. Nous pouvons souligner le même souci du peu de regroupement et donc du peu de différence de score entre tout les segments.

### 2. Désambiguïsation sur le segment anglais du PBM

Comme précédemment nous avons fait la désambiguïsation sur le segment anglais mais nous avons pour cette série de test pris le PBM non réduit ce qui nous permet d'avoir beaucoup plus de segments et donc plus de chance de faire des regroupements.

Nous nous sommes intéressés uniquement à la désambiguïsation de toutes les catégories. En effet la désambiguïsation est une tâche qui prend beaucoup de temps et le PBM non réduit est de grande taille nous ne pouvons donc envisager de faire plusieurs désambiguïsations pour un modèle.

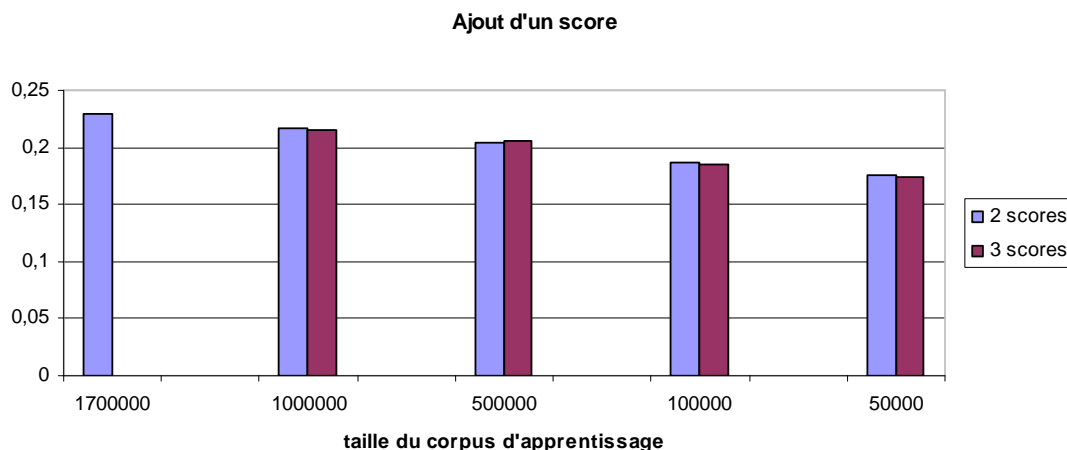


Fig 5.3 : Ajout d'un score dans le modèle entier

Nous pouvons encore une fois voir qu'aucune amélioration significative n'est faite. Seul un score est supérieur à l'original. Nous pouvons encore une fois mettre en évidence le peu de regroupement et donc de l'incapacité au nouveau score à avoir un impact. L'utilisation d'un plus grand nombre de segments n'a donc pas résolu le problème du regroupement.

### 3. Désambiguïsation sur le corpus original

Nous avons dans ce test effectué la désambiguïsation sur le corpus d'apprentissage. Nous avons ensuite introduit les sens trouvés dans le PBM pour ainsi faire les regroupements. Cette méthode permet d'avoir une meilleure désambiguïsation car nous avons un plus grand historique étant donné qu'elle est faite sur une phrase et non plus sur un segment de phrase. Nous avons effectué ce test juste pour le corpus de 1 000 000 phrases pour voir si elle peut avoir un meilleur impact que les précédentes.

2 scores	3 scores
0.2790	0.2766

Tab 5.2 : Score BLEU

Nous pouvons constater encore une fois que le nouveau score n'apporte qu'une légère baisse du score. Cette baisse reste assez faible. Faire la désambiguïsation sur le texte original ne permet donc pas de régler les problèmes rencontrés précédemment. Il y a toujours trop peu de regroupement pour que le nouveau score ait un quelconque impact sur la traduction.

## VI. Conclusion

### A. Bilan de l'étude

Nous avons essayé tout au long de notre étude d'apporter une amélioration grâce à l'utilisation d'informations sémantiques. En effet l'information sémantique nous permet d'obtenir une aide pour trouver le sens d'un mot dans un contexte donné. Cette information est souvent utilisée dans le traitement de la langue naturelle pour apporter des améliorations. Nous avons donc essayé de l'insérer dans notre traducteur automatique. Nous avons mis en place deux méthodes pour effectuer cette tâche.

La première utilise l'enrichissement du corpus d'apprentissage du modèle de traduction par des paraphrases. Au vue de nos différents résultats, nous pouvons dire que cet enrichissement n'apporte aucun gain à notre traduction. Nous avons essayé d'apporter des améliorations supplémentaires, mais celles-ci n'ont rien apporté de plus. Ce résultat est décevant mais un peu prévisible vu que notre méthode peut insérer du bruit. Nous pouvons imaginer pour un travail futur l'insertion permettant d'améliorer l'enrichissement du corpus et ainsi diminuer l'insertion d'erreur.

La deuxième méthode se situe directement dans la construction du modèle de traduction. Nous y insérons un nouveau score. Ce nouveau score est calculé à partir des sens des mots et non plus sur les mots eux mêmes. Cette méthode permet de donner plus de poids aux segments ayant le même sens mais une graphie différente. Cette méthode s'est encore révélée décevante en n'apportant aucune amélioration significative. Le principal problème est la difficulté de faire beaucoup de regroupements et donc d'avoir un score mettant en valeur ces regroupements. Nous n'avons pas totalement fini cette partie lors de l'écriture de notre étude. Il nous reste des pistes à étudier plus ou moins nombreuses selon le temps restant. La première est le calcul de nouveaux scores à ajouter, le premier n'ayant pas un impact assez grand sur le corpus. Nous pouvons aussi envisager un nouveau mode de regroupement plus permissif permettant de faire plus de regroupements, nous pouvons par exemple prendre en compte un changement de position de mots.

Dans un constat général l'utilisation de l'information sémantique dans le modèle de traduction ne nous a pas permis dans notre étude d'améliorer les résultats. Nous pensons tout de même que cette information pourrait quand même permettre une amélioration, peut être en explorant les améliorations proposés dans chaque partie ou bien en utilisant autrement l'information comme par exemple pour corriger la traduction faite par le système.

## B. Bilan personnel

Ce stage a été très intéressant et très enrichissant sur plusieurs niveaux.

Il m'a permis de découvrir le fonctionnement d'un laboratoire de recherche. Il m'a donné une vue du travail de recherche. Il m'a permis d'acquérir une rigueur de travail indispensable dans ce domaine. J'ai pu apprendre à travailler dans le milieu de la recherche en intégrant l'équipe du RALI et bénéficier de leurs compétences. De plus j'ai pu apprendre à travailler sur contraintes notamment sur la contrainte de la taille des fichiers traités.

De plus cette étude m'a permis de mieux connaître la traduction automatique. En effet mon stage m'a permis d'apprendre exactement comment fonctionne un tel système et les plus efficaces. De plus j'ai pu approfondir mes connaissances en linguistique et tous les outils s'y rapportant comme les étiqueteurs ou les désambiguïsateurs. Cette étude m'a permis d'ajouter tous ces éléments à mes connaissances initiales.

Le stage se passant dans le cadre d'une année à l'étranger, il m'a permis de connaître une autre méthode de travail. Le fait d'avoir travaillé dans un environnement différent ne peut être qu'un plus. En effet il permet d'acquérir une capacité d'adaptation et de nouvelles méthodes de travail. Cet élément ne pourra être qu'un plus pour ma future carrière.

# ANNEXE

Correspondance étiquettes du RALI, étiquete des étiqueteurs étudiés

AdjQ	JJ
AdjQ-Compar	JJR
AdjQ-Super	JJS
Adve	RB
Adve-Compar	RBR
Adve-SAhn	RB
Adve-SPjr	RB
Adve-SPqt	RB
Adve-Super	RBS
Adve-XNOT	RB
Adve-nomi	RB
Adve-part	RB
Affi	POS
Affi-gen	POS
ConC	CC
ConC-Dble	DT
ConS	IN
Dete-dart-sgpl-def	DT
Dete-dart-sing-ind	DT
Dete-ddem-plur-def	DT
Dete-ddem-plur-ind	DT
Dete-ddem-sing-def	DT
Dete-ddem-sing-ind	DT
Dete-dpos-femi-sgpl-p3-def-sng	DT
Dete-dpos-mafe-sgpl-p1-def-plu	DT
Dete-dpos-mafe-sgpl-p1-def-sng	DT
Dete-dpos-mafe-sgpl-p2-def-osp	DT
Dete-dpos-mafe-sgpl-p3-def-plu	DT
Dete-dpos-masc-sgpl-p3-def-sng	DT
Dete-dpos-neut-sgpl-p3-def-sng	DT
Dete-dpos-sing-p2-def	DT
Dete-dwdt-sgpl	DT
Disc	SYM
Inte	VH
Ltre-plur	LS
Ltre-sing	LS
NomC	NN
NomC-plur	NNS
NomC-sing	NN
NomP	NP
Ordi-sing	JJ
Post-degr	NN
Post-orgn	NN
Post-time	NN
Prep	IN
Prep-RP	RP

Prep-TO	TO
Pron-pdm-gmfn-plur-p3-cno	DT
Pron-pdm-gmfn-sing-p3-cno	DT
Pron-pex-neut-sgpl-p3-nom	EX
Pron-pid-gmfn-plur-p3-cno	PRP
Pron-pid-gmfn-sing-p3-cno	PRP
Pron-pid-mafe-sing-p3-cno	NN
Pron-pid-neut-sing-p3-cno	NN
Pron-prfl-femi-sing-p3-obj	PRP
Pron-prfl-gmfn-plur-p3-obj	PRP
Pron-prfl-mafe-plur-p1-obj	PRP
Pron-prfl-mafe-plur-p2-obj	PRP
Pron-prfl-mafe-sing-p1-obj	PRP
Pron-prfl-mafe-sing-p2-obj	PRP
Pron-prfl-masc-sing-p3-obj	PRP
Pron-prfl-neut-sing-p3-obj	PRP
Pron-prp-femi-sgpl-p3-gen-sng	PRP
Pron-prp-femi-sing-p3-nom	PRP
Pron-prp-femi-sing-p3-obj	PRP
Pron-prp-gmfn-plur-p3-nom	PRP
Pron-prp-gmfn-plur-p3-obj	PRP
Pron-prp-gmfn-sgpl-p3-gen-plu	PRP
Pron-prp-mafe-plur-p1-nom	PRP
Pron-prp-mafe-plur-p1-obj	PRP
Pron-prp-mafe-sgpl-p1-gen-plu	PRP
Pron-prp-mafe-sgpl-p1-gen-sng	PRP
Pron-prp-mafe-sgpl-p2-gen-osp	PRP
Pron-prp-mafe-sgpl-p2-nom	PRP
Pron-prp-mafe-sgpl-p2-obj	PRP
Pron-prp-mafe-sing-p1-nom	PRP
Pron-prp-mafe-sing-p1-obj	PRP
Pron-prp-masc-sgpl-p3-gen-sng	PRP
Pron-prp-masc-sing-p3-nom	PRP
Pron-prp-masc-sing-p3-obj	PRP
Pron-prp-neut-sgpl-p3-gen-sng	PRP
Pron-prp-neut-sing-p3-nom	PRP
Pron-prp-neut-sing-p3-obj	PRP
Pron-pwhi	WRB
Pron-pwhi-gmfn-sgpl-cno	WP
Pron-pwhi-gmfn-sgpl-p3-ngo	WP
Pron-pwhi-mafe-sgpl-cno	WP
Pron-pwhi-mafe-sgpl-p3-gen	WP
Pron-pwhi-mafe-sgpl-p3-nom	WP
Pron-pwhi-mafe-sgpl-p3-obj	WP
Pron-pwhi-neut-sgpl-cno	WDT
Pron-pwhr	WRB

Pron-pwhr-gmfn-sgpl-p3-cno	WP\$
Pron-pwhr-gmfn-sgpl-p3-gen	WDT
Pron-pwhr-mafe-sgpl-p3-cno	WP
Pron-pwhr-mafe-sgpl-p3-obj	WP
Pron-pwhr-neut-sgpl-p3-ngo	WDT
Punc-pccm	,
Punc-pcco	:
Punc-pcda	:
Punc-pcdd	:
Punc-pcpl	(
Punc-pcpr	SYM
Punc-pcql	``
Punc-pcqr	"
Punc-pcsc	:
Punc-pcst	.
Quan	CD
Quan-cmp-plur-SPhn-ind	RBR
Quan-cmp-sgpl-SPhn-ind	RBR
Quan-cmp-sgpl-SPqt-ind	RBR
Quan-cmp-sing-SPhn-ind	RBR
Quan-cmp-sing-SPqt-ind	RBR
Quan-frc-plur-def	NNS
Quan-frc-sing-def	NN
Quan-ndg-plur-SAhn-ind	DT
Quan-ndg-plur-SPdt-ind	DT
Quan-ndg-plur-SPjr-ind	JJ
Quan-ndg-plur-Sdet-ind	JJ
Quan-ndg-plur-Sh_n-ind	JJ
Quan-ndg-plur-Sprn-ind	JJ
Quan-ndg-sgpl-SAhn-ind	DT
Quan-ndg-sgpl-SPdt-ind	DT
Quan-ndg-sgpl-SPhn-ind	INC
Quan-ndg-sgpl-SPjr-ind	DT
Quan-ndg-sgpl-Sdet-ind	DT
Quan-ndg-sgpl-Sh_n-ind	DT
Quan-ndg-sgpl-Sord-ind	JJ
Quan-ndg-sgpl-Sprn-ind	JJ
Quan-ndg-sing-SPjr-ind	RB
Quan-ndg-sing-SPqt-ind	RB
Quan-ndg-sing-Sdet-ind	DT
Quan-ndg-sing-Sh_n-ind	JJ
Quan-ndg-sing-Sord-ind	JJ
Quan-ndg-sing-Sprn-ind	JJ
Quan-num-plur-def	CD
Quan-pos-plur-SPjr-ind	JJ
Quan-pos-plur-Sh_n-ind	RB
Quan-pos-sing-SPjr-ind	JJ
Quan-pos-sing-SPqt-ind	RB
Quan-sup-plur-SPhn-ind	RBS
Quan-sup-sgpl-SPhn-ind	RBS
Quan-sup-sgpl-SPqt-ind	RBS
Quan-sup-sing-SPhn-ind	RBS

Quan-sup-sing-SPqt-ind	RBS
Verb	VB
Verb-BSE	VB
Verb-PAST	VBD
Verb-PRES-sing-p3	VBZ
Verb-PRP	VBG
Verb-PSP	VBN
Verb-aux-BSE	VB
Verb-aux-PAST	VBD
Verb-aux-PAST-p2	VBD
Verb-aux-PAST-plur-p1	VBD
Verb-aux-PAST-plur-p3	VBD
Verb-aux-PAST-sing-p1	VBD
Verb-aux-PAST-sing-p3	VBD
Verb-aux-PRES-p2	VBP
Verb-aux-PRES-plur-p1	VBP
Verb-aux-PRES-plur-p3	VBP
Verb-aux-PRES-sing-p1	VBP
Verb-aux-PRES-sing-p3	VBZ
Verb-aux-PRP	VBG
Verb-aux-PSP	VBN
Verb-lex-BSE	VB
Verb-lex-PAST	VBD
Verb-lex-PAST-p2	VBD
Verb-lex-PAST-plur-p1	VBD
Verb-lex-PAST-plur-p3	VBD
Verb-lex-PAST-sing-p1	VBD
Verb-lex-PAST-sing-p3	VBD
Verb-lex-PRES	VBP
Verb-lex-PRES-p2	VBP
Verb-lex-PRES-plur-p1	VBP
Verb-lex-PRES-plur-p3	VBP
Verb-lex-PRES-sing-p1	VBP
Verb-lex-PRES-sing-p3	VBZ
Verb-lex-PRP	VBG
Verb-lex-PSP	VBN
Verb-mdl-BSE	MD
Verb-mdl-PAST	MD
Verb-mdl-PRES-sing-p3	MD

## Références bibliographiques

- [1] A.L. Berger, P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. R. Gillett, J. D. Lafferty, R. L. Mercer, H. Printz, and L. Ures, "The Candide System for Machine Translation", in Proc. ARPA Human Language Technology Workshop, Plainsboro, NJ, Morgan Kaufmann Publishers, San Mateo, CA, pp. 152-157, March 1994.
- [2] Brown Peter F., Della Pietra Stephen A., Della Pietra Vincent J., and Mercer Robert L., "The mathematics of statistical machine translation: Parameter estimation", *Computational Linguistics*, 19(2):263–311, 1993.
- [3] Melby Alan, "A Bilingual Concordance System and its Use in Linguistic Studies", in Proceedings of the 8th Lacus Forum, Hornbeam Press, Columbia SC, pp.541-54, 1981.
- [4] Harris Brian, "Bi-text: A New Concept in Translation Theory", *Language Monthly*, no. 54, pp 8-10, 1988a.
- [5] Harris Brian, "Are You Bi-textual?" *Language Technology*, no.7, p. 41, 1988b.
- [6] (Ma & Liberman, 1997)
- [7] W. A. Gale and Kenneth W. Church, "A program for aligning sentences in bilingual corpora", in *Computational Linguistics*, volume 19, pages 75-102, 1993.
- [8] (Moore, 2002)
- [9] A. Stolcke , "SRILM – An Extensible Language Modeling Toolkit", In Proceedings of the International Conference for Speech and Language Processing (ICSLP), Denver, Colorado, September 2002
- [10] R. Kneser and H. Ney, Improved backing-off for m-gram language modeling. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, volume 1, pages 181-184, 1995.
- [11] S. Chen and J. Goodman, An empirical study of smoothing techniques for language modeling, TR-10-98, Harvard University, August, 1998.(Goodman, 1995)
- [12] F.J. Och and H. Ney, "Improved Statistical Alignment Models", in Proceedings of the Conference of the Association for Computational Linguistic (ACL), Hongkong, China, pp. 440-447, 2000 (GIZA++)
- [13] S. Nießen, S. Vogel, H. Ney and C. Tillmann, A DP Based Search Algorithm for Statistical Machine Translation. In Proceedings of COLING-ACL '98: 36th Annual

- Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, 960–967, Montreal, Quebec, Canada, August, 1998.
- [14] K. Knight, A statistical machine translation workbook, USC/ISI, available at <http://www.clsp.jhu.edu/ws99/projects/mt/mt-worbook.htm>, 1999.
- [15] P. Koehn , “Pharaoh: a Beam Search Decoder for Phrased-Based SMT”, To appear in Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA), 2004
- [16] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton} and A. Sangiovanni-Vincentelli, “SIS: {A} system for sequential circuit synthesis”, 1992.
- [17] K. Papineni, S. Roukos, T. Ward, W. Zhu, BLEU : a Method for Automatic Evaluation of Machine Translation, Proceedings of COLING-ACL '02, Philadelphia, USA, pp.311-318, 2002.
- [18] C. Fellbaum, “WordNet: An Electronic Lexical Database”, Cambridge, Mass: MIT Press, c1998 xxii, 423 p., 1998.
- [19] G. Miller, C. Leacock, T. Randee, and R. Bunker, “A semantic concordance”, in Proceedings of the 3rd DARPA Workshop on Human Language Technology, pages 303–308, Plainsboro, New Jersey, 1993.
- [20] Semcor man pages, SemCor - Discussion of semantic concordance of semantically tagged text, <http://www.cosgi.princeton.edu/~wn/man/semcor.7WN.html>, 1995.
- [21] Barbara B. Greene and Gerald M. Rubin, “Automated Grammatical Tagging of English”, Department of Linguistics, Brown University, Providence, Rhode Island, 1971.
- [22] E. Brill, “A simple rule-based part of speech tagger”, in proceedings of the third Conference on Applied Natural Language Processing, Trento, Italy, 1992.
- [23] L. R. Bahl, R. L. Mercer, “Part of speech assignment by a statistical decision algorithm”, in IEEE International Symposium on Information Theory, Ronneby, 88-89, 1976.
- [24] Helmut Schmid, “Probabilistic part-of-speech tagging using decision trees”, in Proceedings of International Conference on New Methods in Language Processing, Manchester, UK, September, 1994.

- [25] Kenneth Church, “A stochastic parts program and noun phrase parser for unrestricted text”, in proceedings of the Second Conference on Applied Natural Language Processing, ACL, Austin, Tx, 1988.
- [26] D. Cutting, J. Kupiec, J. Pedersen and P. Sibun, “A practical part-of-speech tagger”, in Proceedings of the Third Conference on Applied Natural Language Processing, ACL, 1992.
- [27] S. Derose, “Grammatical category disambiguation by statistical optimization” in Computational Linguistics 14, 1988.
- [28] A. Kempe, “A probabilistic tagger and an analysis of tagging errors Technical report”, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart, 1993.
- [29] S. Federici and V. Pirrelli, “Context-sensitivity and linguistic structure in analogy-based parallel networks” in Current Issues in Mathematical Linguistics, pages 353-362, 1994.
- [30] B. Decadt, V. Hoste, W. Daelemans, and A. Van den Bosch, “Gambl, genetic algorithm optimization of memory-based wsd”, in Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, July, 2004.
- [31] D. Yuret, “Some experiments with a naive bayes wsd System”, in Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, July, 2004.
- [32] R. Mihalcea and E. Faruque, “SenseLearner: Minimally supervised word sense disambiguation for all words in open text”, in Proceedings of ACL/SIGLEX Senseval-3, Barcelona, Spain, July, 2004.
- [33] Rada Mihalcea and Andras Csomai, “SenseLearner: Word Sense Disambiguation for All Words in Unrestricted Text”, in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, companion volume (ACL 2005), Ann Arbor, MI, June 2005.
- [34] G. Miller, “Wordnet: A lexical database” in Communication of the ACM, 38(11):39–41, 1995.
- [35] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch, “Timbl: Tilburg memory based learner, version 4.0, reference guide” in Technical report, University of Antwerp, 2001.
- [36] V. Hoste, W. Daelemans, I. Hendrickx, and A. van den Bosch, “Evaluating the results of a memory-based word-expert approach to unrestricted word sense disambiguation”, in

Proceedings of the ACL Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions", Philadelphia, July, 2002.

- [37] R. Mihalcea, "Instance based learning with automatic feature selection applied to Word Sense Disambiguation", in Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), Taipei, Taiwan, August, 2002.
- [38] A. J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2):260–267, April 1967. (The Viterbi decoding algorithm is described in section IV.)

## Liste des tableaux

<i>Tab 1.1 : La répartition des pages web par langues (Funredes)</i>	03
<i>Tab 3.1: Détails données de WordNet (WordNet 2.1 Reference Manual)</i>	18
<i>Tab 3.2: Détails des dossiers de Semcor</i>	20
<i>Tab 3.3: Détails des étiquettes POS des dossiers de Semcor</i>	21
<i>Tab 3.4: Détails des étiquettes POS des corpus</i>	23
<i>Tab 3.5 : Détails des catégories du corpus de test</i>	26
<i>Tab 3.6 : Evaluation des étiqueteurs</i>	26
<i>Tab 3.7 : Détails des erreurs de l'étiqueteur Brill</i>	27
<i>Tab 3.8 : Détails des erreurs de l'étiqueteur Tree Tagger</i>	27
<i>Tab 3.9: Deuxième évaluation des étiqueteurs</i>	28
<i>Tab 3.10: Evaluation des désambigüiseurs</i>	34
<i>Tab 4.1 : Pourcentage de conjugaisons réel</i>	53
<i>Tab 4.2 : Règles de flexion des noms</i>	54
<i>Tab 5.1: Segments de mots regroupés</i>	58
<i>Tab 5.2 : Score BLEU</i>	60

## Liste des figures

<i>Fig 2.1 : Corpus parallèle, "Le bitexte et ses applications"</i>	07
<i>Fig 2.2 : Bitexte , "Le bitexte et ses applications"</i>	07
<i>Fig 2.3 : Exemple d'un modèle de langue d'ordre 2</i>	09
<i>Fig 2.4 : Alignement entre une phrase française et une phrase anglaise</i>	10
<i>Fig 2.5 : Exemple modèle de traduction de segments de phrases</i>	11
<i>Fig 2.6 : Exemple des faiblesses des modèles de traduction PBM</i>	11
<i>Fig 2.7 : L'architecture de la traduction probabiliste (Nießen et al. 1998)</i>	12
<i>Fig 2.8 : Texte à traduire par Pharaoh</i>	14
<i>Fig 2.9 : Texte de référence français</i>	14
<i>Fig 2.10 : Texte traduit par Pharaoh (texte original Fig 2.8)</i>	15
<i>Fig 3.1 : exemple du synset de la catégorie nom du mot "car" de WordNet 2.1.</i>	17
<i>Fig 3.2 : Arbre de concept du premier sens "car" dans WordNet</i>	18
<i>Fig 3.3 : Relations sémantiques de WordNet</i>	19
<i>Fig 3.4 : Méronymes associés au sens "car, auto..." du mot "car"</i>	19
<i>Fig 3.5: Sous hiérarchie de WordNet correspondant au concept "dog" (RFIEC)</i>	19
<i>Fig 3.6 : Extrait du corpus Semcor</i>	22
<i>Fig 3.7 : Exemple de texte étiqueté par Brill</i>	24
<i>Fig 3.8 : Exemple de texte étiqueté par Tree Tagger</i>	25
<i>Fig 4.1 Tableau des synonymes</i>	36
<i>Fig 4.2 : 4 exemples de 4 paraphrases créées par Disambig</i>	37
<i>Fig 4.3 : Extrait du corpus anglais enrichi (phrase original en gras)</i>	38
<i>Fig 4.4 : Extrait du corpus français aligné au corpus anglais enrichi (fig 4.3)</i>	38
<i>Fig 4.5 : Evaluation de l'utilisation de interpolate</i>	42
<i>Fig 4.6 : Perplexité sans option interpolate</i>	43
<i>Fig 4.7 : Perplexité avec option interpolate pour le modèle évaluation</i>	44
<i>Fig 4.8 : Perplexité avec option interpolate pour les deux modèles</i>	45
<i>Fig 4.9 : Score bleu des traductions</i>	48
<i>Fig 4.10 : Traduction avec un corpus d'entraînement de 1,7M de phrases pour le modèle de langue et 50 000 phrases pour le modèle de traduction</i>	50

<i>Fig 4.11 : Traduction avec un corpus d'entraînement de 1,7M de phrases pour le modèle de langue et 100 000 phrases pour le modèle de traduction</i>	51
<i>Fig 4.12 : Utilisation d'un fichier test d'origine différente avec corpus d'apprentissage de 50 000 phrases</i>	52
<i>Fig 4.13 : Utilisation d'un fichier test d'origine différente avec corpus d'apprentissage de 100 000 phrases</i>	52
<i>Fig 4.14 : Utilisation du conjugueur dans les corpus de 50 000 phrases enrichis</i>	54
<i>Fig 4.15 : Utilisation du conjugueur dans les corpus de 100 000 phrases enrichis</i>	55
<i>Fig 5.1 : Ajout d'un nouveau score dans le nouveau modèle</i>	57
<i>Fig 5.2 : Ajout d'un score dans le modèle évalué sur Europarl</i>	59
<i>Fig 5.3 : Ajout d'un score dans le modèle entier</i>	60