

Université Paris 13 Villetaneuse – Institut Galilée

Université de Montréal

Laboratoire RALI

Compression de modèles *phrase based* pour la traduction statistique

par

Chafique Abibouraguimane

Shafiq.82@gmail.com

Septembre 2006

Rapport de stage présenté en vue de l'obtention du diplôme d'ingénieur informatique à

l'université Paris 13 Villetaneuse

Encadrant : Philippe Langlais

Tutrice : Adeline Nazarenko

Remerciements

Je voudrais commencer par remercier Philippe Langlais qui m'a accueilli au sein de son laboratoire et encadré dans mon travail durant mon stage, se montrant toujours disponible pour m'aider et me conseiller, et ce, malgré les nombreux projets qui l'accaparent.

Je remercie également Elliott Macklovitch et tous les membres du RALI qui m'ont permis de travailler dans une ambiance amicale et professionnelle, tout particulièrement Fabrizio Gotti et Alexandre Patry pour toute l'aide qu'ils m'ont apportée.

Mes pensées vont aussi vers mes camarades de classe Lucie Ahfir et Mickael Calvo qui m'ont accompagné à Montréal dans les débuts, et avec qui j'ai passé d'agréables moments.

Je tiens aussi à remercier mes amis de Paris et de Montréal pour tous les bons moments que j'ai passés avec eux et que je n'oublierai pas de sitôt. Sans oublier le groupe Capoeira Camarà Montréal pour m'avoir soutenu durant mon séjour.

Je n'oublie pas ma famille qui m'a soutenu du début jusqu'à la fin lors de ce séjour à l'étranger et pour tout ce qu'elle a fait pour que celui-ci se déroule dans les meilleures conditions.

Mes derniers remerciements sont pour Adeline Nazarenko pour tous ses efforts, sa disponibilité et son soutien, sans lesquels je n'aurais pu faire ce stage.

À Ouways, mon neveu

Résumé

Le domaine de la traduction automatique probabiliste repose principalement sur l'apprentissage automatique de paramètres. Parmi les modèles de traductions utilisés, il y a les PBMs (Phrase-based Models). Ces modèles sont un ensemble de paires de phrases dans la langue source et cible pourvus d'un ou plusieurs scores probabilistes.

Ces PBMs sont générés automatiquement à partir d'algorithmes d'apprentissage sur bitextes. C'est à l'issue de cette phase que l'on observe une surgénération de paramètres dans les PBMs; on dit généralement que nos PBMs sont bruités.

De ce fait, les PBMs prennent une place importante sur le disque dur (de l'ordre du Go pour un modèle typique) et il est difficilement envisageable de livrer ces modèles avec des applications grand public, où parfois l'espace mémoire fait défaut. De plus, la taille des PBMs influe directement sur le temps de traduction.

Cette étude consistera à identifier des approches pour réduire la taille des PBMs avec un minimum de perte de qualité de traduction, en utilisant divers algorithmes et stratégies, comme la programmation dynamique et des heuristiques.

Abstract

Statistical Machine Translation is a process relying on the automatic learning of parameters, which constitute our translation models. Phrase-Based Models (PBM) are one type of translation models. They consist of a set of sentence pairs in source and target languages associated with one or more scores.

Learning algorithms automatically created PBMs from parallel corpora. After that, we usually notice an over-generation of parameters in the PBM.

Because of the fact that our PBMs take a significant space in memory (1 Gb or more of hard disk space, typically) and it's hardly conceivable to provide those models for large, public-purpose applications where memory space is at a premium. Moreover, the size of PBMs influences the translation time of a text.

My study will consist in finding approaches to reduce the size of PBMs with a minimum loss of translation quality by using various algorithms and strategies, like dynamic programming and heuristics.

Table des matières

Remerciements	2
Résumé	4
Abstract	5
Table des matières	6
1 Introduction	8
1.1 Le RALI	8
1.1.1 Présentation du RALI	8
1.1.2 Organisation du RALI	8
2 Introduction à la SMT	9
2.1 Historique de la traduction automatique	9
2.2 La SMT	10
2.2.1 Principe du décodage.....	10
2.2.2 Le modèle de langue (LM)	11
2.2.3 Le modèle de traduction (MT).....	13
2.3 Les PBM	13
2.3.1 Description d'un PBM.....	13
2.3.2 Problématique du PBM.....	16
2.3.3 Filtrage des PBMs	17
2.4 La métrique d'évaluation BLEU	18
3 Présentation du sujet et du contexte	19
3.1 Objectif de mon stage	19
3.2 Travaux antérieurs	20
3.3 Les outils	20
3.3.1 Les outils de développement	20
3.3.2 Les outils de traduction.....	20
3.4 Protocole	22
4 Expérimentations et mise en place d'une baseline	23
4.1 Répartition des paramètres dans un PBM	23
4.1.1 But de l'expérience	23
4.1.2 Déroulement de l'expérience	23
4.1.3 Analyse des résultats	24
4.2 Constitution d'une baseline	28
4.2.1 Heuristique du baseline et procédé	28
5 Algorithme de compressions	32
5.1 Compression par découpages binaires	32
5.1.1 Description de l'algorithme et de l'expérience	32
5.1.2 Résultats	34

5.2	Compression par découpage n-aire incluant les scores	35
5.2.1	Principe.....	35
5.2.2	Résultats	39
5.3	Compression sans perte	41
5.3.1	Principe.....	41
5.3.2	Résultats	42
5.4	Compression heuristique : la ponctuation	42
5.4.1	Principe.....	42
5.4.2	Résultat.....	44
5.5	Compression heuristique : pertinence des paramètres.....	46
5.5.1	Principe.....	46
5.5.2	Résultats	46
6	Conclusion.....	48
6.1	Bilan de l'étude.....	48
6.2	Conclusion personnelle	48
Annexe	50
	Programmation d'outils d'analyse statistique des PBM et des traductions.....	50
	Baselines	52
	Optimisation de l'algorithme du découpage binaire	56
	Découpage n-aire : explications	58
Liste des figures	61
Références	63

1 Introduction

1.1 *Le RALI*

1.1.1 Présentation du RALI

Le RALI¹ (Recherche Appliquée en Linguistique Informatique) est un laboratoire de recherche de l'Université de Montréal fondé en 1996 qui a hérité du programme de recherche en traduction assistée par ordinateur (TAO) poursuivi depuis 1984 par le Centre d'Innovation en Technologie de l'Information (CITI).

Les domaines de compétence du RALI sont les suivants : outils d'aide à la traduction, appariement automatique de textes, génération automatique de textes, réaccentuation automatique, recherche d'information aidée par des outils linguistiques, extraction d'information, identification de la langue et du codage, et transducteurs à états finis.

De ces recherches sont nées plusieurs applications qui ont fait la renommée du RALI, par exemple le projet d'aide à la traduction TransX qui a donné naissance à TransTalk, TransCheck, TransSearch², et TransType, basés sur la recherche en temps réel dans des bases données. Ils se révèlent une véritable boîte à outils pour les traducteurs.

Le RALI est devenu un acteur important dans le domaine de la linguistique informatique et c'est au sein du groupe « traduction » que j'ai réalisé mon stage de fin d'études sous la direction de Philippe Langlais.

1.1.2 Organisation du RALI

Le RALI est sous la responsabilité d'Elliott Macklovitch, qui s'occupe entre autres tâches de la coordination de toute l'équipe. Le RALI compte trois professeurs (Philippe Langlais, Guy Lapalme et Jian-Yun Nie), deux chercheurs (Leila Arras et Fabrizio Gotti), et une vingtaine d'étudiants gradués qui effectuent leurs travaux de recherche sous la direction des professeurs du RALI.

¹ <http://rali.iro.umontreal.ca/>

² <http://www.tsrali.com/>

2 Introduction à la SMT

2.1 Historique de la traduction automatique

La traduction automatique, appelée TA (Traduction Automatique) a véritablement vu le jour pendant la guerre froide, lorsque l'armée américaine s'est intéressée à l'interception et la traduction des messages russes pour en extraire des renseignements stratégiques. Ce projet de traduction automatique donna plus tard naissance à SYSTRAN³ (acronyme pour System Translation), basé principalement sur la consultation de grands dictionnaires bilingues.

En 1966, un rapport prématuré de l'ALPAC⁴ met fin au financement des recherches en TA, ce dernier concluant que la traduction avait un futur assez limité.

Mais le système METEO [Lan05] développé à Montréal en 1977, considéré comme un des plus grands succès en traduction automatique, a prouvé son efficacité dans la traduction de bulletins météorologiques. Les bulletins météo étant écrits dans une langue assez répétitive et formelle, le transfert de l'anglais vers le français est effectué au niveau de chaque mot à l'aide d'un dictionnaire spécialisé pour les idiomes (p. ex. : blowing snow = poudrierie) et pour les sites géographiques (p. ex. : Newfoundland = Terre Neuve). Cette méthode est bien sûre limitée au domaine de la météorologie, mais cette expérience a contribué au renouveau de la traduction automatisée.

Le projet CANDIDE d'IBM lancé dans les années 1980 ouvrit la voie à la traduction automatique statistique, plus communément appelée SMT (Statistical Machine Translation). CANDIDE était fondé sur l'analyse de corpus bilingues afin de trouver des concordances entre les mots.

La SMT repose sur le principe d'apprentissage de paramètres à partir de bitextes nous permettant d'effectuer des traductions sans aucun encodage au préalable de règles ou de base connaissances. Le système s'occupe alors de trouver la meilleure traduction en se basant sur des modèles probabilistes.

³ <http://www.systran.fr/index.html>

⁴ **ALPAC** (Automatic Language Processing Advisory Committee), organisation créée en 1964 pour évaluer les avancées en linguistique informatique et traduction automatique.

2.2 La SMT

2.2.1 Principe du décodage

La traduction automatique statistique repose essentiellement sur le principe du canal bruité de Shannon [Sha49]. L'approche de la traduction automatique statistique est la suivante. Étant donné une phrase française f , nous cherchons la traduction anglaise e qui maximise $p(e/f)$, la probabilité qu'une phrase e soit la traduction de f (on traduira toujours du français f vers l'anglais e dans la section 2.2) :

Équation 1 – Équation de la traduction statistique

$$\arg \max_e (p(e/f)) = \arg \max_e (p(e) \times p(f/e))$$

La formule $p(e/f)$ est décomposée à l'aide la règle de Bayes. On obtient alors l'Équation 1, qui décrit à elle seule le fonctionnement de la traduction automatique statistique :

- $p(e)$ est appelé le modèle de langue
- $p(f/e)$ est appelé le modèle de traduction.
- $argmax$ représente le décodage, l'opération de maximisation.

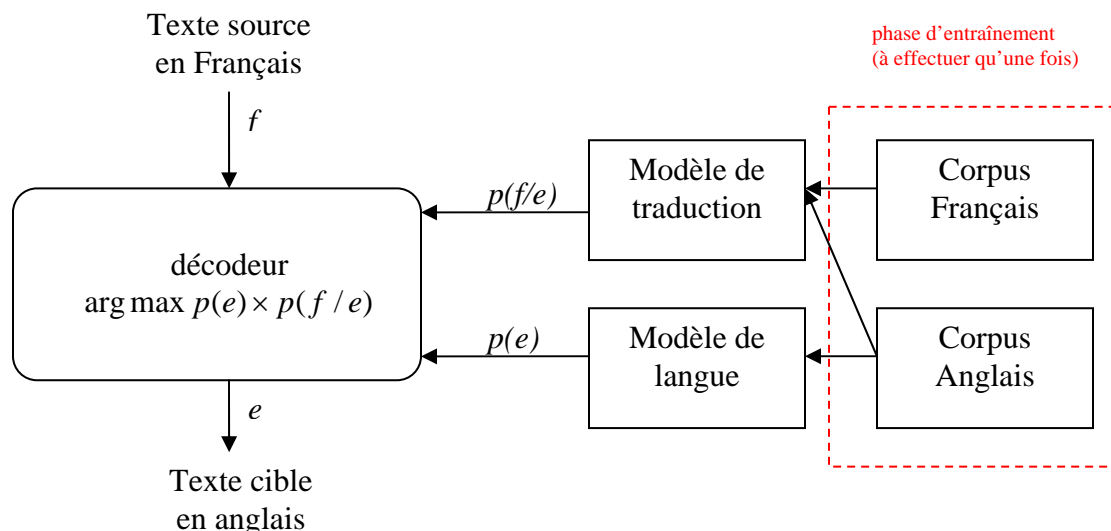


Figure 2-1. Principales composantes de la SMT. Le décodeur prend en entrée le texte source, un modèle de traduction et un modèle de langue pour fournir en sortie le texte traduit. Notons que la langue vers laquelle on veut traduire sera appelé « langue cible »

Pour décrire la SMT de manière plus simple, on peut faire une analogie avec le comportement humain. Lorsqu'une personne est novice dans une langue, la seule technique qu'elle a à sa disposition est de traduire la phrase morceau par morceau, tout en vérifiant au fur à mesure que les parties traduites sont valides. Si elle se rend compte que ce n'est pas le cas, elle revient en arrière en émettant d'autres hypothèses. La conclusion est qu'il y a souvent plusieurs traductions possibles pour une phrase donnée. On se retrouve alors à choisir la meilleure selon son jugement, la notion « d'hypothèse de traduction » est illustrée dans la figure ci-dessous.

Maria	no	daba	una	bofetada	a	la	bruja	verde
Mary	not	give	a	slap	to	the	witch	green
	did not		a	slap	by		green	witch
	no		slap		to the			
	did not give				to			
					the			
			slap			the	witch	

Figure 2-2. Exemples illustrant la notion d'hypothèse de traductions [Koe04]

Lorsque notre personne a traduit les morceaux de la phrase, elle a utilisé sa propre mémoire pour mettre en relation chaque morceau de la langue source en une traduction potentielle dans la langue cible; cette compétence joue le rôle du *modèle de traduction*.

A chaque morceau de phrase traduite, elle a vérifié si tout cela semblait correct par rapport aux traductions précédentes, c'est le rôle que joue *modèle de langue* en traduction automatique.

Enfin, parmi les traductions possibles obtenues, elle a choisi la meilleure; c'est l'opération de maximisation qu'effectue le *décodeur*.

Dans cette partie, nous avons vu qu'il y a trois composantes pour réaliser un décodeur. Nous ne décrivons pas en détail l'opération de maximisation qui ne constitue pas l'objet de notre travail, mais on décrira dans les sections suivantes le modèle de langue et de traduction, pour ensuite aller directement au cœur de notre sujet le PBM.

2.2.2 Le modèle de langue (LM)

Le modèle de langue donne la probabilité d'observer un mot sachant ceux qui le précèdent. Avec cette information, nous pouvons déterminer si une phrase est plus ou moins probable dans la langue cible. Ces modèles sont obtenus à partir d'entraînements (ou d'apprentissage) sur des corpus de la langue cible.

Pour une chaîne e dans la langue cible, on obtient $p(e)$ à l'aide la formule ci-dessous. On utilise la notation suivante, $e = (w_1, w_2, \dots, w_n) = w_1^n$, w_i est le $i^{\text{ième}}$ mot de e et w_i^j est la sous-phrase de e allant du mot i à j .

Équation 2 – formule d'un modèle de langue n-gram

$$p(e = w_1, \dots, w_N) = \prod_{i=1}^N p(w_i / w_{i-n+1}^{i-1}) = \prod_{i=1}^N p(w_i / \underbrace{w_1 \dots w_{i-1}}_h)$$

$p(e)$ donne la probabilité que e soit une phrase plus « probable » dans la langue cible. Le modèle de langue donne la probabilité d'observer un mot étant donné ceux déjà rencontrés.

La probabilité est conditionnée sur l'historique h (l'ensemble des mots déjà vus) mais l'espace des historiques étant trop important, on se contente habituellement de calculer la probabilité d'observer un mot sachant les 2 derniers mots observés (on parle d'un modèle trigramme ou 3-grams). Le modèle trigramme utilise la formule Équation 3.

Équation 3 – formule d'un modèle de langue 3-gram

$$p(e) \approx p(w_1) \times p(w_2 / w_1) \times p(w_3 / w_1 w_2) \times p(w_4 / w_2 w_3) \dots \times p(w_N / w_{N-1} w_{N-2})$$

Pour mieux comprendre le calcul de $p(e)$ avec un modèle de langue 3-gram, voici un exemple d'application sur la phrase « 14 années de traduction en 45 minutes » en Figure 2-3. Exemple d'application du modèle trigramme [Lan06] sur la phrase « 15 années de traduction en 45 minutes ».

$$p(\text{« 15 années de traduction en 45 minutes »}) = p(\text{« 15 »}) \times p(\text{« années »} / \text{« 15 »}) \times p(\text{« de »} / \text{« 15 années »}) \times p(\text{« traduction »} / \text{« années de »}) \times p(\text{« en »} / \text{« de traduction »}) \times p(\text{« 45 »} / \text{« traduction en »}) \times p(\text{« minutes »} / \text{« en 45 »})$$

Figure 2-3. Exemple d'application du modèle trigramme [Lan06] sur la phrase « 15 années de traduction en 45 minutes »

Même si le modèle de langue n'est pas le sujet de notre travail, il est nécessaire à la compréhension du processus de traduction automatique. Par la suite nous allons expliquer ce qu'est un modèle de traduction, ce modèle est le centre d'intérêt de mon stage.

2.2.3 Le modèle de traduction (MT)

Le modèle de traduction donne la probabilité qu'un mot ou un groupe de mots dans la langue source soient traduits par un autre dans la langue cible.

Pour une paire $\langle f, e \rangle$, on attribue une probabilité $p(f/e)$ qui indique quelle est la probabilité que f traduise e .

Le modèle de traduction est obtenu à partir de l'alignement automatique d'un bitexte, c'est-à-dire qu'un algorithme d'apprentissage va se charger de constituer tous les paramètres de notre modèle de traduction à partir d'un corpus bilingue aligné. On appelle cette phase : phase d'entraînement.

Il existe plusieurs types de modèles de traduction. Une approche est de voir un modèle de traduction comme un modèle d'alignement de mots, où chaque mot de la phrase source est aligné avec un (ou plusieurs) mots de la phrase cible. Chaque association a sa probabilité. Nous n'irons pas plus loin sur ce type de modèle; car nous ne les avons pas utilisés pendant le stage.

Une autre approche est de travailler sur des segments de mots (ou « phrases », en anglais). Ces modèles sont appelés « phrase-based models » ou PBM. Mon sujet de stage portera sur ce type de modèles de traduction dit « phrase-based ».

2.3 Les PBM

2.3.1 Description d'un PBM

Les PBMs, comme tous les modèles de traduction, sont entraînés sur des bitextes. Chaque ligne d'un PBM contient des « paramètres » qui sont constitués d'une paire de segments de phrases dans la langue source et cible⁵, munis d'une ou plusieurs probabilités que l'on appelle « score(s) ». Généralement, les PBMs sur lesquels nous travaillerons auront souvent cinq scores appelés « scores de Koehn ⁶ ».

Par abus de langage on appellera dans toute la suite de ce rapport les segments phrases, « phrases ».

La Figure 2-4 montre des exemples de paramètres dont la phrase source est « ambitions »

⁵ Note : la langue source est la langue d'origine. La langue cible est dans laquelle on veut traduire.

⁶ <http://www.iccs.inf.ed.ac.uk/~pkoehn/>

ambitions		achieve		0.000884564	0.0010167	0.0116959	0.0149701	2.718
ambitions		actual ambitions		1	0.267153	0.00584795	0.00164043	2.718
ambitions		aim		0.000436491	0.000373	0.00584795	0.002994	2.718
ambitions		aiming		0.0153846	0.0074349	0.00584795	0.005988	2.718
ambitions		ambition ,		0.04	0.0637813	0.00584795	0.0229168	2.718
ambitions		ambition in		0.0833333	0.0637813	0.00584795	0.002009	2.718
ambitions		ambition		0.0514286	0.0637813	0.105263	0.0838323	2.718
ambitions		ambitions		0.668539	0.533528	0.695906	0.547904	2.718
ambitions		ambitious		0.00486618	0.0101112	0.0233918	0.0299401	2.718
ambitions		aspirations		0.0330579	0.0193237	0.0233918	0.011976	2.718
ambitions		convenience		0.00869565	0.0060976	0.00584795	0.002994	2.718
ambitions		directive		0.000117897	9.96e-05	0.00584795	0.002994	2.718
ambitions		efforts		0.00040032	0.000421	0.00584795	0.005988	2.718
ambitions		goals		0.0121655	0.0125786	0.0292398	0.0239521	2.718
ambitions		objectives		0.00108696	0.0011527	0.0175439	0.011976	2.718
ambitions		of ambition		0.0555556	0.0637813	0.00584795	0.00885678	2.718
ambitions		plans		0.00194175	0.001827	0.0116959	0.008982	2.718

Figure 2-4. Paramètres extrait d'un PBM traduisant le mot « ambitions » du français vers l'anglais.

Dans les PBMs chaque couple phrase source et cible est unique. Une phrase source peut avoir plusieurs traductions possibles, de même pour les phrases cibles. Nous utiliserons cette notation dans le rapport pour décrire un paramètre :

(<phrase source> ||| <phrase cible> ||| <scores>).

Par rapport à l'exemple ci-dessus. Il ne faut pas s'étonner de trouver des paramètres incohérents dans nos PBMs, par exemple « ambitions » ne se traduit pas par « directive », mais généralement les scores qui leurs sont associés les pénalisent grandement lors de la phase de traduction.

Les paramètres ne sont donc pas des traductions au sens strict du terme. Un paramètre donne la probabilité que sa phrase source soit une traduction de sa phrase cible.

La Figure 2-5 est un extrait d'un PBM français-anglais, elle illustre mieux l'idée générale du PBM.

propres ||| individual ||| 0.00120265 0.0006141 0.00130378 0.0009281 2.718
 précis ||| individual ||| 0.00120265 0.0024562 0.00324149 0.0081883 2.718
 préparer ||| individual ||| 0.000601323 0.000307 0.00206612 0.0008598 2.718
 responsabilité ||| individual ||| 0.000601323 0.0018422 0.000380807 0.0013204 2.718
 soi ||| individual ||| 0.000601323 0.000307 0.00154083 0.0006645 2.718
 solo ||| individual ||| 0.000601323 0.0009211 0.333333 0.130435 2.718
 un seul ||| individual ||| 0.00180397 2.94118e-05 0.00547445 0.002025 2.718
 uniquement ||| individual ||| 0.00120265 0.0009211 0.00108167 0.0012616 2.718
 repousser ||| push out ||| 1 0.004561 0.0232558 0.000473372 2.718
 faire face ||| of dealing ||| 0.0555556 0.000467709 0.00255754 2.36037e-05 2.718
 de traiter ||| of dealing ||| 0.0555556 0.0169114 0.00595238 0.0119547 2.718
 de gestion ||| of dealing ||| 0.111111 0.00417294 0.00581395 0.00187233 2.718
 faire un ||| make this an ||| 0.2 0.0202463 0.00384615 6.72556e-05 2.718
 le faire ||| this can be achieved ||| 0.111111 0.000529353 0.00102987 2.91038e-08 2.718
 un groupe de ||| to a group of ||| 0.1 0.10516 0.012987 0.0134743 2.718
 pousser les ||| push the ||| 0.1 0.0041467 0.0714286 0.0260796 2.718
 nous , les ||| we , the ||| 0.227273 0.061982 0.232558 0.231617 2.718
 nous , ||| we , the ||| 0.0909091 0.472375 0.00530504 0.0190754 2.718
 nous ||| we , the ||| 0.0454545 0.65399 1.64974e-05 0.00790044 2.718
 est nécessaire ||| is necessary also ||| 0.5 0.106187 0.000870322 0.00213066 2.718
 du respect des ||| the observation of ||| 0.25 0.000161912 0.016129 0.00019227 2.718
 était , dès ||| was , from ||| 0.5 0.00074995 1 0.0269044 2.718
 précisément ||| precisely because ||| 0.0254237 0.163941 0.00232019 0.00111784 2.718
 fait ||| precisely because ||| 0.00847458 0.0102413 8.78966e-05 3.81431e-06 2.718
 convaincu ||| believe very ||| 0.5 0.0046904 0.00148588 0.000218773 2.718
 nous ||| we , too ||| 0.5 0.65399 8.24872e-06 0.000354562 2.718

Figure 2-5. Extrait d'un PBM

Les avantages des PBMs sont nombreux. Par construction, ils permettent déjà de réordonner les mots (p. ex. : montagne bleue → blue mountain). De plus, de capturer des formes idiomatiques (p. ex. : passer un sapin → pulling a fast one). Ils sont tolérants pour des langues difficiles à segmenter en mots comme le chinois et leur obtention est assez simple car elle se fait à partir de corpus parallèles (le Hansard, les débats parlementaires européens, ...).

2.3.2 Problématique du PBM

Les PBMs font une taille d'environ 1 Go pour 1,7 millions de paramètres. Mais une grande partie de ces paramètres ne serviront pas pendant la traduction, le volume de données est important à manipuler.

D'autre part Les PBMs souffrent de leur simplicité : lors de l'entraînement, on n'a pas de possibilité de généraliser les connaissances et beaucoup de paramètres servent à traduire la même chose comme on peut le voir ci-dessous.

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

Figure 2-6. Exemple de redondance dans un PBM pour les phrases apparentées à « passer un sapin »

La SMT dépend énormément de la qualité des modèles de traduction, de langue mais aussi de la taille des données. Si les modèles augmentent en taille, ce n'est pas seulement l'espace mémoire mais aussi le temps de calcul qui en serait grandement affecté, d'autant plus que celui-ci n'est pas linéaire par rapport à la taille des modèles de traduction.

Avant de rentrer dans les détails du sujet de stage, les sections suivantes introduiront les notions de filtrage et de métrique d'évaluation.

2.3.3 Filtrage des PBMs

On travaillera rarement sur les PBMs d'une taille 1Go, car ils ralentissent les temps de calcul et nécessite un espace disque conséquent. Beaucoup de paramètres sont inutiles quand on traduit d'un texte donné. En pratique, nous filtrons donc un PBM de manière à ne conserver que les seuls paramètres utiles à la traduction d'un texte donné.

Cette opération est appelé *filtrage* (cf. Figure 2-7). Les PBMs auxquels on applique un filtrage seront nommés « PBMs filtrés sur X », X étant le texte à traduire; les PBMs non filtrés seront appelés « PBMs de base » ou simplement « PBMs ».

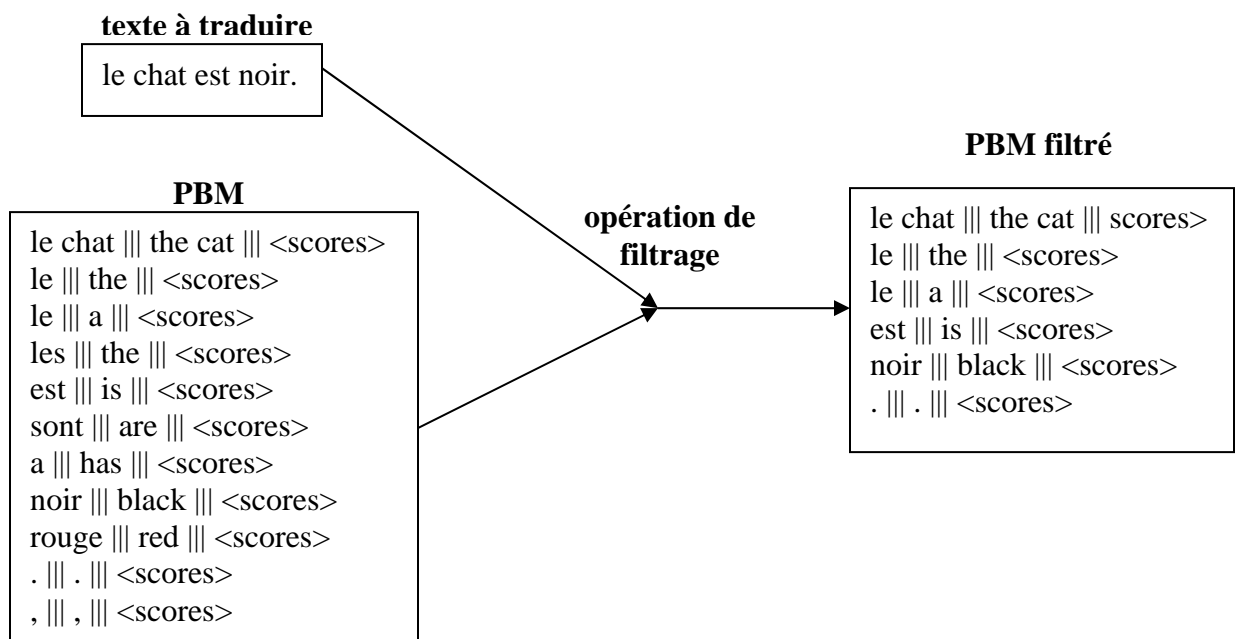


Figure 2-7. Description du filtrage sur le texte « le chat est noir »

Notons que l'opération de filtrage n'influe pas sur la qualité de traduction du PBM, car les paramètres retirés sont totalement inutiles lors de la traduction. Cela ne constitue pas une technique de compression en soi : c'est seulement une restriction du PBM. Lorsque nous utilisons un système de traduction dans un cadre plus général, effectuer le filtrage sur le texte à traduire n'est pas une solution valable. La compression doit s'effectuer au préalable, puis nous utilisons ce même PBM pour toutes nos traductions.

2.4 La métrique d'évaluation BLEU

BLEU (BiLingual Evaluation Understudy) est un outil d'évaluation de traduction automatique inventé par Papinieni [Pap02]; nous l'utiliserons tout au long de nos expériences comme référence pour évaluer et comparer la qualité de nos traductions. Le principe de BLEU est de comparer une traduction de référence (faites par un humain) avec notre traduction automatisée en nous basant sur des séquences de mots n-grams, le but étant de trouver combien de n-grams sont retrouvés dans la traduction de référence. En sortie, BLEU attribue un score entre 0 et 1, sachant que plus on est proche de 1, meilleure est la traduction (nous le ramènerons souvent à un pourcentage). Il a été prouvé que BLEU est fortement corrélé avec le jugement humain.

Cette métrique est l'objet de débats dans la communauté de la SMT, mais elle est souvent utilisée comme une aide à l'évaluation des traductions nous l'utiliserons pour vérifier la qualité de nos approches.

3 Présentation du sujet et du contexte

3.1 Objectif de mon stage

Comme nous l'avons vu précédemment, les PBMs sont des modèles très importants en SMT. Leur taille de l'ordre du Go nous empêche de les utiliser dans des systèmes embarqués. Ceci a un impact par rapport au temps de chargement des modèles, de même que pour le temps de traduction et pour l'espace mémoire requis. Mais pour qu'un système soit performant, il faut que les modèles possèdent un maximum de paramètres chargés. La réduction de la taille du modèle de traduction, lorsqu'elle n'est pas faite correctement, peut entraîner des pertes considérables en terme de qualité de traduction.

Mon sujet consiste à compresser les PBMs avec un minimum de perte en qualité de traduction. Ici « compresser » est un abus de langage, généralement ce terme réfère aux algorithmes et aux outils de compressions comme *gzip*, dans notre contexte on travaille sur la réduction du nombre de paramètres dans nos modèles.

Toutes les méthodes de compression de PBMs testées auront comme approche l'élagage du modèle de traduction, c'est-à-dire retirer des paramètres du PBM. Le but est de généraliser au maximum le critère avec lequel le modèle est élagué afin de maintenir des performances stables d'une traduction à l'autre.

La réduction de paramètre offre des gains en performance, aussi bien au niveau du temps de traduction et que pour l'espace mémoire requis pour stocker les PBM. C'est un atout de la « réduction de paramètre » qui n'est pas offert par les autres approches (algorithme de compression, modification du format des PBM...).

Il n'est pas souhaitable de modifier le format des PBMs (par exemple, modifier le caractère séparateur « ||| ») pour gagner en espace mémoire. Cela impliquerait en effet des modifications au niveau du décodeur Pharaoh [Koe04c], ce dernier étant seulement un programme auquel on a accès, mais que nous ne pouvons modifier.

Dans cette étude, on a commencé par des expériences visant à observer la répartition des paramètres dans un PBM, puis après la constitution d'un baseline, on a évalué diverses approches pour enfin conclure sur le sujet.

3.2 Travaux antérieurs

Beaucoup de chercheurs se sont déjà intéressés à la compression des modèles de langues utilisés dans la recherche d'information et la reconnaissance de la parole en utilisant des approches diverses : l'élagage du modèle (retirer des paramètres), encodage des paramètres dans un format plus compact, ... On peut observer que beaucoup d'efforts ont été faits pour réduire la taille de certains modèles, mais la plupart concernent les modèles de langues et non les PBMs.

Marcello Federico et Nicola Bertoldi [FeBe06] se sont intéressés à la compression de PBMs en discrétisant les probabilités.

Réduire la taille des modèles en SMT est un problème connu, mais la réduction du nombre de paramètres dans un modèle PBM n'a pas été traitée. La plupart des travaux ne portant pas sur les PBMS, utilisant des approches différentes et ne présentant pas les mêmes avantages ; nous ne les reprendrons pas dans nos travaux.

3.3 Les outils

3.3.1 Les outils de développement

Durant mon stage, j'ai eu à ma disposition un poste de travail sous environnement UNIX. Tous les programmes que j'ai réalisés seront écrits en C++ et seront compilés à l'aide du compilateur *g++*.

J'ai utilisé l'environnement *tcsh* pour écrire mes scripts et lancer mes expériences. Généralement, cela consiste à lancer mes propres programmes sur diverses données afin d'observer le maximum de résultats possibles.

Pour la présentation de mes résultats sous forme de graphiques, j'utiliserai le logiciel *gnuplot*.

Les traductions nécessitant beaucoup d'espace mémoire et de temps de calcul, j'ai eu à ma disposition des clusters, un groupe de machines partagées par tous les membres du RALI sur lesquelles on lance la plupart de nos calculs via notre terminal. Ces machines ont notamment les capacités nécessaires pour charger en mémoire des modèles de traductions de grande taille et évitent de surcharger nos propres postes de travail.

3.3.2 Les outils de traduction

Le RALI possède tous les outils nécessaires à la traduction automatique statistique. Dans le cadre de mon stage, j'utiliserai surtout le décodeur *Pharaoh* avec lequel je ferai mes traductions et le script *mteval* pour les évaluer.

3.3.2.1 Le décodeur Pharaoh

*Pharaoh*⁷ est le décodeur écrit par Phillip Koehn [Koe03] à l'université de Californie du Sud. C'est un décodeur « phrase-based » gratuit offrant des performances état de l'art. Son utilisation est très répandue dans la communauté scientifique, y compris au RALI.

Pharaoh prend en entrée un modèle de langue, un modèle de traduction et le texte à traduire dans la langue source. En sortie, nous obtenons le texte traduit dans la langue cible.

Sans rentrer dans les détails, constituer un modèle de langue nécessite l'utilisation du toolkit *SRILM*⁸. Pour le modèle de traduction, nous utilisons *Giza++*⁹. Dans notre travail, nous avons utilisé les bitextes du corpus *Europarl*.

3.3.2.2 Le script mteval

Le script d'évaluation *mteval* me permet d'évaluer mes traductions selon la métrique BLEU. Il prend en entrée le texte original (dans la langue source), la traduction automatisée (dans la langue cible) et une traduction de référence.

⁷ <http://www.isi.edu/publications/licensed-sw/pharaoh/>

⁸ <http://www.speech.sri.com/projects/srilm/>

⁹ <http://www.fjoch.com/GIZA++.html>

3.4 Protocole

Dans cette section, on explique le protocole utilisé pour nos essais de compressions. En situant notre travail par rapport au processus de traduction décrit précédemment (en Figure 2-1), on obtient une nouvelle figure ci-dessous.

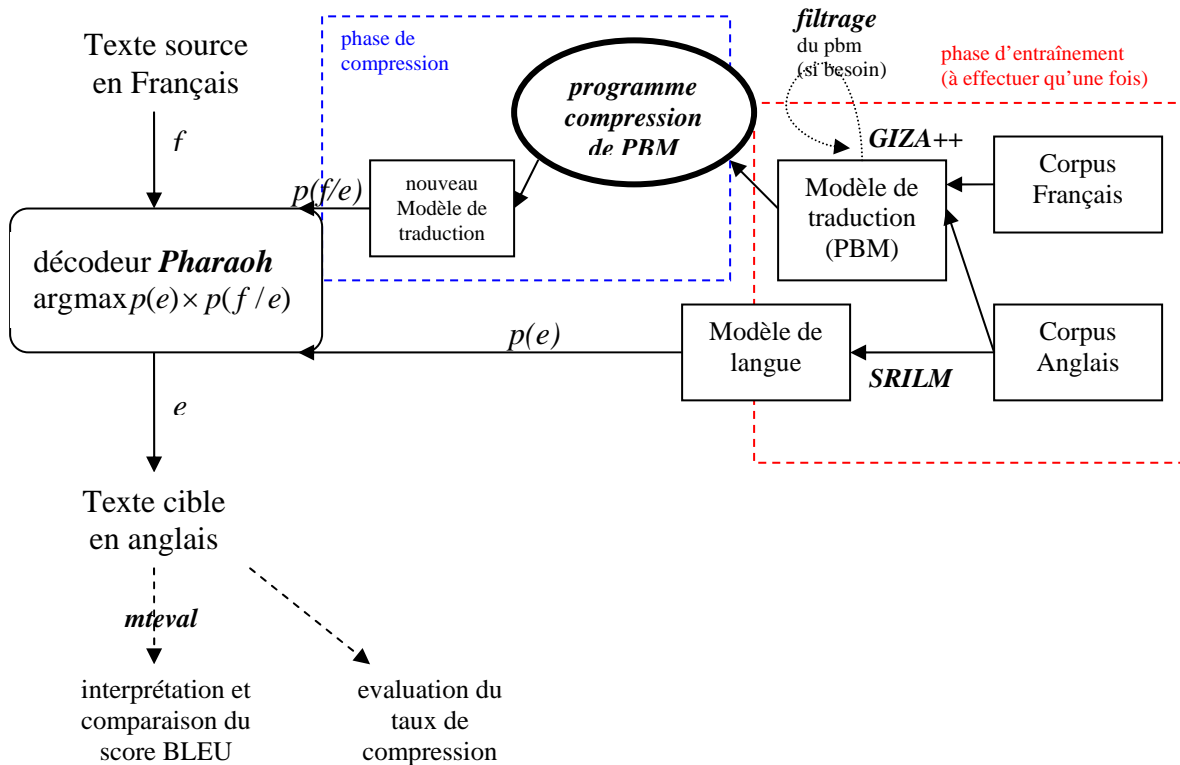


Figure 3-1. Protocole d'un test de compression, notre travail se situe dans la « phase de compression » encadrée en bleu.

La Figure 3-1 illustre le déroulement classique d'un test de compression. La phase d'entraînement encadrée en rouge n'a lieu qu'une seule fois pendant tout le stage, elle consiste à créer nos modèles de langue et de traduction.

Nous filtrons par la suite le PBM si cela s'avère nécessaire pour l'expérience, ce filtrage est effectué qu'une fois.

On applique ensuite un algorithme de compression sur notre PBM (la partie encadrée en bleu), c'est à ce niveau se situe notre travail. À chaque expérience nous proposons un algorithme différent pour compresser le PBM. Notons que le PBM est compressé qu'une fois, il est inutile de réitérer le processus.

On traduit le texte source à l'aide de notre nouveau PBM et du modèle de langue. Au final l'évaluation de la traduction nous permet de conclure sur l'efficacité de notre algorithme de compression.

4 Expérimentations et mise en place d'un baseline

4.1 Répartition des paramètres dans un PBM

Dans cette section on cherche à élargir nos connaissances par rapport au comportement décodeur. La contribution des paramètres des PBM pendant la traduction nous est encore inconnue. À travers des expériences on va étudier cette contribution, pour éventuellement s'en servir plus tard pour établir des stratégies de compression.

4.1.1 But de l'expérience

On a d'abord évalué la répartition des paramètres selon la taille des phrases sources et cibles dans nos PBMs et les PBMs filtrés. De même, il serait intéressant de voir quels types de paramètres sont utilisés pendant la traduction.

Le but de l'expérience étant d'observer la contribution des paramètres avant et pendant du processus de traduction.

Concrètement pour établir nos statistiques, les paramètres ont été répertoriés selon la taille de leur phrase source et cible. Pour symboliser une catégorie, on a utilisé la notation $[i-j]$ pour les paramètres dont la taille de la phrase source est i et celle de la phrase cible est j .

Dans nos PBM les phrases sources et cibles ne dépassent pas 7 mots, ce qui fait varier i et j de 1 à 7 et nous donne 49 catégories de paramètres.

4.1.2 Déroulement de l'expérience

Pour cette expérience, j'ai utilisé les corpus Europarl, une ressource regroupant des débats parlementaires européens en français, allemand, anglais, et espagnol.

Nous avons à notre disposition en français, espagnol et allemand:

- quatre corpus (textes à traduire) nommés : « devtest2006 », « dev2006 », « test2006-ood », « test2006-rest ».
- un PBM.

Ce qui nous fait 3 PBM, 12 PBM filtrés et 12 textes à traduire au total. On effectue toutes les opérations de traductions vers l'anglais.

Nous établissons les statistiques sur les traductions des quatre corpus, les PBMs de base et les PBMs filtrés sur ces corpus. On a programmé « *pbm_stat* » pour établir des statistiques sur les PBM et « *traduction_stat* » pour les traductions (les détails d'implémentation sont en Annexe).

Étant donné le nombre d'opérations de filtrage, de traduction et de statistiques qu'il y a à faire, on lancera toutes ces tâches en parallèle à l'aide de scripts *tsh* sur des clusters du RALI.

Notre tâche est donc de lancer :

- *pbm_stat* sur les PBMs de base et filtrés,
- lancer *traduction_stat* sur les traductions
- Mettre tous les résultats sous forme de graphique *Excel* (avec *gnnumeric*).

4.1.3 Analyse des résultats

On observe avant toutes choses une homogénéité des résultats; les statistiques sur les traductions et sur les PBM se ressemblent aussi bien entre différents corpus que sur différentes langues. On restreindra nos discussions sur la traduction du corpus « test2006-rest.fr », le PBM filtré sur le corpus « test2006-rest.fr » et le PBM français.

Le diagramme en Figure 4-1 donne la répartition des paramètres pour le PBM français, en abscisse on a les paires $[i-j]$ et en ordonnée la fréquence de ces paires.

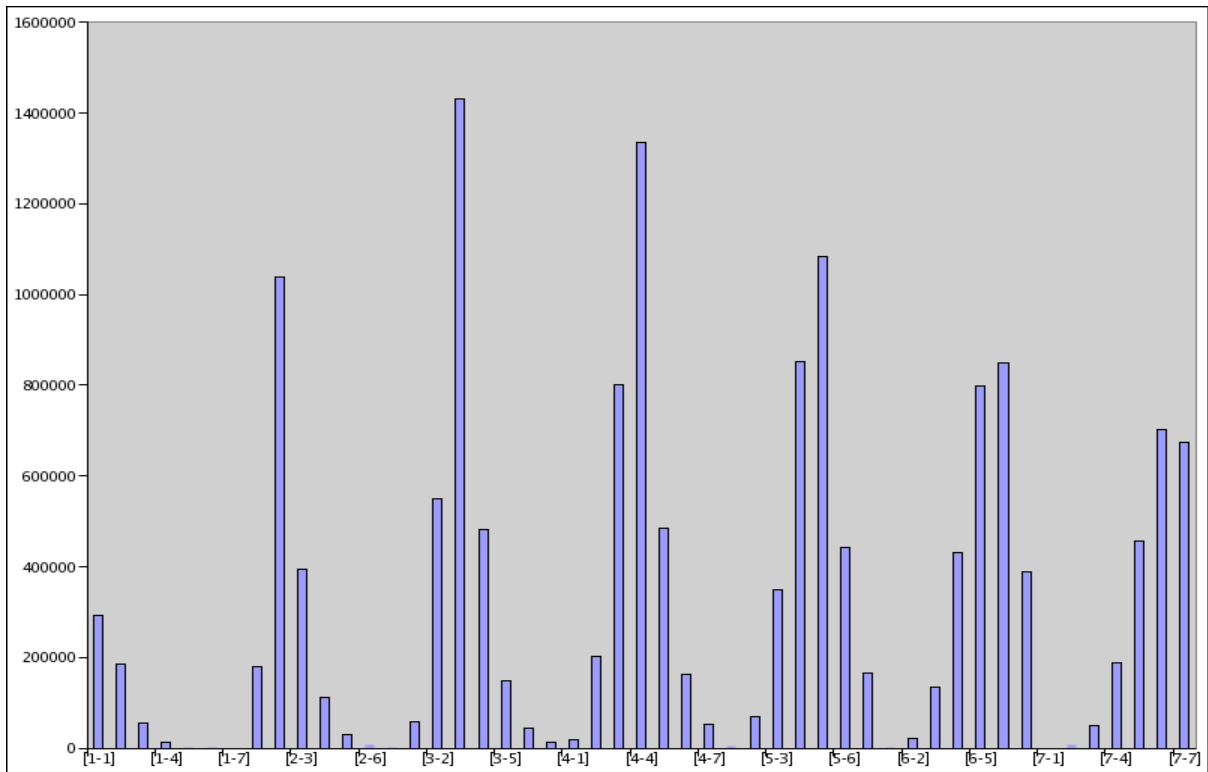


Figure 4-1. Répartition des paramètres dans le PBM français en fonction de la longueur de la phrase source et cible

La plupart des unités sont réparties autour de la médiane ([1-1], [2-2], [3-3] ... [7-7]). En effet, les paramètres dont l'écart de taille entre la phrase source et cible sont rares (par exemple, un mot est rarement traduit par un segment de 7 mots dans une autre langue), d'où le manque de paramètre en dehors de la médiane.

La répartition des paramètres après filtrage du PBM sur le texte « test2006-rest.fr » est illustré ci-dessous, les unités de longueurs sont les même que pour les schémas précédents.

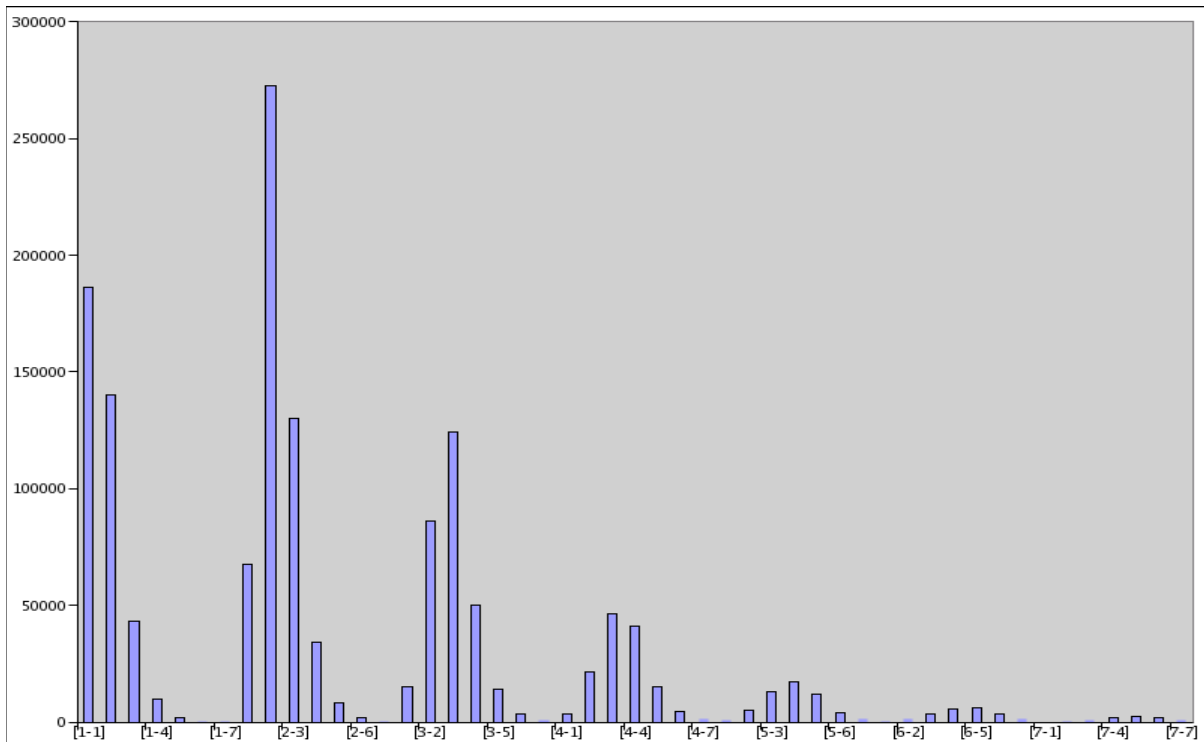


Figure 4-2. Répartition des paramètres dans le PBM français filtré sur le corpus « test-rest » en fonction de la longueur de la phrase source et cible

Après filtrage du PBM sur le corpus « test-rest » français, on observe que lors de cette phase de filtrage, la plupart des paramètres dont les phrases source et cible ont une taille supérieure à 4 est réduit significativement, il ne représente plus que 5% du PBM filtré. La majorité des paramètres sont donc de moyenne et petite taille après un filtrage du PBM.

La Figure 4-3 donne la répartition des paramètres utilisés lors de la traduction, les unités de longueur restent les même.

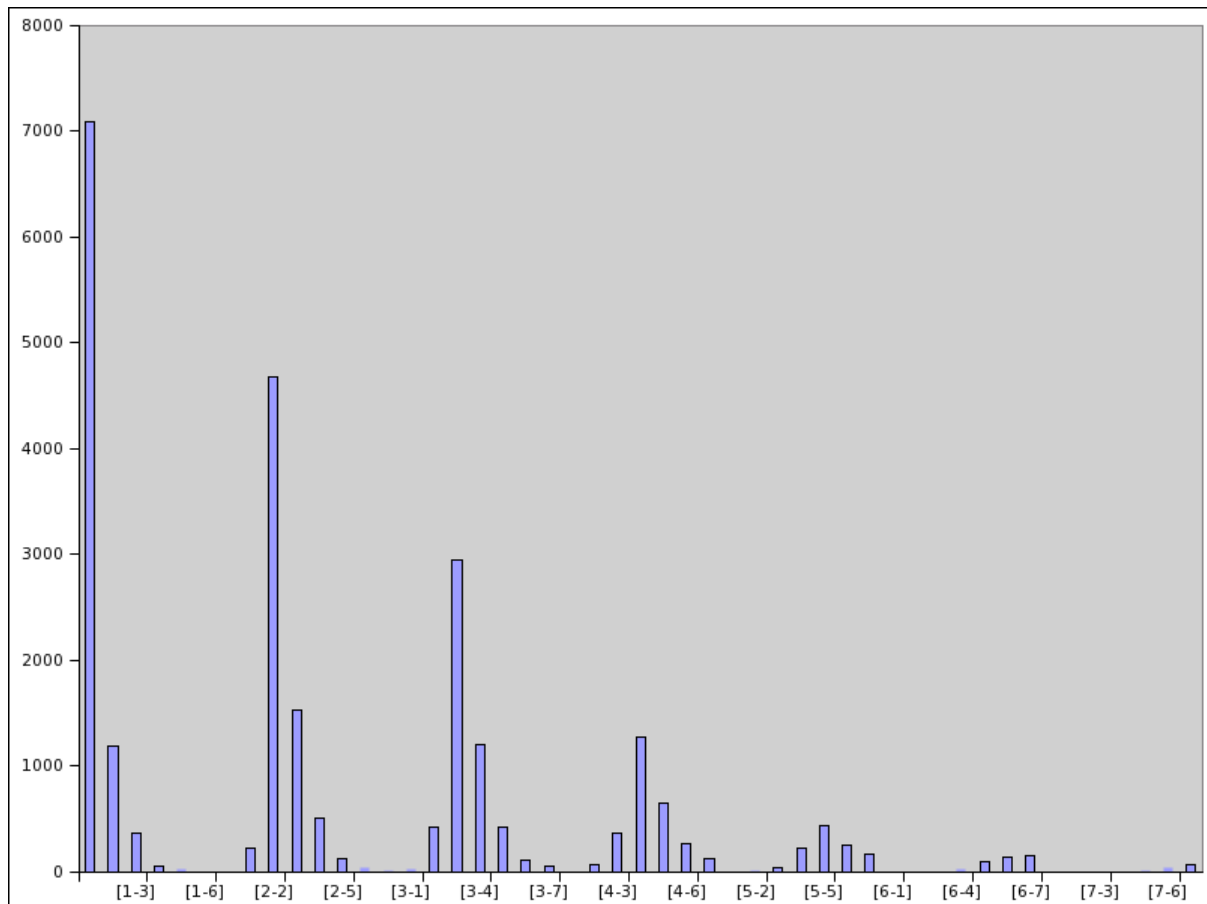


Figure 4-3. Paramètres utilisés lors de la traduction du texte « test2006-rest.fr » en fonction de la longueur de la phrase source et cible

Nous pouvons voir le profil des paramètres utilisés lors du processus de traduction du texte « test2006-rest ». On voit que les paramètres au dessus de [4-4] sont les moins utilisés pendant la traduction. En général plus les paramètres sont petits plus ils sont utilisés pendant la traduction.

On a observé lors de cette expérience que le décodeur favorise les paramètres dont les phrases source et cible sont assez petites (d’une allant de 1 à 3). Lorsque l’on compare ce résultat avec les répartitions précédentes sur le PBM non filtré, on constate qu’à la base les paramètres au dessus de [4-4] représentaient 60% du modèle d’origine.

Dans la prochaine section, nous avons créé un *baseline* à partir d’une heuristique simple pour évaluer l’efficacité de nos algorithmes.

4.2 Constitution d'une baseline

Avant de se lancer dans des expérimentations et des tentatives de compression, il faut pouvoir se constituer un « baseline », c'est le résultat avec lequel on compare nos algorithmes de compression.

Premièrement, nous avons établi d'une heuristique « naïve » qui sert d'algorithme de compression dans ce contexte. Dans un deuxième temps nous avons itéré le processus de tel manière à obtenir plusieurs baselines pour un même PBM.

4.2.1 Heuristique du baseline et procédé

On a décidé de retirer de nos PBMs filtrés certains paramètres, en partant de ce principe : *pour i et j fixé, on garde seulement les paramètres dont la taille de la phrase source et cible sont respectivement inférieur ou égal à i et j .*

Cette heuristique naïve nous sert d'algorithme de « compression », on l'appellera « élagage » dans cette section.

On va se constituer plusieurs baselines pour un PBM donné, on applique notre algorithme pour i et j allant de 1 à 7, on obtient alors 49 PBM.

(Une phrase dans un PBM n'excédant pas 7 mots, appliquer l'algorithme pour $i=7$ et $j=7$ laisse le PBM inchangé).

On fait une traduction par sous-PBMs obtenus puis on évalue le BLEU. On obtient pour chaque élagage différent un score BLEU.

La Figure 4-4 illustre ce procédé pour un PBM filtré sur un texte X .

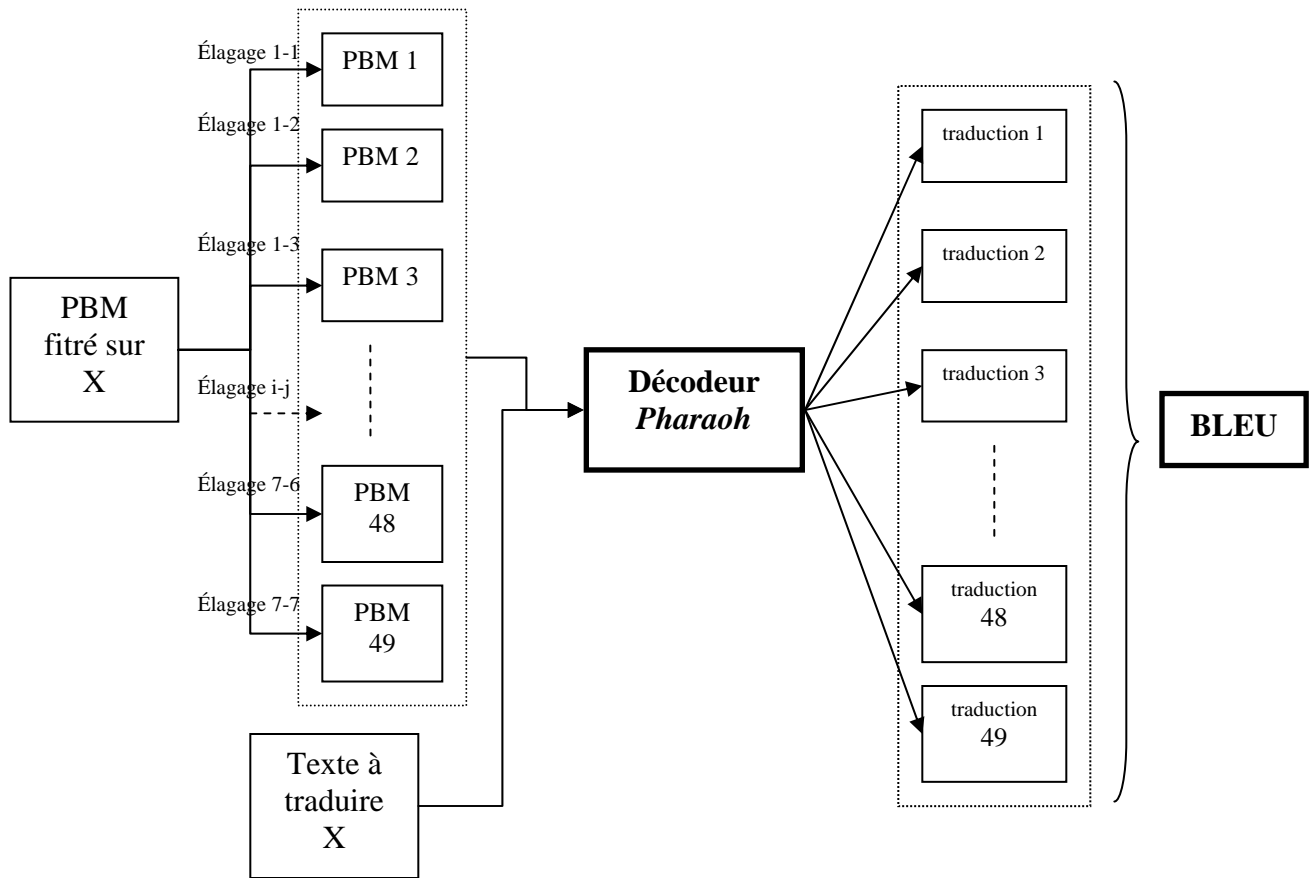


Figure 4-4. Constitution des baselines, les traductions du texte X sont faites pour différents PBM élagués

Les deux graphiques ci-dessous donnent les scores BLEU des traductions des textes « test2006-ood » « test2006-rest » (français) pour différents élagages de leurs PBMs filtrés respectifs.

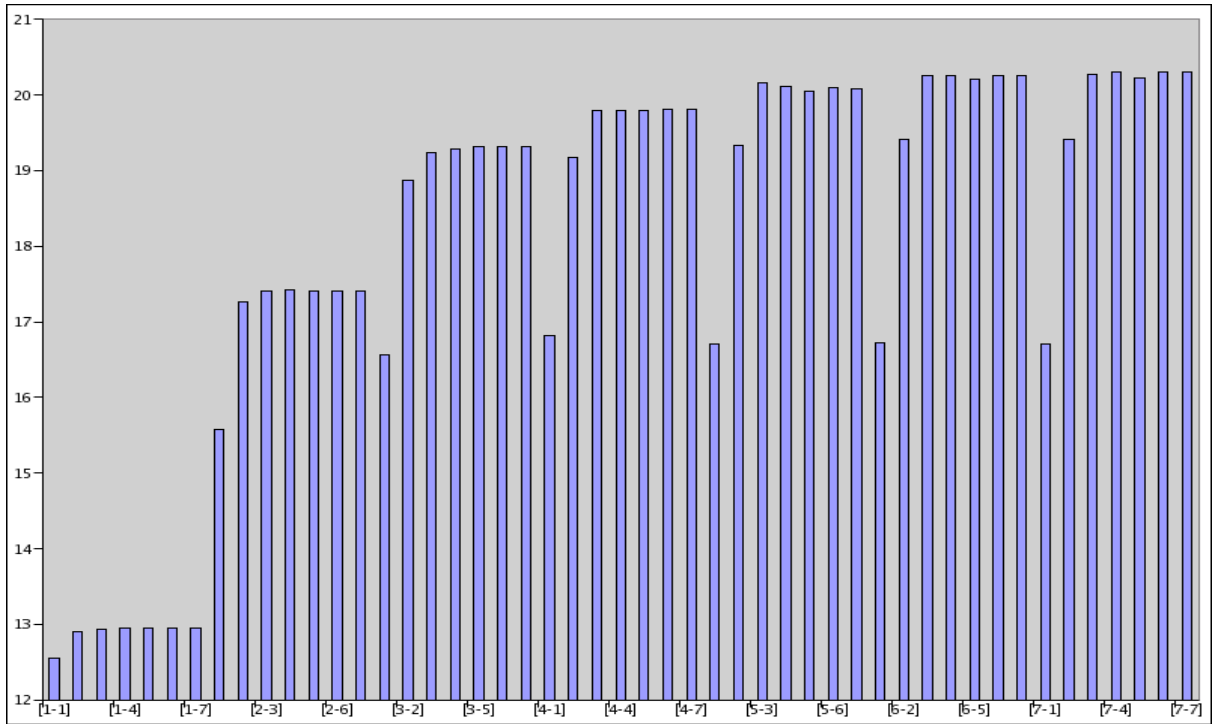


Figure 4-5. Scores BLEU obtenus sur les 49 traductions du corpus « test2006-ood » français, en abscisse nous avons le critère d'élagage et en ordonnée les scores BLEU obtenus

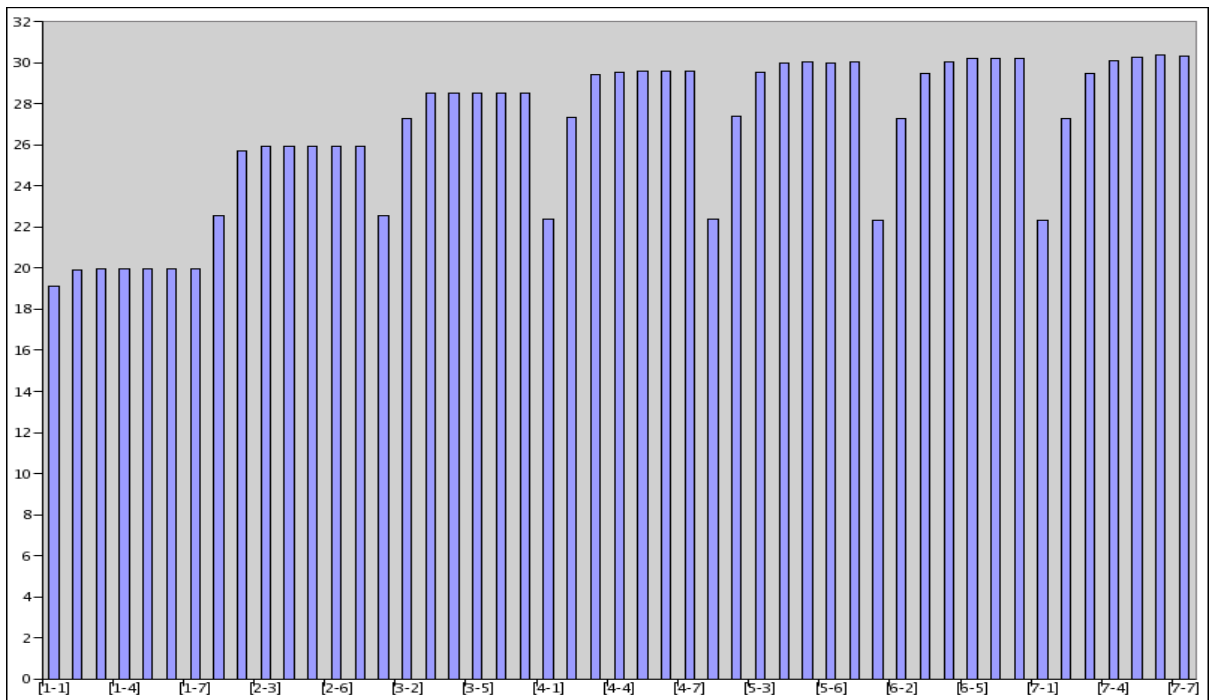


Figure 4-6. Scores BLEU obtenus sur les 49 traductions du corpus « test2006-rest », en abscisse nous avons le critère d'élagage et en ordonnée les scores BLEU obtenus

Nous observons des pertes très importantes lorsque l'écart entre i et j est trop important, car cela revient à garder des paramètres qui sont rares dans nos PBMs. De même, on voit que le score décroît logiquement au fur à mesure que nous restreignons le critère d'élagage de nos PBMs.

Grâce aux résultats obtenus, nous avons plusieurs baselines à notre disposition. Nous pourrions ainsi comparer lors des tests de compression pour des tailles de PBMs fixes quels scores nous aurions dû obtenir ou inversement pour des scores équivalents quels taux de compressions nous aurions dû obtenir au minimum.

Dans la suite du rapport on comparera souvent les baselines avec ceux de nos algorithmes de compression. Les baselines pour chaque corpus sont en Annexe dans la section « Baselines » dans un format différent de celui présenté précédemment.

5 Algorithme de compressions

Dans cette section nous rentrons dans le cœur du sujet : la compression des PBM. Nous avons deux catégories de méthodes dites de « découpage » et les heuristiques. Nous abordons dans un premier lieu deux méthodes de découpage en 5.1 et 5.2 puis nous continuons sur les méthodes heuristiques jusqu'à la fin de la section. Pour ensuite conclure sur le sujet en section 6.

5.1 Compression par découpages binaires

5.1.1 Description de l'algorithme et de l'expérience

Le but d'un algorithme de découpage est de conserver dans le PBM les sous-paramètres qui reconstitue des plus gros paramètres (ex : s'il existe des paramètres « le chat ||| the cat ||| <scores> » et « est noir ||| is black ||| <scores2> » alors on peut supprimer le paramètre « le chat est noir ||| the cat is black »).

Décrivons l'algorithme en Figure 5-2. Pour chaque paramètre du PBM « s ||| t ||| <scores> » on génère tous les découpages binaires de la partie source s (comme dans la Figure 5-1).

le ciel est bleu ||| the sky is blue ||| 0.99

Les découpages binaires de la phrase source donne :

(« le », « ciel est bleu »), (« le ciel », « est bleu »), (« le ciel est », « bleu »).

Figure 5-1. Exemple de découpages binaires possibles pour la partie source « le ciel est bleu »

Pour chaque couple (s_1 , s_2), nous recherchons tous les paramètres dont les parties sources correspondent à s_1 et s_2 (nous obtenons donc deux listes de paramètres, qu'on appellera X et Y). Pour chaque paramètre de X et Y , on vérifie si la concaténation des parties cibles (t_1 et t_2) donne bien t .

Si c'est le cas, nous avons trouvé un découpage du paramètre « s ||| t ||| <scores> », donc nous n'avons plus besoin de ce paramètre dans ce PBM, car il en existe une version « découpée ».

$$\forall (s \ ||| \ t \ ||| \langle scores \rangle) \in PBM \text{ tel que } s1 + s2 = s$$

$$\text{si } \exists (s1 \ ||| \ t1 \ ||| \langle scores1 \rangle) \text{ et } \exists (s2 \ ||| \ t2 \ ||| \langle scores2 \rangle) \in PBM$$

$$\text{tel que } t1 + t2 = t \text{ alors retirer le paramètre } s \ ||| \ t \ ||| \langle scores \rangle \text{ du PBM}$$

Figure 5-2. Algorithme proposé pour la compression par découpage binaire, la concaténation entre deux chaînes est symbolisé par « + »

L'algorithme est illustré dans la Figure 5-3, pour un PBM et un paramètre donné :

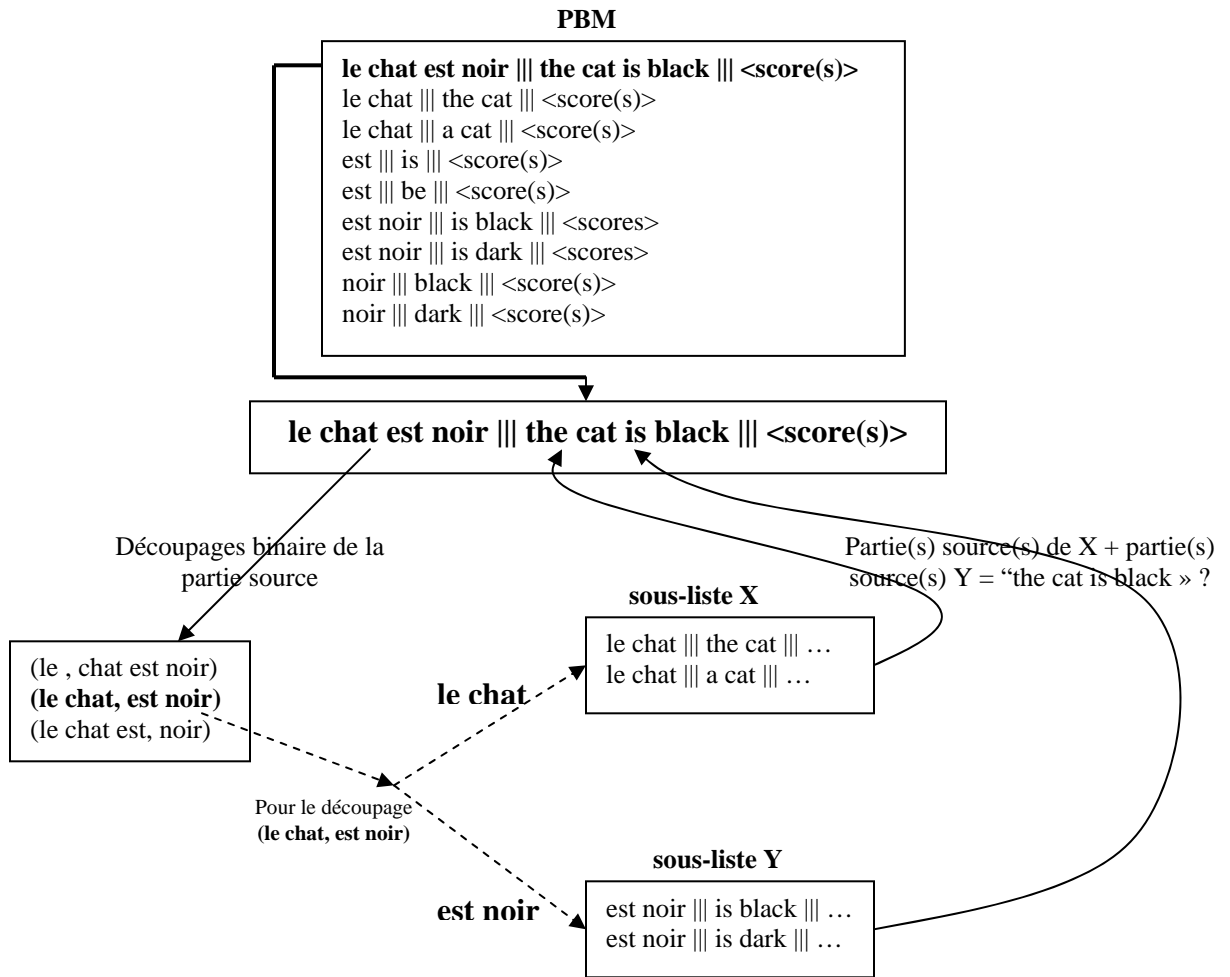


Figure 5-3. Application de l'algorithme du découpage binaire sur le paramètre « le chat est noir »

Avec cette heuristique, nous espérons voir comment le décodeur va se comporter lors de la traduction. Même si l'algorithme semble simple, il sera entre autre intéressant de voir de quel ordre sera la perte de BLEU et quel sera le taux de compression.

5.1.2 Résultats

Nous avons compressés les PBM filtré sur les textes test2006-rest et test2006-ood (en français, espagnol et allemand), ensuite ces textes ont été traduits en utilisant les PBM compressés. Les scores BLEU obtenus à la suite de l'expérience sont présentés en Figure 5-4 et les taux de compression en Figure 5-5.

Sur la colonne de gauche, on peut lire les performances standards (c'est-à-dire sans compression) et sur la colonne de droite avec compression .

texte source	sans compression	Avec compression
Test2006-rest (fr)	30.35	25.28
Test2006-ood (fr)	20.31	18.02
Test2006-rest (es)	30.80	25.56
Test2006-ood (es)	25.17	22.45
Test2006-rest (de)	24.91	21.55
Test2006-ood (de)	15.90	13.91

Figure 5-4. Scores BLEU obtenus sur les traductions après compression binaire

PBM filtrés	sans compression (en Mo)	avec compression (en Mo)
test2006-rest (français)	107	39
test2006-ood (français)	59	28
test2006-rest (espagnol)	108	42
test2006-ood (espagnol)	57	28
test2006-rest (allemand)	74	35
test2006-ood (allemand)	40	23

Figure 5-5. Taille (en Mo) des PBM obtenus après compression binaire

On observe que sur des PBMs filtrés, le taux de compression est de 40% à 60%, mais il s'accompagne d'une perte de BLEU considérable de 2% à 5%.

Nous avons observé par la suite la répartition des paramètres dans les PBM compressés (comme dans la section 4.1). La majorité des paramètres présents après compression sont des paramètres qui ont des longueurs de phrases sources ou cibles d'au maximum 3. Ceci explique la mauvaise qualité des traductions, poussant le décodeur à traduire presque « mot à mot ». De plus on ne s'est pas assuré de retrouver la traduction initiale, par rapport à l'exemple Figure 5-3 « le chat est noir » a pu être traduit en « the cat is dark » au lieu de « the cat is black », ce qui nous pénalise en qualité de traduction.

Les taux de compression sont en moyenne de 50 %, bien que satisfaisants, ils sont accompagnés de mauvais résultats en terme de BLEU.

En dehors des résultats obtenus précédemment, l'algorithme a un autre point faible : son temps de calcul. Pour un PBM d'environ 100 Mo, il faut environ 17h de calcul pour obtenir un PBM compressé. L'idée est d'avoir un algorithme assez rapide pour le lancer sur un PBM non filtré, afin d'obtenir le taux de compression pour des modèles beaucoup plus grands dont la taille est de l'ordre du Go.

On a donc étudié l'algorithme pour y trouver des possibilités d'optimisation. Les résultats sont en Annexe dans la section « Optimisation de l'algorithme ».

5.2 Compression par découpage n-aire incluant les scores

5.2.1 Principe

Le principe du découpage n-aire est de généraliser l'approche du découpage binaire et on inclut les scores lorsqu'on prend la décision de retirer ou pas un paramètre. En utilisant les scores on espère pouvoir retrouver nos découpages lors de la traduction.

Nous avons utilisé la programmation dynamique pour programmer cet algorithme. Mais celui-ci étant compliqué à décrire nous allons partir d'un exemple.

Il faut savoir que l'algorithme est appliqué seulement aux paramètres de longueur supérieure ou égale à 4 mots. La Figure 5-6 est un PBM simple sur lequel on va s'appuyer pour donner une idée générale de l'algorithme.

Mr le président , je veux Mr speaker , I want 0.2 0.1 0.5 0.9 2.718
veux want 0.8 0.3 0.2 0.4 2.718
je veux , I want 0.6 0.3 0.1 0.01 2.718
Mr Mr 0.1 0.01 0.9 0.5 2.718
Mr le Mr 0.4 0.5 0.1 0.1 2.718
le président speaker 0.1 0.7 0.6 0.6 2.718
, je , I 0.5 0.8 0.7 0.3 2.718
Mr le président Mr speaker 0.1 0.1 0.2 0.3 2.718

Figure 5-6. Exemple de PBM auquel on va appliquer le découpage n-aire au paramètre «Mr le président , je veux»

Pour chaque paramètre, on crée une matrice de taille NxN (N étant la longueur de la partie source). La partie source de ce paramètre sera notée $p = (p_1, p_2, \dots, p_N) = p_1^N$. Une cellule (i,j) (avec $i \leq j$) de la matrice représente la sous-phrase $p_i^j = (p_i, p_{i+1}, \dots, p_j)$.

Pour chaque cellule (i, j) de la matrice, s’il existe des paramètres dont la phrase source est p_i^j . On ajoute ces paramètres à la cellule(i,j) en tant qu’hypothèses de traduction. On obtient alors un tableau initialisé comme sur la Figure 5-7.

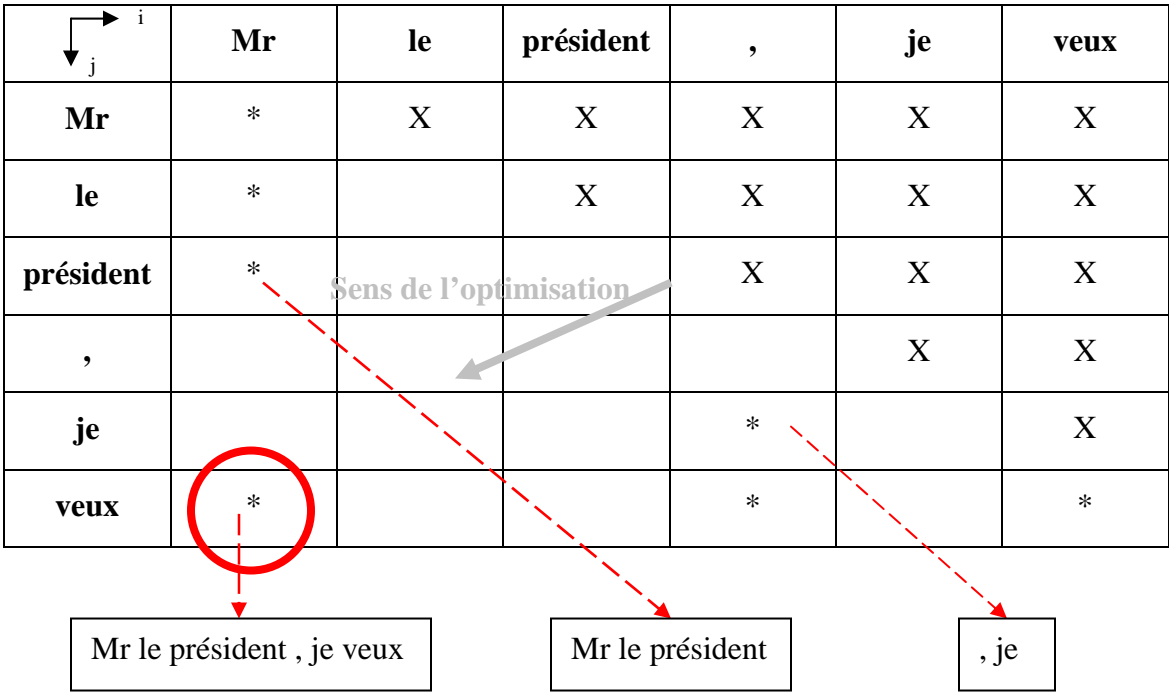


Figure 5-7. Phase d’initialisation de l’algorithme appliquée sur l’exemple le pbm en Figure 5-6, les « * » représentent les cases initialisées, les cases vides ne contiennent pas d’hypothèses et les X sont des cases inexistantes.

Sur le tableau obtenu précédemment, on applique l'algorithme de maximisation en Équation 4.

Équation 4 – Algorithme de maximisation utilisé pour le découpage n-aire. La fonction « cout » renvoie les couts de toutes les hypothèses associés à p_i^j .

$$table(i, j) = \begin{cases} \max(cout(t, p_i^i)), i = j \\ \max\left(\max(cout(t, p_i^j)), \max_{d \in \{i, i+1, \dots, j-1\}}(table[i, d] + table[d + 1, j])\right), \text{sin on} \\ t \in TraductionPossibles \end{cases}$$

Cette équation est récursive, on la lance à partir de la cellule (1,6) (celle entourée en rouge dans la Figure 5-7), car on veut récupérer le résultat dans la case représentant « Mr le président , je veux », le paramètre que l'on cherche à découper. Sans rentrer dans les détails, cette équation permet de trouver le meilleur découpage en en faisant intervenir les scores comme critère de découpage. Un paramètre est retiré seulement si son découpage a un meilleur score que lui.

Nous n'irons pas plus loin dans les explications de cet algorithme, il faut juste retenir l'idée générale de l'algorithme : trouver un découpage du paramètre en faisant intervenir le score pour prendre notre décision. Pour plus de détails voir la section « Découpage n-aire : explications » en Annexe.

Ci-dessous, l'algorithme du découpage n-aire écrit en pseudo-code :

```

Hypothèse cell[][] ;
Pour nbmots 2 à N-1{
    Pour i=1 à N-nbmots+1{
        Pour j=i à i+nbword-2{
            X ← nbmots+i-1;
            Si ∃ cell[i,j] && ∃ cell[j+1,X] {
                match(i,j,j+1,X);
            }
        }
    }
}

```

Figure 5-8. Algorithme général du découpage n-aire

```

float S;
bitset T;
liste d'hypothèse H1,H2,H3
Fonction match(i,j, X){
    k=j+1;
    H1 ← cell[i,j];
    H2 ← cell[k,l];
    H ← cell[i,l];
    Pour h1 ∈ H1{
        Pour h2 ∈ H2{
            T ← target(h1,h2); % si h1 et h2 incompatible STOP
            S ← score(h1) + score(h2);
            ∀ h ∈ H / target(h)=T{
                Si s > score(h){
                    MAJ(h,s,k);
                }
            }
        }
    }
}

```

Figure 5-9. Algorithme de la fonction MATCH

```

Fonction MAJ(hypothese,score,backpointer){
    hypothese→score=score;
    hypothese→backpointer=backpointer;
}

```

Figure 5-10. Algorithme de la fonction MAJ

5.2.2 Résultats

Nous observons sur la Figure 5-11 que les pertes varient selon la taille du PBM en entrée, généralement lorsque le PBM est filtré sur les corpus test2006-ood, résultant en un PBM de petite taille (proche de 60 Mo) on observe des pertes de moins d'1% par rapport au BLEU initial par contre pour les PBMs filtrés sur les corpus test2006-rest nos pertes sont entre 1.33% et 1.63%. Les taux de compression (Figure 5-12) varient entre 5% et 15% et sont encore proportionnels à la taille des PBMs.

Pour une analyse plus précise, comparons les résultats obtenus avec les baselines en Annexe. Nous rappelons que nous avons appliqué nos découpages seulement aux paramètres dont la partie source est supérieure à 4, ce qui revient à comparer avec les modèles élagués dont on a conservé les phrases sources de taille 3 (les lignes « [3-1] » « [3-2] » ... « [3-7] »). En général, nos résultats sont un peu meilleurs pour ce qui est du score BLEU mais nos gains en compression sont légèrement inférieurs.

Cette approche montre d'assez bons résultats, comparés à nos baselines, mais nous ne pouvons aller plus loin car il faudrait à partir de là utiliser le modèle de langue pour s'assurer de la « reconstruction » de nos paramètres lors de la traduction, ce qui est trop complexe et hors de notre portée par rapport au temps qu'il nous reste avant la fin du stage. Nous allons désormais tenter de compresser les PBM à l'aide d'heuristique.

Pbm source	Sans compression	Découpage binaire	baseline (avec un PBM à répartition équivalente)	Avec compression (découpage n-aire)
Pbm filtré sur test2006-rest (fr)	30.35	25.28	28.55	28.72
Pbm filtré sur test2006-ood (fr)	20.31	18.02	19.31	19.48
pbm filtré sur test2006 –rest (es)	30.80	25.56	29.40	29.47
pbm filtré sur test2006 –ood (es)	25.17	22.45	24.70	24.56

Figure 5-11. Evaluation des scores BLEU obtenus par compression n-aire (colonne de droite), comparés avec les performances standards, les résultats du découpage binaire et le baseline.

pbm source	Sans compression	Baseline	avec compression n-aire
Pbm filtré sur test2006.lower-rest (fr)	107	78	90
Pbm filtré sur test2006.lower-ood (fr)	59	49	54
Pbm filtré sur test2006.lower-rest (es)	108	82	95
Pbm filtré sur test2006.lower-ood (es)	57	49	54

Figure 5-12. Taille en Mo des PBMS obtenus par compression n-aire (colonne de droite)

5.3 Compression sans perte

C'est à partir de cette section la « compression sans perte » que nous abandonnons les méthodes de découpage et introduisons les méthodes heuristiques. Elle consiste à établir des critères de compression à partir d'observation sur différents PBMs.

5.3.1 Principe

En étudiant le comportement du décodeur, nous nous sommes rendu compte que certains paramètres peuvent être supprimés dans le PBM car ils ne seront jamais sélectionnés par le décodeur lors des traductions.

Lorsque nous listons pour une source donnée tous les paramètres correspondants, il arrive que certaines parties cibles commencent et finissent de la même manière, ces paramètres ont une propriété spécifique, on les appellera « paramètres concurrents ». Un exemple de « paramètres concurrents » est donné dans la figure ci-dessous.

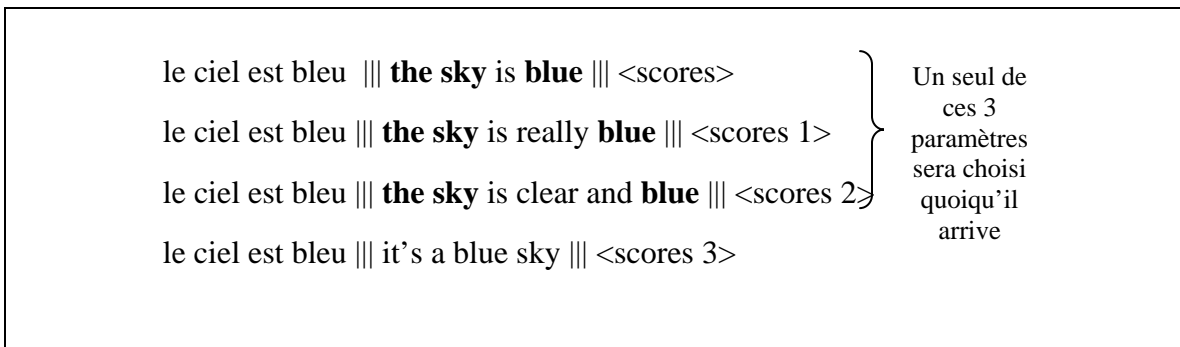


Figure 5-13. Exemple de paramètres concurrents

Par rapport à la Figure 5-13, si le décodeur doit traduire « le ciel est bleu », il choisira parmi les paramètres commençant par « the sky » et finissant par « blue » qu'un seul des trois, on peut alors retirer les 2 autres paramètres.

Supprimer ces paramètres inutiles devrait d'une part nous permettre de gagner en terme de compression mais d'autre part cela n'entraînera aucune perte de qualité de traduction, en théorie.

Il nous reste à savoir dans quelle proportion ce phénomène apparaît dans nos PBMs pour savoir si on doit vraiment s'y intéresser. Dans ce but, nous allons programmer un algorithme qui établira des statistiques, il donnera principalement le pourcentage de phrase sources pour lesquelles on trouve des parties cibles qui sont « concurrentes ». Cet algorithme est donné en Figure 5-14.

```

int occ=0;
Pour chaque paramètre (src,tgt) du PBM {
  Si src appartient à DÉJÀ_VU on passe au prochain paramètre;
  X = lister_paramètre(src);
  Pour tous les paramètres p=(X_src, X_tgt) de X {
    Pour tous les paramètres p=(X_src', X_tgt') de X {
      Si (X_tgt' commence et finit comme X_tgt) alors {
        On ajoute X_src à la liste DÉJÀ_VU;
        occ++;
      }
    }
  }
}
Retourne occ;

```

Figure 5-14. Algorithme pour la compression sans perte en pseudo-code

5.3.2 Résultats

Nous avons effectué nos tests sur les PBM filtrés sur test2006-ood et test2006-rest en allemand, français et espagnol. Les résultats sont décevants, seulement en moyenne 2% des différentes phrases sources d'un PBM peuvent être réduites à un seul paramètre.

Nous n'envisagerons pas de tests de compressions dans cette voie-là, car le gain est trop faible même si cette méthode est dite « sans perte ».

5.4 Compression heuristique : la ponctuation

5.4.1 Principe

Lors de l'apprentissage de nos PBMs, on retrouve beaucoup de symboles de ponctuation dans nos paramètres, en partant du principe que les symboles de ponctuations ne devraient être traduits que par des symboles de ponctuations.

On devrait n'en retrouver que dans des règles du type « <symbole de ponctuation> ||| <symbole de ponctuation> ||| <scores> », or ce n'est pas le cas (cf. Figure 5-15).

% , est % , is	0.2	0.189001	1	0.387482	2.718
% , les % , the	0.2	0.0808288	1	0.291821	2.718
% , % , and	0.047619	0.337495	0.003861	0.0201153	2.718
% , % , that	0.2	0.61601	0.003861	0.0158547	2.718
% , % and	0.0540541	0.0589794	0.023166	0.0304763	2.718
% , % annual	1	0.00345983	0.003861	8.86579e-06	2.718
% , % as	0.181818	0.0282689	0.00772201	0.005347	2.718
! je souhaite , i wish	0.00617284	1.68351e-05	1	0.00166323	2.718
! je , i	0.000191394	0.000139081	0.666667	0.0137175	2.718
! je but i	0.000307031	0.000107753	0.111111	0.000870976	2.718
! !	0.784547	0.757348	0.591233	0.577009	2.718
! ,	0.00018397	0.0002202	0.009467	0.0156755	2.718
! - and here	0.0526316	0.0006014	0.000129685	6.42018e-08	2.718
! - and	0.000510986	0.0006014	0.000129685	2.14557e-05	2.718
! je souhaite , i wish	0.00617284	1.68351e-05	1	0.00166323	2.718
! je , i	0.000191394	0.000139081	0.666667	0.0137175	2.718
! , and	0.000344656	0.00016865	0.00116716	6.82558e-05	2.718
" , les " , the	0.125	0.0577128	0.166667	0.0597613	2.718
" , les ' , the	0.129032	0.0234883	0.666667	0.144463	2.718
" , nous " , we	1	0.28765	0.5	0.0914984	2.718
" , nous ' is	0.008	2.16383e-05	0.125	0.00512641	2.718
" , nous ' we	0.0136986	0.0204718	0.125	0.335108	2.718

Figure 5-15. Exemple de modèle bruité (extrait d'un PBM), on voit que les symboles de ponctuation peuvent prendre une place importante dans les modèles de traduction.

De plus, cette méthode pourrait en théorie améliorer la qualité de nos traductions car les symboles de ponctuations parasitent nos paramètres. Par exemple, pour le paramètre « ! je souhaite ||| , i wish ||| <scores> », il risque d'être rarement utilisé lors d'une traduction car il est peu probable de traduire « ! je souhaite ». Par contre, traduire « je souhaite » l'est beaucoup plus.

Le principe est de prendre chaque paramètre du PBM et retirer la ponctuation qui entoure la phrase cible et source, comme dans l'exemple Figure 5-16. Principe du « nettoyage » de paramètre:

<p><i>./ le ciel est bleu ? * & the sky is blue ; <scores></i></p> <p>➔ <i>le ciel est bleu the sky is blue <scores></i></p>
--

Figure 5-16. Principe du « nettoyage » de paramètre

J'appellerai cette opération le « nettoyage »; elle peut s'appliquer à un paramètre entier (c'est-à-dire à la phrase source et cible) ou juste à une phrase source ou cible.

Puis nous vérifions si la version « nettoyée » du paramètre n'existe pas ailleurs dans le PBM. Si c'est le cas, nous retirons le paramètre courant, sinon nous le conservons, on évite ainsi la création des doublons dans le PBM.

5.4.2 Résultat

Les tableaux Figure 5-17 et Figure 5-18 nous permettent de comparer les scores obtenus grâce à notre heuristique avec ceux du découpage n-aire, des baselines et sans compression. On peut observer une perte beaucoup moins importante en BLEU par rapport aux baseline. Une chose étonnante est qu'on ne s'attendait pas à perdre en BLEU, car cette heuristique contrairement aux autres méthodes est moins risquée par rapport aux méthodes de découpage. Il serait intéressant de voir d'où vient cette perte.

Nos Pertes en BLEU varient de 0.02% à 0.60 %. Cette perte est assez négligeable et les gains en compression sont assez importants, de l'ordre de 15% à 20 % sur des PBMs filtrés. Ce qui prouve que la ponctuation prend une place relativement importante dans nos modèles.

Pbm	sans compression	avec compression (découpage n-aire)	baseline (avec des PBMs de taille équivalente)	avec compression (heuristique sur la ponctuation)
PBM filtré sur test2006.fr.lower-rest	30.35	28.72	29.50	29.82
PBM filtré sur test2006.fr.lower-ood	20.31	19.48	19.24	20.28
PBM filtré sur test2006.es.lower-rest	30.80	29.47	29.39	30.20
PBM filtré sur test2006.es.lower-ood	25.17	24.56	24.70	24.92

Figure 5-17. Scores BLEU obtenus après compression heuristique sur la ponctuation (sur la droite)

Pbm	sans compression	Baseline (avec des scores équivalents)	avec compression (heuristique sur la ponctuation)
PBM filtré sur Test2006.fr.lower-rest	107	100	86
PBM filtré sur Test2006.fr.lower-ood	59	52	48
PBM filtré sur Test2006.es.lower-rest	108	98	91
PBM filtré sur Test2006.es.lower-ood	57	56	49

Figure 5-18. Taux de compression obtenu après compression heuristique sur la ponctuation (sur la droite)

En comparant avec les baselines en annexe nous avons de meilleurs résultats en termes de compression et de BLEU, ce qui n'était pas le cas avec les méthodes précédentes.

Pour aller plus loin, nous avons effectué les tests de compression sur des PBMs de base en français et espagnol (vers l'anglais). Les résultats sont dressés dans le tableau ci-dessous.

PBM	sans compression	avec compression
PBM français	1.4 Go	1.1 Go
PBM espagnol	1.7 Go	1.5 Go

Figure 5-19. compression heuristique sur des PBM de plus de 1Go

Les taux de compressions sont approximativement de 11% pour la langue espagnole et de 20% pour le français.

En conclusion, les méthodes de découpages se révélant pour l'instant assez difficiles à mettre en place, cette méthode ouvre la voie aux méthodes de compression heuristique. Les résultats obtenus sont bons et assez encourageants pour continuer à émettre de nouvelles heuristiques.

5.5 Compression heuristique : pertinence des paramètres

5.5.1 Principe

On a vu lors de l'introduction au PBM que tous les paramètres n'étaient pas pertinents (cf. Figure 5-20). On cherche un critère qui pourrait départager les paramètres qui sont potentiellement utilisables lors de la traduction de ceux qui ne le sont pas. Pour déterminer ce critère, on va se baser sur les scores.

Le principe est de faire la somme des quatre premiers scores (le dernier étant toujours égal à 2.718) et de déterminer un seuil, si le paramètre dépasse ce seuil alors on le garde dans notre PBM sinon il est rejeté. On appellera cette compression « *compression par seuil* ».

En observant plusieurs PBM, on a décidé de fixer le seuil à 0,1.

avoir very	5.93331e-05	0.0001622	0.000181028	0.000466	2.718
avoir want to	0.000225327	0.0034105	0.000181028	0.000428439	2.718
avoir want	0.000940144	0.0034105	0.000543085	0.0039146	2.718
avoir was	0.000429943	0.0003664	0.00162925	0.0017709	2.718
avoir we all	0.00444444	0.0003767	0.000181028	5.10276e-05	2.718
avoir we did	0.00265252	0.0006233	0.000181028	6.42888e-06	2.718

Figure 5-20. Extrait de paramètres « non pertinent » dont la phrase source est « avoir », ces traductions seront probablement pas utilisées pour la traduction.

5.5.2 Résultats

Les figures Figure 5-21 et Figure 5-22 présentent les résultats de nos tests de compressions sur les PBM filtré sur les textes test2006-rest et test2006-ood, ceci en français et en anglais.

Pbm filtré sur	Sans compression	compression par ponctuation	compression par seuil
test2006.lower-rest (français)	30.35	29.82	29.12
test2006.lower-ood (français)	20.31	20.28	20.34
test2006.lower-rest (espagnol)	30.80	30.20	30.16
test2006.lower-ood(espagnol)	25.17	24.92	25.23

Figure 5-21. Score BLEU obtenus après compression par seuil (colonne de droite), on note un gain de bleu pour les corpus « ood » et une légère perte sur les corpus « rest » par rapport aux performances standards

pbm filtré sur	sans compression	compression par ponctuation	compression par seuil
test2006.lower-rest (français)	107	86	60
test2006.lower-ood (français)	59	48	28
test2006.lower-rest (espagnol)	108	91	60
test2006.lower-ood (espagnol)	57	49	28

Figure 5-22. Tailles de PBM obtenus après compression par seuil (colonne de droite), celles des PBM originaux sont sur la colonne « sans compression »

Les résultats de cette expérience sont meilleurs que ceux de l'expérience précédente (cf. Compression heuristique : la ponctuation), les taux de compression ont atteint les 50% pour des scores BLEU proche de ceux observés avec la compression par ponctuation.

Le bilan de cette expérience est positif, avec cette heuristique on a prouvé qu'il était possible de déterminer des critères simple mais qui se révèle efficace pour la compression des PBM.

6 Conclusion

6.1 Bilan de l'étude

Nous avons vu tout au long de nos expériences à quel point la compression de PBM est subtile à mettre en place, chaque modification sur les PBMs détériore plus ou moins la qualité de nos traductions.

En général, nous avons observé des pertes importantes avec les méthodes par découpage, mais il est bon de rappeler qu'à ce niveau nous n'avons pas tous les éléments en main pour retrouver les découpages lors des traductions. Ce qui a conduit à des résultats moins bons.

Les méthodes de compression heuristique ont donné de meilleurs résultats pour l'instant. Mais généralement les critères d'élagages sont difficiles à déterminer, et la proportion des paramètres qui peuvent être élagués varient fortement d'un modèle à un autre. Ce qui rend les résultats assez imprévisibles.

Par rapport au baseline fixé, la dernière approche a donné des résultats positifs. On a même réussi à compresser sans pertes de BLEU avec des taux très intéressants.

Cela ne remet pas en cause la possibilité de compresser les PBMs d'avantages. Il reste encore diverses heuristiques et approches à essayer (et des détails à régler) dans les méthodes que nous avons utilisées.

6.2 Conclusion personnelle

Ce stage a été très enrichissant. D'une part, j'ai eu la chance de découvrir de plus près l'environnement d'un laboratoire de recherche et de voir sous un autre angle l'enseignement informatique que j'ai reçu durant mes années à l'université. J'ai pu voir à quel point un bon équilibre entre l'enseignement théorique et pratique est nécessaire.

J'ai apprécié la découverte du domaine de la traduction automatique statistique, comprendre comment fonctionnent ces systèmes a enrichi ma culture générale et informatique. Les compétences des membres de mon équipe m'ont apporté beaucoup et ont approfondi mes connaissances en informatique. De plus, j'ai appris à programmer avec des contraintes de mémoire et de temps de calcul, ce qui m'a permis d'étudier mon code et de le débayer à un niveau auquel j'avais rarement fait face auparavant.

Enfin cette période de stage m'a donné l'occasion d'être plus au contact de la culture québécoise, et de profiter de Montréal dans un contexte différent de ma période d'étude.

Cette expérience aura été grandement enrichissante aussi bien au niveau personnel que pour ma future carrière d'ingénieur.

Annexe

Programmation d'outils d'analyse statistique des PBM et des traductions

Pour les analyses statistiques des PBMs et des traductions, on a créé respectivement deux programmes effectuant cette tâche : « *pbm_stat* » et « *traduction_stat* ».

- *pbm_stat* est un programme qui prend en entrée un PBM. Sa tâche est d'évaluer la répartition des paramètres selon la longueur des phrases sources et cibles. Autrement dit, pour chaque paramètre $f ||| e ||| p(f/e)$ du modèle, on détermine les longueurs l (en mots) de f et e , on obtient alors une association $(l(f), l(e))$ pour chaque paramètre. Au final, le programme donne la fréquence de chaque association de manière relative et absolue par rapport au PBM.

je veux manger une pomme ||| I want to eat an apple ||| 0.2 0.1 0.001 0.9 2.81

c'est une association [5-6], $l(\text{«je veux manger une pomme »}) = 5$ et $l(\text{« I want to eat an apple »}) = 6$.

Figure 6-1. Exemple d'association dans un PBM

- *traduction_stat* est un programme dont la sortie est similaire à *pbm_stat* mais qui n'opère pas sur le même type d'entrée. Le but est d'analyser la sortie d'une traduction sur laquelle on active l'option $-t$ (trace) de *pharaoh*. Cette option nous permet d'avoir plus d'information sur la manière dont la traduction a été faite, la plus importante d'entre elles étant la position du segment de mots source qui est représenté par un couple d'entier (qu'on appellera (x,y)). À partir de là, on peut calculer la longueur du segment de mot source qui a été traduit par le segment cible.

Phrase à traduire: *Das ist ein kleines haus*

Sortie de pharaoh sans l'option trace: *this is a small house*

Sortie de pharaoh avec l'option trace:

$\underbrace{\text{this is}}_2 | 0.014086 | \overbrace{0|1}^{(x,y)} | \underbrace{a}_1 | 0.188447 | \underbrace{2|2}_1 | \underbrace{\text{small}}_1 | 0.000706353 | \underbrace{3|3}_1 | \underbrace{\text{house}}_1 | 1.46468e-07 | \underbrace{4|4}_1 |$

Interprétation :

« *this is* » a été généré à partir des mots Allemand 0-1 « *das it* » avec un coût de 0.014086

a été généré à partir des mots 2-2 « *ein* » avec un coût de 0.18847

« *small* » a été généré à partir des mots 3-3 « *kleines* » avec un coût de 0.000706353

« *house* » a été généré à partir des mots 4-4 « *haus* » avec un coût de 1.46468e-07

Figure 6-2. Calcul des associations dans une traduction

Par rapport à l'exemple de la Figure 6-2 on obtient des associations [2-2], [1-1], [1-1], [1-1].

Baselines

[1-1]	16.61	137987	9.3M
[1-2]	17.46	256003	18M
[1-3]	17.49	293998	21M
[1-4]	17.48	303049	21M
[1-5]	17.47	304946	22M
[1-6]	17.47	305363	22M
[1-7]	17.47	305458	22M
[2-1]	19.87	183372	13M
[2-2]	22.28	464718	33M
[2-3]	22.43	586971	42M
[2-4]	22.36	618206	44M
[2-5]	22.34	625029	45M
[2-6]	22.33	626453	45M
[2-7]	22.33	626764	45M
[3-1]	20.64	190815	13M
[3-2]	24.13	503476	36M
[3-3]	24.70	676066	49M
[3-4]	24.59	727468	53M
[3-5]	24.53	739377	54M
[3-6]	24.49	741964	54M
[3-7]	24.48	742537	54M
[4-1]	20.65	191773	14M
[4-2]	24.34	508251	36M
[4-3]	25.36	688659	50M
[4-4]	25.18	749493	55M
[4-5]	25.08	764662	56M
[4-6]	25.04	768042	56M
[4-7]	25.04	768835	56M
[5-1]	20.71	191875	14M
[5-2]	24.45	508773	36M
[5-3]	25.41	690077	50M
[5-4]	25.33	752372	55M
[5-5]	25.19	768840	56M
[5-6]	25.12	772662	57M
[5-7]	25.12	773556	57M
[6-1]	20.71	191891	14M
[6-2]	24.45	508838	36M
[6-3]	25.47	690297	50M
[6-4]	25.37	752822	55M
[6-5]	25.29	769537	56M
[6-6]	25.17	773570	57M
[6-7]	25.16	774527	57M
[7-1]	20.71	191892	14M
[7-2]	24.45	508845	36M
[7-3]	25.49	690321	50M
[7-4]	25.36	752896	55M
[7-5]	25.31	769675	56M
[7-6]	25.17	773778	57M
[7-7]	25.17	774781	57M

Figure 6-3. Baseline constitué à partir du PBM filtré sur test2006-ood (espagnol)

[1-1]	19.92	185440	13M
[1-2]	21.00	338825	24M
[1-3]	21.12	389761	27M
[1-4]	21.13	402891	28M
[1-5]	21.14	405918	29M
[1-6]	21.13	406647	29M
[1-7]	21.13	406860	29M
[2-1]	21.35	261217	18M
[2-2]	26.45	693614	50M
[2-3]	26.99	893338	65M
[2-4]	27.03	950206	69M
[2-5]	27.00	964748	70M
[2-6]	27.00	968316	71M
[2-7]	27.00	969279	71M
[3-1]	21.13	278179	20M
[3-2]	27.34	782759	56M
[3-3]	29.33	1111845	82M
[3-4]	29.39	1224910	91M
[3-5]	29.42	1255814	93M
[3-6]	29.40	1263873	94M
[3-7]	29.40	1265983	94M
[4-1]	21.03	281118	20M
[4-2]	27.39	798729	58M
[4-3]	29.77	1156131	85M
[4-4]	30.20	1308016	98M
[4-5]	30.17	1354769	102M
[4-6]	30.18	1367553	103M
[4-7]	30.17	1371151	103M
[5-1]	20.87	281572	20M
[5-2]	27.39	801391	58M
[5-3]	29.78	1163938	86M
[5-4]	30.49	1324545	99M
[5-5]	30.48	1381363	104M
[5-6]	30.52	1398046	106M
[5-7]	30.52	1402834	106M
[6-1]	20.83	281643	20M
[6-2]	27.34	801809	58M
[6-3]	29.79	1165336	86M
[6-4]	30.58	1327733	100M
[6-5]	30.60	1387169	105M
[6-6]	30.65	1406591	107M
[6-7]	30.67	1412422	107M
[7-1]	20.81	281652	20M
[7-2]	27.34	801864	58M
[7-3]	29.79	1165569	86M
[7-4]	30.61	1328290	100M
[7-5]	30.72	1388273	105M
[7-6]	30.76	1408570	107M
[7-7]	30.80	1415260	108M

Figure 6-4. Baseline constitué à partir du PBM filtré sur test2006-rest (espagnol)

[1-1]	12.55	145008	9.7M
[1-2]	12.90	258485	18M
[1-3]	12.94	292958	21M
[1-4]	12.95	300689	21M
[1-5]	12.95	302296	21M
[1-6]	12.95	302621	21M
[1-7]	12.95	302699	21M
[2-1]	15.57	187663	13M
[2-2]	17.26	466835	33M
[2-3]	17.40	576856	41M
[2-4]	17.43	602997	43M
[2-5]	17.41	608511	44M
[2-6]	17.40	609758	44M
[2-7]	17.40	610046	44M
[3-1]	16.57	195747	14M
[3-2]	18.87	518597	37M
[3-3]	19.24	678975	49M
[3-4]	19.29	723122	52M
[3-5]	19.32	733010	53M
[3-6]	19.31	735253	53M
[3-7]	19.31	735827	54M
[4-1]	16.82	197407	14M
[4-2]	19.18	529829	38M
[4-3]	19.80	707296	51M
[4-4]	19.79	762437	56M
[4-5]	19.80	775652	57M
[4-6]	19.81	778657	57M
[4-7]	19.81	779413	57M
[5-1]	16.71	197674	14M
[5-2]	19.33	531786	38M
[5-3]	20.16	713236	52M
[5-4]	20.11	772661	56M
[5-5]	20.05	788013	58M
[5-6]	20.09	791591	58M
[5-7]	20.08	792494	58M
[6-1]	16.72	197726	14M
[6-2]	19.41	532091	38M
[6-3]	20.26	714434	52M
[6-4]	20.25	775084	57M
[6-5]	20.21	791363	58M
[6-6]	20.25	795305	58M
[6-7]	20.26	796308	59M
[7-1]	16.71	197740	14M
[7-2]	19.42	532161	38M
[7-3]	20.27	714711	52M
[7-4]	20.30	775743	57M
[7-5]	20.23	792341	58M
[7-6]	20.31	796465	59M
[7-7]	20.31	797536	59M

Figure 6-5. Baseline constitué à partir du PBM filtré sur test2006-ood (français)

[1-1]	19.12	186229	13M
[1-2]	19.93	326658	23M
[1-3]	19.98	369830	26M
[1-4]	19.98	379996	27M
[1-5]	19.98	382196	27M
[1-6]	19.98	382693	27M
[1-7]	19.98	382835	27M
[2-1]	22.55	253986	18M
[2-2]	25.70	666900	48M
[2-3]	25.93	840032	60M
[2-4]	25.94	884330	64M
[2-5]	25.95	894765	65M
[2-6]	25.96	897137	65M
[2-7]	25.96	897790	65M
[3-1]	22.56	269030	19M
[3-2]	27.31	768365	55M
[3-3]	28.55	1065922	78M
[3-4]	28.53	1160513	86M
[3-5]	28.52	1184976	88M
[3-6]	28.52	1191094	88M
[3-7]	28.52	1192723	88M
[4-1]	22.38	272593	19M
[4-2]	27.36	793586	57M
[4-3]	29.40	1137853	84M
[4-4]	29.55	1273423	95M
[4-5]	29.61	1313055	98M
[4-6]	29.60	1323580	99M
[4-7]	29.60	1326426	100M
[5-1]	22.38	273415	19M
[5-2]	27.42	799489	58M
[5-3]	29.55	1156870	85M
[5-4]	30.00	1309981	98M
[5-5]	30.02	1361537	103M
[5-6]	30.01	1376274	104M
[5-7]	30.04	1380515	104M
[6-1]	22.36	273597	19M
[6-2]	27.30	800900	58M
[6-3]	29.50	1161815	86M
[6-4]	30.07	1320865	99M
[6-5]	30.19	1378650	104M
[6-6]	30.23	1396902	106M
[6-7]	30.22	1402493	107M
[7-1]	22.34	273642	19M
[7-2]	27.28	801296	58M
[7-3]	29.47	1163255	86M
[7-4]	30.08	1324175	99M
[7-5]	30.29	1384247	105M
[7-6]	30.36	1404444	107M
[7-7]	30.35	1411193	107M

Figure 6-6. Baseline constitué à partir du PBM filtré sur test2006-rest (français)

Optimisation de l'algorithme du découpage binaire

Pour mieux cerner notre problème on va passer par une forme algorithmique de notre heuristique. Nous prendrons comme convention « (src, tgt) » pour nommer nos paramètres (« src ||| tgt ||| <scores> »), ici nous ne prenons pas en compte les scores car ils n'interviennent pas dans l'algorithme.

```
Tant qu'il y a des paramètres (s,t) ∈ PBM {
  Pour chaque découpage binaire de s {
    S1 = lister_les_parametres(s1);
    S2 = lister_les_parametres(s2);
    Pour tout paramètres (s1,t1) ∈ S1{
      Pour tout paramètres (s2, t2) ∈ S2{
        Si (t1+t2 = t)
          ne pas afficher( (s,t) )
        Sinon afficher(s,t)
```

Note : +, est le symbole de concaténation entre deux chaînes de caractères

Figure 6-7. Algorithme de la compression binaire en pseudo-code

L'énorme temps de calcul observé lors des premiers tests de l'algorithme est dû aux itérations sur les S1 et S2, car ces deux listes sont souvent très grandes. Pour optimiser l'algorithme on va se concentrer sur les comparaisons entre les éléments de S1 et S2, le but est de voir si parmi ces comparaisons, certaines d'entre elles sont superflues.

En étudiant l'algorithme de plus près, nous pouvons faire 3 optimisations :

- Si t ne commence pas par t1, on peut abandonner toutes les comparaisons avec les t2.
- Si $taille(t1) + taille(t2) \neq taille(t)$ alors on se fait pas $t1+t2=t$
- On peut retirer le test $(t1+t2=t)$ par le test « t se termine-t-il par t2? » car nous avons déjà vérifié si t commence par t1.

Par la suite, nous avons relancé nos tests sur les PBMs filtrés précédents et les résultats sont significatifs :

	Test2006-ood (espagnol)	Test2006-rest (espagnol)	Test2006-ood (français)	Test2006-rest (français)
Version non optimisée	15h02	21h13	14h50	17h27
Version Optimisée	20 min 11 s	38 min 25 s	21 min 08 s	38 min 03 s

Figure 6-8. Comparaison entre les temps de calcul des deux versions de l'algorithme

Alors qu'il fallait en moyenne 17h pour compresser nos PBMs, ce temps moyen est passé à 30 min. De ce fait, la version optimisée de l'algorithme nous a permis de nous intéresser aux résultats obtenus sur un PBM de plus grande taille, nous avons donc pris un PBM de base d'une taille 1.4 Go. Après compression nous obtenons un PBM de 124 Mo.

La taille du PBM a été réduite par 10 alors que ce n'était pas le cas pour les PBM filtrés. Le nombre de découpage n'est donc pas proportionnel à la taille des PBMs.

Ces résultats sont tout de même intéressants car il nous donnent les taux de compression que l'on pourrait avoir si on était capable de retrouver nos paramètres « découpés ».

Découpage n-aire : explications

L'algorithme reprend le principe du « découpage » (cf. Compression par découpages binaires) sauf qu'on le généralise, il ne s'agit plus de faire des découpages binaires mais n-aires. De plus, pour éviter d'itérer le découpage sur tous les paramètres du PBM, nous allons appliquer l'algorithme seulement aux paramètres dont la phrase source compte au moins 4 mots.

Le principe est le suivant pour chaque paramètre « <src> ||| <tgt> ||| <scores> », si le nombre de mots dans la phrase source est supérieur à 4, on crée une matrice carrée de dimension N (N étant le nombre de mots dans la phrase source) où chaque cellule (x,y) représente la phrase source allant du mot x à y.

La Figure 6-10. Phase d'initialisation de l'algorithme appliquée sur l'exemple précédent, les « * » représentent les cases initialisées et les X les cases inexistantes. situe les phrases «*Mr le président , je veux* », « , je », « *Mr le président* » par rapport à la matrice du paramètre « *Mr le président , je veux* ».

Nous allons expliquer l'algorithme en nous aidant des exemples suivants :

- | | | | | | |
|----|---------------------------|--|---------------------|--|-----------|
| 1. | Mr le président , je veux | | Mr speaker , I want | | 1 1 1 1 1 |
| 2. | veux | | want | | 1 1 1 1 1 |
| 3. | , je veux | | , I want | | 1 1 1 1 1 |
| 4. | , je | | , I | | 1 1 1 1 1 |
| 5. | Mr | | Mr | | 1 1 1 1 1 |
| 6. | Mr le | | Mr | | 1 1 1 1 1 |
| 7. | le président | | speaker | | 1 1 1 1 1 |
| 8. | , je | | , I | | 1 1 1 1 1 |
| 9. | Mr le président | | Mr speaker | | 1 1 1 1 1 |

Figure 6-9. un exemple de PBM

On passe sur chacun des paramètres, mais seul le premier « *Mr le président , je veux* ||| *Mr speaker , I want* ||| <scores> » possède une phrase source qui dépasse les 4 mots, on crée donc notre matrice 6x6 en conséquence.

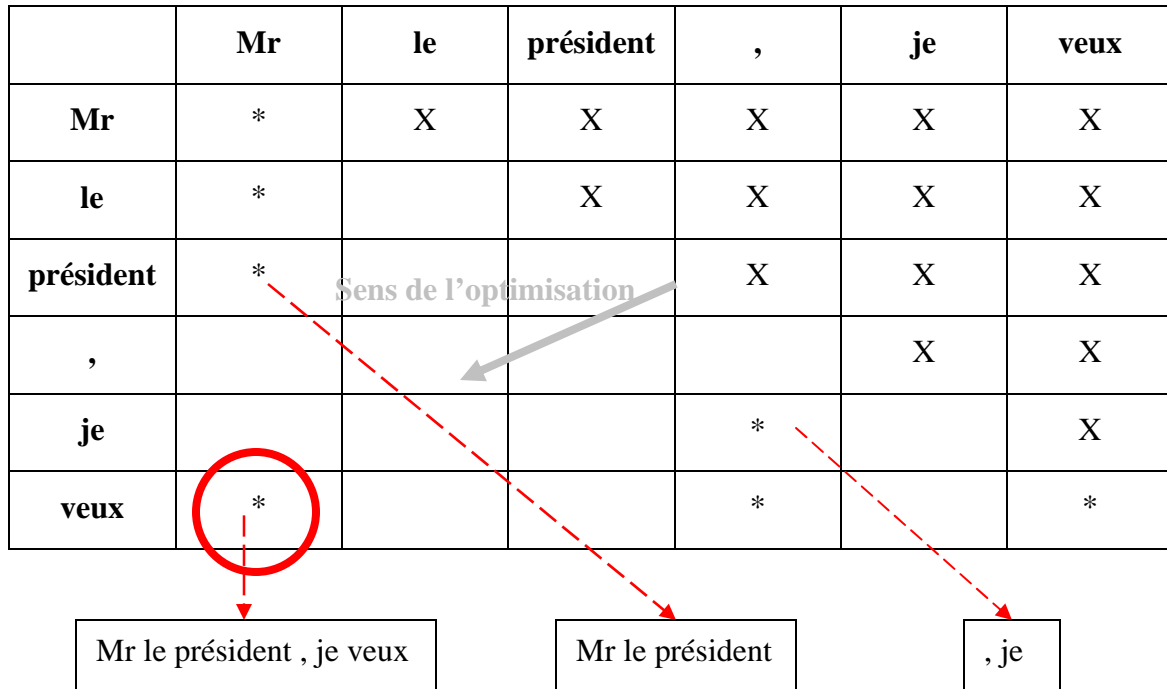


Figure 6-10. Phase d'initialisation de l'algorithme appliquée sur l'exemple précédent, les « * » représentent les cases initialisées et les X les cases inexistantes.

La 1^{ère} phase est l'initialisation (cf. Figure 6-10). On vérifie s'il existe pour chaque cellule (x,y) du tableau, des paramètres dont la phrase source est (x,y). Si c'est le cas, nous ajoutons ces paramètres en tant qu'hypothèses dans cette cellule. Une hypothèse est constituée :

- d'un score.
- d'une couverture de la phrase cible (p. ex. : pour la phrase cible « Mr speaker , I want » le paramètre « je veux ||| I want ||| <scores> » couvre les deux derniers mots de la phrase cible).
- d'un « backpointer » qui nous permet de retrouver quel est le découpage d'une cellule du tableau.

La cellule (1,6) marquée d'un cercle rouge est spécifique, c'est elle qui stockera le résultat final et avec laquelle on prendra la décision de découper ou pas.

Dans notre exemple ci-dessus, les cellules initialisées sont représentées par des « * » et, au final, chaque cellule contient une liste vide ou non vide d'hypothèses.

	Mr	le	président	,	je	veux
Mr	*	X	X	X	X	X
le	*		X	X	X	X
président	*	*		X	X	X
,					X	X
je				*		X
veux	*			*		*

La 2^e phase consiste à lancer l'algorithme de maximisation sur cette matrice. On commence par les cellules de la diagonale inférieure stricte pour finir à la dernière case en bas à gauche. Pour chaque cellule $cell(x,y)$ de la partie triangulaire inférieure du tableau, on vérifie s'il existe deux cellules $cell(x',y)$ et $cell(x,y')$ qui sont un découpage de la partie source :

- On met alors on en concurrence chaque hypothèse h de $cell(x,y)$ avec toutes les hypothèses $h1$ de $cell(x',y)$ et $h2$ de $cell(x,y')$.
- On vérifie si les phrases cibles de $h1$ et $h2$ couvrent bien celles de h et si le cumul des scores $h1$ et $h2$ dépasse celui de h . Si c'est le cas, on met à jour les hypothèses h avec le nouveau score et le « backpointer » pour retrouver notre découpage.

Dans notre exemple précédent, les flèches de même couleur qui pointent sur une même case sont des hypothèses $h1$ et $h2$ qui sont en concurrence avec une ou plusieurs hypothèses h de la case pointée.

À la fin de l'algorithme, s'il existe un découpage dans la case $(1,N)$ (celle qui représente notre paramètre d'origine) alors on supprime ce paramètre.

Liste des figures

FIGURE 2-1. PRINCIPALES COMPOSANTES DE LA SMT. LE DÉCODEUR PREND EN ENTRÉE LE TEXTE SOURCE, UN MODÈLE DE TRADUCTION ET UN MODÈLE DE LANGUE POUR FOURNIR EN SORTIE LE TEXTE TRADUIT. NOTONS QUE LA LANGUE VERS LAQUELLE ON VEUT TRADUIRE SERA APPELÉ « LANGUE CIBLE ».....	10
FIGURE 2-2. EXEMPLES ILLUSTRANT LA NOTION D’HYPOTHÈSE DE TRADUCTIONS [KOE04]	11
FIGURE 2-3. EXEMPLE D’APPLICATION DU MODÈLE TRIGRAMME [LAN06] SUR LA PHRASE « 15 ANNÉES DE TRADUCTION EN 45 MINUTES »	12
FIGURE 2-4. PARAMÈTRES EXTRAIT D’UN PBM TRADUISANT LE MOT « AMBITIONS » DU FRANÇAIS VERS L’ANGLAIS.	14
FIGURE 2-5. EXTRAIT D’UN PBM	15
FIGURE 2-6. EXEMPLE DE REDONDANCE DANS UN PBM POUR LES PHRASES APPARENTÉES À « PASSER UN SAPIN ».....	16
FIGURE 2-7. DESCRIPTION DU FILTRAGE SUR LE TEXTE « LE CHAT EST NOIR »	17
FIGURE 3-1. PROTOCOLE D’UN TEST DE COMPRESSION	22
FIGURE 4-1. RÉPARTITION DES PARAMÈTRES DANS LE PBM FRANÇAIS EN FONCTION DE LA LONGUEUR DE LA PHRASE SOURCE ET CIBLE	25
FIGURE 4-2. RÉPARTITION DES PARAMÈTRES DANS LE PBM FRANÇAIS FILTRÉ SUR LE CORPUS « TEST-REST » EN FONCTION DE LA LONGUEUR DE LA PHRASE SOURCE ET CIBLE.....	26
FIGURE 4-3. PARAMÈTRES UTILISÉS LORS DE LA TRADUCTION DU TEXTE « TEST2006-REST.FR » EN FONCTION DE LA LONGUEUR DE LA PHRASE SOURCE ET CIBLE	27
FIGURE 4-4. CONSTITUTION DES BASELINES, LES TRADUCTIONS DU TEXTE X SONT FAITES POUR DIFFÉRENTS PBM ÉLAGUÉS.....	29
FIGURE 4-5. SCORES BLEU OBTENUS SUR LES 49 TRADUCTIONS DU CORPUS « TEST2006-OD » FRANÇAIS, EN ABCISSE NOUS AVONS LE CRITÈRE D’ÉLAGAGE ET EN ORDONNÉE LES SCORES BLEU OBTENUS.....	30
FIGURE 4-6. SCORES BLEU OBTENUS SUR LES 49 TRADUCTIONS DU CORPUS « TEST2006-REST », EN ABCISSE NOUS AVONS LE CRITÈRE D’ÉLAGAGE ET EN ORDONNÉE LES SCORES BLEU OBTENUS.....	30
FIGURE 5-1. EXEMPLE DE DÉCOUPAGES BINAIRES POSSIBLES POUR LA PARTIE SOURCE « LE CIEL EST BLEU »	32
FIGURE 5-2. ALGORITHME PROPOSÉ POUR LA COMPRESSION PAR DÉCOUPAGE BINAIRE, LA CONCATÉNATION ENTRE DEUX CHÂÎNES EST SYMBOLISÉ PAR « + ».....	33
FIGURE 5-3. APPLICATION DE L’ALGORITHME DU DÉCOUPAGE BINAIRE SUR LE PARAMÈTRE « LE CHAT EST NOIR ».....	33
FIGURE 5-4. SCORES BLEU OBTENUS SUR LES TRADUCTIONS APRÈS COMPRESSION BINAIRE	34
FIGURE 5-5. TAILLE (EN MO) DES PBM OBTENUS APRÈS COMPRESSION BINAIRE.....	34
FIGURE 5-6. EXEMPLE DE PBM AUQUEL ON VA APPLIQUER LE DÉCOUPAGE N-AIRE AU PARAMÈTRE «MR LE PRÉSIDENT , JE VEUX».....	36
FIGURE 5-7. PHASE D’INITIALISATION DE L’ALGORITHME APPLIQUÉE SUR L’EXEMPLE LE PBM EN FIGURE 5-6, LES « * » REPRÉSENTENT LES CASES INITIALISÉES, LES CASES VIDES NE CONTIENNENT PAS D’HYPOTHÈSES ET LES X SONT DES CASES INEXISTANTES.	36
FIGURE 5-8. ALGORITHME GÉNÉRAL DU DÉCOUPAGE N-AIRE	38

FIGURE 5-9. ALGORITHME DE LA FONCTION MATCH	38
FIGURE 5-10. ALGORITHME DE LA FONCTION MAJ	39
FIGURE 5-11. EVALUATION DES SCORES BLEU OBTENUS PAR COMPRESSION N-AIRE (COLONNE DE DROITE), COMPARÉS AVEC LES PERFORMANCES STANDARDS, LES RÉSULTATS DU DÉCOUPAGE BINAIRE ET LE BASELINE.	39
FIGURE 5-12. TAILLE EN MO DES PBMS OBTENUS PAR COMPRESSION N-AIRE (COLONNE DE DROITE)	40
FIGURE 5-13. EXEMPLE DE PARAMÈTRES CONCURRENTS	41
FIGURE 5-14. ALGORITHME POUR LA COMPRESSION SANS PERTE EN PSEUDO-CODE	42
FIGURE 5-15. EXEMPLE DE MODÈLE BRUITÉ (EXTRAIT D’UN PBM), ON VOIT QUE LES SYMBOLES DE PONCTUATION PEUVENT PRENDRE UNE PLACE IMPORTANTE DANS LES MODÈLES DE TRADUCTION.	43
FIGURE 5-16. PRINCIPE DU « NETTOYAGE » DE PARAMÈTRE.....	44
FIGURE 5-17. SCORES BLEU OBTENUS APRÈS COMPRESSION HEURISTIQUE SUR LA PONCTUATION (SUR LA DROITE).....	44
FIGURE 5-18. TAUX DE COMPRESSION OBTENU APRÈS COMPRESSION HEURISTIQUE SUR LA PONCTUATION (SUR LA DROITE).....	45
FIGURE 5-19. COMPRESSION HEURISTIQUE SUR DES PBM DE PLUS DE 1GO.....	45
FIGURE 5-20. EXTRAIT DE PARAMÈTRES « NON PERTINENT » DONT LA PHRASE SOURCE EST « AVOIR », CES TRADUCTIONS SERONT PROBABLEMENT PAS UTILISÉES POUR LA TRADUCTION.	46
FIGURE 5-21. SCORE BLEU OBTENUS APRÈS COMPRESSION PAR SEUIL (COLONNE DE DROITE), ON NOTE UN GAIN DE BLEU POUR LES CORPUS « OOD » ET UNE LÉGÈRE PERTE SUR LES CORPUS « REST » PAR RAPPORT AUX PERFORMANCES STANDARDS	47
FIGURE 5-22. TAILLES DE PBM OBTENUS APRÈS COMPRESSION PAR SEUIL (COLONNE DE DROITE), CELLES DES PBM ORIGINAUX SONT SUR LA COLONNE « SANS COMPRESSION ».....	47
FIGURE 6-1. EXEMPLE D’ASSOCIATION DANS UN PBM	50
FIGURE 6-2. CALCUL DES ASSOCIATIONS DANS UNE TRADUCTION	51
FIGURE 6-3. BASELINE CONSTITUÉ À PARTIR DU PBM FILTRÉ SUR TEST2006-OOD (ESPAGNOL).....	52
FIGURE 6-4. BASELINE CONSTITUÉ À PARTIR DU PBM FILTRÉ SUR TEST2006-REST (ESPAGNOL).....	53
FIGURE 6-5. BASELINE CONSTITUÉ À PARTIR DU PBM FILTRÉ SUR TEST2006-OOD (FRANÇAIS).....	54
FIGURE 6-6. BASELINE CONSTITUÉ À PARTIR DU PBM FILTRÉ SUR TEST2006-REST (FRANÇAIS).....	55
FIGURE 6-7. ALGORITHME DE LA COMPRESSION BINAIRE EN PSEUDO-CODE.....	56
FIGURE 6-8. COMPARAISON ENTRE LES TEMPS DE CALCUL DES DEUX VERSIONS DE L’ALGORITHME	57
FIGURE 6-9. UN EXEMPLE DE PBM.....	58
FIGURE 6-10. PHASE D’INITIALISATION DE L’ALGORITHME APPLIQUÉE SUR L’EXEMPLE PRÉCÉDENT, LES « * » REPRÉSENTENT LES CASES INITIALISÉES ET LES X LES CASES INEXISTANTES.	59

Références

[Lan05] Philippe Langlais, Simona Gandrabur, Thomas Leplus and Guy Lapalme (2005) "The Long-Term Forecast for Weather Bulletin Translation" in Journal of Machine Translation, Vol. 19, Springer, pp. 83-112, 2005.

[Sha49] Shannon, The Mathematical Theory of Information. Urbana, IL:University of Illinois Press, 1949

[Koe04] a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models – User Manual and Description

[Lan06] Philippe Langlais, « 15 années de traduction en 45 minutes », séminaire RALI-OLST, Montréal Janvier, 2006.

[Pap02] 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.

[Koe04b] Phillip Koehn, pharaoh training manual , p.11

[FeBe06] Marcello Federico and Nicola Bertoldi, "How Many Bits Are Needed To Store Probabilities for Phrase-Based Translation?" ITC-irst - Centro per la Ricerca Scientifica e Tecnologica

[Koe04c] 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