Université Paris-Sud 11 — Faculté des sciences d'Orsay
Ecole Doctorale d'Informatique de Paris-Sud
Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur

Thèse

pour le diplôme de Docteur en Sciences, spécialité Informatique
présentée et défendue publiquement
le lundi 17 décembre 2007 à Orsay (91)
par

Daniel Déchelotte

Traduction automatique de la parole par méthodes statistiques
Automatic Speech Translation by statistical methods

Membres du jury : Laurent Besacier (rapporteur)
Roland Kuhn (rapporteur)
Holger Schwenk (directeur)
Jean-Luc Gauvain (co-directeur)
Philipp Koehn (examinateur)
Joseph Mariani (examinateur)

Pour leur écoute,
Pour leur patience,
Pour leurs conseils,
Pour leur gentilesse,

Merci

à ma fiancée,
à ma famille, mes parents et mes sœurs,
à mes directeur et co-directeur de thèse,
aux collègues du groupe TLP et du LIMSI,
aux amis,
à mon kinésithérapeute,
et à toi lecteur !

Cette thèse a été partiellement financé par l'Union Européenne sous le projet TC-Star (IST-2002-FP6-506738), et par le Gouvernement français sous le projet Instar (ANR JCJC06_143038).

Table des matières

Chapitre 1

Introduction

Échanger, collaborer, construire ensemble, tout serait plus simple si l'on pouvait parler toutes les langues. Ou, à défaut, disposer d'un outil le permettant. Même si le rêve de converser librement avec tous les habitants de la planète n'est pas encore accessible, de nombreuses applications de la traduction de la parole sont envisageables. On songe notamment à une utilisation touristique : un traducteur informatique serait une aide précieuse pour s'informer et communiquer dans un pays dont la langue est inconnue. Parmi les applications professionnelles, des systèmes de dialogue pourraient faciliter l'échange et la négociation avec des partenaires internationaux. D'autres applications incluent la traduction de journaux télévisés et l'indexation cross-lingue de contenus multimédias. Ceci permettrait aux entreprises, par exemple, de savoir dans quels contextes elles ont été citées, et aux citoyens d'avoir un accès impartial à des journaux étrangers. Citons également le nombre croissant de langues officielles au sein de l'Union Européenne, qui multiplie les besoins en traducteurs et en interprètes. Par ailleurs, les agences gouvernementales veulent déployer cette technologie auprès des troupes en mission à l'étranger, afin d'établir un contact avec les populations autochtones et recueillir des informations.

Depuis plus d'un demi siècle, des chercheurs se passionnent pour la reconnaissance de la parole et la traduction automatique. La reconnaissance de la parole exige une modélisation acoustique des variations liées au locuteur et aux conditions d'enregistrement, mais aussi une modélisation linguistique pour former des phrases cohérentes à partir des sons reconnus. Pour y parvenir, tous les systèmes de reconnaissance de la parole à grand vocabulaire utilisent aujourd'hui des modèles statistiques, dont les paramètres sont appris sur des corpus de parole et de textes. La traduction doit quant à elle capturer les régularités et les irrégularités de *deux* langues, et décrire le passage de l'une à l'autre. De nombreuses approches de la traduction automatique ont été proposées — les plus importantes sont décrites au prochain chapitre. Les approches statistiques, qui peuvent sembler intuitivement inadaptées au problème, sont aujourd'hui incontournables dans la recherche sur la traduction automatique.

La traduction de la parole cumule les difficultés liées à la reconnaissance et à la traduction. En effet, tout système de traduction de la parole sera confronté à un style de langue différent de l'écrit, à des mots répétés, des phrases grammaticalement fausses, et

bien sûr à d'inévitables erreurs de reconnaissance. Des techniques robustes doivent être développées pour intégrer au mieux les différentes technologies nécessaires et répondre aux besoins de l'utilisateur.

Dans cette optique, cette thèse porte sur la traduction automatique par méthodes statistiques, et se concentre plus particulièrement sur la traduction de la parole reconnue automatiquement. Le travail effectué au cours du doctorat s'articule autour de deux axes :

1. le développement et l'amélioration de systèmes de traduction ;
2. l'étude des spécificités de la traduction de la parole, quelle soit transcrite manuellement ou automatiquement, avec l'accent porté sur ce dernier cas.

Plus précisément, cette thèse s'organise comme suit :

– Le chapitre 2 présente l'état de l'art des modèles et algorithmes développés depuis cinquante ans pour permettre aux ordinateurs de traduire automatiquement. Ce chapitre explique ensuite ce qui a motivé nos choix quant aux deux systèmes de traduction développés au cours de cette thèse. Il décrit enfin les tâches de traduction retenues et les procédures d'évaluation.
– Les chapitres 3 et 4 décrivent les deux systèmes de traduction développés au cours de cette thèse. Le premier a été entièrement créé par l'auteur, tandis que le second repose sur un décodeur *open source*. Les deux systèmes sont évalués sur les mêmes données et des expériences sont menées pour combiner les deux.
– Le chapitre 5 propose un algorithme réalisant un apprentissage discriminant de la *table de traduction*, qui est le modèle central des beaucoup de systèmes de traduction et en particulier de celui décrit au chapitre 4.
– Le chapitre 6 est le premier chapitre de la seconde partie, qui a trait aux spécificités de la traduction de la *parole*. Ce chapitre énumère les principales difficultés rencontrées par les systèmes de traduction de la parole et dresse l'état de l'art des solutions décrites dans la littérature.
– Les chapitres 7 et 8 présentent les résultats de notre recherche à ce sujet. Le chapitre 7 aborde le problème de la traduction d'un flux de mots produit par un système de reconnaissance que l'on ne peut contrôler. Ce cas de figure est fréquent, par exemple lorsque deux laboratoires travaillent, l'un sur la reconnaissance automatique et l'autre sur la traduction. Le chapitre 8 considère au contraire que le système de reconnaissance peut être modifié pour le bénéfice de la traduction et décrit plusieurs expériences décrivant l'intégration de la reconnaissance automatique de la parole et de la traduction automatique.
– Enfin, le chapitre 9 conclut ce document en rassemblant les faits marquants appris au cours de cette thèse et propose quelques directions de recherche prometteuses.

Cette thèse a été partiellement financée par le projet européen TC-Star (*Technology and Corpora for Speech to Speech Translation*, technologie et corpus pour la traduction parole à parole), et par le projet ANR Instar (Algorithmes d'apprentissage innovants pour la traduction automatique par approches statistiques).

Première partie

Modèles et algorithmes pour la traduction automatique

Chapitre 2

Introduction à la traduction automatique

2.1 État de l'art de algorithmes de traduction automatique

2.1.1 Approches à la traduction automatique

Presque immédiatement après l'apparition des « calculateurs électroniques » qui allaient devenir les futurs ordinateurs, des chercheurs ont tenté d'exploiter leur puissance de calcul croissante pour la traduction automatique. Les différents systèmes conçus jusqu'à ce jour ont pour but d'assister le traducteur humain dans son travail, ou plus ambitieusement de produire de façon automatique une traduction utilisable. Cette section présente brièvement plusieurs approches qui ont jalonné la recherche sur la traduction automatique, du milieu du siècle dernier à nos jours. Les sections suivantes décrivent plus en détail l'approche *statistique*, qui est celle mise en œuvre dans cette thèse.

Les premiers programmes d'ordinateurs relatifs à la traduction étaient destinés à servir d'aide à la traduction. Quelques règles et surtout un dictionnaire bilingue composaient le cœur du système, bien évidemment encore très rudimentaire. Le 7 janvier 1954, un système conçu par l'Université de Georgetown et IBM est présenté au public. Le système, qui dispose d'un vocabulaire de 250 mots et de 6 règles de grammaire, parvient à traduire une soixantaine de phrases soigneusement choisies du russe à l'anglais. Cette démonstration soulève d'immenses espoirs et marque le début d'importants efforts de recherche.

Les années suivantes voient les dictionnaires grossir et le nombre de règles régissant le réordonnement des mots augmenter. La nécessité d'automatiser l'acquisition des règles et d'améliorer leur généricité participe au développement de la linguistique informatique et par exemple des grammaires formelles.

Le triangle présenté à la figure 2.1 est attribué à Vauquois [1968]. Il présente de manière synthétique une analyse du processus de traduction encore pleinement pertinente et employée de nos jours. La traduction peut s'opérer à plusieurs niveaux. Au niveau

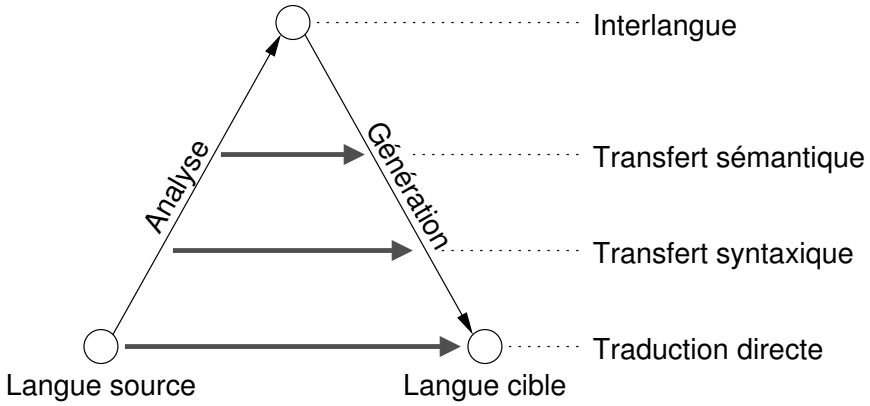


FIG. 2.1 – Le triangle dit « de Vauquois » pour la traduction

le plus bas, on retrouve la traduction directe, qui passe directement des mots de la langue source aux mots de la langue cible. Si l'on effectue une analyse syntaxique de la phrase source, le transfert à la langue cible devrait être simplifié. À ce niveau, les détails spécifiques à la constitution des groupes nominaux, par exemple, n'ont pas besoin d'être connus des règles régissant le transfert. Avec une analyse plus approfondie de la phrase source, au niveau sémantique, le transfert devient uniquement sémantique et, espère-t-on, plus simple. En revanche, la génération des mots après le transfert est plus complexe qu'au niveau inférieur. Enfin, une analyse totale de la phrase source peut aboutir à une représentation de son sens dans une « inter-langue » artificielle, de laquelle on dérive ensuite les mots cible.

L'approche reposant sur une inter-langue est séduisante car elle remplace le problème de la traduction par deux problèmes monolingues, d'analyse et de synthèse. L'avantage est que les modules monolingues sont *a priori* réutilisables. Pour couvrir tous les sens de traduction entre n langues, il suffit de n modules d'analyse et de n modules de synthèse, contre $n(n-1)$ systèmes de transfert [Hiroshi and Meiying, 1993]. La définition même de l'inter-langue est cependant un problème qui reste non-résolu, en particulier lorsque l'on souhaite obtenir un système de traduction général, pour un domaine large. L'inter-langue KANT [Mitamura et al., 1991] se limite par exemple à un vocabulaire et une grammaire contrôlés. L'approche inter-langue reste toutefois un domaine de recherche actif ([Besacier et al., 2001, Lee and Seneff, 2005], par exemple).

Revenons au milieu des années soixante. Quel que soit le niveau d'analyse effectuée par les systèmes de traduction, tous semblent se heurter à une « barrière sémantique » et les espoirs soulevés sont déçus. En 1966, un rapport de l'ALPAC (*Automatic Language Processing Advisory Committee*, comité consultatif sur le traitement automatique des langues) conclut que la traduction par ordinateur est lente, imprécise et chère et qu'elle ne deviendra utile ni dans un futur proche, ni dans un futur lointain.

Les financements tarissent et les systèmes n'évoluent que lentement. Les systèmes à base de règles (et de dictionnaires bilingues) s'enrichissent progressivement de règles, et sont spécialisés sur des tâches précises. À mesure que le nombre de langues couvertes croît, la modularisation des systèmes en analyse, transfert et génération devient cru-

ciale, de même que l'inversibilité de l'analyse et de la génération [Surcin et al., 2007]. L'approche de la traduction à base de règles est aujourd'hui l'objet de nombreuses publications [Amtrup et al., 2000, Charoenpornsawat et al., 2002] et est employée par le plus connu des systèmes commerciaux de traduction [Senellart et al., 2001].

Le début des années 1990 voit le développement d'autres types d'approches. Les ordinateurs se répandent et gagnent en puissance, ce qui permet l'émergence de stratégies qui se fondent sur de grandes quantités de données (« corpus-based approaches »). On distingue en particulier deux grands types d'approches : la traduction automatique à base d'exemples et la traduction automatique par méthodes statistiques.

La traduction automatique à base d'exemples (« example-based machine translation », ou EBMT) repose sur un ensemble « d'exemples » préalablement traduits : un corpus *parallèle* de phrases traductions l'une de l'autre. Nagao [1984] est considéré être à l'origine de la traduction automatique à base d'exemples, et Somers [1999] présente un tour d'horizon approfondi de cette approche.

Lorsqu'on lui présente une phrase à traduire, le système parcourt sa base d'exemples et produit trivialement une traduction si la phrase s'y trouve. Dans le cas général, la phrase n'apparaît pas dans la base et le système s'emploie alors à rassembler des exemples qui contiennent des fragments communs (des groupes de mots) avec la phrase à traduire. Pour chaque fragment d'exemple dans la langue source, il s'agit ensuite de retrouver sa traduction dans la langue cible : c'est la phase d'alignement. Enfin, la phase de génération assemble les fragments dans la langue cible et produit la traduction. À chacune des trois étapes, il est possible d'utiliser des sources externes de connaissances, telles que des lexiques bilingues, des listes de synonymes, des étiquettes ou des arbres syntaxiques, etc. [Nakazawa et al., 2006] est un exemple de développement récent dans le domaine, et emploie entre autres un lexique bilingue et un analyseur morphologique pour déterminer les structures de dépendance dans les phrases anglaises et japonaises.

La traduction automatique par méthodes statistiques doit ses origines aux travaux de Brown et al. [1990] et en particulier au prototype *Candide* [Berger et al., 1994], un système de traduction construit à partir de discours disponibles en français et en anglais de parlementaires canadiens. En effet, comme la traduction à base d'exemples, la traduction par méthodes statistiques repose sur un corpus parallèle. Un modèle statistique de traduction est défini, comprenant une ou plusieurs lois de probabilités. Le corpus est traité afin d'estimer ces lois, qui sont souvent constituées de plusieurs milliers voire millions de paramètres. Les systèmes de traduction conçus et utilisés au cours de cette thèse suivent l'approche statistique, aussi est-elle décrite plus en détail à la section suivante.

Concluons ce tour d'horizon des différentes approches de la traduction automatique en mentionnant les systèmes hybrides. Leur idée directrice est qu'une approche unique du problème de la traduction, aussi perfectionnée soit-elle, ne parviendra pas à produire une traduction satisfaisante dans tous les cas. Au contraire, une approche par règle peut s'avérer particulièrement adaptée à certaines phrases, tandis que d'autres phénomènes linguistiques sont correctement traités par une approche reposant sur des corpus. Un

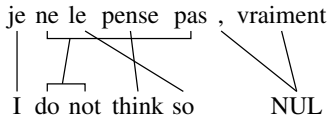


FIG. 2.2 – Exemple d'alignement entre deux phrases

système hybride pourrait parvenir à tirer profit des forces de chaque approche. Une première stratégie pour mettre en œuvre un système hybride est d'utiliser les différentes approches en parallèle. Par exemple Paul et al. [2005] font traduire le texte source par huit systèmes et sélectionnent ensuite la traduction finale. Une autre possibilité est d'utiliser deux systèmes en cascade. Ainsi, Simard et al. [2007] et Dugast et al. [2007] tentent de corriger automatiquement les erreurs d'un système à base de règles par un système statistique. Enfin, dans un système statistique, il est courant de traiter par un système de règles spécialisées certains fragments de phrases, typiquement les nombres, les dates, etc. Les morceaux de phrases ainsi identifiés et traduits en isolation par le système à base de règles peuvent être transmis au système statistique *via* un fragment de code XML [Koehn, 2004].

2.1.2 L'approche statistique

Il pourrait paraître surprenant au premier abord de vouloir traiter un processus linguistique comme la traduction par des méthodes statistiques. Toutefois, la traduction d'un texte nécessite la prise de décisions : choisir un mot, une locution ou tournure de phrase en prenant en considération de dépendances souvent difficiles à quantifier. L'approche statistique rend compte de ces dépendances floues et est en mesure de les combiner de façon multiplicative ou additive. En outre, le traitement statistique permet de garantir que pour toute phrase source, une phrase traduite sera générée — même si la syntaxe de cette phrase n'est pas correcte, elle permettra très probablement de transmettre le sens de la phrase originale. On peut donc résumer la traduction statistique comme la combinaison d'une modélisation linguistique et d'une *prise de décision statistique*.

Notion d'alignement

Parmi les nombreux modèles statistiques existants, la quasi-totalité d'entre eux introduisent une variable cachée $\mathcal{A}$, appelée *alignement*, qui décrit une correspondance entre les mots d'une phrase et ceux de sa traduction (parmi les traductions possibles). La figure 2.2 montre un exemple d'un tel alignement. Les alignements de groupes de mots à d'autres groupes de mots sont *a priori* autorisés, de même que l'alignement à un mot spécial appelé NUL utilisé lorsqu'un ou plusieurs mots d'une phrase n'ont pas de correspondance dans l'autre phrase (formellement, il y a un mot NUL dans chacune des langues).

Approximation au meilleur alignement

Un modèle statistique de traduction évalue par la quantité $\Pr(\mathbf{f}|\mathbf{e})$ la probabilité que la phrase $\mathbf{f} = f_1 \dots f_J$ soit une traduction de la phrase $\mathbf{e} = e_1 \dots e_I$. On introduit l'ensemble des alignements $\mathcal{A}$, puis en pratique seul l'alignement le plus probable est considéré :

$$\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathcal{A}} \Pr(\mathbf{f}, \mathcal{A}|\mathbf{e}) \approx \max_{\mathcal{A}} \Pr(\mathbf{f}, \mathcal{A}|\mathbf{e}) \tag{2.1}$$

Relation de Bayes

On suppose maintenant qu'il faille trouver la phrase cible $\mathbf{e}$, traduction d'une phrase source $\mathbf{f}$ donnée. La quasi-totalité des modèles statistiques $\Pr(\mathbf{e}|\mathbf{f})$ ne permettent pas seuls de déterminer une phrase $\mathbf{e}$ qui soit correcte syntaxiquement et grammaticalement. La relation de Bayes[1] est donc utilisée, afin de faire apparaître un modèle de langage de la langue cible $\Pr(\mathbf{e})$:

$$\operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \tag{2.2}$$

Cette transformation, inspirée de la reconnaissance automatique de la parole, reprend le principe « source/canal de transmission » et modifie le rôle du traducteur, qui doit maintenant retrouver la phrase $\mathbf{e}$ qui a *produit* la phrase $\mathbf{f}$ observée.

Modèle log-linéaire

En pratique, il est souvent bénéfique de pondérer les différentes sources d'information que sont le modèle de langage $\Pr(\mathbf{e})$ et le modèle de traduction $\Pr(\mathbf{f}|\mathbf{e})$. La quantité à maximiser devient ainsi :

$$\operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) \approx \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e})^{\alpha} \Pr(\mathbf{f}|\mathbf{e})^{(1-\alpha)} \tag{2.3}$$

avec un $\alpha \in [0, 1]$ à choisir judicieusement.

En outre, si le modèle de traduction $\Pr(\mathbf{f}|\mathbf{e})$ est le produit de plusieurs composantes, celles-ci peuvent être pondérées de la même façon. L'expression maximisée par le traducteur peut alors s'écrire sous la forme suivante :

$$\operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) \approx \operatorname*{argmax}_{\mathbf{e}} \prod_i h_i(\mathbf{e}, \mathbf{f})^{\lambda_i} \tag{2.4}$$

$$= \operatorname*{argmax}_{\mathbf{e}} \exp\left(\sum_i \lambda_i \log h_i(\mathbf{e}, \mathbf{f})\right) \tag{2.5}$$

L'équation 2.2 pouvait sembler contraignante : le principe « source/canal de transmission » justifiait chacun des deux termes et n'en autorisait *a priori* aucun autre. Avec

[1] Dans l'équation 2.2, le terme $\Pr(\mathbf{f})$ au dénominateur est supprimé car il n'a pas d'influence sur l'opération $\operatorname{argmax}_{\mathbf{e}}$.

l'équation 2.5, il s'agit maintenant de *caractériser* le processus de traduction au moyen d'une *combinaison log-linéaire* de fonctions caractéristiques $h_i(\mathbf{e}, \mathbf{f})$. Toute fonction aidant à produire une traduction correcte peut être incluse, sans autre justification théorique. Les fonctions caractéristiques usuelles peuvent inclure un ou plusieurs modèles de langage $h(\mathbf{e}, \mathbf{f}) = \Pr(\mathbf{e})$ et tout modèle de traduction $h(\mathbf{e}, \mathbf{f}) = \max_{\mathcal{A}} \Pr(\mathbf{f}, \mathcal{A}|\mathbf{e})$ (l'alignement $\mathcal{A}$ faisant partie des variables internes maintenues par le traducteur). Un système de traduction compte en général entre cinq et une douzaine de ces fonctions caractéristiques.

Entraînement contre développement

Les fonctions caractéristiques $h_i(\mathbf{e}, \mathbf{f})$ non-triviales sont apprises sur un corpus d'entraînement. Le détail de leur entraînement sera décrit dans les deux prochains chapitres, suivant le modèle de traduction choisi.

Une fois les fonctions caractéristiques calculées, l'on dispose d'une famille de modèles paramétrés par les λ_i :

$$\Pr(\mathbf{e}|\mathbf{f}) \propto \exp\left(\sum_i \lambda_i \log h_i(\mathbf{e}, \mathbf{f})\right) \tag{2.6}$$

Les λ_i sont estimés sur un ensemble de développement différent de l'ensemble d'entraînement : sans cela, il suffirait de favoriser les fonctions caractéristiques qui apprennent le mieux « par cœur » le corpus d'entraînement, sans considération pour leur aptitude à généraliser.

L'équation 2.6 à la forme des modèles dit « de maximum d'entropie » [Berger et al., 1996], mais les λ_i ne sont pas appris par les méthodes classiques (GIS [Darroch and Ratcliff, 1972], IIS [Della Pietra et al., 1997]). En effet, leur nombre restreint permet d'optimiser directement la mesure finale plutôt que de maximiser la vraisemblance des données de développement [Och and Ney, 2002]. Cette optimisation peut employer toute méthode numérique générique et peut viser à optimiser toute mesure automatique de la qualité de la traduction, par exemple parmi celles décrites à la section 2.4.

2.1.3 Modèles à base de mots

Les modèles « IBM »

Brown et al. [1993] ont défini cinq modèles statistiques de traduction de complexité croissante et proposé un algorithme pour leur apprentissage. Il s'agit de modèles à base de mots, c'est-à-dire que l'unité de traduction qui apparaît dans les lois de probabilité est le mot. Nous allons décrire en détail le modèle « IBM-4 », pour lequel nous avons écrit un décodeur (chapitre suivant), et présenter succinctement les quatre autres modèles.

Dans la suite de cette section, la phrase *source* est notée $\mathbf{e} = e_1 \dots e_I$ et la phrase *cible* $\mathbf{f} = f_1 \dots f_J$. Les modèles présentés proposent chacun une expression de la quantité

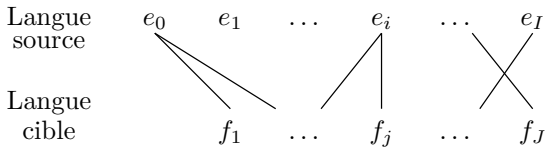


FIG. 2.3 – Exemple d'alignement autorisé par les modèles IBM-2 à IBM-5

$\Pr(\mathbf{f}|\mathbf{e})$. Ce sens de traduction est conforme avec celui présent dans le membre de droite de l'équation 2.2 (relation de Bayes).

Le modèle « IBM-1 » est une exception : c'est le seul modèle à ne pas aligner les mots source aux mots cible. Plus exactement, il considère que tous les mots source peuvent être alignés à tous les mots cible avec la même probabilité. Le modèle IBM-1 repose sur une seule loi de probabilité, une loi *lexicale* notée $t(f|e)$.

La seule raison d'être de ce modèle est de permettre l'entraînement des modèles suivants.

Le modèle « IBM-2 » et les suivants imposent des restrictions sur l'alignement $\mathcal{A}$ entre les mots des phrases source et cible. Au lieu d'être aussi général qu'à la figure 2.2, il doit être de la forme $\mathcal{A} = a_1 \dots a_J$, où, pour tout j de l'intervalle $[1, J]$, $a_j \in [0, I]$. $a_j = i > 0$ signifie que le mot cible f_j est aligné à e_i, tandis que $a_j = 0$ signifie que f_j n'est pas aligné, ou est aligné au mot NUL e_0. Ainsi, un alignement de cette forme autorise l'alignement de plusieurs f_j à un seul e_i, mais pas l'inverse : un mot cible f_j est aligné à 0 ou 1 mot source. Le modèle IBM-2 et les suivants sont donc *asymétriques*. La figure 2.3 présente un exemple d'alignement respectant les contraintes ci-dessus. Notons qu'il est acceptable qu'un e_i n'ait généré aucun f_j, c'est-à-dire que pour ce i et $\forall j \in [1, J]$, $a_j \neq i$.

En plus de la loi de traduction lexicale $t(f|e)$, le modèle IBM-2 dispose d'une loi d'*alignement* ou de *distorsion* de la forme $p(a_j|j)$.

Le modèle « IBM-3 » intègre en plus une loi de *fertilité*, de la forme $n(\phi|e)$. Pour chaque position source $i \in [1, I]$, ϕ_i est le nombre de mots cible alignés à e_i, soit

$$\phi_i = \text{Card}\,\{j|a_j = i\} \tag{2.7}$$

Le modèle IBM-3 considère que les mots cible f alignés à aucun mot source apparaissent spontanément entre les autres mots cible. Le modèle définit ainsi une probabilité $p_1 = 1 - p_0$ de génération spontanée d'un mot cible aligné à e_0 après toute génération de mot cible aligné à un mot source. La loi $t(f|e_0)$ détermine alors quel mot cible sera généré. Ces mots « spontanés » ne portent pas de sens ; en pratique, ils sont insérés de façon à respecter la grammaire de la langue cible.

Le modèle « IBM-4 » ne diffère d'IBM-3 que par son modèle de distorsion, qui repose sur deux lois de probabilité. La première, $d_{=1}(\Delta j|\mathcal{E}, \mathcal{F})$, permet de positionner le

premier mot cible f généré par un mot source e (les notations sont expliquées plus bas). La deuxième loi de distorsion, notée $d_{>1}(\Delta j|\mathcal{F})$, permet de positionner les éventuels mots cible supplémentaires générés par un même mot source e dont la fertilité serait strictement supérieure à 1.

Le modèle IBM-4 est donc composé des quatre sous-modèles suivants :

1. le modèle de traduction lexicale $t(f|e)$;
2. le modèle de fertilité $n(\phi|e)$;
3. le modèle de distorsion, avec les lois $d_{=1}(\Delta j|\mathcal{E},\mathcal{F})$ et $d_{>1}(\Delta j|\mathcal{F})$;
4. et le modèle de « génération spontanée », avec la probabilité $p_1 = 1 - p_0$ et la loi $t(f|e_0)$.

IBM-4 modélise la production de la phrase cible par la phrase source en trois temps. Dans un premier temps, la fertilité ϕ_i de chaque mot source e_i est déterminée selon la loi $n(\phi|e)$. Dans un deuxième temps, les mots source e_i dont la fertilité est non nulle produisent chacun respectivement ϕ_i mots cible, dont le choix suit la loi $t(f|e)$ et qui sont placés suivant les lois de distorsion $d_{=1}(\Delta j|\mathcal{E},\mathcal{F})$ et $d_{>1}(\Delta j|\mathcal{F})$. Dans l'équation 2.8, $\tau_{i,k} \in [1, J]$ est la position du $k^{\text{ème}}$ mot cible généré par e_i, $\mathcal{F}(f)$ est la classe d'un mot cible f, $\mathcal{E}_{i-1}$ est la classe du dernier mot source fertile précédent e_i et $\overline{\tau_{i-1}}$ est la moyenne des positions des mots cible générés par ce dernier mot source fertile. Enfin dans le troisième et dernier temps, pour chaque mot cible déjà produit, une décision binaire d'introduire ou non un mot cible aligné à NUL est prise, avec une probabilité p_1 d'introduire un tel mot (selon la loi $t(f|e_0)$).

Finalement, l'équation 2.8 fournit l'expression de la probabilité $\Pr(\mathbf{f}, \mathcal{A}|\mathbf{e})$ telle qu'elle est modélisée par IBM-4 [Brown et al., 1993, page 16] :

$$
\begin{aligned}
\Pr(\mathbf{f}, \mathcal{A}|\mathbf{e}) = & \\
& \prod_{i=1}^{I} n(\phi_i|e_i) \times \\
& \quad t(f_{\tau_{i,1}}|e_i) d_{=1}(\tau_{i,1} - \overline{\tau_{i-1}}|\mathcal{E}_{i-1}, \mathcal{F}(f_{\tau_{i,1}})) \times \\
& \quad \prod_{k=2}^{\phi_i} t(f_{\tau_{i,k}}|e_i) d_{>1}(\tau_{i,k} - \tau_{i,k-1}|\mathcal{F}(f_{\tau_{i,k}})) \times \\
& \binom{J - \phi_0}{\phi_0} p_0^{J-2\phi_0} (1 - p_0)^{\phi_0} \frac{1}{\phi_0!} \times \\
& \quad \prod_{k=1}^{\phi_0} t(f_{\tau_{0,k}}|e_0)
\end{aligned}
\tag{2.8}
$$

Les lois de distorsion sont conditionnées par des classes de mots dans le but de modéliser certains phénomènes de réordonnement. Par exemple, les adjectifs, qui en anglais précèdent le nom, se trouvent souvent après le nom en français. Cette propriété semble effectivement capturée par le modèle $d_{=1}(\Delta j|\mathcal{E},\mathcal{F})$ [Brown et al., 1993, page 19].

Le modèle de traduction IBM-4 obtient de très bonnes performances comparé à d'autres modèles à base de mots [Och and Ney, 2003]).

Le modèle « IBM-5 » est identique au modèle IBM-4 à ceci près qu'il corrige sa « déficience ». En effet, les lois de distorsion du modèle IBM-4 ne prennent pas en compte les positions cible déjà couvertes et allouent une masse de probabilité à des évènements impossibles, comme des pseudo-phrases où plusieurs mots occuperaient la même position. Le modèle IBM-5 est plus juste mathématiquement mais son entraînement est significativement plus long que celui d'IBM-4 et ses performances similaires, voire légèrement inférieures [Och and Ney, 2003].

Le modèle « HMM »

Le modèle de traduction dit « HMM » [Vogel et al., 1996] est similaire au modèle IBM-2. Comme ce dernier, le modèle HMM est composé d'une loi lexicale $t(f|e)$ et d'une loi de distorsion. Toutefois, alors que la loi de distorsion d'IBM-2 était une loi d'ordre 0, de la forme $p(a_j|j)$, celle du modèle HMM est d'ordre 1, de la forme $p(a_j|a_{j-1}) = p(a_j - a_{j-1})$. Och and Ney [2003] ont montré que cette différence rendait le modèle HMM nettement meilleur que le modèle IBM-2, tout en restant moins performant que le modèle IBM-4.

Entraînement des paramètres de ces modèles

Un programme appelé Giza++ réalise l'entraînement des paramètres des modèles présentés ci-dessus et est disponible publiquement[2]. Giza++ est une extension du programme Giza, qui fait partie de la boîte à outils pour la traduction automatique statistique Egypt [Al-Onaizan et al., 1999].

L'entraînement est réalisé à partir d'un corpus parallèle, aligné par phrases. Aucune information *a priori* est nécessaire, pas même un lexique. Giza++ entraîne successivement des modèles de complexité croissante, comme proposé par Brown et al. [1993] et évalué de façon systématique par Och and Ney [2003]. Chaque itération consiste en une étape de l'algorithme « Expectation-Maximization »[Baum et al., 1970]. Le détail de la séquence d'entraînement dépend de l'application et sera détaillé aux chapitres suivants.

Modèles dépendant du contexte

Afin de limiter la complexité des modèles, Brown et al. [1993] sont amenés à supprimer la plupart des dépendances des lois de traduction lexicale, de fertilité et de distorsion. Ainsi, dans [Brown et al., 1993, page 13], $\Pr(F_{i,k} = f|\cdot)$ devrait être conditionnée notamment par l'ensemble des mots cibles déjà produits et par la phrase source entière $\mathbf{e}$. La quantité $\Pr(F_{i,k} = f|\cdot)$ est en fait approchée par $t(F_{i,k} = f|e_i)$, notée plus simplement $t(f|e)$, qui ne conserve que le conditionnement sur le mot source e_i aligné à f.

Berger et al. [1996] proposent d'apprendre des modèles à maximum d'entropie pour les mots source e les plus fréquents. Pour ces mots, la loi $t(f|e)$ de [Brown et al., 1993] est remplacée par une loi de la forme $t_e(f|x)$, où x représente un *contexte* autour du mot

[2] Giza++ : `http://www.fjoch.com/GIZA++.html`.

source e. Berger et al. [1996] présentent deux phrases dont la traduction est améliorée par l'utilisation de tels modèles.

Varea et al. [2002] reproduisent et étendent ces expériences. Le *contexte* x inclut les trois mots à gauche et les trois mots à droite du mot source considéré, comme dans [Berger et al., 1996], ainsi que les classes respectives de ces mots. Par ailleurs, les modèles lexicaux contextuels sont intégrés à la procédure d'entraînement décrite à la sous-section précédente, ce qui permet d'envisager que les meilleurs modèles lexicaux aident à obtenir également de meilleurs modèles de distorsion et de fertilité. Enfin, afin d'éviter le sur-apprentissage, le modèle lexical dépendant du contexte est interpolé avec le modèle lexical « basique » :

$$t'_e(f|x) = \lambda t_e(f|x) + (1 - \lambda)t(f|e) \tag{2.9}$$

L'intérêt des modèles contextuels a été évalué en terme de qualité d'alignement du corpus d'entraînement. Varea et al. [2002] ont constaté une amélioration significative des taux de rappel et de précision par rapport à un alignement manuel du corpus servant à l'évaluation. Toutefois, à notre connaissance, cette amélioration de la qualité d'alignement ne s'est pas traduite par une amélioration de la qualité de traduction.

Critique

Les modèles à base de mots que nous venons de décrire utilisent le mot comme unité de traduction. Notons que ces modèles permettent une meilleure traduction que par une technique « mot à mot », puisqu'ils autorisent et modélisent des phénomènes comme le fait qu'un mot se traduise en plusieurs mots ou que les mots soient réordonnés. De plus, le programme réalisant la traduction emploie toujours un modèle de langage : de ce fait, le choix de la traduction de chaque mot n'est pas pris en isolation mais en considérant son impact sur l'ensemble de la phrase produite.

Supposons que nous traduisions la phrase `it isn't true` (`ce n' est pas vrai`) avec un modèle IBM-4. La traduction du mot `isn't` est problématique : il nous faut espérer que le modèle de fertilité préfère une fertilité de 3 aux fertilités plus faibles, et que le modèle de traduction lexicale génère les trois mots `n'`, `est` et `pas`. Ceci est tout à fait en mesure d'émerger de l'apprentissage, mais le risque demeure de produire `c' est vrai`, que le modèle de langage est susceptible de préférer à la phrase plus longue `ce n' est pas vrai`. Pourtant, le couple ⟨`isn't`, `n' est pas`⟩ est probablement apparu de nombreuses fois dans le corpus d'entraînement : ceci suggère que même entraînés sur un corpus adéquat, les modèles à base de mots sont susceptibles de mal traduire des expressions courantes.

Un autre problème des modèles à base de mots est l'asymétrie des alignements qu'ils imposent (figure 2.3). Dans l'exemple ci-dessus, un modèle IBM-4 entraîné du français vers l'anglais n'est pas en mesure de modéliser correctement la traduction de `n' est pas` en `isn't`. En effet, IBM-4 ne permet d'aligner le mot `isn't` qu'à un mot français au plus, or aucun mot parmi `n'`, `est` et `pas` n'est ici un choix satisfaisant.

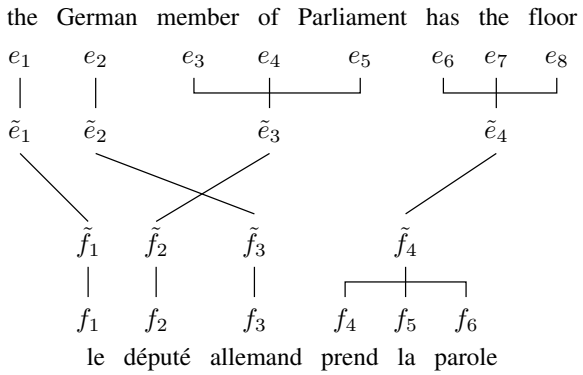


FIG. 2.4 – Alignement autorisé par la plupart des modèles par groupes de mots (certains modèles ne prévoient pas le réordonnement des groupes de mots)

2.1.4 Modèles par groupes de mots

Principe

Les modèles par groupes de mots (*phrase-based translation models*) corrigent ces défauts en symétrisant l'alignement entre les mots source et cible et en extrayant plus d'information du corpus d'entraînement. Comme leur nom l'indique, l'unité de traduction de ces modèles est le groupe de mots. Un groupe de mots peut compter un ou plusieurs mots. Les groupes de mots sont symbolisés avec un tilde : par exemple, $\tilde{e} = e_i, \ldots, e_{i+l-1}$ regroupe l mots, avec $l \geq 1$. Nous décrivons ici les principes communs à tous les systèmes de traduction par groupes de mots.

Le processus de traduction est illustré à la figure 2.4. La phrase source $\mathbf{e}$ est d'abord segmentée en K groupes de mots : $\mathbf{e} = \tilde{e}_1 \ldots \tilde{e}_K$. Chaque groupe de mots source $\tilde{e}_k$ est ensuite traduit en un groupe de mots cible $\tilde{f}_k$. Ces groupes de mots sont éventuellement réordonnés selon une permutation $\rho(\cdot)$ de $[1, K]$ puis sont simplement accolés pour constituer la phrase cible finale $\mathbf{f} = \tilde{f}_{\rho(1)} \ldots \tilde{f}_{\rho(K)}$.

Utiliser des groupes de mots comme unité de traduction permet d'aligner n mots source à m mots cible et d'éviter les alignements parfois peu satisfaisants qu'imposaient les modèles à base de mots. Dans l'exemple de la figure 2.4, `Member of Parliament` est aligné à `député` et `has the floor` à `prend la parole`.

Par ailleurs, un autre alignement valide pour cette paire de phrases pourrait aligner les quatre mots `German Member of Parliament` à `député allemand` et éviter ainsi tout réordonnement. Ceci est une propriété importante des modèles par groupes de mots : ils sont en mesure de traduire directement, par exemple, des groupes nominaux ou des ensembles nom+adjectif *observés sur l'ensemble d'apprentissage* et ainsi parvenir à préserver certaines contraintes locales sur l'ordre des mots. Un algorithme d'extraction de paires de groupes de mots est présenté au chapitre 4.

Fonctions caractéristiques usuelles

Comme les modèles à base de mots, les modèles par groupes de mots sont souvent exprimés sous la forme d'un produit pondéré de sous-modèles (équation 2.6). La « table de traduction » (*phrase table*) est la loi de probabilité centrale d'un modèle par groupes de mots. Elle assigne une probabilité $t(\tilde{f}|\tilde{e})$ à chaque couple $\langle \tilde{e}, \tilde{f} \rangle$. En pratique, la table de traduction leur attribue souvent plusieurs scores, qui sont eux-mêmes pondérés et multipliés dans le cadre de l'équation 2.6.

Les modèles qui autorisent le réordonnement des groupes de mots disposent en plus d'un sous-modèle de distorsion, déjà employé par les modèles à base de mots. Nous décrivons ici un modèle de distorsion simple et répandu inspiré de celui du modèle de traduction HMM [Vogel et al., 1996].

Notons a_k la position du début de $\tilde{f}_{\rho(k)}$ et b_{k-1} celle de la fin de $\tilde{f}_{\rho(k-1)}$. Le terme de distorsion correspondant au positionnement de $\tilde{f}_{\rho(k)}$ est défini par $d(a_k - b_{k-1}) = \alpha^{|a_k - b_{k-1} - 1|}$, où α est choisi judicieusement. Ce modèle autorise ainsi *a priori* tous les réordonnements, tout en favorisant les sauts courts et, en particulier, les traductions monotones. En cela, il est particulièrement adapté à un couple de langues comme l'anglais et l'espagnol. D'autres modèles de distorsion sont parfois mis en œuvre, par exemple pour la traduction de langues plus éloignées, comme l'arabe et l'anglais [Al-Onaizan and Papineni, 2006] ou le chinois et l'anglais [Kuhn et al., 2006].

Enfin, tous les systèmes de traduction statistiques utilisent un ou plusieurs modèles de langage de la langue cible.

Systèmes décrits dans la littérature

Le modèle à patrons d'alignement (*alignment template model*) [Och et al., 1999, Och and Ney, 2004] fut le premier modèle de traduction qui permette de traduire plusieurs mots d'un coup. Il a inspiré de nombreux systèmes par groupes de mots, qui ne diffèrent que sur les détails de leur entraînement, sur leur modélisation des réordonnements et sur le nombre de fonctions caractéristiques de leur modèle log-linéaire (équation 2.6). [Zens et al., 2002] et [Koehn et al., 2003] sont d'autres articles fondateurs de la traduction par groupes de mots.

En ce qui concerne les décodeurs, Pharaoh [Koehn, 2004] a joué un rôle important dans la communauté du fait de sa disponibilité, même sous des conditions restrictives[3]. Il a depuis inspiré le développement de Moses [Koehn et al., 2007], qui lui est un logiciel libre[4]. Au cours de cette thèse, nous avons mis en œuvre un système de traduction reposant sur Moses, dont l'algorithme de décodage sera décrit succinctement au chapitre 4. Signalons également le décodeur Marie [Crego et al., 2005], lui aussi un logiciel libre[5] et qui a la particularité de pouvoir tirer parti d'un modèle de « langage » n-gramme construit sur les couples $\langle \tilde{e}, \tilde{f} \rangle$, appelés *tuples*. Ainsi, en plus des probabilités $t(\tilde{e}|\tilde{f})$ que la quasi-totalité des traducteurs par groupes de mots utilisent, Marie peut

[3] Pharaoh est disponible depuis `http://www.isi.edu/licensed-sw/pharaoh/`.

[4] Moses est disponible sous licence LGPL depuis `http://www.statmt.org/moses/`.

[5] Marie est disponible sous licence GPL depuis `http://gps-tsc.upc.es/veu/soft/soft/marie/`.

utiliser des probabilités de la forme $t(\langle \tilde{e}_k, \tilde{f}_k \rangle | \langle \tilde{e}_{k-2}, \tilde{f}_{k-2} \rangle, \langle \tilde{e}_{k-1}, \tilde{f}_{k-1} \rangle)$.

Le lecteur intéressé et motivé est invité à se reporter aux descriptions des systèmes de traduction d'IBM [Tillmann, 2003], d'ITC-IRST [Cettolo et al., 2005], de RWTH [Mauser et al., 2006], de CMU [Vogel et al., 2003], du NRC [Sadat et al., 2005] ou du LIMSI [Déchelotte et al., 2007b], cette liste n'étant pas exhaustive.

2.1.5 Modèles statistiques syntaxiques

Tandis que les modèles par groupes de mots représentent aujourd'hui l'approche dominante parmi les modèles statistiques de traduction, plusieurs de leurs faiblesses motivent une recherche soutenue pour développer des modèles qui s'appuient sur la *syntaxe*. Voici quelques raisons pour lesquelles prendre en compte explicitement la syntaxe des deux langues est susceptible d'être bénéfique :

- Pour certaines paires de langues, la plupart des réordonnements ne sont motivés que par la syntaxe. Si certains réordonnements locaux, par exemple entre adjectifs et noms, peuvent être appris par une approche par groupes de mots, il n'en demeure pas moins qu'il sont appris « par cœur », sans généralisation : si un couple nom+adjectif n'a jamais été observé lors de l'entraînement, il sera traduit mot à mot et risque d'être mal traduit. De plus, seuls les modèles syntaxiques semblent en mesure de traiter les réordonnements sur de plus longues distances, nécessaires par exemple pour le placement des verbes en allemand ou pour traduire du chinois vers l'anglais.
- Les modèles syntaxiques peuvent avoir une connaissance explicite des « mots de fonction » nécessaires à la construction d'une phrase correcte. Un modèle syntaxique peut notamment disposer de règles pour le choix des déterminants, ou savoir avec quelles prépositions et quel auxiliaire un verbe se construit.
- Les modèles syntaxiques peuvent conditionner la traduction d'un mot sur un autre avec qui il a un lien syntaxique. Par exemple, la traduction d'un verbe peut dépendre explicitement de son sujet. Ceci représente une différence cruciale avec les systèmes non-syntaxiques. Les systèmes par groupes de mots font par exemple l'hypothèse que les probabilités de traduction sont indépendantes entre elles[6] : $t(\tilde{f}_k|\tilde{e}_k)$ ne dépend pas du choix de $\tilde{f}_1, \ldots, \tilde{f}_{k-1}, \tilde{e}_1, \ldots, \tilde{e}_{k-1}$. Les systèmes syntaxiques peuvent quant à eux conserver des dépendances sur les mots ou les groupes de mots pertinents, et être ainsi plus à même à assurer une cohérence entre les choix de traduction.
- Finalement, l'utilisation de modèles de traduction ou de langage syntaxiques permet d'espérer que le texte produit soit grammaticalement correct.

Parmi les nombreux systèmes de traduction reposant sur la syntaxe, citons par exemple [Yamada and Knight, 2001] pour la traduction de l'anglais au japonnais. La phrase à traduire est dans un premier temps analysée pour construire son arbre syntaxique. Les nœuds de l'arbre sont ensuite réordonnés selon des règles apprises de façon non-supervisée sur le corpus d'entraînement. C'est à cette étape, notamment, que des structures de phrase du type SVO (Sujet Verbe Objet) de l'anglais peuvent être transformées en structures SOV (Sujet Objet Verbe), pour le japonais. Dans une troisième étape, des mots sont éventuellement insérés avant ou après chaque nœud. L'objectif est ici

[6] Cette limitation est en partie corrigée par l'utilisation d'un modèle de langage n-gramme sur les tuples $\langle \tilde{e}, \tilde{f} \rangle$ [Crego et al., 2005].

d'insérer les particules qui existent en japonais pour marquer le *cas*, qui est déterminé en anglais et en français par l'ordre des mots. La quatrième et dernière étape est la traduction proprement dite des feuilles de l'arbre. Il suffit alors de lire les feuilles traduites pour obtenir la phrase cible. Yamada and Knight [2001] montrent que cette l'entraînement de ce modèle produit des alignements de mots plus satisfaisants que le modèle IBM-5.

L'approche précédente construit un arbre syntaxique dans une langue, effectue des opérations dessus puis produit la traduction. Un autre type de modèle syntaxique repose sur la construction d'arbres syntaxiques dans les deux langues. [Wu, 1997] fait l'hypothèse qu'il est possible de construire un arbre syntaxique qui corresponde simultanément aux deux phrases, source et cible, avec un formalisme de grammaire hors-contexte. Ceci est rendu possible en autorisant systématiquement l'inversion des règles de production. Ainsi, une règle comme VP $\rightarrow$ V NP prévue pour le chinois conduit à la création d'une règle VP $\rightarrow$ [V NP]$|\langle$V NP$\rangle$ capable d'analyser l'anglais et le chinois simultanément. Wu [1997], Fung et al. [2004] ont montré que ce formalisme était utilisable même avec des ressources réduites en temps de calcul et en mémoire, tout en obtenant une bonne qualité de traduction.

Les inconvénients des approches syntaxiques sont liés à la complexité des modèles et de leur mise en œuvre. Les analyses effectuées pendant l'entraînement et le décodage dépendent des deux langues traitées, voire du sens de traduction. Par ailleurs, ces analyses nécessitent de bons outils, or il n'est pas aisé de juger à partir de quel niveau de performance un analyseur syntaxique peut commencer à effectivement aider la traduction. Toutefois, il est encourageant de constater que les approches syntaxiques sont dorénavant compétitives dans des évaluations internationales[7].

2.2 Motivations et choix pour la conception de nos systèmes de traduction

Avant le commencement de cette thèse, le LIMSI ne disposait pas de système de traduction. Un des objectifs de la thèse était donc de développer un traducteur. Le cahier des charges pour le futur système comprenait les points suivants :

- le système de traduction devait être complet et autonome, nous devions connaître son fonctionnement précis et pouvoir aisément le modifier pour les besoins de notre recherche ;
- en particulier, parmi les thématiques qui nous intéressaient, l'apprentissage discriminant du modèle de traduction et les modèles de traduction contextuels sont susceptibles de nécessiter des modifications importantes du décodeur ;
- par ailleurs, puisque cette thèse porte sur la traduction *de la parole*, le traducteur devait pouvoir s'intégrer à la reconnaissance automatique de la parole (RAP), même si les modalités de cette intégration n'étaient pas encore connues.

[7] Par exemple, le système syntaxique d'ISI [Galley et al., 2006] a obtenu de très bons résultats du chinois à l'anglais à l'évaluation NIST de 2006 : `http://www.nist.gov/speech/tests/mt/doc/mt06eval_official_results.html`.

Comme nous l'avons vu, l'approche statistique de la traduction repose sur un modèle de langage et un modèle de traduction. Le groupe TLP dispose d'une solide expertise dans le domaine des modèles de langages (entre autres) et d'outils performants : à la fois publics, comme la boîte à outils SRILM [Stolcke, 2002], et développés au LIMSI, comme la boîte à outils STK et le modèle de langage neuronal [Schwenk, 2007]. Concernant le modèle de traduction, nous avions le choix entre mettre en œuvre un modèle décrit dans la littérature et définir notre propre modèle de traduction statistique. Nous avons opté pour la première solution : nos résultats seraient alors plus facilement comparables à ceux d'autres laboratoires, et cela fournirait également un point de comparaison si nous devions par la suite « inventer » un autre modèle. Restait à choisir le modèle parmi ceux existants. Au moment de la rédaction de cette thèse, le modèle par groupes de mots est clairement le plus populaire parmi les modèles statistiques non-syntaxiques. Trois à cinq ans plus tôt, la variété des modèles de traduction décrits dans la littérature était pléthorique. Plusieurs publications faisaient état des bonnes performances des modèles traduisant les mots groupes par groupes [Och and Ney, 2004, Marcu and Wong, 2002, Koehn, 2004] mais certains systèmes à base de mots savaient rester compétitifs [Akiba et al., 2004]. De plus, au début de sa thèse, l'auteur considérait volontiers que la traduction de groupes de mots hors contexte était un palliatif temporaire avant l'avènement de modèles de traduction *contextuels*.

Notre choix s'est donc porté sur l'écriture d'un décodeur « maison » pour le modèle IBM-4. En plus des points cités plus haut, plusieurs éléments ont décidé ce choix :

- De nombreuses descriptions de systèmes et de décodeurs sont disponibles, que ce soit pour le modèle IBM-4 ou pour d'autres modèles. L'écriture de mon propre décodeur paraissait pleinement réalisable dans le cadre de cette thèse.
- À notre connaissance, deux décodeurs étaient disponibles mi-2004 : ReWrite[8] [Germann, 2003] et Pharaoh[9] [Koehn, 2004]. ReWrite est un décodeur pour le modèle IBM-4 tandis que Pharaoh est un décodeur pour modèle par groupes de mots. Leurs algorithmes sont décrits dans les articles cités ci-dessus mais les programmes ne sont pas libres et, en particulier, ils sont distribués sous forme binaire seule, ce qui ne convient pas à une utilisation pérenne.
- L'architecture retenue pour notre décodeur est inspirée de la littérature et comporte notamment de fortes similarités y compris avec certains décodeurs par groupes de mots comme Pharaoh. Ainsi, après avoir développé un décodeur pour le meilleur modèle à base de mots qu'est IBM-4, deux perspectives d'évolution sont envisageables : le raffinement du modèle IBM-4 par l'ajout de dépendances contextuelles, ou le passage à un modèle de groupes de mots.

En ce qui concerne l'entraînement des modèles, le programme Giza++ (déjà mentionné plus haut) était disponible librement pour l'entraînement des paramètres d'IBM-4 et nous avons décidé de l'utiliser tel quel. Le décodeur pour IBM-4 a été entièrement développé par l'auteur au cours de cette thèse et est décrit au chapitre 3.

En février 2006, ce système de traduction a participé à l'évaluation internationale TC-Star. Le système a été évalué pour la traduction de l'espagnol vers l'anglais. Même s'il ne faisait pas partie des meilleurs systèmes, il a obtenu des résultats similaires ou

[8] ReWrite : `http://www.isi.edu/licensed-sw/rewrite-decoder/`.
[9] Pharaoh : `http://www.isi.edu/licensed-sw/pharaoh/`.

meilleurs que certains autres sites qui avaient pourtant une expérience bien plus longue dans ce domaine, ce qui était encourageant. Le développement de Moses l'été suivant allait toutefois être décisive.

Moses est distribué sous licence libre GPL, et a obtenu de très bonnes performances dès nos premières expériences sur les mêmes données que notre traducteur interne. De plus, Moses est activement développé et dispose de nombreuses caractéristiques intéressantes, comme la possibilité d'exploiter des modèles de traduction factorisés ou des modèles de distorsion lexicalisés, de traduire des réseaux de consensus, de spécifier la traduction de certains mots *via* un fragment de code XML, etc. Nous avons donc décidé de créer un second système de traduction, cette fois centré sur Moses. Ce système est décrit au chapitre 4.

Un algorithme innovant d'apprentissage discriminant de la table de traduction a ensuite été développé à partir de Moses. C'est le sujet du chapitre 5.

2.3 Tâches de traduction considérées

2.3.1 Traduction de discours parlementaires

Cette thèse a été réalisée dans le cadre du projet européen TC-Star [Tc-Star, 2006]. Ce projet, financé par la Commission Européenne, a pour objectif d'améliorer significativement la traduction automatique de la parole et de réduire l'écart de performance entre la traduction manuelle et la traduction automatique. En effet, les barrières linguistiques ont été identifiées comme « le dernier obstacle au commerce des services informatifs en Europe » (*ibid.*, page 11). L'Union Européenne compte 23 langues officielles[10], dans lesquelles sont traduits les traités et les lois qui la régissent. Au sein même du parlement européen, durant les sessions plénières à Strasbourg et à Bruxelles, des interprètes traduisent les interventions de chaque député européen de sa langue natale dans chacune des 22 autres langues officielles, pour un total de 506 sens de traduction différents. Il n'existe pas toujours d'interprète pour chaque sens de traduction. Il est parfois nécessaire d'avoir recours à une langue pivot : par exemple, pour traduire du bulgare au maltais, un interprète traduit le bulgare en français (ou anglais), et un autre le retraduit en maltais.

La tâche considérée pour le projet TC-Star est la traduction de la parole conversationnelle non contrainte dans les trois langues suivantes : l'anglais européen, l'espagnol européen et le mandarin (chinois).

Nous avons choisi de travailler sur la paire de langue anglais-espagnol, dans les deux sens de traduction. Les données à traduire se composent de discours d'euro-députés qui se sont exprimés en anglais ou en espagnol et dont c'était leur langue natale. Comme les données à traduire de l'espagnol vers l'anglais étaient en quantité insuffisante, des discours tirés du parlement espagnol ont été ajoutés aux données provenant du parlement européen.

[10] Au 1er janvier 2007.

let me say this now and let me say this with all due seriousness ; I take these allegations very very seriously indeed which are being made in order to undermine my integrity and my reputation .
it is crucial that you don't agree but I urge but I urge you to respond or ask a German government representative to do so .
we collected more than twenty-three thousand signatures in two months .

(a) Exemple de données « Verbatim »

[... on any evidence] let me say this now let me say this with all due seriousness .
I take these allegations very very seriously indeed which are being made in order to undermine my integrity and my reputation .
it is crucial that you do not agree but Europe but I urge you to respond or ask a German government representatives to do so .
[... the context of the that of] the collected more than twenty three thousand signatures in two months [the latest trade ...]

(b) Exemple de données « RAP »

I must say at once and with all due seriousness that I take these allegations, which are aimed at undermining my integrity and reputation, very seriously indeed.
It is significant that you do not agree, but I urge you to respond or to ask a German Government representative to do so.
We collected more than 23 000 signatures in two months.

(c) Exemple de données « TF »

FIG. 2.5 – Extraits des données à traduire pour l'évaluation de la traduction de l'anglais vers l'espagnol

2.3.2 Détails des conditions de traduction

Le projet TC-STAR propose de traduire trois types de données :

1. les transcriptions **Verbatim** des interventions des euro-députés aux sessions plénières. Tous les artefacts liés à la parole sont conservés : répétitions de mots, agrammaticalités, phrases interrompues, ...
2. les transcriptions produites par des systèmes de **RAP**, semblables aux transcriptions Verbatim, aux erreurs de reconnaissance près.
3. et le **Texte Final (TF)**, qui est le texte final officiel disponible notamment depuis le site web du parlement européen[11]. Ce texte est rédigé en corrigeant les « défauts » des transcriptions Verbatim.

La figure 2.5 montre plusieurs extraits des données de la campagne d'évaluation de 2007, pour la traduction de l'anglais vers l'espagnol. À la sous-figure (a) se trouvent deux phrases tirées de la condition Verbatim. Les données sont en « vraie casse », c'est-à-dire que les majuscules respectent les règles orthographiques (« I » et « German »,

[11]Le site du parlement européen : `http://www.europarl.europa.eu`

dans les exemples) mais la première lettre de la phrase n'est pas systématiquement mise en majuscule. Les phrases ont été ponctuées et la ponctuation est séparée explicitement des mots, et les nombres sont en toutes lettres (« twenty-three thousand »). Enfin, les éventuelles erreurs ou hésitations du locuteur sont conservées (« but I urge but I urge you to »).

La sous-figure (b) présente la sortie d'un système de reconnaissance automatique de la parole. Les conventions relatives à la casse et la ponctuation sont les mêmes que pour la condition Verbatim. La segmentation du flux de mots en phrases est déterminée automatiquement et peut donc différer de celle des données Verbatim (signalé par les crochets). La transcription étant automatique, elle peut naturellement contenir des erreurs, comme « but Europe but I urge you to », ou « the context of the that of » qui aurait du être « the context of the World Cup ». Signalons également que certains mots, même reconnus correctement, sont susceptibles d'être transcrits sous différentes formes, comme « do not » de la condition RAP contre la forme contractée « don't » de la condition Verbatim.

Enfin la sous-figure (c) montre la version TF des mêmes phrases. La casse et la ponctuation suivent ici les règles usuelles : la première lettre de la phrase est en majuscule, et les ponctuations sont collées aux mots de façon appropriée. Les quantités (nombres, montants, dates, etc) sont représentées avec des chiffres lorsque cela aide la lecture (« 23 000 signatures » mais « two months »). Par ailleurs, les phrases sont remaniées, parfois sensiblement, et le niveau de langue en condition TF est généralement plus élevé que dans les deux autres conditions. Notons que les phrases relatives à la gestion de la séance (accueil des députés, début et fin de prise de parole, etc), présentes dans les conditions Verbatim et RAP, sont supprimées de la version TF.

Nous nous intéressons dans cette thèse à la traduction des données Verbatim et RAP, ce qui est conforme à notre intérêt spécifique pour la traduction de la parole. Afin d'éviter la charge de travail liée au développement d'un système pour une condition supplémentaire, nous avons choisi de ne pas traiter la condition TF.

2.3.3 Données servant à l'entraînement, au développement et à l'évaluation

Le tableau 2.1 rassemble des informations sur les données d'apprentissage, de développement et d'évaluation. Les seules données d'apprentissage parallèles sont les textes du parlement européen, avec environ 1,2 M de phrases parallèles. Les textes des trois parlements (européen, espagnol et britannique) respectent les normes usuelles de casse et de ponctuation, ce qui correspond au format TF de la section précédente. Les « transcriptions acoustiques fines » sont un ensemble de données nécessaire au développement des systèmes de reconnaissance de la parole. Ces transcriptions sont au format Verbatim et représentent ainsi les seules données d'entraînement dans ce format.

Concernant les données de développement et d'évaluation des systèmes de traduction, les spécifications ont évolué au cours du projet. Les ensembles Dev05 et Eval05 ne contiennent pas de ponctuations, et si les données Verbatim étaient en vraie casse, les données RAP étaient elles en minuscules. Les données des années 2006 et 2007

	Espagnol	Anglais
Données d'apprentissage parallèles		
Textes du parlement européen	36,6 M	35,3 M
Données d'apprentissage monolingues		
Textes du parlement espagnol	48,9 M	—
Textes du parlement britannique	—	55,1 M
Transcriptions acoustiques	777 k	1,5 M
Données de développement et d'évaluation		
Dev05	46,2 k	49,4 k
Eval05	18,9 k	18,9 k
Dev06	54,2 k	30,3 k
dont parlement espagnol	29,0 k	—
Eval06	61,3 k	30,5 k
dont parlement espagnol	31,0 k	—
Eval07	61,4 k	28,5 k
dont parlement espagnol	32,8 k	—

Tab. 2.1 – Données d'apprentissage, de développement et d'évaluation (en nombre de mots). Pour les données de développement et d'évaluation, les nombres indiqués correspondent aux nombres de mots dans la langue source et pour la condition Verbatim.

respectent les critères de casse et de ponctuation décrits à la section précédente. Toutefois, en ce qui concerne la condition RAP, les sorties des systèmes de reconnaissance de la parole ont été combinées par ROVER [Fiscus, 1997] en ignorant la casse et la ponctuation, qui ont été ensuite réintroduites par post-traitement.

Enfin, en 2007, les sites de recherche participant à l'évaluation de la reconnaissance automatique de la parole étaient invités à rendre disponibles leurs treillis de mots, de façon à explorer des interfaces dites « riches » entre la reconnaissance et la traduction de la parole.

2.4 Mesures automatiques de la qualité des traductions

Déterminer la qualité d'une traduction est un problème difficile et ouvert. Étant donnés un texte source et une traduction candidate, seule une personne bilingue, voire connaissant les intentions de l'auteur du texte source, peut véritablement juger de la qualité de la traduction candidate. Les critères de qualité peuvent être multiples et inclure par exemple des critères de correction grammaticale (*fluency*) et de fidélité au sens du texte (*adequacy*). L'ARPA (*Advanced Research Projects Agency*, agence pour les projets de recherche) y a ajouté un critère quantifiant l'information effectivement transmise (*informativeness*), critère déterminé à l'aide de questions à choix multiples [White, 1995]. D'autres critères peuvent intervenir selon la tâche considérée : aide à la traduction, traduction de pages web, surveillance, etc.

Ces critères de qualité constituent la vraie mesure de l'adéquation du système de traduction à la tâche visée, mais requièrent une coûteuse intervention humaine. Par ailleurs, toute évaluation subjective souffre des problèmes de non-reproductibilité et de variabilité inter-annotateur. C'est en particulier le cas des critères *fluency* et *adequacy* cités plus haut, dont l'évaluation sur une échelle absolue de 1 à 5 est longue et difficile [Callison-Burch et al., 2007].

C'est pourquoi plusieurs mesures automatiques ont été développées au fil des années. Leur objectif est d'être corrélées avec les scores que produirait une évaluation manuelle. Ceci est un problème difficile, car une même phrase peut être traduite de nombreuses façons possibles et également acceptables. Les mesures automatiques doivent autoriser les variations légitimes et pénaliser les erreurs [Babych and Hartley, 2004].

Les mesures présentées ci-dessous sont parmi les plus courantes dans la communauté de la traduction automatique, du moins en ce qui concerne les approches qui se fondent sur de grandes quantité de données. Elles nécessitent toutes une ou plusieurs traductions de *référence* pour chaque phrase source. On peut distinguer les mesures en deux classes : celles qui mesurent une distance entre la phrase candidate et les références, et celles qui mesurent au contraire une ressemblance.

2.4.1 Mesures reposant sur des taux de mots erronés

Score WER

Le score WER (*Word Error Rate*, taux de mots erronés) correspond à la distance d'édition ou distance Levenshtein. Lorsqu'une seule traduction de référence $\mathbf{e}_r$ est disponible, le score WER d'une traduction candidate $\mathbf{e}_c$ est calculé comme suit :

$$\mathrm{WER}(\mathbf{e}_c) = \frac{n_{\mathrm{ins}} + n_{\mathrm{sup}} + n_{\mathrm{sub}}}{|\mathbf{e}_r|} \tag{2.10}$$

où n_{ins}, n_{sup} et n_{sub} sont respectivement les nombres minimums d'insertions, de suppressions et de substitutions pour modifier $\mathbf{e}_c$ en $\mathbf{e}_r$, et $|\mathbf{e}_r|$ dénote la taille de $\mathbf{e}_r$ en nombre de mots.

Lorsque plusieurs références sont disponibles, le numérateur est remplacé par le plus petit nombre d'éditions obtenu en comparant la candidate à chacune des références, et le dénominateur devient (dans l'outil que nous utilisons) la moyenne des longueurs des références.

Score PER

Le score WER est utilisé dans beaucoup de domaines, comme par exemple la reconnaissance de la parole. Il est là particulièrement adapté du fait de la monotonicité entre le signal audio et les mots transcrits. Pour la traduction, en revanche, le score WER peut pénaliser injustement des traductions correctes si elles organisent les mots différemment des traductions de référence. Cette inflexibilité du score WER a motivé la création du score PER (*Position-independent word Error Rate*, taux de mots erronés indépendemment des positions).

Appelons s_c le « sac de mots » contenant les mots de la traduction candidate ; les mots apparaissant plusieurs fois dans $\mathbf{e}_c$ apparaissent exactement autant de fois dans s_c, et en particulier $|s_c| = |\mathbf{e}_c|$. De même, soit s_r le sac de mots de la traduction de référence $\mathbf{e}_r$. Le score PER de $\mathbf{e}_c$ est calculé comme suit :

$$\mathrm{PER}(\mathbf{e}_c) = \frac{\max\left(|s_c \setminus s_r|, |s_r \setminus s_c|\right)}{|\mathbf{e}_r|} \tag{2.11}$$

où $s_c \setminus s_r$ est le sac de mots s_c privé des mots également présents dans s_r. Par exemple, si s_c contient quatre fois le mot w et que s_r ne le contient que deux fois, $s_c \setminus s_r$ le contient encore deux fois et $s_r \setminus s_c$ ne le contient pas du tout.

Comme précédemment pour le score WER, lorsque l'on dispose de plusieurs références, on utilise le plus petit numérateur constaté sur l'ensemble des références, et la moyenne des longueurs des références au dénominateur.

Score TER

Le score TER (*Translation Edit Rate*, taux d'édition de la traduction) est une évolution du score WER qui ajoute l'opération de *décalage* aux trois opérations d'éditions déjà citées (insertion, suppression et substitution) [Shapira and Storer, 2007]. Un décalage permet de déplacer un groupe de mots contigus vers la gauche ou la droite ; tout décalage compte comme une seule édition quels que soient le nombre de mots déplacés et l'amplitude du déplacement. Le nombre de décalages est noté $n_{\text{déc}}$.

Le score TER se calcule de façon similaire au score WER :

$$\mathrm{WER}(\mathbf{e}_c) = \frac{n_{\text{ins}} + n_{\text{sup}} + n_{\text{sub}} + n_{\text{déc}}}{|\mathbf{e}_r|} \tag{2.12}$$

Le lecteur aura pressenti que lorsque plusieurs références sont disponibles, le score TER est déterminé par le nombre d'éditions entre la phrase candidate et la référence la plus proche, et par le nombre moyen de mots des références.

2.4.2 Mesures de ressemblances aux traductions de référence

Score Bleu

Le score Bleu (*Bilingual Evaluation Understudy*, littéralement doublure d'évaluation bilingue) mesure une ressemblance de la traduction candidate à une ou plusieurs traductions de référence [Papineni et al., 2002]. Pour le calculer, il est nécessaire d'introduire la notion de précision n-gramme modifiée p_n.

Considérons dans un premier temps le cas d'une phrase isolée. Si l'on note N est le nombre de mots de la traduction candidate $\mathbf{e}_c$, $\mathbf{e}_c$ contient N unigrammes (par forcément distincts), $N-1$ bigrammes et, plus généralement $N-n+1$ n-grammes. Pour un n donné et pour un n-gramme *n-gram* donné, soient $Compte(n\text{-}gram)$ le nombre de fois qu'il apparaît dans $\mathbf{e}_c$ et $Compte_{clip}(n\text{-}gram)$ le nombre maximal de fois que ce n-gramme apparaît dans une référence, parmi toutes les références disponibles, sans

toutefois dépasser $Compte(n\text{-}gram)$. On peut alors définir la précision n-gramme modifiée :

$$p_n = \frac{\sum_{n\text{-}gram \in \mathbf{e}_c} Compte_{clip}(n\text{-}gram)}{\sum_{n\text{-}gram \in \mathbf{e}_c} Compte(n\text{-}gram)} \tag{2.13}$$

Lorsque ces comptes s'appliquent à une phrase isolée, on remarque que le dénominateur vaut toujours $N-n+1$. Pour calculer la précision n-gramme modifiée sur un document entier, on accumule simplement les comptes pour chacune des phrases. On note encore cette quantité p_n.

Le score Bleu est une moyenne géométrique de ces précisions n-grammes, multipliée par une pénalité de brièveté. Cette dernière est destinée à pénaliser les systèmes qui essaieraient d'augmenter artificiellement leurs scores de précisions en produisant des phrases délibérément courtes. L'expression du score Bleu est ainsi :

$$\text{Bleu}_i(\mathbf{e}_c, \mathbf{e}_r) = \text{PB} \cdot \exp\left(\sum_{n=1}^{i} w_n \log p_n\right) \tag{2.14}$$

où w_n est constant et vaut $\frac{1}{i}$ et la pénalité de brièveté PB est calculée comme suit. Soient $c = |\mathbf{e}_c|$ la taille de la candidate et r la taille de la référence la plus proche c. Alors :

$$\text{PB} = \begin{cases} 1 & \text{si } c \geq r \\ e^{1-r/c} & \text{si } c < r \end{cases} \tag{2.15}$$

Dans la suite de cette thèse, lorsqu'un score Bleu est mentionné, il s'agit du score Bleu_4, qui utilise les précisions unigrammes jusqu'à quadrigrammes.

Bleu étant un score de précision, plus il est élevé, meilleure est la traduction. La bonne corrélation entre Bleu et l'évaluation humaine a été constatée a plusieurs reprises [Papineni et al., 2002, Coughlin, 2003, Doddington, 2002] et Bleu a gagné le statut de mesure automatique de référence au sein de la communauté de la traduction automatique. Cependant, Bleu est connu pour favoriser les systèmes utilisant des approches statistiques aux dépens notamment des systèmes à base de règles [Callison-Burch et al., 2006].

Autres mesures automatiques

Le score NIST [Doddington, 2002] reprend le principe du score Bleu et l'adapte légèrement. La modification la plus notable est que, dans le score NIST, les n-grammes sont pondérés par leur quantité d'information, notion déterminée par leur fréquence : les n-grammes rares contribuent plus au score final que les n-grammes fréquents. Par ailleurs, l'expression de la pénalité de brièveté est légèrement différente de celle de Bleu, et enfin le score NIST prend en compte les précisions unigrammes jusqu'à pentagrammes, alors que Bleu se contentait des quadrigrammes.

Le score METEOR [Banerjee and Lavie, 2005] introduit plusieurs concepts intéressants. Pour calculer ce score, il est nécessaire de calculer un alignement entre la

phrase candidate et la référence. Un algorithme à plusieurs passes est employé. À la première passe, seuls les mots identiques sont alignés. La deuxième passe tente d'aligner les mots non-alignés et utilise cette fois un module de *dérivation* (*stemming*) : à cette passe, `joli` et `jolie` pourront être alignés. La troisième passe emploie elle un module de synonymie reposant sur WordNet[12]. Cette passe pourra aligner par exemple `joli` à `beau`. Une fois l'alignement entre la candidate et la référence déterminé, les taux de précision et de rappel (tous deux unigrammes) sont calculés et combinés par moyenne harmonique. METEOR se distingue là à nouveau de Bleu et de NIST, qui ne considéraient que les taux de précisions. Enfin, METEOR intègre une pénalité dont le but est de favoriser une traduction qui a de longs segments consécutifs alignés avec la référence.

Le nombre important de mesures automatiques est significatif de la difficulté d'évaluer la qualité d'une traduction, même en présence d'une ou de plusieurs traductions de référence. Dans cette thèse, les performances sont présentées en terme de taux WER et PER et de scores Bleu et NIST.

[12]WordNet : `http://wordnet.princeton.edu/`.

Chapitre 3

Système de traduction à base de mots

3.1 Description générale

Le traducteur décrit dans ce chapitre est un décodeur pour le modèle de traduction IBM-4, avec classes de mots. Ce traducteur, dont le fonctionnement est détaillé à la section 3.2, a été intégralement réalisé au cours de cette thèse. Il est au cœur du système de traduction complet dont l'architecture est illustrée à la figure 3.1. Le modèle de traduction est entraîné à partir de corpus parallèles, tandis que les modèles de langage peuvent en plus exploiter des données monolingues ; l'entraînement des modèles est décrit à la section 3.3. Le traducteur supporte les modèles de langage bi-, tri- et quadrigrammes. Il est capable de générer un treillis de traduction correspondant à l'espace de recherche exploré, ce qui permet ensuite d'appliquer efficacement des modèles (de langage ou de traduction) plus complexes. Habituellement, le traducteur utilise un modèle de langage trigramme à repli pour générer le treillis de traduction. Ce treillis est alors réévalué à l'aide d'un modèle de langage neuronal quadrigramme afin de produire la traduction cible. Ce chapitre évalue en détail l'impact de plusieurs des caractéristiques spécifiques du traducteur à la section 3.4. Enfin, les résultats obtenus avec ce système pour la traduction entre l'espagnol et l'anglais sont présentés.

Ce système a participé à l'évaluation internationale TC-Star de 2006, la première participation du LIMSI à une campagne d'évaluation en traduction automatique. Les articles suivants reposent sur le système de traduction décrit dans ce chapitre : [Déchelotte et al., 2005], [Déchelotte et al., 2006b], [Déchelotte et al., 2006a] et [Schwenk et al., 2006].

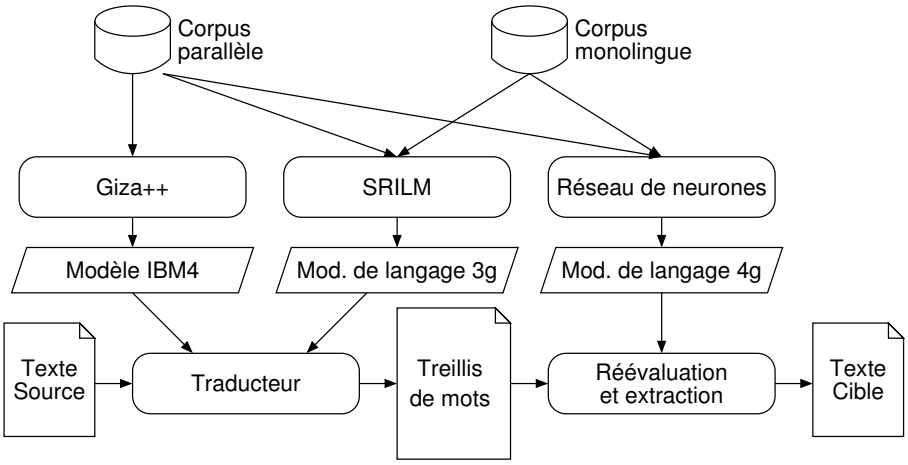


FIG. 3.1 – Entraînement et utilisation typique du traducteur à base de mots au sein d'un système de traduction complet

3.2 Moteur de traduction

Supposons que l'on souhaite traduire de $\mathbf{f}$ vers $\mathbf{e}$. L'expression que doit maximiser le traducteur est de la forme de l'équation 2.5, avec cinq fonctions caractéristiques :

$$\underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}|\mathbf{f}) \approx \underset{\mathbf{e}}{\operatorname{argmax}} \exp\left(\sum_{i=1}^{5} \lambda_i \log h_i(\mathbf{e}, \mathbf{f})\right) \tag{3.1}$$

où $h_1(\mathbf{e}, \mathbf{f}) = \Pr(\mathbf{e})$ est le modèle de langage cible et $h_2, \ldots, h_5$ les sous-modèles de fertilité, traduction lexicale, distorsion et de génération spontanée (section 2.1.3). Conformément à la relation de Bayes (équation 2.2), le modèle IBM-4 est utilisé dans le sens opposé au sens de traduction de « l'application » et le produit des fonctions h_2 à h_5 vaut $\Pr(\mathbf{f}, \mathcal{A}|\mathbf{e})$ (équation 2.8, page 20). Nous voulons traduire de $\mathbf{f}$ vers $\mathbf{e}$, et il s'agit pour cela de retrouver la phrase $\mathbf{e}$ qui, au sens d'IBM-4, a *généré* la phrase $\mathbf{f}$.

À cette fin, le traducteur va progressivement ajouter des mots cibles e_i qui vont chacun « expliquer », ou *couvrir*, un ou plusieurs mots f_j de la phrase source à traduire. L'objectif du décodeur est de couvrir toutes les positions source. Les positions source sont couvertes dans n'importe quel ordre, tandis que les mots cible sont ajoutés dans l'ordre de lecture, de gauche à droite.

Dans la suite du chapitre, le *coût* d'une décision est égal à l'opposé du logarithme de la probabilité de cette décision. À titre d'exemple, le coût de la traduction du mot e en f est $-\log(t(f|e))$.

3.2.1 Stratégie de recherche

Algorithme « A* »

La recherche de la meilleure traduction $\mathbf{e}^*$ parmi l'ensemble des phrases cible $\mathbf{e}$ est réalisée en suivant une stratégie de type « A* » avec élagage, inspirée par [Och et al.,

2001], [Wang and Waibel, 1997] et [Germann et al., 2001]. Le décodeur maintient une liste d'hypothèses partielles qui évolue comme suit :

1. La liste d'hypothèses partielles est initialisée avec une hypothèse vide (aucune position source couverte, aucun mot cible produit).
2. Le décodeur sélectionne la meilleure hypothèse partielle et l'enlève de la liste.
3. Si cette hypothèse a traduit tous les mots source, il s'agit de la traduction prédite par le modèle et l'algorithme est terminé.
4. Sinon, le décodeur étend l'hypothèse en couvrant une position source de plus. Ce faisant, il génère autant de nouvelles hypothèses partielles qu'il y a de mots non-traduits et qu'il y a de traductions possibles pour ces mots.
5. Les nouvelles hypothèses sont incorporées à la liste et triées par coût croissant (donc par probabilité décroissante), puis l'on revient à l'étape 2.

Attributs d'une hypothèse partielle

La structure de donnée d'une hypothèse contient juste l'information nécessaire pour l'étendre et de quoi reconstruire la phrase finale lorsque le décodage est terminé. Ainsi, elle ne contient que les attributs suivants :

1. un pointeur arrière sur l'hypothèse qu'elle étend ;
2. les $n-1$ derniers mots cible produits, pour un modèle de langage cible n-gramme ($n = 2$, 3 ou 4) ;
3. la dernière position source couverte ;
4. le nombre ϕ_{actuel} de mots source actuellement alignés au dernier mot cible produit ;
5. la somme des positions de ces mots source (nécessaire pour le calcul de $d_{=1}(\Delta j|\mathcal{E},\mathcal{F})$) ;
6. un drapeau (vrai/faux) indiquant si la dernière extension a couvert un mot en l'associant à NUL ;
7. le nombre $\phi_{0_{\text{actuel}}}$ de mots source déjà associés à NUL ;
8. la position source la plus à droite associée à NUL.

Les derniers mots cible produits (deuxième item de la liste) sont indispensables au modèle de langage. Les items 3 à 8 sont quant à eux nécessaires pour calculer les probabilités associées aux sous-modèles de fertilité, de distorsion et de génération spontanée.

Extension d'une hypothèse partielle

Étendre une hypothèse partielle signifie couvrir une position source supplémentaire. Ceci est généralement accompli en ajoutant un mot cible supplémentaire, mais peut éventuellement ne produire aucun mot cible, ou en produire plusieurs. Le décodeur dispose des cinq « opérateurs » suivants pour étendre une hypothèse :

– l'opérateur *Ajout* qui traduit un mot source supplémentaire en ajoutant un mot cible ;

– l'opérateur *NZFertAjout* qui ajoute un ou plusieurs mots cible de fertilité nulle (alignés à aucun mot source et donc ne couvrant aucune position source) avant d'ajouter un mot cible qui traduise un mot source supplémentaire ;
– l'opérateur *Extension* qui aligne un mot source supplémentaire au dernier mot cible produit : aucun mot cible n'est donc ajouté avec cet opérateur mais une position source est bien couverte ;
– l'opérateur *CompletementNul* est particulier : il complète l'hypothèse partielle en alignant toutes les positions source non-couvertes au mot cible NUL (aucun mot cible produit) ;
– l'opérateur *AlignementNul* qui aligne un mot source au mot spécial NUL. Comme avec l'opérateur *Extension*, aucun mot cible n'est ajouté mais une position source est couverte.

Les opérateurs *Ajout* et *NZFertAjout* peuvent être employés pour couvrir n'importe quelle position source non-couverte, au contraire de l'opérateur *Extension* qui ne peut couvrir qu'un mot source à droite du dernier mot source couvert. L'opérateur *CompletementNul* ne peut être utilisé que si au moins la moitié des positions source est déjà couverte (en effet, IBM-4 implique que $\phi_0 \leq \sum_{i=1}^{I} \phi_i = J - \phi_0$).

Les quatre premiers opérateurs (*Ajout*, *NZFertAjout*, *Extension* et *CompletementNul*) suffisent en théorie à envisager toutes les traductions admises par le modèle. Avec ces quatre opérateurs, les mots source qui sont le mieux expliqués en les alignant au mot spécial NUL restent non-traduits jusqu'à l'avant-dernière extension, et sont traduits tous ensemble à la dernière extension. Cependant, pour des raisons de vitesse d'exécution, des restrictions au réordonnement des mots (détaillées à la sous-section suivante) ont été introduites et ont nécessité l'ajout du cinquième opérateur, *AlignementNul*.

Limites de réordonnement

Le modèle IBM-4 n'impose pas de limite théorique au réordonnement des mots lors de la traduction : un mot e en début de phrase dans une langue peut produire un mot f en fin de phrase dans l'autre, et inversement. De plus, à la fin de l'entraînement avec Giza++, une masse de probabilité conséquente reste allouée à des « sauts » qu'IBM-4 ne permet pas de modéliser correctement. Par exemple, pour le premier couple de classes de mots $(\mathcal{E}, \mathcal{F})$ du fichier de distorsion créé par Giza++, la probabilité de placer un nouveau mot f 21 positions avant le dernier mot f placé est de 2,31 %, alors qu'elle n'est que de $8{,}98 \times 10^{-5}$ % pour un saut de seulement 4 positions en arrière.

Par ailleurs, lors de la traduction d'une phrase, autoriser des réordonnements arbitraires provoque une explosion combinatoire qui ralentit considérablement l'algorithme. C'est pourquoi deux seuils ont été introduits pour limiter cette explosion combinatoire tout en autorisant les réordonnements « légitimes ».

La première restriction sur les réordonnements est la suivante : à tout moment, parmi les mots source non-traduits, l'algorithme n'envisage de traduire que les n_1 mots non-traduits les plus à gauche, au lieu de considérer *a priori* tous les mots restant à traduire. Ainsi, dans la figure 3.2 où les mots en position 1 et 3 sont traduits, seules les mots en position 2, 4 et 5 peuvent être traduits à l'extension suivante. Cette restriction permet des réordonnements tout en limitant leur amplitude (dans la plupart de nos

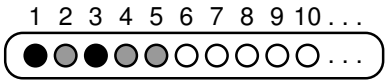


FIG. 3.2 – Première limite de réordonnement : seules les n_1 premières positions source non-traduites sont candidates pour être traduites (ici $n_1 = 3$). Les positions noires, grisées et blanches sont respectivement traduites, non-traduites candidates à la traduction, et non-traduites (et non-candidates).

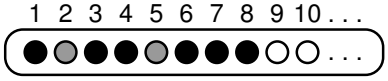


FIG. 3.3 – Seconde limite de réordonnement : seules les positions source non-traduites au plus n_2 positions à droite de la première position non-traduite sont candidates pour être traduites (ici $n_2 = 6$). Les positions noires, grisées et blanches sont respectivement traduites, non-traduites candidates à la traduction, et non-traduites (et non-candidates).

expériences, $n_1 = 3$).

La figure 3.3 illustre la seconde restriction sur les réordonnements : un mot source ne peut être traduit s'il est plus de n_2 positions à droite du premier mot source non-traduit. Ainsi, dans l'exemple de la figure 3.3, couvrir la position 9 n'est pas envisagé car elle est trop éloignée de la première position non-couverte, c'est-à-dire la position 2. Sans ce seuil, l'algorithme pouvait laisser, par exemple, deux mots source non-traduits en début de phrase et tenter de traduire le reste de la phrase de façon quasi-monotone — avec $n_1 = 3$, deux mots source non-traduits en début de phrase interdisent tout réordonnement pour la suite de la phrase. Cela pose deux problèmes :

– d'une part, le risque d'une traduction nettement sous-optimale puisque l'algorithme n'explore pas tous les réordonnements autorisés par le modèle ;
– d'autre part, en terme de temps de traduction, conserver deux mots non-traduits oblige à répéter les mêmes tests à chaque extension de traduction. Augmenter n_1 permet de limiter l'impact du problème précédent mais aggrave ce problème de temps de traduction.

L'introduction du seuil n_2 résout ces problèmes mais a nécessité l'ajout d'un cinquième opérateur, *AlignementNul*, qui étend une hypothèse partielle en alignant un mot source à NUL. Dans le cas de l'hypothèse partielle de la figure 3.3, l'opérateur *AlignementNul* pourrait être employé pour traduire le mot en position 2, et rendre ainsi « accessibles » les mots en positions 9 et 10 à l'extension suivante.

3.2.2 Organisation des hypothèses en files

Un décodeur de type « A* » peut regrouper les hypothèses partielles en une ou plusieurs files. N'utiliser qu'une seule file a certes l'avantage d'une gestion plus simple des hypothèses mais présente le risque de faire des erreurs de décodage, c'est-à-dire de produire une phrase $\mathbf{e}$ qui ne soit pas $\mathbf{e}^*$. En effet, l'optimalité de la recherche n'est garantie qu'avec une capacité de stockage et un temps de calcul infinis, or en pratique,

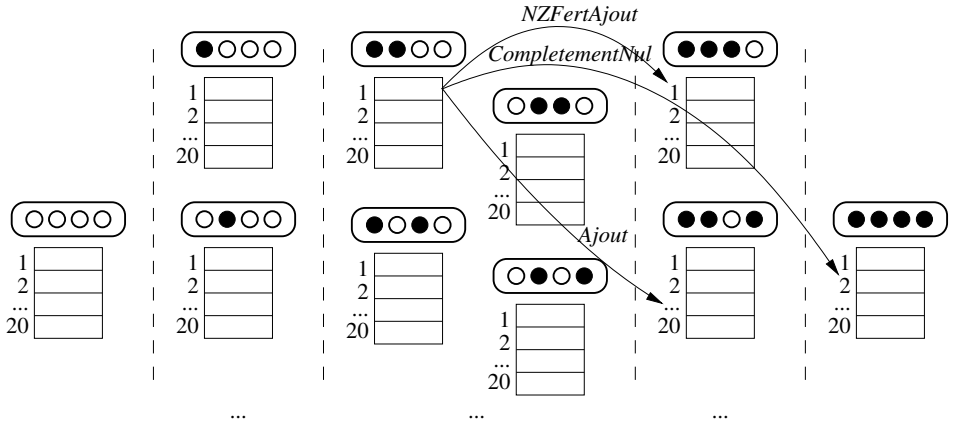


FIG. 3.4 – Les files d'hypothèses partielles dans le traducteur, pour une phrase source de $J = 4$ mots. La figure montre un exemple d'expansion de la meilleure hypothèse de la file `1100`. Les opérateurs *Ajout* et *NZFertAjout* ont été utilisés pour produire des hypothèses dans les files `1110` et `1101`. De plus, l'opérateur *CompletementNul* a pu être employé pour couvrir la totalité des positions source vacantes avec le mot spécial NUL, sans produire de « vrai » mot cible.

un élagage sur le nombre maximal d'hypothèses partielles traitées ou en attente est nécessaire. En mettant en compétition dans une même file des hypothèses qui ont traduit un nombre différent de mots, un décodeur à file unique prend le risque que de nombreuses hypothèses peu prometteuses n'ayant traduit que peu de mots finissent par provoquer l'élagage de bonnes hypothèses partielles qui ont déjà traduit de nombreux mots.

C'est pourquoi le traducteur décrit dans cet article maintient plusieurs files et plus exactement une par sous-ensemble de positions source, soit 2^J files, où J est le nombre de mots dans la phrase source, comme illustré à la figure 3.4. Ce nombre élevé de files (et augmentant de façon exponentielle avec le nombre de mots à traduire) permet de réduire leur taille et de ne mettre en compétition que des hypothèses partielles qui ont couvert exactement les mêmes positions source. Limiter la taille des files à une valeur finie revient à faire un élagage de l'espace de recherche, dont les effets sur le temps nécessaire à la traduction et sur sa qualité sont décrits à la section 3.4.3.

Les limites de réordonnement décrites auparavant (section 3.2.1) permettent d'éviter la croissance exponentielle du nombre de files ; leurs effets sur les temps de traduction et les performances sont discutés à la section 3.4.4.

Fusion des hypothèses « congruentes »

Il arrive que deux ou plusieurs hypothèses partielles d'une même file aient des attributs (comme détaillés à la sous-section 3.2.1) identiques. Ces hypothèses peuvent avoir produit des mots cible différent mais ne pourront plus être distinguées par les futures extensions. Elles sont alors fusionnées et seule celle de coût minimal sera susceptible d'être étendue dans la suite de l'algorithme.

3.2.3 Heuristiques

Disposer de plusieurs files d'hypothèses complique la recherche de la meilleure hypothèse partielle à étendre à chaque itération. Une possibilité consiste à choisir l'hypothèse de moindre coût parmi toutes les hypothèses partielles de toutes les files. De cette façon, une mauvaise hypothèse H_{mvs} n'ayant couvert qu'une position source sera étendue avant une bonne hypothèse ayant couvert de nombreuses positions. Toutefois, à l'inverse d'un décodeur à file unique, les hypothèses produites par H_{mvs} ne pourront provoquer l'élagage des bonnes hypothèses, puisque les multiples files assurent une comparaison juste entre les hypothèses. Cette gestion, qui n'utilise aucune heuristique d'estimation du coût futur, correspond à l'algorithme de Dijkstra [1959]. En pratique, elle conduit à un décodage lent et parfois sous-optimal, malgré les files multiples.

Une heuristique fournit une estimation du coût futur de complètement d'une hypothèse, et une heuristique *admissible* donne toujours un *minorant* de ce coût futur. L'optimalité d'une recherche de type « A* » n'étant assurée que si les heuristiques sont admissibles, une telle heuristique a été développée en s'inspirant de [Och et al., 2001].

L'heuristique retenue comprend deux composantes :
– la première ne dépend que des positions source couvertes. Cette composante est ainsi particulièrement adaptée à une organisation des hypothèses en files indicées par le sous-ensemble des positions source couvertes. En effet, toutes les hypothèses partielles d'une même file vont partager le même coût futur estimé, calculé une seule fois à la création de la file, et ce coût va permettre de mieux comparer les files d'hypothèses entre elles, en favorisant les files qui ont traduit plus de mots.
– la seconde composante, d'expression plus simple, est calculée pour chaque hypothèse partielle.

Heuristique ne dépendant que des positions source non-traduites

L'approche retenue consiste à évaluer un coût minimal de couverture de chaque position source, noté $h_{\text{opt}}(j)$. La composante de l'heuristique qui ne dépend que des positions source couvertes est ainsi simplement obtenu en sommant $h_{\text{opt}}(j)$ sur les positions non-couvertes.

Plusieurs composantes ont été successivement ajoutées à $h_{\text{opt}}(j)$. La première et la plus naturelle prend simplement en compte la probabilité de traduction $t(f|e)$. Cette première heuristique est notée $h^T(j)$:

$$h^T(j) = \min_e - \log t(f_j|e) = - \log \max_e t(f_j|e) \tag{3.2}$$

Il est possible d'incorporer à l'heuristique la loi de fertilité $n(\phi|e)$ lorsque le mot e qui produit f n'est pas e_0. La nouvelle heuristique est notée $h^{TF}(j)$:

$$h^{TF}(j) = - \log \max \left\{ t(f_j|e_0), \max_{e \neq e_0, \phi} t(f_j|e) \sqrt[\phi]{n(\phi|e)} \right\} \tag{3.3}$$

La racine $\phi^{\text{è}}$ est nécessaire pour ne pas comptabiliser plusieurs fois la probabilité $n(\phi|e)$ dans le cas où plusieurs mots source f_j sont alignés au même mot cible e.

Enfin, le modèle de distorsion est pris en compte en ajoutant le terme suivant :

$$h^D(j) = -\log \max \left\{ \max_{j',\mathcal{E}} d_{=1}\left(j|j',\mathcal{E},\mathcal{F}(j)\right), \max_{j>j'} d_{>1}\left(j|j',\mathcal{F}(j)\right) \right\} \tag{3.4}$$

Pour le calcul du coût minimum de distorsion $h^D(j)$, il semblerait possible intuitivement d'obtenir une expression plus précise en conditionnant sur les positions sources déjà couvertes. Malheureusement, la forme de la loi $d_{=1}$ (qui dépend de la différence $j-j'$, où j' est une moyenne d'indices de mots couverts), rend difficile sinon impossible d'éviter une opération de maximisation sur l'ensemble des positions source j' (équation 3.4). Ainsi, $h^D(j)$ ne dépend de la position j que par l'intermédiaire de la classe $\mathcal{F}(j)$ du $j^{\text{ème}}$ mot source.

L'heuristique finalement retenue vaut :

$$h_{\text{opt}}(j) = h^{TF}(j) + h^D(j) \tag{3.5}$$

et est calculée en début de décodage pour chacune des J positions source. Le gain de temps pour le décodage compense largement le temps passé à évaluer ces quantités. En revanche, faute de moyen efficace pour calculer pour chaque mot cible e son plus faible coût de modèle de langage $\max_{e',e''} \Pr(e|e'',e')$, même en se restreignant aux seuls mots e, e' et e'' susceptibles d'être recrutés pour traduire un mot de $\mathbf{f}$, celui-ci n'est pas inclus dans l'heuristique retenue.

Heuristique dépendant de l'hypothèse

Pour chaque hypothèse partielle, en plus du coût heuristique de couverture des mots non-traduits (décrit plus haut), deux termes supplémentaires spécifiques à l'hypothèse partielle sont calculés.

Lorsque l'on vient de couvrir un mot source f_j supplémentaire à l'alignant à un « vrai » mot cible e (pas NUL), les coûts exacts de distorsion et de traduction lexicale peuvent être calculés, mais pas le coût de fertilité. En effet, la fertilité du dernier mot cible n'est pas encore définitive : elle sera peut-être incrémentée à l'extension suivante. Cependant, un coût minimal de fertilité était inclus dans l'heuristique $h_{\text{opt}}(j)$ (équation 3.3). Parce que l'algorithme s'interrompt (pour des raisons de débugage) si une baisse de coût est détectée, il est nécessaire d'inclure un coût de fertilité minimal pour le mot cible en cours, calculé comme suit :

$$h_1(\phi_{\text{actuel}}, e) = -\log \max_{\phi \geq \phi_{\text{actuel}}} n(\phi|e)^{\frac{\phi_{\text{actuel}}}{\phi}} \tag{3.6}$$

Enfin, il est possible de calculer un minorant d'un coût relatif aux mots source alignés à NUL :

$$h_2(\phi_{0\text{actuel}}) = -\log \max_{\phi_0 \geq \phi_{0\text{actuel}}} \binom{J-\phi_0}{\phi_0} p_0^{J-2\phi_0}(1-p_0)^{\phi_0} \tag{3.7}$$

p_0 étant fixé dans nos expériences à 0,98, cette expression permet notamment de pénaliser les hypothèses partielles qui ont aligné de nombreux mots (comptés par $\phi_{0\text{actuel}}$) à NUL.

3.2.4 Caractéristiques spécifiques du traducteur

Production de treillis de mots

Le traducteur est capable de fonctionner dans trois modes distincts. Dans le mode « meilleure hypothèse », lorsque la fin de l'algorithme est détectée (troisième étape de l'algorithme présenté page 39), le traducteur part de l'hypothèse complète qu'il vient de dépiler et suit les pointeurs arrière vers les hypothèses partielles précédentes pour former une phrase cible, résultat de l'algorithme.

Dans le mode « espace de recherche » en revanche, le traducteur ne s'arrête plus à la troisième étape de son algorithme comme ci-dessus, mais lorsqu'il a étendu toutes les hypothèses partielles. Il écrit alors un treillis de mots qui représente l'espace de recherche de l'algorithme en partant des diverses hypothèses complètes et en remontant comme avant les pointeurs arrières. La figure 3.5 présente un exemple de treillis de mots issu de la traduction d'une petite phrase. Sur chaque arc du treillis, les cinq scores (scores lexical, de fertilité, de distorsion, de génération spontanée et du modèle de langage) sont accessibles, mais ne sont pas représentés sur la figure.

La possibilité de produire des treillis sera mise à profit de plusieurs façons :
– pour ajuster les cinq λ_i de l'équation 3.1 (section 3.4.1) ;
– pour traiter les phrases les plus longues (section 3.4.5) ;
– et pour permettre l'utilisation efficace de modèles de langage à portée élevée (section 3.4.6).

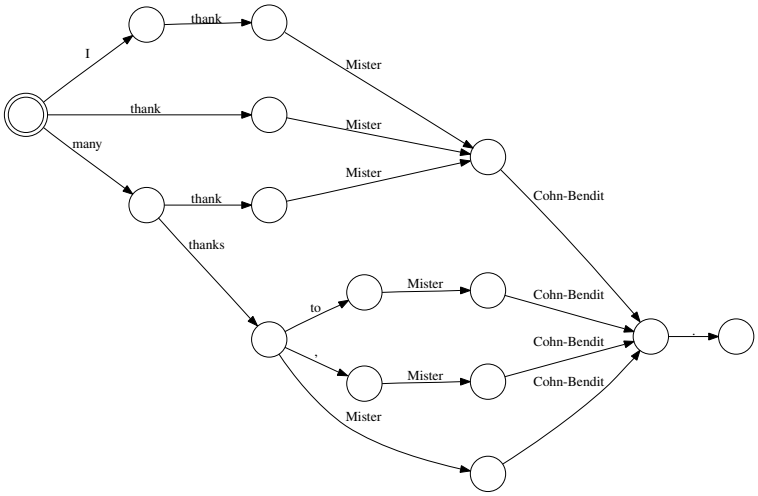
Alignement forcé

Le troisième mode du décodeur permet d'obtenir un « alignement forcé » entre une phrase source et une phrase cible, censée être une traduction potentielle de la phrase source. Dans ce mode, le décodeur cherche l'alignement qui, au sens du modèle IBM-4, explique le mieux la génération de la phrase source $\mathbf{f}$ par la traduction potentielle $\mathbf{e}$. Soit formellement :

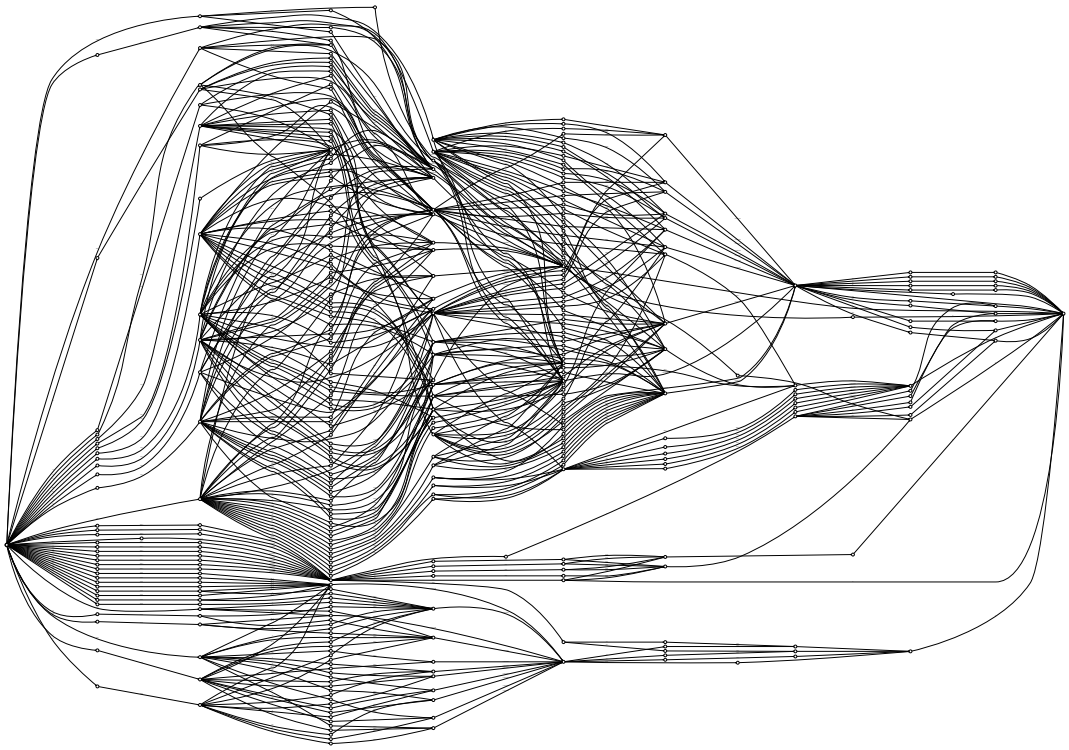
$$\begin{aligned} \operatorname*{argmax}_{\mathcal{A}} \Pr(\mathcal{A}|\mathbf{f},\mathbf{e}) &= \operatorname*{argmax}_{\mathcal{A}} \frac{\Pr(\mathbf{f},\mathcal{A}|\mathbf{e})}{\Pr(\mathbf{f}|\mathbf{e})} && (3.8)\\ &= \operatorname*{argmax}_{\mathcal{A}} \Pr(\mathbf{f},\mathcal{A}|\mathbf{e}) && (3.9) \end{aligned}$$

Le dénominateur peut être supprimé (équation 3.9) car $\mathbf{e}$ et $\mathbf{f}$ sont connus. L'équation 3.9 signifie que pour trouver le meilleur alignement entre $\mathbf{e}$ et $\mathbf{f}$, il suffit d'effectuer un décodage classique en mode « meilleure hypothèse » en s'assurant que la phrase cible décodée corresponde à la phrase cible imposée. C'est exactement comme cela que ce mode de fonctionnement est programmé. En cas de succès, le décodeur produit l'alignement $\mathcal{A}$ trouvé et les cinq scores qui composent la quantité $\Pr(\mathbf{f},\mathcal{A}|\mathbf{e})$. Il arrive que le modèle de traduction fourni au décodeur ne puisse pas expliquer la génération de $\mathbf{f}$ par $\mathbf{e}$; dans ce cas, la recherche échoue car $\Pr(\mathbf{f}|\mathbf{e}) = 0$.

L'utilisation du décodeur pour de l'alignement forcé sera envisagée au chapitre suivant, en vue d'une collaboration entre le traducteur à base de mots et celui par groupes de mots.



(a) Treillis de traduction fortement élagué



(b) Treillis de traduction complet

FIG. 3.5 – Treillis de traduction (élagué et complet) obtenus en traduisant la phrase « muchas gracias señor Cohn-Bendit . »

Influence sur la traduction des extrémités de la phrase

Il est possible de préciser deux mots cible qui, respectivement, précèdent et suivent la traduction attendue d'une phrase source. Ces mots n'ont d'influence sur la traduction que part l'intermédiaire du modèle de langage cible. Ainsi, si l'on indique au traducteur que la traduction à produire sera suivie du mot `abc`, le score du modèle de langage inclura un terme de la forme $\Pr(\texttt{abc}|e_{-2}, e_{-1})$, où e_{-2} et e_{-1} sont les deux derniers mots de la traduction générée. Si rien n'est précisé, le traducteur utilise par défaut les mots spéciaux `<s>` et `</s>`, qui sont les symboles conventionnellement attendus, respectivement en début et fin de phrase, par les modèles de langages créés avec la boîte à outils SRILM [Stolcke, 2002].

Influencer la traduction des extrémités des segments à traduire, combiné à la production de treillis de mots, permettra de traiter les phrases longues sans dégrader notablement la qualité de traduction, comme le montrent les expériences de la section 3.4.5.

3.3 Modèles de langage et de traduction

3.3.1 Entraînement du modèle de traduction

Les classes de mots nécessaires aux modèles de distorsion ont été apprises avec le programme `mkcls`[1] [Och, 1995], qui utilise un critère de maximum de vraisemblance pour déterminer des classes (monolingues) de mots. Cinquante classes ont été utilisées pour chacune des deux langues.

Le programme Giza++ réalise l'entraînement du modèle de traduction. Le processus d'entraînement commence avec cinq itérations d'entraînement du modèle IBM-1. À l'issue de cette phase, une table de traduction — quoique rudimentaire — a été apprise. Elle sert d'initialisation aux cinq itérations d'entraînement du modèle HMM, suivies de cinq itérations d'entraînement du modèle IBM-3. Cinq itérations d'entraînement du modèle IBM-4 concluent la séquence d'entraînement. Les résultats des étapes précédentes sont ignorés. Réaliser plus d'itérations pour chaque modèle, par exemple jusqu'à dix, augmentait de façon linéaire le temps d'entraînement tout en procurant un gain de performance négligeable.

3.3.2 Construction du modèle de langage

Un modèle de langage n-gramme [Jelinek, 1976] est nécessaire au traducteur. Celui-ci supporte les modèles bi-, tri- et quadrigrammes. Cependant, nos expériences ont montré qu'il était préférable, tant pour la qualité de traduction que vis à vis du temps de décodage, de produire un treillis de traduction avec un modèle trigramme. Il est ensuite toujours possible de réévaluer ce treillis avec un modèle à dépendance plus longue, par exemple un modèle quadrigramme. Ainsi, dans les expériences qui suivent, le traducteur utilise toujours un modèle de langage trigramme.

[1] mkcls : http ://www.fjoch.com/mkcls.html

	Anglais	Espagnol
Vocabulaire des modèles	82,6 k	132,5 k
Perplexités des modèles de langage interpolés :		
Modèle trigramme à repli	134,5	69,7
Modèle quadrigramme à repli	123,4	64,0
Modèle quadrigramme neuronal	102,8	54,6

TAB. 3.1 – Taille du vocabulaire retenu et perplexités obtenues sur les données de développement.

Tous les modèles à repli classique (« back-off ») ont été créés par la boîte à outils SRILM [Stolcke, 2002] et appliquent le lissage dit de Kneser-Ney modifié. Tous les n-grammes ont été conservés pour construire les modèles trigrammes, tandis que les quadrigrammes singletons (observés une seule fois) ont été supprimés pour construire les modèles quadrigrammes. Nous obtenons les perplexités les plus basses et, comme nous le verrons plus tard, les performances de traduction les meilleures, en réévaluant les treillis de traduction par un modèle quadrigramme neuronal. Celui-ci demeure un modèle n-gramme mais estime les probabilités dans un espace *continu*, dans lequel sont projetés les mots du vocabulaire. Le lecteur est invité à se reporter à [Schwenk, 2007] pour une description des spécificités du modèle de langage neuronal.

Les quelque 30 millions de mots du corpus TC-STAR paraissent bien peu en comparaison des quantités de données disponibles, par exemple, dans le domaine des « journaux télévisés » (Broadcast News). Pour donner un ordre de grandeur, deux corpus couramment utilisés pour la reconnaissance automatique de la parole sont le « GigaWord »[2], qui rassemble des dépêches en anglais d'agences de presse et compte 1,7 milliards de mots, et les « Google n-grams »[3], construits à partir de plus de mille milliards de mots extraits de pages web en anglais. Aussi avons-nous identifié les sources de données monolingues détaillées au tableau 2.1, page 31.

L'interpolation de modèles de langage permet de tirer parti de la couverture de modèles appris sur de grandes quantités de données, même « hors-domaine », tout en conservant une bonne spécialisation pour le domaine voulu, à savoir les discours parlementaires. L'interpolation fournit en règle générale de meilleurs résultats que l'entraînement d'un seul modèle sur l'ensemble des données disponibles. Pour chaque source de données, un modèle de langage est donc créé. Ces modèles sont ensuite interpolés linéairement pour constituer un modèle de la langue cible. Les coefficients d'interpolation sont estimés par un algorithme *Expectation-Maximization* de manière à minimiser la perplexité sur les données de développement. Dans nos expériences, ajouter des textes de type « journaux télévisés » en plus des données du tableau 2.1 permettait bien d'abaisser légèrement les perplexités mais tendait à dégrader les performances de traduction.

Le tableau 3.1 présente la taille des vocabulaires et la perplexité sur les données de développement Dev06 des différents modèles employés dans ce chapitre. Pour l'anglais comme pour l'espagnol, la perplexité baisse d'environ 8 % en passant d'un modèle tri-

[2] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05`

[3] `http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html`

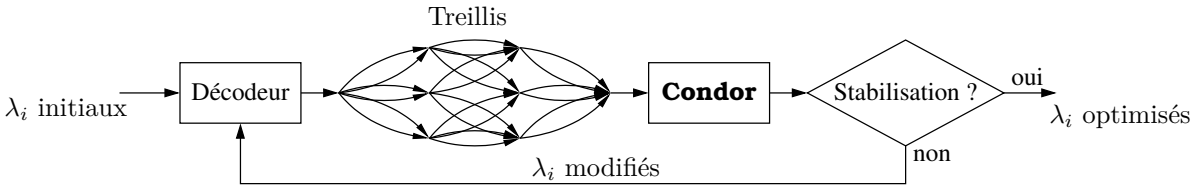


FIG. 3.6 – Utilisation des treillis de traduction pour l'ajustement des λ_i

gramme à un modèle quadrigramme. En passant du modèle quadrigramme à repli au modèle quadrigramme neuronal, on observe une baisse supplémentaire de la perplexité de l'ordre de 15 %. Nous vérifierons ensuite que ces baisses de perplexité produisent effectivement une amélioration de la traduction. Il est à noter que les données de développement en anglais, donc concernant la traduction de l'espagnol vers l'anglais, proviennent de deux sources différentes (parlements européen et espagnol). Ceci explique les perplexités pour l'anglais relativement élevées rapportées dans le tableau 3.1. Les perplexités sur les données du parlement européen seules sont plus basses : 85,0, 77,8 et 64,3 pour, respectivement, le trigramme, le quadrigramme à repli et le quadrigramme neuronal.

3.4 Expériences et résultats

3.4.1 Utilisation des treillis pour régler le système

Une première utilité de la production des treillis de traduction est de permettre l'ajustement efficace des cinq λ_i qui pondèrent les différents modèles de l'équation 3.1. Comme illustré à la figure 3.6, des treillis de traduction sont générés à partir d'un jeu initial de λ_i. L'optimiseur numérique Condor [Berghen and Bersini, 2005] propose alors plusieurs jeux de λ_i, avec lesquels les treillis sont réévalué, jusqu'à obtenir un optimum local. Dans nos expériences, l'ajustement des λ_i visait à maximiser le score BLEU obtenu sur l'ensemble de développement. Vient le test de stabilisation : si les λ_i après optimisation sont significativement différents des λ_i initiaux, il est nécessaire de regénérer des treillis avec ces nouveaux coefficients, et de relancer Condor. Pour ce genre de procédures, les treillis sont plus riches et donc plus efficaces que des listes de n meilleures hypothèses, qui contiennent généralement beaucoup de redondances.

Plus précisément, le cadre expérimental est le suivant :

1. Les cinq λ_i valent initialement tous 1.
2. Le traducteur traite l'ensemble du corpus de développement en mode « espace de recherche », c'est-à-dire qu'il produit un treillis de traduction pour chaque phrase source. Il utilise la valeur courante des λ_i et les contraintes de réordonnement $n_1 = 3$ et $n_2 = 6$ (section 3.2.1).
3. Un script est écrit pour extraire la meilleure traduction de chaque treillis étant donné un jeu de λ_i, et calculer le score BLEU correspondant. Grâce à ce script, Condor peut déterminer un jeu de λ_i qui maximise BLEU. Condor met en œuvre

(a) Résultats pour la traduction de l'anglais vers l'espagnol

	Dev06		Eval07	
	Avant	Après	Avant	Après
Bleu (%)	36,59	39,84	33,96	37,90
NIST	8,50	9,05	7.97	8,67
WER (%)	48,05	47,13	48,99	47,14
PER (%)	37,45	35,14	39,39	35,85

(b) Résultats pour la traduction de l'espagnol vers l'anglais

	Dev06		Eval07	
	Avant	Après	Avant	Après
Bleu (%)	30,24	37,83	34,59	39,17
NIST	7,98	9,26	8,50	9,32
WER (%)	53,42	48,08	50,28	47,43
PER (%)	41,16	34,47	36,72	32,92

Tab. 3.2 – Performances du traducteur à base de mots avant et après réglage des 5 paramètres λ_i, avec un modèle de langage trigramme. Tous les scores prennent en compte la casse et la ponctuation.

une extension de l'algorithme UOBYQA de Powell [2002]. Il lui faut généralement une centaine d'évaluations, soit environ 1 heure, pour trouver l'optimum.

4. Le jeu optimum est normalisé pour que la somme des cinq λ_i vaille 5 et ce jeu normalisé devient la valeur courante des λ_i. Si la nouvelle valeur d'un des λ_i diffère de plus de 0,01 de sa précédente valeur, l'on recommence à l'étape 2. Dans nos expériences, au plus cinq itérations de décodage étaient nécessaires.

Le tableau 3.2 présente les scores obtenus avec plusieurs mesures automatiques avant et après l'ajustement des cinq λ_i, pour les deux sens de traduction entre l'anglais et l'espagnol. Les λ_i ont été adaptés sur l'ensemble de développement uniquement.

Intéressons-nous tout d'abord au sens « anglais vers espagnol ». Le réglage des paramètres améliore tous les scores automatiques, et pas seulement le score Bleu. Le jeu de λ_i optimisés est relativement équilibré : respectivement 1,44, 0,96, 0,58, 1,22 et 0,80[4] pour les modèles de traduction lexicale, de fertilité, de distorsion, de génération spontanée et de langage. Il est remarquable que les gains sur les données de test soient plus importants que sur les données de développement : près de 4 points Bleu sur le test contre 3,25 sur le développement, et plus de 3,5 points PER sur le test contre seulement 2,3 sur le développement.

Dans le sens « espagnol vers anglais », le réglage apporte à nouveau des gains très significatifs pour tous les scores. Les λ_i trouvés par l'optimisation sont respectivement 2,63, 0,57, 0,45, 0,95 et 0,40, soit un jeu de poids très différent de celui avant optimisation (cinq fois 1,0). De plus, les gains sur le développement sont extrêmement importants : +7,5 points Bleu et −6,7 points PER, par exemple. Ceci fait craindre un

[4]Par souci de lisibilité, les λ_i ont été arrondis à la deuxième décimale.

sur-entraînement sur les données de développements. Pourtant, même s'il se produit peut-être un sur-entraînement partiel, les gains sur les données de test restent très appréciables, avec plus de 4,5 points Bleu et une diminution de 3,8 points PER.

Pour les deux sens de traduction, notons que l'optimisation des λ_i attribue un rôle prépondérant au modèle de traduction lexicale $t(f|e)$, tandis que le modèle de distorsion apparaît comme moins important. Ce constat n'est pas surprenant, puisque le modèle de traduction lexicale est intuitivement le modèle qui apporte le plus d'information sur, précisément, la traduction. En comparaison, le modèle de distorsion est pratiquement « aveugle », malgré ses dépendances sur des classes de mots. De plus, les deux langues considérées ici emploient un ordre de mots similaire, ce qui confère une importance limitée au modèle de distorsion. Quant au poids relativement faible accordé au modèle de langage, il s'explique peut-être par une dynamique différente des probabilités produites par le modèle de langage d'une part et par les quatre sous-modèles de traduction d'autre part.

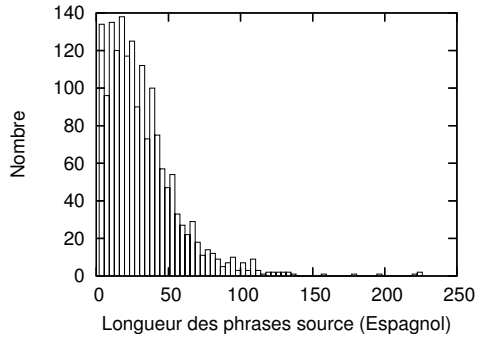
Les différences observées entre les deux sens de traduction s'expliquent par le fait que les rôles respectifs des modèles de traduction et de langage ne sont pas identiques dans les deux sens de traduction. L'espagnol est une langue fortement fléchie, alors que l'anglais l'est peu. Pour traduire de l'anglais vers l'espagnol, le modèle de traduction ne peut souvent que proposer les différentes formes des mots, et le modèle de langage a un grand rôle à jouer pour choisir la forme correcte. Pour traduire dans l'autre sens en revanche, le modèle de traduction est plus souvent en mesure d'indiquer la forme adéquate, ce qui laisse un rôle moindre au modèle de langage. Le poids de ce dernier n'est que de 0,40 pour la traduction vers l'anglais, contre 0,80 pour la traduction vers l'espagnol.

Pour finir, le réglage des λ_i apparaît comme crucial pour les performances des deux systèmes, et les gains réalisés sur les données de test sont similaires : environ 4 points Bleu, 0,7 point NIST, entre 2 et 3 points WER et environ 3,5 points PER.

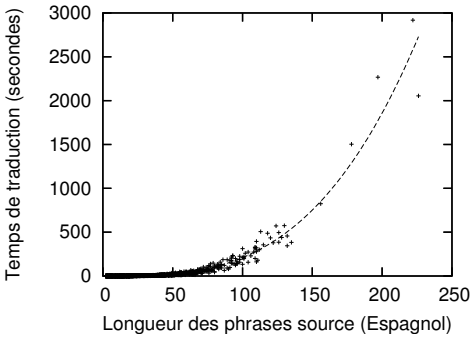
3.4.2 Influence de la longueur des phrases à traduire

Les expériences de cette section sont réalisées pour la traduction de l'espagnol vers l'anglais, avec les λ_i optimisés obtenus à la section précédente. La figure 3.7(a) montre l'histogramme des longueurs des phrases source de l'ensemble de développement. La longueur moyenne des 1712 phrases à traduire est d'un peu moins de 32 mots, même si 293 phrases (soit plus de 17 %) contiennent plus de 50 mots et 36 (soit 2,1 %) plus de 100 mots. La phrase la plus longue compte 226 mots.

Le nuage de points de la figure 3.7(b) représente, pour chaque phrase, le temps en secondes nécessaire à sa traduction, en fonction de sa longueur (en nombre de mots). Une fonction du type $g(x) = mx^n$ a été ajustée au nuage de points par minimisation de l'erreur au carré et a abouti à $n \approx 3{,}1$, c'est-à-dire que le temps de traduction croît de façon cubique — et même légèrement plus vite encore — avec la longueur de la phrase. Pour éviter des temps de traduction prohibitifs, il est nécessaire de régler l'élagage des files d'hypothèses (section suivante), de restreindre les réordonnements autorisés (section 3.4.4), voire de couper les phrases les plus longues avant la traduction



(a) Distribution des longueurs des phrases source



(b) Temps de traduction : nuage de points et régression de la forme $g(x) = mx^n$ avec $m \approx 1{,}1 \cdot 10^{-4}$ et $n \approx 3{,}1$

FIG. 3.7 – Longueur des phrases source et temps de traduction (traduction de l'espagnol vers l'anglais)

(section 3.4.5).

3.4.3 Influence de la taille des files d'hypothèses

Le traducteur organise les traductions partielles en files de taille finie (section 3.2.2). Ceci provoque un élagage de l'espace de recherche et est susceptible de provoquer des erreurs de recherche : la traduction produite peut ne pas être celle prédite par le modèle. Il est attendu qu'augmenter la taille de ces files améliore la qualité des traductions produites au détriment du temps de traduction.

Les expériences menées à ce sujet ont confirmé cette intuition ; leurs résultats sont résumés à la figure 3.8. L'axe des abscisses (représentant la taille des files) et l'axe des ordonnées de droite (pour le temps moyen de traduction, en secondes) sont représentés selon une échelle logarithmique. La relation entre le temps moyen de traduction et la taille des files est presque parfaitement linéaire : le calcul d'une régression de la forme $g(x) = mx^n$ aboutit à $m \approx 0{,}9$ et $n \approx 1{,}1$. La qualité de traduction quant à elle atteint un plateau à partir d'une taille de file de 50, avec un score BLEU de 37,96. Les expériences menées dans cette thèse utilisent en général une taille de file de 20, qui obtient le score BLEU de 37,83 déjà rapporté à la section 3.4.1. Réduire la taille des files en deçà de 10 dégrade sévèrement la qualité de traduction.

3.4.4 Influence des limitations de réordonnement

Calcul d'un majorant du nombre de files utilisées

Le traducteur répartit les traductions partielles en files, et crée une file par sous-ensemble de mots source traduits. Ainsi, pour une phrase source de longueur J, le traducteur est susceptible de créer 2^J files d'hypothèses (section 3.2.2). Cette section étudie les effets des restrictions de réordonnements décrites à la section 3.2.1.

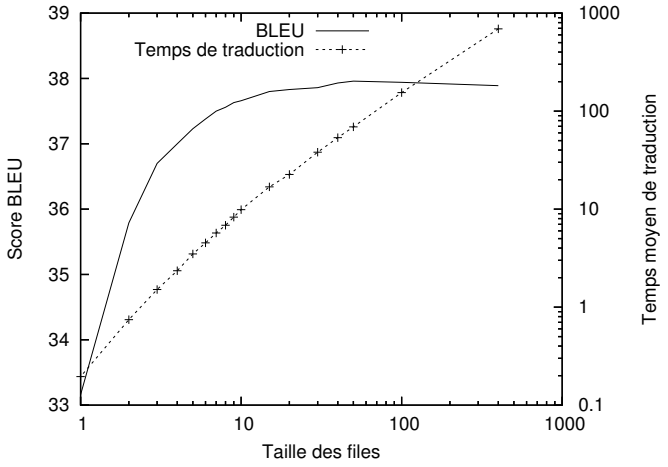


FIG. 3.8 – Influence de la taille des files d'hypothèses sur la qualité de la traduction (score BLEU, sur l'axe de gauche) et le temps de traduction (temps moyen de traduction, sur l'axe de droite)

Appelons $S(j)$ le nombre maximal de files qui peuvent être créées en respectant les contraintes de réordonnement et dont la première position source non-traduites est j. Soit S_{total} le nombre total maximal de files qui peuvent être créées sous la même contrainte. S_{total} s'obtient donc en sommant les $S(j)$ pour toutes les positions source j de 1 à J, et en y ajoutant 1 pour la file finale où tous les mots sont traduits, qui n'est comptabilisée dans aucun $S(j)$. On a ainsi :

$$S_{\text{total}} = 1 + \sum_{j=1}^{J} S(j) \tag{3.10}$$

Lorsqu'aucune contrainte de réordonnement n'intervient, il est possible de calculer exactement $S(j)$. En effet, $S(j)$ compte le nombre de files dont les $j-1$ premières positions sont traduites, la $j^{\text{è}}$ position est non-traduite et les $J-j$ dernières peuvent être traduites ou non, d'où :

$$S(j) = 2^{J-j} \tag{3.11}$$

et donc :

$$S_{\text{total}} = 1 + \sum_{j=1}^{J} 2^{J-j} = 1 + \sum_{j'=0}^{J-1} 2^{j'} = 2^J \tag{3.12}$$

On retrouve à l'équation 3.12 que sans restriction de réordonnement, 2^J files sont nécessaires.

Intéressons-nous dans un premier temps à l'effet de la contrainte « n_1 » (illustrée à la figure 3.2, page 41). Nous notons alors $S(j)$ et S_{total} respectivement $S_{n_1}(j)$ et S_{n_1}. Pour de grandes valeurs de n_1, plus précisément lorsque $n_1 \geq J$, cette contrainte de réordonnement est sans effet. $S_{n_1}(j)$ est alors égal à $S(j) = 2^{J-j}$.

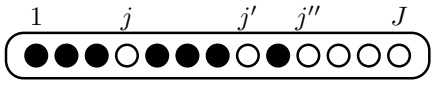


FIG. 3.9 – Exemple de file autorisée lorsque $n_1 = 3$. La trois premières positions non-traduites sont j, j' et j''.

Étudions maintenant le comportement de S_{n_1} pour de petites valeurs de n_1. Fixer $n_1 = 1$ impose au traducteur de couvrir à chaque fois le prochain mot source non-traduit, c'est-à-dire de produire une traduction monotone. Dans ce cas, $\forall j \in [1, J]$, $S_1(j) = 1$ et finalement :

$$S_1 = 1 + \sum_{j=1}^{J} 1 = J + 1 \tag{3.13}$$

Pour un n_1 quelconque, le calcul exact de $S_{n_1}(j)$ est complexe mais il est possible d'en donner une valeur approchée. Pour cela, remarquons que chaque $\{n_1 - 1\}$-uplet j', j'', …, $j^{(n_1-1)}$ de positions source entre $j + 1$ et J détermine une unique file, en considérant que les toutes les positions à gauche de $j^{(n_1-1)}$ sont traduites, sauf les positions j, j', …, $j^{(n_1-1)}$, comme à la figure 3.9. La file qui y est représentée respecte bien la contrainte de réordonnement : tous les mots traduits ont pu être couverts en ne laissant au maximum que $n_1 - 1 = 2$ mots non-traduits. En revanche, couvrir une position à droite de j'' est interdit par la contrainte de réordonnement. Ainsi, il y a au moins autant de files dont la première position non-traduite est j qu'il y a de façons de tirer $n_1 - 1$ entiers distincts entre $j + 1$ et J, soit[5] :

$$S_{n_1}(j) \geq \binom{n_1 - 1}{J - j} = \frac{(J - j) \ldots (J - j - n_1)}{(n_1 - 1)!} \tag{3.14}$$

Pour les cas usuels où $n_1 \ll J$ et tels que $j + n_1 < J$, l'équation 3.14 implique que :

$$S_{n_1}(j) = O\left((J - j)^{n_1 - 1}\right) \tag{3.15}$$

d'où l'on tire que :

$$S_{n_1} = O(1) + \sum_{j=1}^{J} O\left((J - j)^{n_1 - 1}\right) = O\left(J^{n_1}\right) \tag{3.16}$$

Ainsi, la restriction de réordonnement « n_1 » seule permet de contrôler la complexité de l'algorithme de traduction, estimée ici par le nombre de files d'hypothèses partielles susceptibles d'être créées, en la ramenant de $O(2^J)$ à $O(J^{n_1})$, avec un n_1 réglable (et dépendant *a priori* de la paire de langues).

Intéressons-nous à présent à l'effet de la contrainte « n_2 » (figure 3.3, page 41), sans la contrainte « n_1 ». $S(j)$ et S_{total} sont notées respectivement $S_{n_2}(j)$ et S_{n_2}.

[5] L'équation 3.14 est valable lorsque $j + n_1 < J$; on a vu plus haut que, dans le cas contraire, $S_{n_1}(j) = 2^{J-j}$.

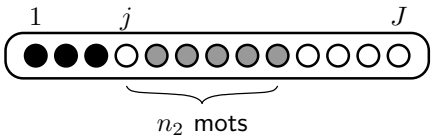


FIG. 3.10 – Files autorisées lorsque $n_2 = 6$.

Pour tout j tel que $j + n_2 \leq J$, comme à la figure 3.10, le nombre de files respectant la contrainte « n_2 » est $S_{n_2}(j) = 2^{n_2-1}$, qui est le nombre de combinaisons binaires « traduits/non-traduits » pour les $n_2 - 1$ mots suivant le mot j. Ainsi, pour les cas habituels où $n_2 \ll J$, on obtient comme ordre de grandeur pour S_{n_2} :

$$S_{n_2} = O(1) + \sum_{j=1}^{J} O(2^{n_2-1}) = O(J \cdot 2^{n_2}) \tag{3.17}$$

La contrainte « n_2 » utilisée seule permet donc de réduire le nombre de files susceptibles d'être créées de 2^J à un nombre de l'ordre de $J \cdot 2^{n_2}$.

Expériences

Cette sous-section présente dans le tableau 3.3 les performances obtenues par le traducteur en combinant les deux restrictions dites « n_1 » et « n_2 », ainsi que le temps moyen de traduction d'une phrase. La tâche considérée est la traduction de l'ensemble de développement dev06, de l'espagnol vers l'anglais. Nous nous sommes limités aux intervalles $[1, 5]$ pour n_1 et $[1, 7]$ pour n_2 en raison du plafonnement des performances et de la forte croissance des temps de traduction. Lorsque $n_1 > n_2$, la restriction « n_1 » n'impose aucune contrainte supplémentaire (tout se passe comme si $n_1 = n_2$).

On observe que le temps de traduction augmente régulièrement avec l'augmentation de n_2, et d'autant plus fortement que n_1 est élevé. Par exemple, lorsque $n_1 = 3$, chaque incrément de n_2 double le temps de traduction. Sur l'ensemble des expériences, les écarts sont très importants : le temps moyen de traduction est de l'ordre de 1 seconde lorsque seule la traduction des deux premiers mots non-traduits est autorisée ($n_1 = 2$), contre un temps moyen supérieur à 10 minutes lorsque l'on élargit la fenêtre de réordonnement ($n_1 = 5$ et $n_2 = 7$).

Concernant les performances de traduction, on constate qu'un plateau est atteint dans ces expériences lorsque $n_1 \geq 3$ et $n_2 \geq 3$. Ce résultat dépend *a priori* des langues considérées et du sens de traduction, ici de l'espagnol vers l'anglais. Le maximum absolu est atteint pour $n_1 = 3$ et $n_2 = 6$, mais l'écart de performances n'est évidemment pas significatif statistiquement. Pour des valeurs plus élevées de n_1 et de n_2, on observe une très légère dégradation des résultats. Cette dégradation, elle aussi non-significative, s'explique peut-être par le fait que le modèle de distorsion ne modélise correctement que les « sauts » d'amplitude réduite. Il est possible que le modèle attribue accidentellement une probabilité élevée à une traduction « farfelue » qui effectue de longs réordonnements. Dans ce cas, limiter les réordonnements avec n_1 et n_2 aide à limiter

n_2 \ n_1	1	2	3	4	5
7	**32,83** *(0,20)*	**37,58** *(2,5)*	**37,83** *(41)*	**37,64** *(251)*	**37,65** *(700)*
6	**32,83** *(0,18)*	**37,57** *(2,0)*	**37,83** *(23)*	**37,69** *(105)*	**37,63** *(234)*
5	**32,83** *(0,18)*	**37,58** *(1,6)*	**37,82** *(12)*	**37,72** *(39)*	**37,69** *(63)*
4	**32,83** *(0,18)*	**37,57** *(1,3)*	**37,81** *(5,5)*	**37,67** *(12)*	
3	**32,83** *(0,17)*	**37,57** *(1,0)*	**37,82** *(2,7)*		
2	**32,83** *(0,17)*	**37,50** *(0,76)*			
1	**32,83** *(0,17)*				

TAB. 3.3 – Influence des contraintes de réordonnement décrites à la page 41) sur le score BLEU (en gras) et le temps moyen de traduction pour une phrase (en secondes, entre parenthèses). La cellule grisée ($n_1 = 3$, $n_2 = 6$) correspond aux paramètres utilisés dans le reste des expériences.

l'espace de recherche à un domaine « raisonnable » pour lequel les lois apprises pendant l'entraînement sont fiables.

Dans ces expériences, le meilleur compromis entre temps de traduction et performance est clairement obtenu pour $n_1 = n_2 = 3$. Cependant, le réglage utilisé pour les deux sens de traduction dans les expériences de ce chapitre correspond à $n_1 = 3$ et $n_2 = 6$, qui est le réglage « historique » du système.

3.4.5 Traitement des longues phrases

Il a été observé plus haut que le temps de traduction augmentait exponentiellement avec la longueur de la phrase, ce qui amène à élaguer et restreindre la recherche dans le but de conserver un temps de traduction raisonnable. Dans cette section, les phrases les plus longues (notion déterminée par un seuil S, en nombre de mots) sont découpées en plusieurs « morceaux de phrases » et traduites, avant que la traduction complète ne soit reconstituée. La possibilité (décrite à la section 3.2.4) d'influer sur la traduction des extrémités de ces morceaux est ici utilisée.

Au début du développement du traducteur, il ne mettait pas en œuvre les restrictions de réordonnements décrites à la sous-section 3.2.1. Il était alors indispensable de couper les phrases les plus longues avant de les traduire. Influer sur la traduction des extrémités des segments était bien meilleur que simplement concaténer les traductions de chaque

segment [Déchelotte et al., 2006b].

Maintenant que le traducteur peut traduire des phrases de longueur arbitraire, traiter les phrases les plus longues comme décrit plus bas s'avère être un très bon compromis temps/performance.

Critère de coupure

Il peut arriver que le segment à traduire contienne plusieurs phrases au sens grammatical. Si le segment compte plus de S mots, il est segmenté en phrases, aux ponctuations usuelles (point, points d'interrogation et d'exclamation, etc). Les phrases ainsi obtenues qui dépassent encore le seuil S sont segmentées aux autres signes de ponctuation, en particulier *avant* les parenthèses ouvrantes et *après* les parenthèses fermantes et les virgules. Cette étape peut générer de nombreux fragments ; ceux-ci sont refusionnés tant que leur longueur totale ne dépasse pas le seuil. Si un fragment reste plus long que le seuil retenu, il est segmenté de façon uniforme en le plus petit nombre de segments qui permette de satisfaire au critère de longueur.

Les critères de coupure énoncés ci-dessus reposent sur l'hypothèse que des mots à gauche d'une ponctuation (une virgule, par exemple) peuvent être traduits indépendamment des mots à droite de la ponctuation, ou du moins que leur traduction se trouvera à gauche de la traduction de la suite de la phrase. Notons que le reste de la procédure n'est pas spécifique à ces critères : d'autres critères, comme ceux décrits par Berger et al. [1996], pourraient être essayés de façon interchangeable.

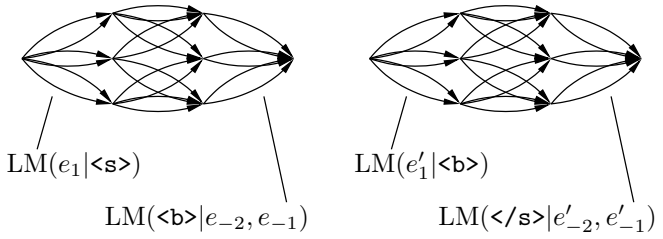
Mise en œuvre

Les phrases qui n'ont pas été coupées sont traduites comme dans les sections précédentes. Pour celles qui ont du être coupées en revanche, la traduction s'effectue comme illustrée par la figure 3.11 :

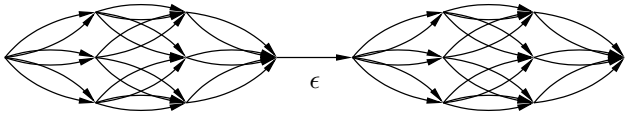
1. Il est demandé au traducteur de produire des treillis et d'utiliser, le cas échéant, le symbole spécial `<b>` (pour *break*) au lieu des mots spéciaux de début et de fin de phrase que sont `<s>` et `</s>`. Ainsi, dans le treillis de gauche de la sous-figure 3.11(a), les arcs arrivant au nœud final n'ajoutent pas de mots cible mais portent des probabilités du modèle de langage de la forme $\mathrm{LM}(\texttt{<b>}|e_{-2}, e_{-1})$. Réciproquement, dans le treillis de droite, les arcs issus du nœud initial portent des probabilités de la forme $\mathrm{LM}(e'_1|\texttt{<b>})$. Un modèle de langage spécial (décrit ci-après) doit donc être utilisé pour cette phase.

2. Les treillis de traduction sont ensuite fusionnés à l'aide d'une transition instantanée ϵ, comme à la sous-figure 3.11(b).

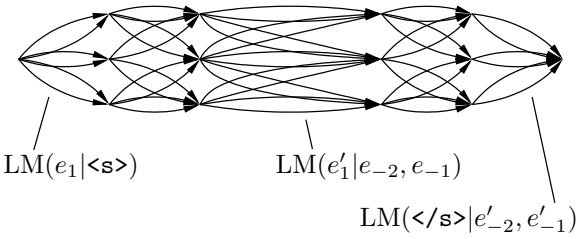
3. Le treillis ainsi constitué est enfin réévalué à l'aide d'un modèle de langage normal, comme le montre la sous-figure 3.11(c).



(a) Traductions indépendantes des deux fragments de phrase



(b) Fusion des deux treillis de traduction par une transition instantanée n'ajoutant pas de mot cible.



(c) Expansion (ajout d'arcs) du treillis par un modèle de langage n-gramme.

FIG. 3.11 – Traduction d'une phrase en utilisant la possibilité d'influer sur la traduction des extrémités des phrases. La phrase est ici segmentée en deux fragments, qui sont traduits indépendamment et dont les treillis de traduction sont ensuite fusionnés.

	Bleu (%)	Temps de traduction (s)
Phrases entières	37,83	23
Coupure à 80 mots	37,84	12
Coupure à 60 mots	37,83	7,8
Coupure à 50 mots	37,81	5,8
Coupure à 40 mots	37,86	4,0
Coupure à 30 mots	37,81	2,6
Coupure à 20 mots	37,64	1,6
Coupure à 16 mots	37,70	1,4
Coupure à 12 mots	37,71	1,2
Coupure à 8 mots	37,23	1,0

Tab. 3.4 – Performance (% Bleu) et temps moyen de traduction (en secondes) pour la traduction de l'espagnol vers l'anglais

Constitution d'un modèle de langage spécial

Un modèle de langage « spécial », qui connaisse le mot `<b>`, doit être construit. Pour cela, les données (dans la langue cible) servant à l'entraînement du modèle « normal » sont traitées comme le sont les phrases à traduire (dans la langue source). Toutefois, au lieu de couper les phrases les plus longues, deux mots[6] `<b>` sont insérés à l'endroit où la coupure aurait eu lieu. Le modèle spécial est alors obtenu par la même interpolation de modèles que le modèle normal (section 3.3.2).

Expériences

Plusieurs seuils de coupure ont été essayés et comparés avec les performances obtenues en traduisant les phrases entières (tableau 3.4). Le temps de fusion des treillis est négligeable et n'est pas pris en compte dans les temps présentés.

Comme attendu, couper les phrases les plus longues accélère fortement leur traduction, puisque cela permet d'éviter la partie « coûteuse » de la croissance cubique du temps de traduction avec la longueur de la phrase (section 3.4.2). Plus surprenant, la qualité de traduction est pratiquement inchangée que l'on coupe les phrases en segments de 30 mots maximum ou qu'on ne les coupe pas (les variations de 0,03 % Bleu ne sont pas significatives). On observe une légère baisse de performance en choisissant un seuil entre 12 et 20 mots, et la dégradation n'est plus sévère que pour un seuil — intuitivement très bas — de 8 mots.

[6] Ce nombre correspond à la taille de l'historique d'un modèle de langage trigramme.

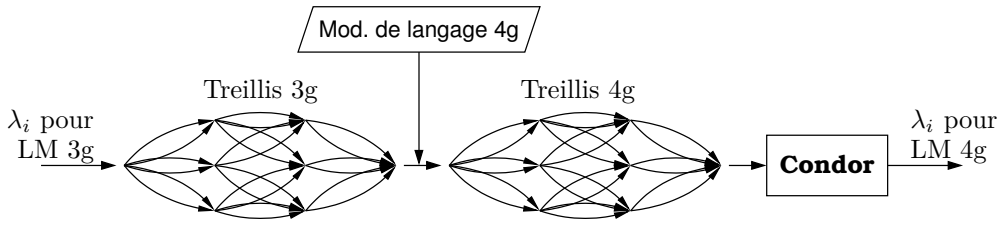


FIG. 3.12 – Ajustement des λ_i de seconde passe

3.4.6 Utilisation de meilleurs modèles de langage

Un autre avantage des treillis de traduction est qu'ils se prêtent particulièrement bien à l'utilisation de meilleurs modèles de langage. Les mêmes outils que ceux existants pour la reconnaissance de la parole purent être employés pour réévaluer les treillis de mots produits par le traducteur. Dans cette dernière section, nous réévaluons les treillis de traduction avec deux types de modèles quadrigrammes : un modèle à repli « classique » et un modèle dit « neuronal ». Tous les modèles de langage sont appris sur les mêmes données, comme décrit à la section 3.3.2.

Le modèle de langage est l'une des cinq fonctions caractéristiques du traducteur : changer de modèle de langage nécessite de déterminer un autre jeu de λ_i (équation 3.1). La figure 3.12 illustre comment le nouveau jeu de λ_i est calculé.

1. Pour chaque phrase source de l'ensemble de développement, le traducteur produit un treillis de traduction, noté « treillis 3g » dans la figure.
2. Ce treillis est alors réévalué par un des deux modèles quadrigrammes. Cette étape peut nécessiter une expansion du treillis et produit le « treillis 4g » de la figure. Même si le cadre mathématique de l'équation 2.5 autorise l'ajout du modèle de langage quadrigramme à la liste des fonctions caractéristiques, nous avons choisi de *remplacer* la probabilité du modèle trigramme par celle du modèle quadrigramme, *sans augmenter* le nombre de fonctions caractéristiques.
3. L'optimisation numérique des λ_i est confiée à Condor et a pour but la maximisation du score BLEU sur l'ensemble de développement, comme lors de l'optimisation des λ_i utilisés pour la production de treillis (figure 3.6, page 49). Toutefois, un seul lancement de Condor suffit ici, puisque nous avons choisi de générer les treillis de traduction une fois pour toute avec le jeu de λ_i déterminé pour le modèle de langage trigramme. Ceci peut être sous-optimal mais beaucoup plus rapide que l'optimisation jointe des deux jeux de λ_i.

Les performances des systèmes de traduction décrits dans ce chapitre sont détaillées ci-dessous. Le processus de traduction est le suivant. Toutes les phrases sont traduites par le décodeur en mode « espace de recherche » avec le premier jeu de λ_i. Ceci est appelé la « première passe » et aboutit à la création de treillis de traduction. Les phrases comptant plus de $S = 40$ mots sont coupées avant d'être traduites et leurs treillis de traduction sont fusionnés après, comme décrit à la section précédente.

Une traduction notée « ML 3g » est obtenue en réévaluant ces treillis avec le modèle trigramme et les mêmes λ_i. À cause du traitement des longues phrases, il est normal que

(a) Résultats pour la traduction de l'anglais vers l'espagnol

	Dev06			Eval07		
	ML 3g	ML 4g	MLN 4g	ML 3g	ML 4g	MLN 4g
Bleu (%)	39,82	40,58	41,41	37,96	38,34	39,52
NIST	9,05	9,10	9,24	8,68	8,69	8,88
WER (%)	47,13	46,72	45,54	46,98	46,95	45,58
PER (%)	35,02	34,96	34,24	35,84	35,94	35,03

(b) Résultats pour la traduction de l'espagnol vers l'anglais

	Dev06			Eval07		
	ML 3g	ML 4g	MLN 4g	ML 3g	ML 4g	MLN 4g
Bleu (%)	37,86	38,36	39,04	39,31	39,48	40,39
NIST	9,28	9,36	9,40	9,34	9,37	9,47
WER (%)	47,89	47,43	47,30	47,16	46,93	46,37
PER (%)	34,38	34,06	34,13	32,77	32,73	32,32

Tab. 3.5 – Performances du traducteur à base de mots sur la traduction de débats parlementaires (TC-Star) entre l'anglais et l'espagnol.

les résultats soient légèrement différents de ceux présentés au tableau 3.2. Une autre traduction notée « ML 4g » est obtenue avec le modèle quadrigramme à repli et « son » jeu de λ_i, déterminé sur l'ensemble de développement. Enfin, le modèle quadrigramme neuronal, utilisé lui aussi avec son jeu de λ_i spécifique, permet d'obtenir la traduction notée « MLN 4g ». Trois résultats (ML 3g, ML 4g et MLN 4g) sont donc présentés par direction de traduction.

Les tableaux 3.5(a) et 3.5(b) présentent les performances du traducteur à base de mots, respectivement pour la traduction vers l'espagnol et vers l'anglais, sur l'ensemble de développement dev06 et l'ensemble d'évaluation eval07 (voir le tableau 2.1 pour les caractéristiques de ces données).

Intéressons-nous tout d'abord au sens espagnol vers anglais (tableau 3.5(b)). Par rapport au modèle trigramme, le modèle quadrigramme à repli améliore les performances sur les données de développement d'environ un demi point, tant pour le score Bleu que pour le taux d'erreurs WER. Cependant, les gains sont plus que divisés par deux sur les données d'évaluation : le score Bleu est accru de moins de 0,2 point, et le WER n'est diminué que 0,23 point. Au contraire, le modèle quadrigramme neuronal conserve sur les données d'évaluation la quasi-totalité de ce qu'il faisait gagner sur les données de développement. Ainsi, le modèle neuronal dépasse le modèle trigramme de près 1,2 points Bleu sur dev06, et de 1,1 points sur eval07. En terme de WER, le modèle neuronal ne gagnait « que » 0,6 point sur les données de développement mais parvient à gagner près de 0,8 point sur eval07.

Pour le sens anglais vers espagnol, les gains obtenus par les deux modèles de langage quadrigramme sont en général plus importants que dans l'autre sens de traduction. Ainsi, sur l'ensemble de développement, le modèle quadrigramme à repli augmente le

score BLEU de 0,8 point et diminue le WER de près de 0,4 point. Mais à nouveau, les gains sont divisés par deux (seulement +0,4 en BLEU) voire négligeables (NIST, WER) sur les données d'évaluation. Comme dans l'autre sens de traduction, le modèle neuronal ne souffre pas de ce problème de généralisation à l'ensemble d'évaluation. Les gains obtenus sur l'ensemble de développement se retrouvent presque exactement sur l'ensemble d'évaluation, à savoir 1,6 points BLEU, 1,4 points WER et 0,8 point PER.

En conclusion, employer des modèles de langages quadrigrammes améliore substantiellement les résultats dans les deux sens de traduction. Le sens de l'anglais vers l'espagnol bénéficie particulièrement des modèles de langage d'ordre supérieur. Cela est sans doute dû à la richesse morphologique de l'espagnol, pour lequel le modèle de langage joue un rôle crucial dans la sélection de la traduction correcte. Il est remarquable par ailleurs que le modèle de langage neuronal obtienne dans tous les cas des performances supérieures au modèle quadrigramme à repli. De plus, les gains obtenus sur l'ensemble de développement se retrouvent presque entièrement sur l'ensemble d'évaluation.

Chapitre 4

Système de traduction par groupes de mots

4.1 Description générale

Le système de traduction décrit dans ce chapitre repose sur Moses [Koehn et al., 2007], un traducteur libre pour un modèle de traduction par groupes de mots. La figure 4.1 illustre l'architecture du système de traduction. À partir d'un texte source, le traducteur emploie un modèle de langage trigramme à repli et une table de traduction pour générer une liste de n meilleures traductions. Celles-ci sont ensuite réévaluées à l'aide d'un modèle de langage neuronal quadrigramme afin de sélectionner la traduction cible. Cette architecture est volontairement similaire à celle du système à base de mots (voir en particulier la figure 3.1, page 38). Les deux systèmes exploitent les mêmes données pour entraîner leurs modèles et mettent tous les deux en œuvre une stratégie à deux passes. Ce chapitre présente la liste des fonctions caractéristiques du modèle de traduction, puis compare la stratégie de recherche employée par Moses à celle du traducteur du chapitre précédent. La procédure d'extraction de la table de traduction est ensuite décrite. Nous présentons alors les performances du système après chacune des deux passes et montrons l'importance du choix du modèle de langage cible. Enfin, nous décrivons des expériences visant à intégrer des fonctions caractéristiques supplémentaires, notamment en exploitant le décodeur à base de mots conçu précédemment.

Les articles suivants reposent sur le système de traduction décrit dans ce chapitre : [Schwenk et al., 2007], [Bonneau-Maynard et al., 2007], [Déchelotte et al., 2007a] et [Déchelotte et al., 2007b].

4.2 Modèle de traduction

Supposons qu'il faille trouver la meilleure traduction d'une phrase source $\mathbf{f}$. Le traducteur Moses cherche la phrase cible $\mathbf{e}$ qui maximise une combinaison log-linéaire de fonctions caractéristiques, de la forme de l'équation 2.6. Les fonctions caractéristiques utilisées dans ce système sont les suivantes :

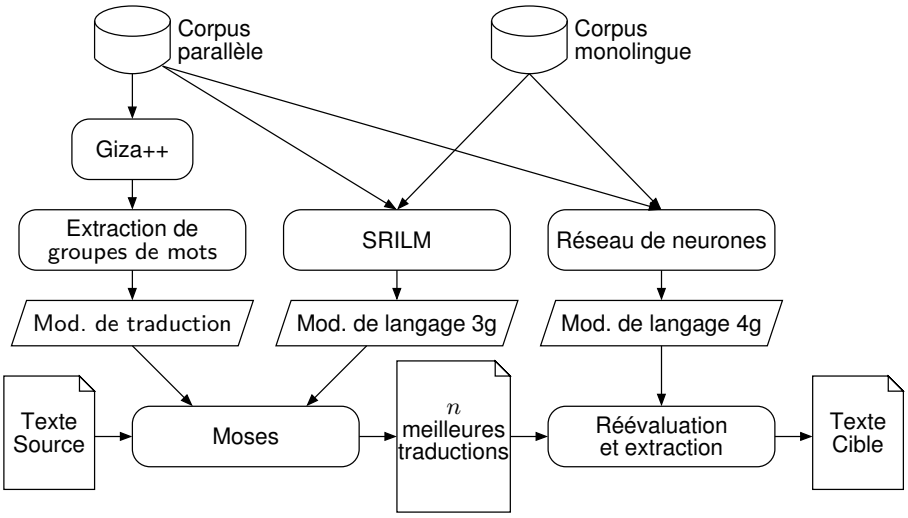


FIG. 4.1 – Entraînement et utilisation typique de Moses, traducteur par groupes de mots (*phrases*) au sein d'un système de traduction complet.

– Les m scores que la table de traduction attribue à tout couple de groupes de mots $\langle \tilde{e}, \tilde{f} \rangle$. Ces scores sont notés $h_i(\tilde{e}, \tilde{f})$, pour $i \in [1, m]$. Dans nos expériences, $m = 5$. Le détail du calcul de ces scores est présenté à la section 4.4.2.
– Le score d'un modèle de langage. Nos expériences emploient un modèle trigramme à repli.
– Le score du modèle de distorsion. Le modèle retenu est celui décrit à la section 2.1.4, de la forme $d(a_k - b_{k-1}) = \alpha^{|a_k - b_{k-1} - 1|}$.
– L'exponentielle du nombre de mots cible générés. Cette « fonction caractéristique », appelée pénalité de mot, permet simplement de contrebalancer la tendance du système à préférer les phrases courtes.

Finalement, l'expression que doit maximiser Moses est la suivante :

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \sum_{i=1}^{m} \lambda_i \sum_k \log h_i(\tilde{e}_k, \tilde{f}_k) + \lambda_D \log \left(\prod_k d(a_k - b_{k-1}) \right) + \lambda_{ML} \log \Pr(\mathbf{e}) + \lambda_{PM} |\mathbf{e}| \quad (4.1)$$

où $\lambda_1, \ldots, \lambda_m$ pondèrent les scores de la table de traduction, λ_D pondère le modèle de distorsion, λ_{ML} le modèle de langage et λ_{PM} la pénalité de mot. L'opération argmax de l'équation 4.1 porte explicitement sur l'ensemble des phrases cible $\mathbf{e}$ mais aussi implicitement sur les variables cachées nécessaires au décodage, à savoir la segmentation de $\mathbf{f}$ en groupes de mots et leur traduction.

4.3 Moteur de traduction

Moses est un décodeur heuristique pour le modèle par groupes de mots décrit ci-dessus. Nous avons choisi d'utiliser Moses car c'est un logiciel libre[1] et à l'état de l'art en matière de traduction par groupes de mots [Shen et al., 2006]. Son algorithme est très proche de celui de Pharaoh, décrit en détail par Koehn [2004]. Cette section en reprend les points principaux et souligne les ressemblances et les différences avec le décodeur pour le modèle IBM-4 décrit au chapitre précédent.

4.3.1 Stratégie de recherche

Moses cherche la meilleure traduction de façon similaire au décodeur pour IBM-4, en gérant des traductions partielles. Le décodage commence par l'hypothèse « vide », qui ne traduit aucun mot source et ne produit aucun mot cible. Puis, le traducteur choisit itérativement une hypothèse partielle et l'étend en traduisant un groupe de mots supplémentaire (d'un ou plusieurs mots source) à l'aide d'entrées de la table de traduction. Il est à noter qu'avec ce modèle, tout groupe de mots doit être aligné à un groupe de mots non vide, alors que le modèle IBM-4 forçait le décodeur à envisager que certains mots aient une fertilité nulle et que d'autres soient insérés spontanément. Ceci simplifie l'algorithme de décodage en évitant les multiples opérateurs décrits page 39.

Le résultat final de l'algorithme est obtenu comme avec le traducteur à base de mots en partant de la meilleure hypothèse complète et en suivant récursivement les pointeurs arrières vers les hypothèses partielles précédentes.

Organisation des hypothèses en files

Moses rassemble en une file toutes les hypothèses qui ont traduit le même nombre de mots, soit exactement $J + 1$ files pour une phrase de J mots. Les hypothèses *congruentes*, c'est-à-dire telles que leurs futures extensions respectives seront identiques, sont fusionnées comme à la section 3.2.2.

[Koehn, 2004, page 30] fournit une estimation d'une borne supérieure au nombre d'hypothèses partielles à considérer, nombre qui croît de façon exponentielle avec la taille J de la phrase à traduire. Moses doit donc élaguer les files d'hypothèses, et il utilise pour cela deux critères : un seuil pour un élagage relatif (les hypothèses de probabilité inférieure à, par exemple, 10^{-3} fois la probabilité de la meilleure hypothèse de la file sont supprimées) et une taille fixe pour les files (de l'ordre de 1000, par exemple).

Cette gestion des hypothèses partielles est similaire mais différente de celle employée par le traducteur à base de mots, qui était susceptible de créer 2^J files (hors contrainte de réordonnement) mais dont la taille n'était de l'ordre que de 10 ou 20.

[1] Le code source de Moses est disponible depuis `http://www.statmt.org/moses/`.

Recherche en faisceau

Moses emploie une recherche dite en faisceau (*beam search*), par opposition à la recherche « A* » du traducteur du chapitre précédent. Moses passe en revue les $J+1$ files *dans l'ordre*, et pour chaque file, étend toutes ses hypothèses partielles. Cette stratégie et le choix de l'heuristique d'estimation du coût futur (section suivante) ne permettent pas de garantir l'optimalité de la recherche mais s'avèrent nécessaires pour produire rapidement une traduction qui soit « presque » la traduction optimale.

4.3.2 Heuristiques

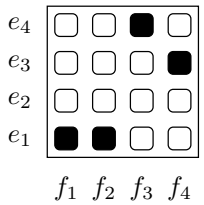
Dans les files que Moses maintient, certaines hypothèses ont pu commencer à traduire les mots « faciles », tandis que d'autres ont traduit des mots plus difficiles. Afin de comparer équitablement ces hypothèse, Moses doit estimer pour chaque hypothèse partielle le coût futur de traduction. Ce coût ne dépend que des positions sources restant à traduire. La différence cruciale avec l'heuristique retenue pour le traducteur à base de mots est que Moses calcule une heuristique qui peut *surestimer* le coût futur réel de traduction, générant ainsi éventuellement des erreurs de recherche. En pratique, le gain de temps est tel que ce risque permet de réaliser un bon compromis entre temps et qualité de traduction.

Le détail du calcul de la contribution au coût futur des trois modèles (modèles de traduction, de distorsion et de langage) n'est accessible à notre connaissance qu'en inspectant le code source. Nous ne mentionnons ici que l'estimation du coût futur pour le modèle de langage car c'est le seul dont le traitement soit radicalement différent par rapport au traducteur à base de mots.

La difficulté de l'estimation du coût futur du modèle de langage réside dans le fait que l'historique (bigramme, si le modèle est trigramme) n'est pas connu. Dans le cas du traducteur à base de mots, il était nécessaire de maximiser $\max_{e',e''} \Pr(e|e'', e')$, dont le calcul était trop long pour être intégré à l'heuristique. Au contraire, avec un modèle par groupes de mots, et pour tous les groupes de mots cible d'au moins trois mots, Moses est en mesure de connaître le coût exact qu'assignera le modèle de langage au troisième mot et aux suivants. Comme de plus l'heuristique de Moses n'a pas la contrainte d'être admissible, il est acceptable d'utiliser une probabilité unigramme pour le premier mot, et une probabilité bigramme pour le deuxième.

4.4 Modèles de langage et de traduction

La tâche considérée dans ce chapitre est, comme au chapitre précédent, la traduction de discours parlementaires entre l'anglais et l'espagnol. Les modèles de langage sont exactement les mêmes que ceux décrits au chapitre précédent, aussi cette section ne traitera que de l'entraînement du modèle de traduction, et plus précisément de la table de traduction.



Couples extraits :
$\langle e_4, f_3\rangle$, $\langle e_3, f_4\rangle$,
$\langle e_1, f_1 f_2\rangle$,
$\langle e_1 e_2, f_1 f_2\rangle$,
$\langle e_3 e_4, f_3 f_4\rangle$,
$\langle e_2 e_3, f_4\rangle$,
$\langle e_2 e_3 e_4, f_3 f_4\rangle$,
$\langle e_1 e_2 e_3 e_4, f_1 f_2 f_3 f_4\rangle$.

FIG. 4.2 – Extraction de tous les couples de paires de groupes de mots compatibles avec l'alignement de mots.

4.4.1 Extraction des paires de groupes de mots

La première étape est l'alignement mot à mot de chaque couple de phrases du corpus parallèle d'entraînement. Pour cela, deux modèles IBM-4, de l'anglais vers l'espagnol et inversement, sont entraînées par Giza++ dans le seul but de conserver les alignements mot à mot les plus probables selon ces deux modèles. Rappelons que le modèle IBM-4 n'est pas symétrique (section 2.1.3) : l'alignement de l'anglais vers l'espagnol ne sera pas le même que celui de l'espagnol vers l'anglais. Afin d'obtenir un alignement symétrique, nous partons des éléments de l'intersection des deux alignements et leur ajoutons des éléments de l'union des deux alignements. Le détail de l'algorithme est décrit dans [Koehn et al., 2003] sous le nom « diag-and ».

La seconde étape est l'extraction proprement dite des paires de groupes de mots. Elle est illustrée à la figure 4.2, où l'alignement (après symétrisation) entre les deux phrases est figuré par les cases noires. Une paire de groupes de mots est dite compatible avec l'alignement si tous les mots source de la paire sont alignés à des mots cible appartenant eux aussi à la paire, et réciproquement [Och et al., 1999]. Toutes les paires compatibles avec l'alignement par mot calculé ci-dessus sont extraites [Koehn et al., 2003]. À la figure 4.2, ces règles conduisent à l'extraction des couples $\langle e_4, f_3\rangle$ et $\langle e_3, f_4\rangle$, mais l'alignement de e_1 à f_2 empêche l'extraction de $\langle e_1, f_1\rangle$. Par ailleurs, le mot e_2 n'étant aligné à aucun mot f_j, il peut être ajouté à tous les couples extraits impliquant e_1 ou e_3.

4.4.2 Estimation des scores de chaque paire de groupes de mots

Savoir estimer la qualité d'une paire de groupes de mots $\langle \tilde{e}, \tilde{f}\rangle$ est le problème central de la traduction par groupes de mots. L'extraction des paires de groupes de mots décrite précédemment détermine l'ensemble des paires qui auront un score non nul. Les paires qui n'ont pas été extraites se voient assigner un score nul ; plus exactement, elles ne seront pas considérées lors de la traduction.

Plutôt que de calculer un unique score à chaque paire $\langle \tilde{e}, \tilde{f}\rangle$, plusieurs scores sont calculés en vue d'être pondérés et combinés (équation 2.6). Notre système emploie un ensemble de cinq scores.

Les deux premiers scores sont de simples *fréquences relatives*. Pour chaque couple $\langle \tilde{e}, \tilde{f}\rangle$,

l'on calcule :

$$h_1(\tilde{e}, \tilde{f}) = \frac{\mathcal{C}(\tilde{e}, \tilde{f})}{\sum_{\tilde{e}'} \mathcal{C}(\tilde{e}', \tilde{f})} \tag{4.2}$$

et

$$h_2(\tilde{e}, \tilde{f}) = \frac{\mathcal{C}(\tilde{e}, \tilde{f})}{\sum_{\tilde{f}'} \mathcal{C}(\tilde{e}, \tilde{f}')} \tag{4.3}$$

où $\mathcal{C}(\tilde{e}, \tilde{f})$ est le nombre de fois que le groupe de mots $\tilde{e}$ a été aligné à $\tilde{f}$ dans le corpus d'entraînement. On remarque qu'aucun lissage n'est ici effectué explicitement. [Foster et al., 2006] étudie différentes techniques de lissage, avec des gains entre 1 et 2 points Bleu suivant les paires de langue.

Les deux scores suivants de *pondération lexicale* peuvent toutefois être interprétés comme une forme de lissage.

$$h_3(\tilde{e}, \tilde{f}) = \prod_{i=1}^{|\tilde{e}|} \frac{1}{|\{j, (i,j) \in \mathbf{a}\}|} \sum_{j,(i,j)\in\mathbf{a}} t(f_j|e_i) \tag{4.4}$$

et

$$h_4(\tilde{e}, \tilde{f}) = \prod_{j=1}^{|\tilde{f}|} \frac{1}{|\{i, (i,j) \in \mathbf{a}\}|} \sum_{i,(i,j)\in\mathbf{a}} t(e_i|f_j) \tag{4.5}$$

Intéressons-nous à l'expression de $h_3(\tilde{e}, \tilde{f})$ (équation 4.4). L'alignement $\mathbf{a}$ correspond à l'alignement le plus fréquent entre les mots de $\tilde{e}$ et $\tilde{f}$. Les lois lexicales $t(f_j|e_i)$ et $t(e_i|f_j)$ sont estimées par fréquence relative sur les alignements de mots symétrisés. Signalons enfin que dans le calcul de h_3, si un mot e_i n'est aligné à aucun f_j, c'est-à-dire $\{j, (i,j) \in \mathbf{a}\} = \emptyset$, le terme $t(f_0|e_i)$ est utilisé, ce qui n'est pas indiqué dans l'équation 4.4 par souci de lisibilité.

Un dernier score produit pour chaque paire de groupes de mots vaut invariablement $h_5(\tilde{e}, \tilde{f}) = e \approx 2,718$. Utilisé au sein de l'équation 4.1, ce score permet de compter le nombre de groupes de mots recrutés pour construire la traduction et joue donc comme une pénalité de groupe de mots. En faisant varier la valeur de λ_5, on peut influer sur la propension du système à utiliser de nombreuses paires de groupes de mots (donc courts) ou au contraire lui faire préférer des groupes de mots plus longs.

4.5 Expériences et résultats

4.5.1 Réglage de la première passe

Comme décrit en introduction de ce chapitre, et de façon similaire au chapitre précédent, notre système emploie une stratégie à deux passes pour produire une traduction. Chacune des deux passes emploie son propre jeu de λ_i pour l'équation 4.1, ce qui est une pratique courante [Mauser et al., 2006, Cettolo et al., 2005, par exemple].

Les poids utilisés en première passe sont optimisés sur l'ensemble de développement dev06 de manière à maximiser le score Bleu. Ceci présente éventuellement le risque

(a) Résultats pour la traduction de l'anglais vers l'espagnol

	Dev06		Eval07	
	Avant	Après	Avant	Après
BLEU (%)	45,49	48,27	45,75	49,11
NIST	9,62	10,17	9,51	10,14
WER (%)	44,05	40,05	42,72	38,60
PER (%)	32,84	29,89	32,30	29,43

(b) Résultats pour la traduction de l'espagnol vers l'anglais

	Dev06		Eval07	
	Avant	Après	Avant	Après
BLEU (%)	41,43	46,19	44,35	47,84
NIST	9,65	10,52	9,98	10,66
WER (%)	46,28	41,23	43,91	40,11
PER (%)	32,96	29,49	30,13	27,75

TAB. 4.1 – Performances de la première passe du traducteur par groupes de mots avant et après réglage des 8 paramètres λ_i, avec un modèle de langage trigramme. Tous les scores prennent en compte la casse et la ponctuation.

de spécialiser les réglages pour cette mesure et aux dépens des autres mesures automatiques. Par exemple, Costa-jussà et al. [2006] proposent de maximiser une combinaison non-linéaire des scores BLEU et NIST, plus précisément $10 \cdot \log_{10}(100 \cdot \text{BLEU}+1)+\text{NIST}$, et signalent une plus grande cohérence des mesures automatiques en optimisant ce critère composite. Pour [Cettolo and Federico, 2004] en revanche, utiliser BLEU (plutôt que NIST) comme seul critère d'optimisation conduit à une optimisation plus rapide, meilleure en terme de score BLEU et parfois même meilleure en terme de score NIST.

L'outil MERT (pour *Minimum Error Rate Training*, entraînement minimisant le taux d'erreur) est distribué au sein de la boîte à outils Moses. Il met en œuvre l'optimisation de la façon suivante. MERT lance le décodeur Moses avec un premier jeu de λ_i et récupère ses n meilleures hypothèses. MERT détermine alors un autre jeu de λ_i, qui permette de maximiser BLEU. Il est à noter que pour ce faire, MERT *connaît* la forme particulière de la fonction donnant le score BLEU en fonction des λ_i : c'est une fonction constante par morceaux, du fait du caractère discret des listes de n-meilleures hypothèses [Och, 2003, Papineni, 1999]. Si les nouveaux λ_i sont différents des anciens, MERT relance Moses. Une douzaine de lancements de Moses est en général nécessaire pour atteindre la stabilisation. Par ailleurs, les différents optimums obtenus au cours du développement du système sont conservés et comparés, car il arrive que MERT « s'égare » dans un maximum local.

Le tableau 4.1 présente les scores obtenus avec plusieurs mesures automatiques avant et après l'ajustement des huit λ_i, pour les deux sens de traduction entre l'anglais et l'espagnol. Les λ_i ont été adaptés sur l'ensemble de développement uniquement.

Dans le sens anglais vers espagnol, le réglage des λ_i permet d'améliorer le score BLEU de près de 3 points sur l'ensemble de développement, le score NIST d'un demi point, et les taux WER et PER de 4 et 3 points respectivement. Sur l'ensemble d'évaluation, ces gains sont encore légèrement supérieurs, ce qui est *a priori* surprenant. Cela suggère que le système n'est pas sur-adapté sur l'ensemble de développement. Souvenons-nous que nous avions observé un comportement similaire du traducteur à base de mots sur les mêmes données (tableau 3.2, page 50).

Dans le sens espagnol vers anglais, les gains sur l'ensemble de développement sont importants : près de 5 points BLEU et autant de points WER. À nouveau, ceci n'est pas sans rappeler le comportement du traducteur à base de mots. Sur l'ensemble d'évaluation, les gains sont moindres mais restent conséquents : 3,5 points BLEU, près de 4 points WER et plus de 2 points PER.

4.5.2 Seconde passe : définition et réglage

La seconde passe est souvent vue comme une opportunité de calculer plusieurs scores supplémentaires sur les n meilleures hypothèses. Cela permet d'intégrer des fonctions caractéristiques que l'on ne pouvaient pas exploiter pendant le décodage. Par exemple, des modèles de langage n-grammes où $n \geq 5$ imposeraient un coût prohibitif en terme de temps de calcul lors du décodage de première passe, mais sont exploitables en seconde passe. Par ailleurs, certaines fonctions caractéristiques ne sont calculables que lorsque la phrase cible entière est connue ; voir par exemple les cinq modèles de réévaluation de [Mauser et al., 2006], ou la *kyrielle* de fonctions de [Och et al., 2004].

Après plusieurs expériences, nous n'avons pas suivi cette voie. En effet, ajouter de nombreuses fonctions présente un double risque de sur-adaptation. D'une part, les fonctions caractéristiques inspirées d'une analyse manuelle des erreurs sur le corpus de développement risquent d'avoir un impact sur ce corpus qui ne sera pas représentatif de leur impact sur des données inconnues. D'autre part, augmenter le nombre de fonctions caractéristiques complique l'optimisation numérique des λ_i, en particulier en considérant la nature hautement discontinue du score BLEU. Le système final décrit ici est donc volontairement simple, dans l'espoir qu'il soit aussi performant sur de nouvelles données que sur les données de développement.

Le réglage de la seconde passe est effectuée par Condor, déjà utilisé dans cette thèse, et est illustré à la figure 4.3. Moses traduit l'ensemble de développement, avec un modèle de langage trigramme et les poids obtenus à la section précédente, et produit pour chaque phrases ses n meilleures hypothèses. Dans nos expériences, $n = 1000$ et il est demandé à Moses de produire des hypothèses distinctes deux à deux. Les huit scores de chaque traduction potentielle sont accessibles. Les traductions sont réévaluées par un modèle quadrigramme, dont le score *remplace* celui du modèle trigramme de la première passe. Un court script-outil est alors écrit : étant donné un jeu de λ_i, il extrait la meilleure hypothèse pour chaque phrase de l'ensemble de développement et calcule le score BLEU correspondant. Ce script est nécessaire à Condor pour produire le jeu de λ_i optimisés. Une centaine d'itérations sont nécessaires, c'est-à-dire une centaine de lancements du script-outil : Moses n'est lancé qu'une seule fois, au début du processus.

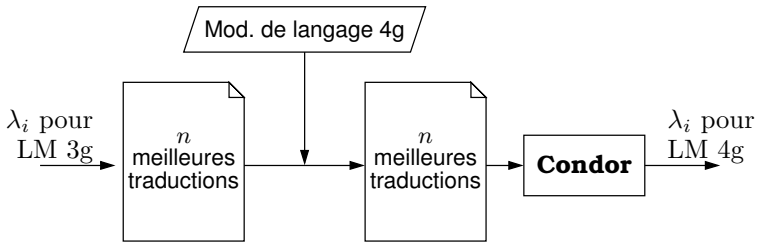


FIG. 4.3 – Ajustement des λ_i de seconde passe

Comme au chapitre précédent, deux modèles de langage quadrigramme sont comparés : le modèle à repli et le modèle neuronal. Les résultats sont présentés aux tableaux 4.1(a) et 4.1(b).

C'est pour la traduction de l'anglais vers l'espagnol que l'amélioration du modèle de langage a le plus d'impact sur la qualité de la traduction. Le score BLEU sur l'ensemble de développement progresse d'un point grâce au modèle quadrigramme à repli, ce qui est une amélioration sensible. Étonnamment, les taux de mots erronés WER et PER sont quasiment inchangés voire très faiblement à la *hausse*. Une sur-adaptation au score BLEU n'est pas à exclure. Les résultats sur les données d'évaluation sont légèrement plus rassurants : le modèle quadrigramme à repli fait à nouveau gagner un point BLEU et améliore toutes les mesures automatiques.

Le modèle quadrigramme neuronal montre un comportement encore meilleur. D'une part, il améliore le score BLEU d'environ 1,8 points par rapport au modèle trigramme, pour les deux ensembles de données considérés. D'autre part, il améliore aussi toutes les autres mesures, que ce soient le score NIST, de 0,15 point, ou les taux WER et PER, de 0,6 point. Remarquons que les gains sur les données d'évaluation sont toujours au moins aussi importants que ceux obtenus sur les données de développement. Au cours de la campagne TC-STAR, les différents sites ont échangé leurs résultats sur l'ensemble de développement et le système de ce chapitre n'avait pas les meilleurs. Lors de l'évaluation en revanche, il a obtenu le meilleur[2] score BLEU, le deuxième score NIST et taux WER, et le meilleur taux PER. Ces bons résultats suggèrent que la relative simplicité du système proposé, notamment en terme de nombre de fonctions caractéristiques, a permis d'éviter une éventuelle sur-adaptation à l'ensemble de développement

Les performances du modèle quadrigramme à repli pour la traduction de l'espagnol vers l'anglais sont similaires à celles observées pour l'autre sens de traduction. Le score BLEU croît de 1 point sur l'ensemble de développement, mais les scores NIST et WER sont dégradés. Sur l'ensemble d'évaluation, le tableau est encore plus contrasté. Le score BLEU ne gagne qu'un demi point par rapport au modèle trigramme, et toutes les autres mesures déclinent.

Le modèle neuronal parvient à faire mieux que les modèles à repli presque dans tous les cas. Sur l'ensemble de développement, il permet d'améliorer le score BLEU de 1,8 points

[2]Les classements cités sont les classements officiels en utilisant les mesures automatiques sensibles à la casse, sans compter la combinaison des systèmes TC-STAR. Le rapport d'évaluation est disponible publiquement depuis `http://www.tc-star.org/documents/D30.pdf`.

(a) Résultats pour la traduction de l'anglais vers l'espagnol

	Dev06			Eval07		
	ML 3g	ML 4g	MLN 4g	ML 3g	ML 4g	MLN 4g
Bleu (%)	48,27	49,26	50,03	49,11	50,08	50,91
NIST	10,17	10,22	10,30	10,14	10,20	10,29
WER (%)	40,05	40,08	39,42	38,60	38,57	37,94
PER (%)	29,89	29,99	29,34	29,43	29,08	28,80

(b) Résultats pour la traduction de l'espagnol vers l'anglais

	Dev06			Eval07		
	ML 3g	ML 4g	MLN 4g	ML 3g	ML 4g	MLN 4g
Bleu (%)	46,19	47,17	47,93	47,84	48,37	48,93
NIST	10,52	10,49	10,58	10,66	10,57	10,64
WER (%)	41,23	41,58	41,03	40,11	40,47	39,96
PER (%)	29,49	29,40	29,09	27,75	27,76	27,57

Tab. 4.2 – Performances du traducteur par groupes de mots après réévaluation par plusieurs modèles de langage.

mais aussi les autres mesures. Sur l'ensemble d'évaluation, le modèle neuronal obtient un score Bleu 1,1 points supérieur à celui du modèle trigramme, et les taux WER et PER sont légèrement améliorés. Seul le score NIST est dégradé de façon négligeable. À l'évaluation de février 2007, ce système de traduction s'est classé troisième selon les mesures Bleu, NIST et WER, et premier pour le taux PER.

Par ailleurs, nous pouvons comparer ce système à celui présenté au chapitre précédent et dont les performances sont consignées aux tableaux 3.5(a) et 3.5(b), page 61. Les deux systèmes sont entraînés et évalués sur les mêmes données et utilisent les mêmes modèles de langage. Une comparaison directe des modèles de traduction est donc possible, et elle est sans appel : de l'ordre d'une dizaine de points Bleu, de 7 points WER et de 6 points PER. Cet écart entre le modèle à base de mots et celui par groupes de mots est conforme à l'ordre de grandeur constaté dans la littérature [Ney, 2005].

4.5.3 Fonctions caractéristiques supplémentaires et intégration avec le traducteur à base de mots

Score IBM-1

Une fonction caractéristique ajoutée couramment aux systèmes de traduction à deux passes est le score IBM-1 du couple $(\mathbf{f}, \mathbf{e})$, pour chacune des n meilleures traductions de $\mathbf{f}$. Déjà introduit page 19, le modèle IBM-1 aligne tous les mots source à tous les mots

cible avec la même probabilité, et la formule suivante permet de calculer $\Pr(\mathbf{f}|\mathbf{e};\text{IBM-1})$ efficacement :

$$\Pr(\mathbf{f}|\mathbf{e};\text{IBM-1}) = \frac{\epsilon}{(I+1)^J}\prod_{j=1}^{J}\sum_{i=0}^{I} t(f_j|e_i) \tag{4.6}$$

où I et J sont les tailles respectives de $\mathbf{e}$ et $\mathbf{f}$.

Pour chaque traduction potentielle $\mathbf{e}$ de la phrase source $\mathbf{f}$, les deux scores $\Pr(\mathbf{f}|\mathbf{e};\text{IBM-1})$ et $\Pr(\mathbf{e}|\mathbf{f};\text{IBM-1})$ sont ajoutés, pour un total de dix fonctions caractéristiques — le score du modèle de langage est celui du modèle quadrigramme neuronal. Condor est ensuite lancé afin de déterminer un jeu optimal de dix poids. Plusieurs points de départ ont été essayés : les huit λ_i déjà présents en première passe ont été initialisés à leurs valeurs à la première passe ou à la seconde après optimisation ; les poids des deux scores IBM-1 ont été initialisés à 0 et à 0,1. Malheureusement, les gains en score Bleu n'ont jamais dépassé 0,1 point (sur l'échelle allant de 0 à 100) et les poids attribués aux fonctions caractéristiques IBM-1 étaient insignifiants. Les scores IBM-1 n'ont donc pas été retenus dans le système final.

Ces résultats négatifs avec les scores IBM-1 sont quelque peu surprenants, car ce score fait partie des fonctions caractéristiques « classiques » des systèmes de traduction statistique. Dans [Och et al., 2004], c'est même celle qui a le plus gros impact sur le système de référence. L'exploitation du modèle de langage neuronal n'est probablement pas ce qui annule l'effet du score IBM-1, car ces deux fonctions caractéristiques tendent à corriger des problèmes différents, voire orthogonaux. Le modèle de langage tend à améliorer la *fluidité* de la langue cible, sans considérer l'adéquation avec le sens de la phrase source, tandis que le score IBM-1 tend justement à mesurer cette adéquation, sans regard pour l'ordre des mots dans la langue cible. On peut avancer que les pondérations lexicales de chaque paire de groupes de mots (les fonctions caractéristiques h_3 et h_4 de la page 68) apportent déjà une information similaire à la « pondération lexicale globale » effectuée par le score IBM-1. Reconnaissons cependant que le système de [Och et al., 2004] incluait aussi un score, appelé *word selection*, qui ressemble aux pondérations lexicales h_3 et h_4.

Score IBM-4

D'autres expériences ont été menées pour ajouter un score IBM-4 à chaque couple traduction potentielle $\mathbf{e}$. Contrairement au score IBM-1, il n'existe pas de façon simple de calculer $\Pr(\mathbf{f}|\mathbf{e};\text{IBM-4})$ à partir de $\mathbf{e}$ et $\mathbf{f}$, même en l'approchant par la quantité obtenue avec le meilleur alignement :

$$\Pr(\mathbf{f}|\mathbf{e};\text{IBM-4}) \approx \max_{\mathcal{A}} \Pr(\mathbf{f},\mathcal{A}|\mathbf{e};\text{IBM-4}) \tag{4.7}$$

Un mode « alignement forcé » a donc été ajouté au décodeur pour IBM-4 du chapitre précédent (section 3.2.4). Deux modèles IBM-4 ont été entraînés sur les mêmes données que le système par groupes de mots, dans les deux sens entre l'anglais et l'espagnol. Pour ces deux sens également, les poids de première passe des cinq fonctions caractéristiques du système IBM-4 ont été ajustés. Pour l'alignement forcé, le poids

du modèle de langage est bien sûr inutile, puisque la phrase cible est connue, mais les quatre autres poids ont un impact sur le choix du meilleur alignement $\mathcal{A}$.

Chaque traduction potentielle $\mathbf{e}$ est alignée à $\mathbf{f}$ par le décodeur et se voit attribuée quatre scores supplémentaires, correspondant aux scores de traduction lexicale, de fertilité, de distorsion et de génération spontanée du modèle IBM-4. Pour le sens de traduction de $\mathbf{f}$ vers $\mathbf{e}$ du point de vue du traducteur par groupes de mots, nous réalisons l'alignement forcé avec le traducteur à base de mots entraîné pour le même sens de traduction. Ce traducteur, par la relation de Bayes, utilise le modèle IBM-4 dans le sens inverse, de $\mathbf{e}$ vers $\mathbf{f}$, et les scores produits par l'alignement forcé correspondent donc à la quantité $\Pr(\mathbf{f}|\mathbf{e};\text{IBM-4})$. Avec les huit scores habituels, cela représente un total de douze fonctions caractéristiques.

Contrairement au score IBM-1, le score IBM-4 ne peut pas toujours être calculé : il arrive qu'aucun alignement $\mathcal{A}$ ne permette d'expliquer la production de $\mathbf{f}$ par $\mathbf{e}$ au sens du modèle IBM-4. Plusieurs traitements sont possibles pour gérer ces cas :

– N'utiliser le score IBM-4 que si toutes les traductions potentielles ont pu être alignées, et ne conserver sinon que la « première hypothèse », c'est-à-dire l'hypothèse de la liste des n meilleures qui aurait été choisie en fin de seconde passe si le score IBM-4 n'était pas utilisé.
  Cette stratégie est trop stricte car elle élimine la liste des n meilleures hypothèses la plupart du temps. À titre d'exemple, pour la traduction de 1712 phrases de l'espagnol vers l'anglais, seules 123 phrases ont généré des listes dont toutes les traductions potentielles ont pu être alignées.
– Éliminer les traductions potentielles qui ne peuvent être alignées. Si aucune traduction potentielle ne peut être alignée, ne conserver que la première hypothèse, définie comme précédemment.
  Cette stratégie fait courir le risque de perdre irrémédiablement la première hypothèse, si elle n'a pas pu être alignée mais qu'une de ses suivantes a pu l'être.
– La première hypothèse est toujours conservée. Si elle a pu être alignée, conserver toutes les traductions alternatives qui comme elle ont un score IBM-4. Si l'alignement forcé de la première hypothèse a échoué, ignorer toutes les traductions alternatives. Cette stratégie est celle que nous avons employée. Plus « libérale » que la première, elle ne conserve cependant les listes de n meilleures traduction que pour 193 phrases. En moyenne, ces listes compte 293 traductions alternatives, contre en moyenne 788 pour les listes complètes.
  Pour la traduction de l'anglais vers l'espagnol, ces problèmes d'alignement sont moins sévères : sur 1194 phrases à traduire, cette stratégie conserve en moyenne 84 traductions alternatives pour 970 phrases, et les 224 autres phrases perdent leur liste de n meilleures hypothèses. Pour cette expérience, des listes de 100 meilleures hypothèses avaient été demandées à Moses, et contenaient en moyenne 90 traductions alternatives.

Après que les listes de n meilleures hypothèses ont été ainsi traitées, les poids des douze fonctions caractéristiques ont été optimisés par Condor sur l'ensemble de développement. Aucun gain significatif ne s'est dégagé. Notre interprétation de ces mauvais résultats est que, comme pour le modèle IBM-1, l'information extraite du corpus d'entraînement par le modèle IBM-4 est déjà contenue dans le modèle par groupes de mots.

Chapitre 5

Adaptation discriminante de la table de traduction

Nous considérons dans ce chapitre le traducteur par groupes de mots, décrit au chapitre 4. Pour ce traducteur, la table de traduction joue un rôle essentiel. C'est en effet elle qui attribue un ou plusieurs scores à chaque couple de groupes de mots $\langle \tilde{e}, \tilde{f} \rangle$. Ces scores sont quant à eux de simples fréquences relatives d'évènements observés sur le corpus parallèle d'entraînement (section 4.4.2), éventuellement lissées [Foster et al., 2006]. Ces scores heuristiques trouvent une justification intuitive mais sont *a priori* sous-optimaux. On pourrait envisager d'apprendre ces scores de manière à maximiser la vraisemblance des données d'entraînement, mais à notre connaissance, les recherches dans ce sens n'ont pas réussi à faire mieux que les scores heuristiques [DeNero et al., 2006]. Une autre approche consiste à apprendre ou modifier ces scores sur la base des erreurs commises par le système. Dans ce chapitre, nous proposons de modifier les scores de la table de traduction de façon discriminante sur un ensemble d'adaptation.

5.1 Apprentissage discriminant

5.1.1 Notations

La table de traduction peut être vue comme une matrice très haute H :

$$H = \begin{bmatrix} h_{1,1} & \dots & h_{1,m} \\ \vdots & \vdots & \vdots \\ h_{i,1} & h_{i,k} & h_{i,m} \\ \vdots & \vdots & \vdots \\ h_{I,1} & \dots & h_{I,m} \end{bmatrix} \tag{5.1}$$

Chaque couple de groupes de mots $\langle \tilde{e}, \tilde{f} \rangle$ est repéré dans la table de traduction par un indice $i \in [1, I]$ et se voit attribué m scores. Par la suite, on notera indifféremment $h_{i,k}$ ou $h_k(\tilde{e}, \tilde{f})$ le k$^{\text{ème}}$ score du $i^{\text{ème}}$ couple. Ces notations sont résumées au tableau 5.1.

I	Nombre d'entrées dans la table de traduction	$I \approx 50$ millions
m	Nombre de scores par entrée	$m = 5$
i	Indice désignant un couple $\langle \tilde{e}, \tilde{f} \rangle$	$i \in [1, I]$
k	Indice désignant un des scores	$k \in [1, m]$
$h_{i,k}$	k$^{\text{ème}}$ score du $i^{\text{ème}}$ couple	aussi noté $h_k(\tilde{e}, \tilde{f})$
$\lambda_1, \ldots, \lambda_m$	Pondération des m scores de la table de traduction	
M	Nombre total de fonctions caractéristiques	$M = 8$
$\lambda_{m+1}, \ldots, \lambda_M$	Pondération des autres fonctions caractéristiques	

TAB. 5.1 – Résumé des notations

Chacun des m scores de la table de traduction est pondéré par un λ_k estimé sur un ensemble de développement (section 2.1.2). Le traducteur dispose d'autres fonctions caractéristiques, pondérées par $\lambda_{m+1}, \ldots, \lambda_M$. Pour notre système, ces fonctions caractéristiques supplémentaires correspondent au modèle de distorsion, au modèle de langage et à la pénalité de mots, et $M = 8$.

5.1.2 Cadre « classique »

Les λ sont regroupés dans un vecteur $\mathbf{w}$ de taille M qui joue le rôle de pondérateur :

$$\mathbf{w} = \left[\begin{array}{ccc:ccc} \lambda_1 & \ldots & \lambda_m & \lambda_{m+1} & \ldots & \lambda_M \end{array} \right] \tag{5.2}$$

Les pointillés servent uniquement à distinguer visuellement les λ qui ont trait à la table de traduction, à gauche, des autres λ, à droite.

Au cours du décodage, le système de traduction produit les scores suivants[1] pour chaque traduction potentielle $\mathbf{e}$ qu'il considère :

$$\Phi(\mathbf{e}) = \left[\begin{array}{c} \sum_p h_1(\tilde{e}_p, \tilde{f}_p) \\ \vdots \\ \sum_p h_m(\tilde{e}_p, \tilde{f}_p) \\ \hdashline h_{m+1}(\mathbf{e}, \mathbf{f}) \\ \vdots \\ h_M(\mathbf{e}, \mathbf{f}) \end{array} \right] \tag{5.3}$$

où le signe de sommation $\sum_p$ représente la somme sur tous les couples de groupes de mots utilisés pour produire $\mathbf{e}$ à partir de $\mathbf{f}$. À nouveau, les pointillés séparent les scores relatifs à la table de traduction, en haut, des autres scores, en bas. Le score pondéré de $\mathbf{e}$ est le produit $\mathbf{w} \cdot \Phi(\mathbf{e})$.

Lorsque M est faible, il est envisageable d'optimiser $\mathbf{w}$ directement de façon à maximiser le score du système, par exemple son score BLEU sur l'ensemble de développement

[1] Le traducteur utilise chaque score après en avoir calculé le logarithme ; cette opération n'est pas indiquée pour de raisons de lisibilité.

[Och, 2003]. Cela revient à réaliser un apprentissage discriminant mettant en compétition les M fonctions caractéristiques, et en particulier les m colonnes de la table de traduction H. Nous allons écrire différemment le score final $\mathbf{w} \cdot \Phi(\mathbf{e})$ afin de réaliser un apprentissage discriminant mettant en compétition les couples de groupes de mots *eux-mêmes*.

5.1.3 Cadre pour l'apprentissage discriminant de la table de traduction

On considère maintenant que le système dispose de I fonctions caractéristiques, une par ligne de la table de traduction H. Chacune de ces fonctions caractéristique ϕ est associée à un couple de groupes de mots $\langle \tilde{e}, \tilde{f} \rangle$ et compte le nombre de fois $\mathcal{C}(\tilde{e}, \tilde{f})$ que ce couple a été utilisé pour la production de $\mathbf{e}$ à partir de $\mathbf{f}$.

Le score avant pondération $\Phi(\mathbf{e})$ (équation 5.3) que le traducteur attribue à chaque hypothèse est modifié et est appelé $\Phi'(\mathbf{e})$. Les m premières lignes de $\Phi(\mathbf{e})$ sont remplacées par I lignes de comptes :

$$\Phi'(\mathbf{e}) = \left[\begin{array}{c} \mathcal{C}(\tilde{e}_1, \tilde{f}_1) \\ \vdots \\ \mathcal{C}(\tilde{e}_i, \tilde{f}_i) \\ \vdots \\ \mathcal{C}(\tilde{e}_I, \tilde{f}_I) \\ \hdashline h_{m+1}(\mathbf{e}, \mathbf{f}) \\ \vdots \\ h_M(\mathbf{e}, \mathbf{f}) \end{array}\right] \tag{5.4}$$

$\mathbf{w}$ est modifié en $\mathbf{w}'$ de façon duale, récupérant les scores de la table de traduction $h_{k,i}$ que $\Phi'(\mathbf{e})$ a perdus :

$$\mathbf{w}' = \left[\begin{array}{ccc:ccc} \ldots & \sum_{k=1}^{m} \lambda_k h_{i,k} & \ldots & \lambda_{m+1} & \ldots & \lambda_M \end{array}\right] \tag{5.5}$$

On vérifie aisément que $\mathbf{w} \cdot \Phi(\mathbf{e}) = \mathbf{w}' \cdot \Phi'(\mathbf{e})$, c'est-à-dire que cette réécriture ne modifie pas la recherche effectuée par le traducteur. Il est à noter qu'ici, les m scores de la table de traduction sont agrégés en un seul. On notera par la suite w_i le score (unique) du couple de groupe de mots d'indice i :

$$w_i = \sum_{k=1}^{m} \lambda_k h_{i,k} \tag{5.6}$$

En effet, l'apprentissage décrite plus bas discrimine les couples de groupes de mots, mais ne cherche plus à discriminer les m fonctions caractéristiques comme à la section précédente (cadre « classique »). Ceci a des conséquences sur la façon dont sera utilisé l'apprentissage discriminant : celui-ci commencera *après* que les M λ_i aient été déterminés sur un ensemble de développement. Autrement dit, l'apprentissage discriminant partira du meilleur système que le « cadre classique » permet d'obtenir.

Apprentissage de type « Perceptron »

Dans sa forme la plus simple, le Perceptron est un classifieur binaire [Rosenblatt, 1958]. Une particularité de son algorithme d'apprentissage est que les exemples de l'ensemble d'entraînement sont présentés itérativement au classifieur. Les paramètres de ce dernier ne sont modifiés qu'en cas d'erreur. Nous voulons adapter cet apprentissage aux scores de la table de traduction. Pour chaque phrase source de l'ensemble d'entraînement, si le traducteur produit l'hypothèse $\mathbf{e_h}$ et que la sortie désirée était $\mathbf{e_d}$, la règle de mise à jour des paramètres inspirée par le perceptron est :

$$\mathbf{w}' \leftarrow \mathbf{w}' + \alpha \left(\Phi'(\mathbf{e_d}) - \Phi'(\mathbf{e_h}) \right) \tag{5.7}$$

où α est une constante telle que $0 < \alpha < 1$. Par souci de clarté, nous l'omettons temporairement des équations suivantes et en rediscuterons plus bas. L'équation 5.7 se traduit dans notre cas en le système de mises à jour suivant :

$$\left\{ \begin{array}{ccccc} \vdots & & \vdots & & \vdots \\ w_i \leftarrow w_i & + & \mathcal{C}(\tilde{e}_{i,d}, \tilde{f}_i) & - & \mathcal{C}(\tilde{e}_{i,h}, \tilde{f}_i) \\ \vdots & & \vdots & & \vdots \\ \hdashline \lambda_{m+1} \leftarrow \lambda_{m+1} & + & h_{m+1}(\mathbf{e_d}, \mathbf{f}) & - & h_{m+1}(\mathbf{e_h}, \mathbf{f}) \\ \vdots & & \vdots & & \vdots \\ \lambda_M \leftarrow \lambda_M & + & h_M(\mathbf{e_d}, \mathbf{f}) & - & h_M(\mathbf{e_h}, \mathbf{f}) \end{array} \right. \tag{5.8}$$

Illustrons ce système d'équations par un exemple. Supposons que pour traduire la phrase `the kitty pursues the mouse`, un système de traduction propose la traduction $\mathbf{e_h}$ = *`le petit chat pourchasse le souris` (l'astérisque indique que la phrase n'est pas correcte grammaticalement). Pour produire $\mathbf{e_h}$, le système utilise les couples de groupes de mots suivants : $\mathcal{C}$(`the`, `le`) = 2, $\mathcal{C}$(`kitty`, `petit chat`) = 1, $\mathcal{C}$(`pursues`, `pourchasse`) = 1 et $\mathcal{C}$(`mouse`, `souris`) = 1. Supposons de plus que l'on ait identifié comme sortie désirée $\mathbf{e_d}$ = `le petit chat pourchasse la souris`, qui utilise comme couples de groupes de mots $\mathcal{C}$(`the`, `le`) = 1, $\mathcal{C}$(`kitty`, `petit chat`) = 1, $\mathcal{C}$(`pursues`, `pourchasse`) = 1 et $\mathcal{C}$(`the mouse`, `la souris`) = 1. Dans ce cas, la mise à jour modifiera les scores w_i des trois couples de groupes de mots suivants :

$$\left\{ \begin{array}{lcccll} w_i \leftarrow w_i & + & 1 & - & 2 & \text{pour le couple } \langle \texttt{the}, \texttt{le} \rangle \\ w_i \leftarrow w_i & + & 0 & - & 1 & \text{pour le couple } \langle \texttt{mouse}, \texttt{souris} \rangle \\ w_i \leftarrow w_i & + & 1 & - & 0 & \text{pour le couple } \langle \texttt{the mouse}, \texttt{la souris} \rangle \end{array} \right.$$

En plus des w_i, le système de mises à jour (équations 5.8) préconise de modifier les poids $\lambda_{m+1}, \ldots, \lambda_M$ pondérant le modèle de distorsion, de langage et la pénalité de mots. Toutefois, mettre à jour ces paramètres pourrait sérieusement compromettre le bon fonctionnement du système de traduction, par exemple si les mises à jour sont d'un ordre de grandeur supérieur à celui des poids. Aussi nous avons préféré dans un premier temps ne mettre à jour que les scores de la table de traduction.

Les scores w_i sont mis à jour comme suit. On considère que la table de traduction est pourvue d'une colonne supplémentaire initialisée à 0 :

$$\forall i \in [1, I], \qquad h_{i,0} = 0 \tag{5.9}$$

Pour cette colonne, λ_0 est fixé à 1 (pondération neutre). Pour chaque exemple de l'ensemble d'entraînement et pour chaque couple de groupe de mots, le score $h_{i,0}$ est modifié, le cas échéant, par la différence de comptes $\mathcal{C}(\tilde{e}_{i,d}, \tilde{f}_i) - \mathcal{C}(\tilde{e}_{i,h}, \tilde{f}_i)$. La forme des mises à jour assure que $h_{i,0}$ est toujours entier.

À mesure que les exemples sont présentés au système, la valeur de w_i dépendra de moins en moins des valeurs des $h_{i,1}, \ldots, h_{i,m}$ initiaux et de plus en plus de la valeur de $h_{i,0}$, appris de façon discriminante. En pratique toutefois, il reste un problème « d'ordre de grandeur ». La valeur initiale de w_i est de l'ordre de -10 à -1, devant lesquels des variations de $+1$ ou -1 sont trop importantes : le risque est de devoir présenter les exemples un nombre de fois prohibitif avant d'atteindre la convergence de l'apprentissage. Le choix d'une valeur appropriée pour α (équation 5.7) permet d'éviter cet écueil. $h_{i,0}$ est alors modifié par $\alpha \cdot \Delta\mathcal{C}(\tilde{e}_i, \tilde{f}_i)$. Dans nos expériences, $\alpha = 0{,}05$.

Détermination de la sortie désirée

Les calculs ci-dessus mentionnent une sortie désirée $\mathbf{e_d}$ en direction de laquelle les paramètres du système sont modifiés. Plusieurs alternatives sont à notre disposition quant au choix de cette sortie désirée. Il peut s'agir :

1. de la traduction de référence, ou de l'une des traductions de référence s'il y en a plusieurs ;
2. ou d'une traduction qui appartienne au voisinage de l'hypothèse $\mathbf{e_h}$.

Le voisinage de l'hypothèse est typiquement défini comme une liste de n-meilleures traductions explorées par le traducteur. La première solution, la mise à jour des paramètres en direction de la traduction de référence, n'est pas toujours possible. Pour modifier les paramètres selon le système d'équation 5.8, il faut en effet que la référence soit *productible* par le modèle de traduction, ce que des contraintes sur le réordonnement des mots peuvent empêcher, par exemple.

Liang et al. [2006] ont comparé les deux alternatives ci-dessus à une troisième, appelée hybride : la traduction de référence sert de sortie désirée si elle est productible par le modèle, sinon c'est la meilleure traduction du voisinage de $\mathbf{e_h}$ qui joue ce rôle. Il en ressort que la meilleure stratégie est la mise à jour dite *locale*, vers une traduction du voisinage de $\mathbf{e_h}$. Liang et al. [2006] suggèrent que modifier le système de traduction de façon à directement produire la référence fait courir le risque de favoriser un alignement déraisonnable, dont la seule qualité serait de permettre la traduction parfaite de l'exemple considéré. Il a donc été décidé de choisir la sortie désirée $\mathbf{e_d}$ parmi les n-meilleures traductions produites par le système.

Il reste à identifier la meilleure traduction parmi ces n-meilleures. Comme le réglage des poids du modèle log-linéaire maximise Bleu (section 4.5.1), il est naturel de sélectionner l'hypothèse de score Bleu maximal. Toutefois, il est courant qu'un score Bleu calculé à l'échelle d'un segment soit nul, ce qui apporte peu d'information sur la

qualité de la traduction proposée. Aussi avons-nous *lissé* ce score en le combinant avec d'autres scores BLEU_i, pour $i < 4$. Rappelons que BLEU_i est un score BLEU calculé en prenant en compte les unigrammes jusqu'aux i-grammes, et que le score BLEU habituel correspond à BLEU_4 (section 2.4.2). Le score $\text{BLEU}_{\text{lissé}}$ se calcule de la façon suivante :

$$\text{BLEU}_{\text{lissé}} = \sum_{i=1}^{4} \frac{\text{BLEU}_i(\mathbf{e}, \mathbf{e_r})}{2^{5-i}} \tag{5.10}$$

Nous suivons en cela [Liang et al., 2006]. Tous les scores présentés par la suite sont des scores BLEU = BLEU_4 « classiques ».

5.2 Présentation des approches alternatives

Les expériences menées dans ce chapitre portent sur la traduction entre l'espagnol et l'anglais. À cause de la taille importante du corpus d'entraînement du modèle de traduction, nous avons choisi de réaliser l'apprentissage discriminant sur l'ensemble de développement dev06 uniquement. L'apprentissage discriminant est donc utilisé ici comme une technique d'*adaptation* du modèle de traduction. Aussi nous comparons cette technique à d'autres techniques d'adaptation, que ce soit du modèle de traduction ou de langage. Tous les systèmes décrits ci-dessous, sauf le système de référence, sont adaptés sur l'ensemble dev06. L'ensemble eval06 est utilisé le cas échéant pour contrôler l'adaptation. L'ensemble eval07 sert dans tous les cas pour l'évaluation en aveugle des différentes stratégies (voir le tableau 2.1 pour les caractéristiques de ces ensembles de données).

Les systèmes comparés sont les suivants :

Système de référence : le système de référence est le système décrit au chapitre 4, en se limitant à la première passe. Pour ce système, l'ensemble de développement dev06 sert uniquement à déterminer les coefficients d'interpolation du modèle de langage et les poids λ_i de la combinaison log-linéaire des fonctions caractéristiques. L'ensemble eval06 n'est pas utilisé.

Adaptation du modèle de langage : un modèle de langage trigramme est appris sur dev06, en incluant les deux références, soit un peu plus de 120 000 mots. Puis ce modèle de langage est interpolé linéairement avec celui du système de référence, en minimisant la perplexité de l'ensemble eval06. Les coefficients d'interpolation sont 0,70 pour le modèle de référence et 0,30 pour le modèle appris sur dev06. Le modèle de traduction est le même que celui du système de référence.

Apprentissage discriminant du modèle de traduction : la table de traduction du système de référence est modifiée conformément à l'algorithme décrit à la section 5.1.3, en présentant plusieurs fois l'ensemble dev06 au système. L'ensemble eval06 permet de contrôler l'adaptation afin de prévenir un éventuel sur-apprentissage.

Modèle de traduction appris sur toutes les données : les données d'apprentissage et les données dev06 sont concaténées et un modèle de traduction est appris sur l'ensemble, en suivant exactement la même procédure que pour le système

de référence. Afin de tirer parti des deux traductions de référence proposées par dev06, son texte source est ajouté deux fois au corpus parallèle d'entraînement, et les deux références chacune une fois. Nous appelons cette opération l'ajout (une fois) de l'ensemble dev06 au corpus d'entraînement.

Nous avons également essayé d'ajouter n fois l'ensemble dev06 au corpus d'entraînement. Pour ce faire, nous avons comparé deux approches légèrement différentes. La première, appelée « AjoutDirect », consiste à ajouter n fois dev06 au corpus parallèle et de relancer toute la procédure d'entraînement, y compris l'alignement des mots. La seconde approche, notée « RecupAlig », récupère les alignements de mots calculés par Giza++ lorsque dev06 est ajouté une seule fois, réplique n fois l'alignement de dev06 et relance uniquement la procédure d'extraction et d'estimation des scores des couples de groupes de mots. L'approche RecupAlig est donc nettement plus rapide qu'AjoutDirect. Remarquons que tous les systèmes obtenus avec RecupAlig extraient exactement les mêmes groupes de mots que l'ajout direct de dev06 une fois : seuls changent les scores des couples de groupes de mots. Lorsque n augmente, un score poids croissant est accordé aux couples qui apparaissent dans l'ensemble de développement.

Modèle de traduction appris sur dev06 seul : comme nous le verrons plus tard, l'ajout de n fois l'ensemble dev06 est toujours bénéfique. Par acquis de conscience, un modèle de traduction a été appris uniquement sur dev06. Ce modèle *remplace* le modèle de traduction du système de référence.

De façon similaire à l'approche RecupAlig mentionnée ci-dessus, seule l'extraction des groupes de mots et l'estimation des couples a été refait, l'alignement par mots sur dev06 étant récupéré du lancement de Giza++ sur les données d'entraînement et de développement concaténées. Les résultats ainsi obtenus sont plus de 2 points Bleu meilleurs qu'en relançant Giza++ sur dev06 seul.

Deux modèles de traduction : la table de traduction du système de référence et celle entraînée sur dev06 sont utilisées en parallèle. Dans cette configuration, Moses est libre de choisir les couples de groupes de mots dans l'une *ou* l'autre des tables de traduction. Si un couple $\langle \tilde{e}, \tilde{f} \rangle$ apparaît dans les deux tables, Moses prend les scores les plus avantageux des deux tables.

Ce dernier système compte cinq fonctions caractéristiques de plus que le système de référence : les cinq colonnes de la table de traduction supplémentaire. Pour cette raison, il n'est pas possible d'utiliser les λ_i du système de référence. Puisque l'ensemble dev06 est « connu » de la deuxième table de traduction, l'ajustement des $8 + 5 = 13$ poids est effectué sur l'ensemble eval06.

Réajustement des poids du modèle log-linéaire

Tous les systèmes présentés ci-dessus sauf le dernier comptent autant de fonctions caractéristiques que le système de référence et peuvent donc réutiliser les mêmes λ_i. Toutefois, afin de permettre une comparaison juste avec le dernier système, les résultats seront également présentés après réadaptation des λ_i sur l'ensemble eval06 pour chaque système. L'outil MERT, déjà utilisé pour l'optimisation des λ_i de première passe au chapitre 4, réalise cette optimisation.

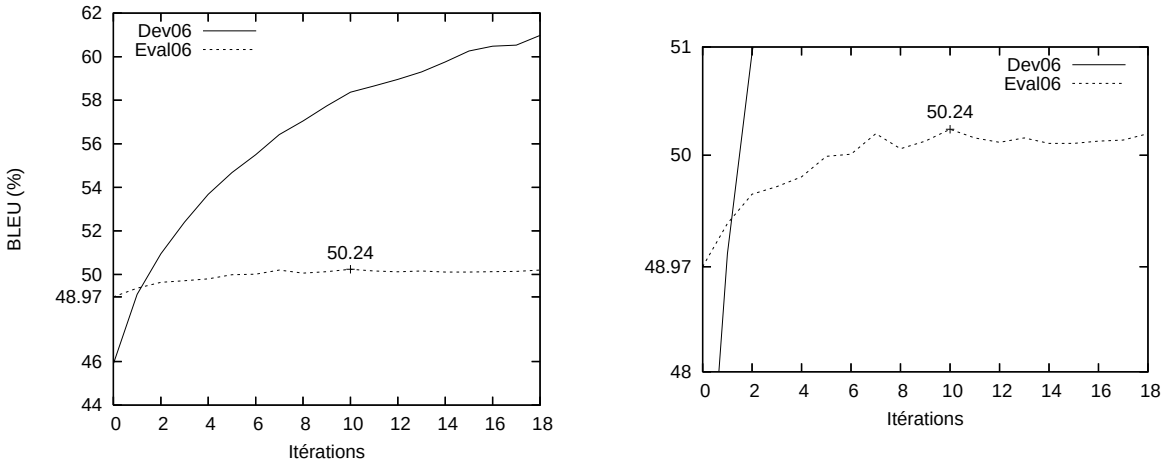


FIG. 5.1 – Apprentissage discriminant de la table de traduction sur dev06, et évolution sur eval06 (à droite : détail).

5.3 Résultats

Dans un premier temps, nous détaillons les résultats des différentes techniques d'adaptation pour la traduction de l'espagnol vers l'anglais.

5.3.1 Performances de l'adaptation discriminante

La figure 5.1 présente l'évolution des performances du système « apprentissage discriminant du modèle de traduction » au fur et à mesure des itérations. Rappelons que l'on présente au système l'ensemble dev06 ; la figure présente ses performances sur dev06 et eval06.

La courbe « dev06 » confirme uniquement que l'algorithme décrit ci-dessus est bien capable d'apprendre à traduire l'ensemble qui lui est présenté. Le score BLEU est amélioré d'une quinzaine de points après 18 itérations et semble pouvoir continuer sa progression avec plus d'itérations. La courbe « eval06 » est plus intéressante car elle montre la capacité de l'apprentissage à *généraliser* à un nouvel ensemble. Ce que l'apprentissage discriminant apprend sur dev06 semble utile pour traduire eval06, comme le montre l'évolution positive du score BLEU. L'amélioration maximale est atteinte à la dixième itération, avec un gain de plus de 1,2 points BLEU, ce qui est un résultat très encourageant.

On traduit alors l'ensemble eval07 avec le système de traduction obtenu après dix itérations d'apprentissage discriminant de son modèle de traduction, sans autre modification ni réglage. Le score BLEU s'élève alors à 48,67 %, soit près de 0,9 point BLEU de plus que le système de référence (tous les résultats sont regroupés à la table 5.2).

5.3.2 Ajouts de dev06 aux données d'apprentissage

Cette section compare les performances obtenues en ré-apprenant la table de traduction sur l'ensemble d'entraînement augmenté de n fois les données dev06. Les deux méthodes

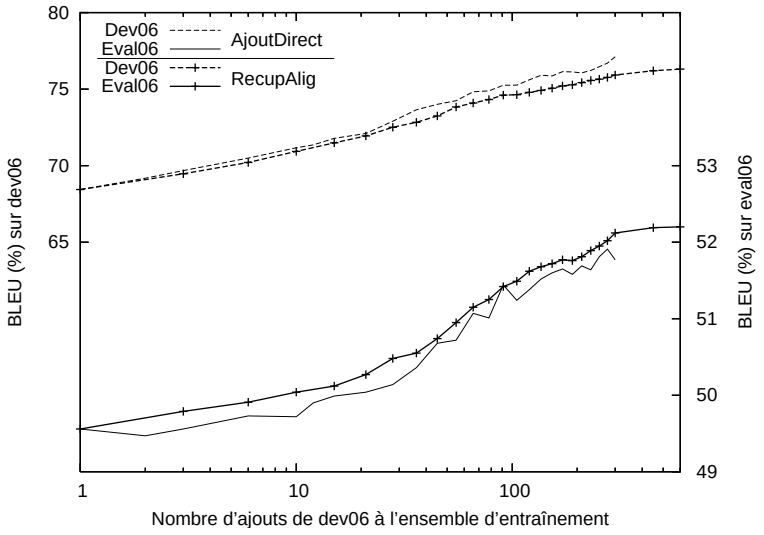


FIG. 5.2 – Ajouts de l'ensemble dev06 à l'ensemble d'entraînement

AjoutDirect et RecupAlig sont comparées, ainsi que plusieurs valeurs de n, de 1 à 600. Les systèmes ainsi obtenus sont évalués sur les données dev06 et eval06, et leurs performances sont représentées à la figure 5.2. Nous attirons l'attention du lecteur sur le fait que l'axe de gauche représente les scores BLEU obtenus sur dev06, pour les deux courbes du haut, tandis que l'axe de droite représente les scores BLEU obtenus sur eval06, pour les deux courbes du bas.

Sur l'ensemble dev06, les scores BLEU sont évidemment très élevés. La méthode AjoutDirect est très légèrement plus performante que la méthode RecupAlig, mais l'étape d'alignement par Giza++ prend un temps croissant à mesure que le nombre n d'ajouts de dev06 augmente. Le dernier point calculé avec la méthode AjoutDirect correspond à $n = 300$, ce qui représente approximativement autant de mots ajoutés que l'ensemble d'entraînement initial. L'alignement par Giza++ a duré près de 45 heures sur un processeur cadencé à 2,6 GHz. Le dernier point calculé avec la méthode RecupAlig correspond à $n = 600$. Les scripts d'entraînement doivent être adaptés si l'on souhaite tester $n = 1000$ ou au-delà.

Les performances des systèmes sur eval06 sont surprenantes à deux titres. D'une part, on n'observe aucun sur-apprentissage, même après avoir ajouté *six cents fois* l'ensemble dev06 au corpus parallèle initial. La répétition de dev06 donne un poids croissant aux couples de groupes de mots qu'il apporte, et est bénéfique au moins jusqu'à $n = 600$. La répétition des phrases semble en revanche nuisible à Giza++ : la méthode RecupAlig est toujours meilleure[2] que la méthode AjoutDirect, et montre un comportement plus régulier sur l'intervalle de variation de n. Dans les résultats décrits ci-dessous, le système « MT train+600 dev » correspond à l'ajout par la méthode RecupAlig de 600 dev06 aux données d'entraînement initiales.

D'autre part, l'autre résultat surprenant tient à l'ampleur de l'amélioration constatée

[2]La seule exception a lieu pour $n = 91$, où la méthode AjoutDirect est 0,02 point meilleure.

	Eval06		Eval07	
Poids λ_i	tels quels	réoptimisés	tels quels	réoptimisés
Référence	48,97	50,26	47,81	48,22
ML adapté	49,78	51,51	48,24	48,87
Appr. disc. du MT	50,24	51,26	48,67	48,90
MT train+600 dev	52,20	52,94	49,66	49,90
MT dev	42,64	43,01	39,58	39,85
MT train + MT dev	—	52,56	—	49,17

TAB. 5.2 – Performances (% BLEU) des différentes stratégies d'adaptation. L'adaptation des modèles est effectuée sur dev06. L'optimisation des λ_i est effectuée sur dev06 (poids « tels quels ») ou sur eval06 (poids « réoptimisés »).

sur les données eval06. Le système « MT train+600 dev » y atteint un score BLEU de 52,20 %, soit 2 points meilleur que le système après apprentissage discriminant et plus de 3 points meilleur que le système de référence. Sur l'ensemble eval07, l'amélioration est moindre mais reste significative, avec 1,8 points de plus que le système de référence.

Par sa proximité ou sa ressemblance avec les ensembles eval06 et eval07, l'ensemble dev06 est donc en mesure d'améliorer très significativement le système de référence. Nous allons maintenant comparer les stratégies d'adaptation énumérées plus haut et tenter d'isoler ce qui permet les gains observés.

5.3.3 Résultats complets pour le sens espagnol vers anglais

Le tableau 5.2 permet de comparer les performances des stratégies d'adaptation décrites à la section 5.2. Le système de référence obtient des scores BLEU de 48,97 % sur l'ensemble eval06 et de 47,81 % sur l'ensemble eval07. Les fonctions caractéristiques du système sont ici pondérées par des λ_i optimisés sur dev06. Ces poids peuvent être ignorés et réoptimisés sur eval06. Les résultats sont alors sans surprise meilleurs sur eval06. Le score BLEU passe à 50,26 %, soit un gain important de 1,3 points qui peut faire craindre des poids trop « spécialisés » sur cet ensemble. Pourtant, les résultats sur eval07 s'améliorent légèrement eux aussi pour atteindre 48,22 %.

Avant réoptimisation des λ_i, l'adaptation du modèle de langage ne fait gagner que 0,8 point BLEU sur eval06 et moitié moins sur eval07. Puisqu'un modèle est modifié, réoptimiser les poids semble raisonnable. L'écart avec le système de référence s'accroît alors : 1,25 points sur eval06, et à nouveau presque exactement la moitié sur eval07.

Les performances du système après apprentissage discriminant de la table de traduction ont déjà été données lorsque les λ_i ont leurs valeurs initiales, optimisées sur dev06. Le cadre théorique de l'apprentissage discriminant fusionne les m scores de la table de traduction en un (noté w_i) et c'est ce score qui est modifié au cours des itérations. Réoptimiser les poids des m scores après n'a plus vraiment de sens. C'est toutefois possible, puisque les modifications apportées aux scores w_i sont stockées dans une colonne supplémentaire de la table de traduction. On peut donc considérer que le système

dispose de *neuf* fonctions caractéristiques : les cinq scores initiaux de la table de traduction, un sixième score issu de l'apprentissage discriminant, le modèle de langage, le score de distorsion et la pénalité de mot. L'optimisation sur eval06 des neuf poids n'améliore le score BLEU que de 1 point sur eval06, et de 0,2 point sur eval07.

Analysons maintenant les différentes stratégies d'ajout de dev06 aux données d'entraînement du modèle de traduction. Comme nous l'avons déjà vu plus haut, simplement ajouter 600 fois l'ensemble dev06 au corpus parallèle initial permet d'obtenir des résultats étonnamment bons : 52,20 % sur eval06, 49,66 % sur eval07, soient les meilleurs résultats jusqu'à présent. Ajuster les λ_i sur eval06 ne fait gagner qu'environ 0,7 point BLEU sur eval06, et un peu moins que 0,3 point sur eval07. Puisque donner un poids toujours plus important aux couples de groupes de mots de dev06 est bénéfique, un modèle a été appris uniquement sur dev06. Ses performances sont très inférieures à tous les autres systèmes, montrant l'importance de la *couverture* apportée par les couples extrait du corpus parallèle d'entraînement. Enfin, le système utilisant deux tables de traductions en parallèle obtient de bons résultats sur eval06, quoique moins bons que ceux de « MT train+600 dev ». De plus, le système « MT train + MT dev » généralise moins bien sur eval07 que tous les autres systèmes, peut-être à cause du nombre élevé de ses fonctions caractéristiques. Signalons que c'est là notre première expérience avec plusieurs modèles de traduction en parallèle.

Analyse

Les gains constatés montrent que, pour la tâche considérée, le système de référence a beaucoup à apprendre des données de développement. Autrement dit, l'ensemble dev06 est significativement plus proche des données d'évaluation que ne le sont les données d'entraînement. Au moins deux facteurs pourraient l'expliquer :

1. Les traductions proviennent de sources différentes. D'une part, les données d'entraînement ont été produites par le service de traduction de la Commission Européenne. D'autre part, le comité d'évaluation a fait traduire à des traducteurs professionnels les données de développement et d'évaluation. Il est donc possible que le style des traductions de référence soit sensiblement différent de celui du corpus parallèle.

2. Les textes à traduire proviennent de sources différentes. Comme le rappelle le tableau 2.1, les données de développement et d'évaluation contiennent des transcriptions du parlement espagnol, tandis que le corpus parallèle ne contient que des données du parlement européen. Même si le modèle de langage est en partie appris sur le parlement espagnol, cette différence de source pourrait aussi en partie expliquer les gains observés.

Afin d'isoler ces deux facteurs, les expériences ont été partiellement reproduites sur la même tâche mais dans l'autre sens de traduction, anglais vers espagnol. Dans ce sens, l'éventuelle différence de style de traduction existe *a priori* autant que dans le sens que nous venons de considérer, mais les données en anglais proviennent entièrement du parlement européen, comme les données d'entraînement.

	Eval06		Eval07
Poids λ_i	tels quels	réoptimisés	les meilleurs sur eval06
Référence	37,15	37,10	49,09
Appr. disc. du MT	37,29	37,27	48,88
MT train+1 dev	37,10	37,20	48,84
MT train+300 dev	37,73	37,69	48,59

Tab. 5.3 – Performances (% Bleu) des différentes stratégies d'adaptation. L'adaptation des modèles est effectuée sur dev06. L'optimisation des λ_i est effectuée sur dev06 (poids « tels quels ») ou sur eval06 (poids « réoptimisés »).

5.3.4 Résultats pour le sens anglais vers espagnol

Pour des raisons de temps, seuls quatre systèmes ont été comparés :

1. le système de référence ;
2. le système obtenu après apprentissage discriminant de la table de traduction ;
3. les systèmes obtenus après ajout, par la méthode RecupAlig, de 1 et 300 fois l'ensemble dev06. Ces nombres ont été choisis arbitrairement en considérant les résultats de la section 5.3.2.

Les résultats sur les ensembles eval06 et eval07 sont détaillés au tableau 5.3.

Les mauvais résultats de l'optimisation des λ_i sont surprenants : la différence de performance avant et après réoptimisation est très faible et, surtout, négative trois cas sur quatre. Cela signifie que, pour trois systèmes, MERT trouve un maximum local qui n'est pas aussi bon que les poids pourtant obtenus en optimisant sur dev06. Comme dans toutes nos expériences, MERT a utilisé comme point de départ de l'initialisation celui par défaut, et pas les λ_i notés « tels quels ». Au cours de nos recherches, nous avons occasionnellement fait l'expérience des défauts d'optimalité des poids trouvés par MERT. Pour la traduction de l'ensemble eval07 et pour chaque système, ce sont les poids qui obtenaient les meilleurs résultats sur eval06 qui sont utilisés.

Comme précédemment, les meilleures performances sur eval06 pour l'apprentissage discriminant sont atteintes après dix itérations. Cependant, le gain par rapport au système de référence demeure très faible : à peine 0,14 point Bleu. Sur l'ensemble eval07, l'apprentissage discriminant de la table de traduction dégrade les performances de 0,2 point Bleu.

Le système « MT train+300 dev » est le meilleur sur eval06, avec un gain de 0,7 point Bleu, qui reste très inférieur au gain observé de l'espagnol vers l'anglais. En revanche, sur eval07, l'ajout des données dev06 fait perdre 0,5 point Bleu.

5.3.5 Analyse et discussion

Comme l'ont montré les expériences dans le sens espagnol vers anglais, l'algorithme d'apprentissage discriminant de la table de traduction présenté dans ce chapitre a été employé pour adapter un système à un ensemble de développement et améliorer ses performances de 1,2 points Bleu sur l'ensemble de contrôle eval06 et de 0,8 point sur

l'ensemble eval07. Deux expériences complémentaires hypothèquent ces bons résultats, cependant :
– les gains peuvent être reproduits et même dépassés simplement en ajoutant un grand nombre de fois l'ensemble de développement aux données servant à l'entraînement de la table de traduction ;
– les gains n'ont au contraire pas pu être reproduits dans l'autre sens de traduction.
Nous avons recommencé ces expériences sur la tâche BTEC (*Basic Travel Expression Corpus*, corpus de phrases simples du domaine touristique, Takezawa et al. [2002]), pour la traduction de l'arabe vers l'anglais. Cette tâche a la caractéristique de proposer un ensemble d'entraînement et plusieurs ensembles de développement et d'évaluation tirés d'un même corpus homogène. La taille restreinte du corpus a permis de considérer l'apprentissage discriminant sur l'ensemble d'entraînement comme sur les ensembles de développement. Mais à nouveau, aucune amélioration n'a été constatée sur un ensemble de contrôle.

5.4 Autres travaux et perspectives

L'adaptation du modèle de traduction a fait l'objet d'un petit nombre de publications. Dans [Hildebrand et al., 2005], les modèles de traduction et de langage sont réappris spécifiquement pour chaque ensemble à traduire. Les données sur lesquelles ces modèles sont appris sont déterminées par la combinaison d'un critère de similarité avec l'ensemble à traduire et d'un critère de perplexité. Cette méthode permet de sélectionner les phrases utiles parmi des données hors-domaine et améliore les performances pour les deux paires de langues testées.

L'évaluation WMT de 2007 portait sur la traduction de journaux télévisés et proposait un petit corpus parallèle du « domaine » et un gros corpus « hors-domaine » (Europarl, qui est similaire aux données TC-Star). Koehn and Schroeder [2007] comparent de nombreuses techniques simples d'adaptation du système : interpolation du modèle de langage, entraînement du modèle de traduction sur les deux corpus séparés ou fusionnés, etc. Contrairement à nos résultats, c'est l'utilisation de deux modèles de traduction en parallèle qui obtient les meilleures performances.

À l'évaluation NIST de 2006, pour la traduction du chinois vers l'anglais, les participants pouvaient utiliser de nombreux corpus couvrant plusieurs « genres » (journaux télévisés, dépêches, discours parlementaires, etc). Foster and Kuhn [2007] entraînent un modèle de traduction et un modèle de langage d'une part sur l'ensemble des données et d'autre part sur chaque corpus, appelé composant. L'objectif est de combiner ces modèles en fonction des données à traduire. Foster and Kuhn [2007] envisagent d'adapter le modèle de langage comme le modèle de traduction. Deux méthodes d'interpolation, linéaire et log-linéaire, sont comparées. Par ailleurs, plusieurs mesures sont proposées pour déterminer les poids d'interpolation. Parmi les enseignements de cet article, notons que les meilleurs gains sont de l'ordre de 1 point Bleu et ont été obtenus en pondérant linéairement les modèles de langage appris sur chaque composant.

Quant à l'apprentissage discriminant du modèle de traduction, les travaux présentés dans ce chapitre prolongent ceux de [Liang et al., 2006], déjà cité plus haut. Dans cet article, l'apprentissage inspiré du perceptron sert dans un premier temps à apprendre

les poids relatifs des modèles de langage et de traduction. Des millions de fonctions caractéristiques dites *lexicales* sont ensuite proposées. Ces fonctions sont de deux types. Les premières reconnaissent un couple de groupes de mots particulier et sont analogues à la colonne supplémentaire que nous adjoignons à la table de traduction. Les secondes s'activent pour un n-gramme cible particulier et reviennent à modifier de façon discriminante le modèle de langage [Roark et al., 2004]. Les expériences sont menées sur le corpus Europarl, du français à l'anglais. Pour des raisons de performance cependant, seules les phrases entre 5 et 15 mots sont considérées et une traduction monotone est imposée. L'ajout des fonctions caractéristiques lexicales permet un gain de 1,6 points BLEU sur leur ensemble de développement et moitié moindre sur l'ensemble de test. Mais le gain n'est que de 0,5 point BLEU par rapport à un système utilisant Pharaoh et réglé avec MERT.

Tillmann and Zhang [2006] appliquent un apprentissage discriminant du type perceptron à leur modèle de « suites de blocs » et obtiennent des résultats encourageants, sans toutefois dépasser leur système par groupes de mots classique.

Récemment, Arun and Koehn [2007] ont mené des expériences du tchèque vers l'anglais sur le petit corpus PCEDT (*Prague Czech-English Dependency Treebank*). Comme [Liang et al., 2006], des fonctions caractéristiques bilingues et monolingues sont considérées, et Arun and Koehn [2007] y ajoutent des fonctions qui prennent en compte le réordonnement des groupes de mots. Les auteurs confirment certains résultats déjà avancés par Liang et al. [2006], mais les systèmes obtenus après apprentissage discriminant sont très nettement *moins bons* que le système de référence.

Pour conclure, les recherches sur l'apprentissage discriminant des scores de la table de traduction n'ont pas encore réussi à améliorer de façon fiable les scores calculés à l'aide d'heuristiques, comme les fréquences relatives et les pondérations lexicales. Pourtant, l'amélioration des systèmes par groupes de mots passe probablement par des scores mieux estimés, et l'apprentissage discriminant est un bon candidat pour y parvenir. On peut envisager plusieurs raisons pour lesquelles à ce jour, les recherches n'ont pas été concluantes. La méthode décrite ci-dessus est peut-être sensible à certains « réglages », comme la valeur de la constante α réglant l'amplitude des mises à jours des scores. α pourrait au contraire dépendre du nombre d'itérations ou de l'écart entre la sortie désirée et l'hypothèse. Il est aussi possible que le cadre théorique lui-même soit en cause : comment définir la sortie désirée ? Comment prendre en compte le fait qu'une hypothèse peut être correcte *et* différente de la référence ? Faut-il chercher toutes les segmentations qui aboutissent à la même hypothèse, ou à la sortie désirée ? Une autre piste de réflexion pourrait être celle du *lissage* des mises à jour. En effet, dans l'exemple de la page 78, seuls trois couples sont modifiés : $\langle$`the`, `le`$\rangle$, $\langle$`mouse`, `souris`$\rangle$ et $\langle$`the mouse`, `la souris`$\rangle$. Les scores des couples contenant ces mots au sein de contextes plus longs sont inchangés. À l'itération suivante, si un couple de groupes de mots plus long produit la même erreur, celle-ci sera à nouveau corrigée. Mais si la traduction produite par le système est correcte, plus aucun score ne sera modifié. Ce cas est en réalité loin d'être idéal : en modifiant ainsi les scores, l'apprentissage discriminant pourrait parvenir à traduire parfaitement les données d'entraînement mais au prix de la *cohérence* entre les scores, autrement dit de leur lissage.

Des recherches complémentaires sont nécessaires pour améliorer l'algorithme proposé et permettre de réaliser un apprentissage discriminant performant.

Deuxième partie

Spécificités de la traduction de la parole

Chapitre 6

Motivation

6.1 Introduction

Les différentes approches de la traduction décrites au chapitre 2 ont presque toutes été développées dans le but de traduire des textes : manuels pour utilisateur, bulletins météorologiques, rapports de parlements, pages web, dépêches d'agences de presse, rapports de bogue, etc. Cependant, la parole reconnue automatiquement et les systèmes de dialogue représentent un « marché » potentiel important en termes de besoin en traduction.

Plusieurs prototypes ont d'ailleurs été développés durant ces vingts dernières années. Le plus ancien est peut-être celui commencé en 1986 à ATR pour s'inscrire à des conférences et réserver un hôtel par téléphone [Kurematsu and Morimoto, 1996]. Le projet Janus portait lui sur la planification de voyages [Levin et al., 2000]. Le projet Verbmobil avait pour objectif le développement d'un système de traduction mobile pour la traduction de la parole spontanée [Wahlster, 2000]. Un prototype d'aide à la prise de rendez-vous d'affaires a été créé, traduisant de l'allemand et du japonais vers l'anglais, censé servir de *lingua franca*. Dans le projet NESPOLE !, un client anglais, allemand ou français emploie un système de traduction parole à parole pour organiser un voyage auprès une agence de tourisme italienne [Metze et al., 2002]. Plus récemment, le projet MASTOR est un système de traduction parole à parole dans les deux sens entre l'arabe iraquien et l'anglais américain, dont 1000 unités seront prochainement déployées sur le terrain. Ce système peut reconnaître et traduire de la parole non-contrainte dans le domaine médical et a été porté sur un ordinateur portable et même un organiseur numérique personnel (*PDA*) [Gao et al., 2006, Zhou et al., 2004]. Mentionnons également le prototype développé par [Bach et al., 2007] pour une tâche similaire. Enfin, cette thèse a été effectuée dans le cadre du projet TC-STAR, qui a pour objectif d'améliorer significativement les performances de la traduction parole à parole.

Ce chapitre va tenter de montrer en quoi la traduction de la parole, qu'elle soit transcrite manuellement ou automatiquement, est un problème sensiblement différent de la traduction de texte. Dans la section 6.2, nous présentons les difficultés spécifiques rencontrées par les systèmes de traduction de la parole, *même en supposant une reconnaissance parfaite* des mots prononcés. Dans la section 6.3, nous envisageons l'inter-

action étroite de la reconnaissance automatique de la parole et de sa traduction afin de réduire l'impact des erreurs de reconnaissance. Chacun des problèmes soulevés est accompagné de solutions tirées de la littérature, tandis que nos recherches sur le sujet sont décrites aux chapitres 7 et 8.

6.2 Différences entre texte et parole du point de vue de la traduction

6.2.1 Niveau de langue et disfluences

Le niveau de langue varie très sensiblement d'un document à l'autre et le type de support, écrit ou oral, n'est pas le seul critère. Parmi les documents écrits, le style et le niveau de langue attendus ne sont pas les mêmes dans un roman, un article de quotidien, sur une page web ou dans un *blog*[1]. De même, un homme ou une femme politique prononçant un discours préparé commettra moins d'hésitations et de répétitions qu'un journaliste de journal télévisé, ou que lors d'une interview. À l'extrême, une conversation téléphonique entre deux personnes se connaissant bien contient souvent de nombreuses ellipses, interruptions, reprises, etc.

La figure 2.5, page 29, proposait quelques extraits de discours prononcés par des députés européens (données « Verbatim ») et leurs réécritures pour le site web du parlement européen (données texte final « TF »). Pour le premier segment, la réécriture TF est grammaticalement plus juste mais aussi complexe, avec une incise (which are aimed at undermining my integrity and reputation) au sein d'une phrase déjà relativement longue. De ce point de vue, la version Verbatim est légèrement plus simple à analyser et, peut-être, à traduire.

La figure 6.1 propose d'autres extraits de parole prononcée dans des situations quotidiennes, et leur traduction par un traducteur en ligne généraliste. Ces extraits sont des transcriptions manuelles servant de référence pour l'évaluation des systèmes de reconnaissance de la parole. On remarque qu'elles ne contiennent pas de signes de ponctuation mais suivent ici les règles orthographiques en matière de casse : les premières lettres des phrases sont en majuscule.

Les deux premiers exemples correspondent à du texte rédigé à l'avance et lu sur un prompteur par la présentatrice. On constate que certaines expressions qui relèvent du jargon du métier ne sont pas correctement traduites (« *coming up* » pourrait être traduit par « à venir dans ce journal »), mais une adaptation du système au domaine devrait corriger ce problème. Dans le deuxième exemple, l'utilisation du style direct pour rapporter les discours et l'absence de ponctuation provoquent une mauvaise traduction de « We are ready » (nous sommes prêts). Insérer un « : » dans la phrase anglaise permet d'obtenir une traduction correcte.

Les troisième et quatrième exemples sont extraits d'un dialogue au cours d'une interview dans la même émission. Le style de parole est ici moins soutenu, et des mots

[1] Journal personnel en ligne, où l'auteur communique ses idées et ses impressions (source : Mediadico).

Extraits de l'émission « The World » : journal télévisé

Coming up a twenty-seven year veteran of the FBI is arrested and charged with spying for the Russians	Monter des vingt-sept vétérans d'an du FBI est arrêté et chargé de l'espionnage pour les Russes
Beijing's mayor had a message today for a visiting inspection team from the International Olympic Committee We are ready	Le maire de Pékin a eu un message aujourd'hui pour une équipe d'inspection de visite du Comité olympique international que nous sommes prêts

Extraits de l'émission « The World » : interview

So is there a chance that if indeed he is guilty that he has put the lives of Americans in danger	Y a ainsi il une chance que si en effet il est coupable qu'il ait mis les vies des Américains en danger
Well we are still spying on each other because things happen in governments that other governments think they need to know about	Bien nous remarquons toujours sur l'un l'autre parce que les choses se produisent dans les gouvernements que d'autres gouvernements pensent qu'ils doivent savoir

Extraits d'une conversation téléphonique

They didn't have any snow on the ground but boy it was eh	Ils n'ont eu aucune neige sur la terre mais le garçon qu'elle était hein
yeah but it's just that it's just that clear cold that's the way it is been here it's just been cold	ouais mais c'est juste qu'il fait juste ce froid clair qui est la manière qu'il est été ici il est juste été froid

FIG. 6.1 – Parole transcrite manuellement extraite de différents corpus. À droite, la traduction en français obtenue à l'aide d'un traducteur en ligne.

comme « So », « Well » ou « I mean » (non présent dans ces extraits) sont fréquemment insérés. De plus, on peut déceler quelques entorses à la grammaire, comme par exemple le « that » répété dans le troisième exemple. Il est remarquable que le traducteur en ligne ait « réussi » à traduire cette faute très exactement, en répétant le « que », et qu'il ait ainsi produit une traduction au style parlé convaincant. Toutefois, lorsque de tels cas se présentent dans les données TC-STAR, les systèmes de traduction doivent produire une phrase correcte et donc *corriger* ce genre de fautes. Dans le quatrième exemple, l'interviewé est gêné. Sa réponse compte huit hésitations, qui n'ont pas été reproduites dans la figure 6.1. La structure de la phrase est globalement fausse et il est difficile de bien la traduire.

Enfin, les deux derniers exemples sont typiques de conversations téléphoniques informelles. Le lexique est clairement différent des exemples précédents (« boy », « yeah »). Des morceaux de phrases sont parfois répétés (« it's just that it's just that »). Avec ce type de parole spontanée, il est parfois difficile de comprendre l'énoncé, même à partir d'une transcription exacte.

Ces exemples montrent les différences entre les différents styles de parole et, *a fortiori*, entre la parole et le texte. On peut supposer que les méthodes de traduction qui reposent sur une analyse syntaxique de la phrase source auront plus de difficultés à traduire de la parole que des méthodes qui ne considèrent la phrase à traduire que comme une suite de mots.

6.2.2 Segmentation en phrases et ponctuation

La traduction de la parole se heurte à une autre difficulté que l'on ne retrouve pas dans le texte : le signal audio est un flux continu qui n'est pas segmenté explicitement. De même, la parole transcrite ne contient « naturellement » ni ponctuation, ni délimiteur de phrase. Le signal contient bien des informations relatives à la segmentation et, dans une moindre mesure, à la ponctuation choisie par le locuteur : ces informations font partie de la *prosodie*. Mais un système de RAP classique tente en général de s'affranchir des variations de rythme, d'intonation, etc.

Certes, les transcriptions de référence sont parfois ponctuées — c'est du moins le cas au sein de projet TC-Star. Mais les règles de placement des ponctuations n'étant pas clairement définies, les systèmes de RAP n'ont pas de « référence en or » à reproduire. C'est pourquoi le score *de facto* standard de la RAP reste le taux de mots erronés WER calculé sans considérer la ponctuation ni la segmentation en phrases, et les systèmes de reconnaissance ne produisent en général aucune ponctuation.

Avant de traduire le flux de mots source, il est ainsi nécessaire au minimum de le segmenter en phrases, voire de le reponctuer.

Dans le projet Verbmobil, un module exploitait les informations prosodiques pour segmenter le signal audio et guider la traduction des phrases [Strom et al., 1997]. Plusieurs types de frontières acoustiques (*boundaries*) sont introduits, des frontières segmentant le signal en phrases entières aux frontières entre les mots, en passant par des marqueurs délimitant des sous-phrases. Ces frontières sont automatiquement détectées et servent à limiter la taille de l'espace de recherche des modules analysant les mots reconnus. Par ailleurs, des caractéristiques d'insistance (*focus*) sont extraites automatiquement du contour de la fréquence fondamentale et peuvent influencer la structure de la traduction. Toutefois, ces techniques ne sont évaluées qu'en terme de taux de traductions acceptables. De plus, il s'agit d'une tâche restreinte de planification de rendez-vous, où la traduction repose sur une détection des types de tours de parole dans le dialogue.

Plus proche de nous, [Matusov et al., 2006] proposent un algorithme de segmentation du flux de mots et comparent trois stratégies pour passer d'une suite de mots source non-ponctués à une phrase cible ponctuée. Concernant la segmentation en phrases, un modèle de langage quadrigramme et la longueur de la durée des pauses entre les mots sont les deux critères les plus importants. En particulier, la modélisation explicite de la longueur des phrases apporte des gains très modestes. Quant à la production d'un texte cible ponctué, les trois stratégies considérées sont les suivantes :

1. Effectuer la traduction du texte source non-ponctué en un texte cible non-ponctué, puis ajouter la ponctuation dans le texte cible en post-traitement. Un inconvénient de cette méthode est que la reponctuation peut difficilement s'aider

des informations prosodiques contenues dans le signal source, et qu'elle s'effectue sur un texte qui contient plus d'erreurs que le texte source reconnu.

2. Faire produire la ponctuation implicitement par le modèle de traduction. L'article détaille comment l'extraction de la table de traduction est aménagée. L'avantage de cette technique est qu'aucun pré-traitement ni post-traitement n'est nécessaire.

3. Ponctuer la phrase source, et la traduire avec un système de traduction prévu par exemple pour la condition Verbatim. Cette méthode peut poser deux problèmes : les inévitables erreurs de l'algorithme d'insertion de la ponctuation risquent de dégrader la traduction, et effectuer ce prétraitement sur un treillis de mots issu de la RAP n'est pas trivial.

[Matusov et al., 2006] évaluent ces trois stratégies sur la tâche BTEC (déjà mentionnée au chapitre 5) du chinois à l'anglais et sur la tâche TC-Star de l'anglais vers l'espagnol. Pour la tâche BTEC, la deuxième stratégie apparaît clairement meilleure que les deux autres. Pour la tâche TC-Star, les performances des trois approches sont similaires, mais la troisième est toujours légèrement meilleure. C'est celle que nous suivions dès la l'évaluation TC-Star de 2006 [Déchelotte et al., 2006b] ; des travaux plus récents ont montré que la reponctuation pouvait apporter des gains de performance très importants ([Déchelotte et al., 2007a], et chapitre 7).

Cattoni et al. [2007] intègrent la ponctuation en traduisant des réseaux de confusion ; leur approche sera détaillée plus bas dans ce chapitre.

Matsoukas et al. [2007] décrivent plusieurs traitements qui interviennent entre la reconnaissance automatique et la traduction. La tâche considérée est celle de GALE, soit la traduction de l'arabe vers l'anglais de journaux télévisés et de parole conversationnelle. C'est une tâche « difficile », où les scores Bleu dépassent rarement 15. Matsoukas et al. [2007] mettent en évidence l'importance de la segmentation automatique, et notamment l'importance de ne pas produire des segments trop courts. En revanche, le taux de mots erronés du système de reconnaissance et la ponctuation secondaire (virgules) n'ont qu'un impact limité sur la qualité de la traduction, un résultat surprenant attribué au niveau de performance du système de traduction.

6.2.3 Du point de vue des techniques d'apprentissage

La traduction de la parole pose un autre problème, lui plus spécifique aux approches qui reposent sur de grandes quantités de données : il s'agit d'une tâche pour laquelle l'on dispose justement de peu de données d'entraînement. En effet, les corpus parallèles les plus importants sont des textes de parlements, qui correspondent à des interventions orales mais qui ont été édités pour suivre les conventions de l'écrit. D'autres sources de données parallèles usuelles sont des textes issus d'agences de presse. Mais les corpus parallèles constitués de parole transcrite sont rares ; une exception notable est la tâche BTEC.

Pour résoudre ce problème, plusieurs approches sont possibles et peuvent être combinées :

– Les données d'entraînement peuvent être transformées de façon à être plus proches de la forme des données à traduire. À notre connaissance, ceci est rarement décrit dans la littérature mais est nécessaire pour beaucoup de systèmes. Par exemple, pour les systèmes décrits dans cette thèse, les données d'entraînement étant au format TF, il a fallu transformer tous les nombres représentés en chiffres en mots, comme ils apparaissent dans les données d'évaluation.
– Réciproquement, les données à traduire doivent être transformées pour « ressembler » aux données d'entraînement. Dans [Matsoukas et al., 2007] par exemple, un module reconnaît les nombres dans la parole transcrite et les transforme en chiffres, car c'est ainsi que le modèle de traduction a été appris.
  De façon similaire mais moins évidente, l'on peut vouloir changer la casse de la transcription à traduire. En effet, même si certains systèmes de RAP produisent une transcription qui distingue les majuscules des minuscules, des problèmes de normalisation subsistent pour certains mots, comme par exemple `Projet`, `Programme` ou `Comité`. Nous abordons ce problème, ainsi que le cas des mots composés et d'autres problèmes de normalisation, dans [Déchelotte et al., 2007a] et au chapitre 7.
– Une façon de traiter le problème de la traduction de la parole est de partir d'un système entraîné sur du texte et de l'*adapter* sur le type de données voulu. Le modèle de langage peut être adapté, par exemple en l'interpolant avec un modèle de langage appris spécifiquement sur des transcriptions audio (section 3.3.2, qui décrit les modèles de langage utilisés dans nos systèmes). Concernant le modèle de traduction, différentes techniques d'adaptation été mentionnées à la section 5.4.
– Terminons par l'étape la plus évidente pour un système statistique : le réglage du système spécifiquement aux données de développement disponibles. À notre connaissance, tous les sites le font ; Matsoukas et al. [2007] gagnent environ deux points Bleu en passant d'un réglage obtenu pour la traduction de dépêches à un réglage pour les transcriptions de journaux télévisés.

6.3 Étude de l'interaction avec la reconnaissance automatique

6.3.1 Aspects théoriques

Nous avons vu que la traduction de la parole nous confronte à plusieurs problèmes difficiles : il est possible d'observer des phénomènes linguistiques dans un discours oral, surtout spontané, qui ne sont pas observés dans du texte écrit, comme les reprises, les corrections et hésitations ou même des erreurs importantes de syntaxe et sémantique. Contrairement à certaines autres approches, les approches statistiques de la traduction automatique présentent l'avantage de n'interdire *a priori* aucun « évènement » dans la langue source, aussi improbable ou incorrect soit-il.

Les erreurs commises par le système de reconnaissance de la parole compliquent aussi la traduction. Afin de limiter leur impact, il peut être utile d'intégrer étroitement la reconnaissance et la traduction, et cette intégration ne peut être que facilitée lorsque les mêmes approches sont utilisées dans les deux modules.

La reconnaissance automatique de la parole continue emploie un formalisme probabiliste : étant donné un signal acoustique $\mathbf{X}$, le décodeur tente de retrouver la phrase $\mathbf{f}^*$ la plus probable qui est responsable de l'observation acoustique [Bahl et al., 1983] :

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmax}} \Pr(\mathbf{f}|\mathbf{X}) = \underset{\mathbf{f}}{\operatorname{argmax}} \Pr(\mathbf{f}) \Pr(\mathbf{X}|\mathbf{f}) \tag{6.1}$$

Dans cette équation, $\Pr(\mathbf{f})$ représente le modèle de langage et $\Pr(\mathbf{X}|\mathbf{f})$ le modèle acoustique. En modélisant de façon statistique la traduction, c'est-à-dire en disposant de modèles de la forme $\Pr(\mathbf{e}|\mathbf{f})$, il est possible de considérer la traduction de la parole dans un cadre théorique unifié. La dérivation suivante provient de [Ney, 1999] :

$$\begin{aligned}
\mathbf{e}^* &= \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}|\mathbf{X}) \\
&= \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}) \Pr(\mathbf{X}|\mathbf{e}) \\
&= \underset{\mathbf{e}}{\operatorname{argmax}} \left(\Pr(\mathbf{e}) \sum_{\mathbf{f}} \Pr(\mathbf{X}, \mathbf{f}|\mathbf{e}) \right) \\
&= \underset{\mathbf{e}}{\operatorname{argmax}} \left(\Pr(\mathbf{e}) \sum_{\mathbf{f}} \Pr(\mathbf{f}|\mathbf{e}) \Pr(\mathbf{X}|\mathbf{f}, \mathbf{e}) \right) \\
&\approx \underset{\mathbf{e}}{\operatorname{argmax}} \left(\Pr(\mathbf{e}) \sum_{\mathbf{f}} \Pr(\mathbf{f}|\mathbf{e}) \Pr(\mathbf{X}|\mathbf{f}) \right) \qquad (6.2) \\
&\approx \underset{\mathbf{e}}{\operatorname{argmax}} \left(\Pr(\mathbf{e}) \max_{\mathbf{f}} \Pr(\mathbf{f}|\mathbf{e}) \Pr(\mathbf{X}|\mathbf{f}) \right) \qquad (6.3)
\end{aligned}$$

L'équation 6.2 est exacte à cette approximation près :

$$\Pr(\mathbf{X}|\mathbf{f}, \mathbf{e}) \approx \Pr(\mathbf{X}|\mathbf{f}) \tag{6.4}$$

qui revient à considérer que lorsque l'on connaît la phrase $\mathbf{f}$ dans la langue source, connaître sa traduction $\mathbf{e}$ ne donne pas plus d'information sur le signal acoustique $\mathbf{X}$. Cette approximation semble pleinement justifiée. Remarquons que jusqu'à l'équation 6.2 incluse, la phrase $\mathbf{f}$ n'est qu'une variable cachée, dont la détermination n'est pas nécessaire pour obtenir la traduction $\mathbf{e}$ du message contenu dans le signal $\mathbf{X}$.

L'approximation suivante est plus lourde de conséquences. À l'équation 6.3, la somme sur toutes les phrases source, c'est-à-dire sur toutes les transcriptions possibles du signal, a été remplacée par la quantité maximale obtenue par une seule transcription. À ce moment là seulement l'on peut parler de phrase source reconnue. Toutefois, il ne s'agit pas forcément de la phrase qu'aurait produit un système de reconnaissance automatique. En effet, en plus du modèle acoustique $\Pr(\mathbf{X}|\mathbf{f})$, la RAP utilise un modèle de langage source $\Pr(\mathbf{f})$, alors que l'équation 6.3 fait intervenir le produit $\Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$.

Cette interprétation de l'équation 6.3 est la justification théorique des expériences de *couplage* de la reconnaissance et de la traduction menées ces dernières années par de nombreux sites de recherche. On peut tenter d'en donner une autre interprétation, intuitive : si le modèle de langage source ne permet pas de trancher entre plusieurs hypothèses de scores voisins, il peut être préférable pour la traduction de préserver

cette ambiguïté, de traduire chaque phrase $\mathbf{f}$ possible et de produire comme résultat la phrase $\mathbf{e}$ qui maximise $\Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e})$, plutôt que de prendre une décision irrévocable sur $\mathbf{f}$ en fin de reconnaissance.

L'amélioration de l'intégration entre reconnaissance et traduction a été l'objet de nombreuses publications. Quelques unes sont présentées ci-dessous, regroupées en trois thèmes :

1. comment limiter l'impact des erreurs de reconnaissance sur la traduction, en traduisant « plus » que simplement la meilleure hypothèse de reconnaissance ;
2. quel impact ont les erreurs de reconnaissance sur les performances de la traduction, et en particulier quelle est l'importance relative des modèles acoustique et de langage source ;
3. et comment éventuellement modifier la reconnaissance de la parole et les erreurs qu'elle commet de façon à améliorer les performances globales de la reconnaissance suivie de la traduction.

6.3.2 Traduction de la sortie ambiguë de la reconnaissance automatique

Dans cette section, les différentes méthodes permettant le couplage entre la reconnaissance et la traduction sont présentées. L'objectif est à chaque fois de capturer une partie de l'ambiguïté de la sortie de la reconnaissance automatique et de prendre la décision sur la traduction en tenant compte des incertitudes sur la phrase source.

Traduction de listes de n meilleures hypothèses

Il s'agit du couplage le plus simple entre la reconnaissance et la traduction. Le décodeur de RAP produit ses n meilleures hypothèses, accompagnées de leurs scores respectifs. De préférence, les différents scores du modèle de langage, du modèle acoustique, et des éventuels modèles supplémentaires utilisés lors du décodage, sont accessibles séparément. Chaque hypothèse est alors traduite. Pour finir, une traduction est extraite parmi les n alternatives grâce à un jeu de poids pondérant les scores issus de la reconnaissance et les scores de traduction regroupés en un seul score global. Aucune modification du traducteur n'est nécessaire pour ce type de couplage, mais le temps de traduction est exactement multiplié par la taille des listes de meilleures hypothèses.

Le plus souvent, pour chacune des n hypothèses de reconnaissance, on demande au traducteur m hypothèses de traduction, pour un total de $n \times m$ traductions potentielles pour chaque phrase. C'est par exemple l'approche décrite dans [Quan et al., 2005]. Dans un premier temps, des poids sont appris séparément, d'une part pour produire les n hypothèses de reconnaissance, et d'autre part pour produire les m hypothèses de traduction. Ensuite, pour l'extraction de la traduction finale parmi les $n \times m$ hypothèses, les poids du système de traduction et ceux du système de reconnaissance sont optimisés conjointement. Quan et al. [2005] observent une amélioration du score Bleu de 1,2 points par rapport à la traduction de la meilleure hypothèse de reconnaissance, sur la tâche de traduction BTEC et avec le modèle à base de mots IBM-4.

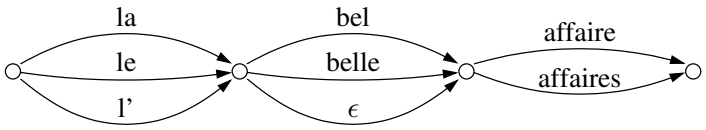


FIG. 6.2 – Exemple de réseau de confusion

Nos recherches sur le sujet ([Déchelotte et al., 2005], et chapitre 8) emploient aussi un modèle IBM-4 mais portent sur la tâche TC-STAR. Elles n'utilisent que la meilleure hypothèse de traduction, et pas les m meilleures.

Traduction de réseaux de confusion

La traduction de listes de n meilleures hypothèses a comme principal inconvénient d'être coûteuse en temps : n fois le temps de traduction de la meilleure hypothèse. Or ces listes sont généralement très redondantes, les hypothèses ne différant l'une de l'autre que de quelques mots. Le réseau de confusion permet de factoriser cette redondance en une représentation compacte des mots entre lesquels le système de RAP hésite. Le réseau de confusion est obtenu à partir du treillis de mots produit par le système de RAP en effectuant un décodage par consensus [Mangu et al., 1999]. Un exemple (fictif) de réseau de confusion est présenté à la figure 6.2. Dans cet exemple, le premier mot de la phrase peut être `le`, `la` ou `l'`, tandis que l'arc portant ϵ signifie que le deuxième mot peut être vide. En plus des mots, les arcs portent habituellement des scores, qui ne sont pas représentés sur la figure.

[Bertoldi and Federico, 2005, Bertoldi et al., 2007] décrivent comment l'algorithme de traduction est adapté pour gérer en entrée un réseau de confusion plutôt qu'une phrase. Il est intéressant de remarquer qu'en fait, l'algorithme est relativement *peu* modifié. La différence principale réside dans la recherche de toutes les traductions possibles pour chaque intervalle source $[i, j]$: dans le cas d'une phrase simple, il suffit de chercher dans la table de traduction toutes les traductions du groupe de mots $f_i \dots f_j$, alors que dans le cas d'un réseau de confusion, il faut faire cette recherche pour tous les groupes de mots source que l'on peut former entre les positions i et j. Bertoldi et al. [2007] montrent que la traduction de réseaux de confusion est meilleure que la traduction de listes de n meilleures hypothèses (jusqu'à $n = 10$ au moins) tout en n'étant que deux fois plus lente que la traduction de la meilleure hypothèse, soit cinq fois plus rapide que la traduction des 10 meilleures hypothèses. Conjointement à ce travail, Moses a été modifié pour pouvoir traduire des réseaux de confusion.

En revanche, [Al-Onaizan and Mangu, 2007], que nous citerons à nouveau plus bas, signalent des « gains insignifiants » obtenus lors d'expériences préliminaires en traduisant des réseaux de confusion.

[Cattoni et al., 2007] a récemment montré comment la traduction de réseaux de confusion pouvait être adaptée de façon à apporter une solution au problème des ponctuations soulevé plus tôt dans ce chapitre. Poursuivant le travail de [Bertoldi et al., 2007], un réseau de confusion est construit à partir du treillis de mots issu de la RAP et la meilleure hypothèse du réseau de confusion est extraite. Entre les mots de cette

hypothèse, des ponctuations sont proposées et pondérées. Ces ponctuations et leurs pondérations respectives sont alors réinsérées dans le réseau de confusion initial. Le traducteur, en l'occurence Moses, choisit de façon jointe la meilleure ponctuation et la meilleure traduction. Sur la tâche TC-STAR, de l'anglais vers l'espagnol, la traduction de réseaux de confusion est meilleure que la traduction de la meilleure hypothèse de reconnaissance de 0,6 point BLEU. Par ailleurs, insérer les ponctuations sous forme de réseaux de confusion et laisser Moses choisir la meilleure ponctuation produit un gain supplémentaire de 0,4 point BLEU par rapport à l'insertion déterministe des ponctuations.

Traduction de treillis de mots

Un couplage encore plus fin avec la reconnaissance de la parole est possible à condition de pouvoir traduire directement des treillis de mots, qui représentent fidèlement l'espace de recherche de la reconnaissance. Ceci complique significativement l'algorithme de traduction, même par rapport à la traduction de réseaux de confusion.

Par exemple, Zhang and Kikui [2006] traduisent des treillis de mots à l'aide d'un algorithme à deux passes. Lors de la première passe, le treillis de mots source est transformé en un treillis de mots cible ; seules les probabilités lexicales du type $t(f|e)$, *à la* IBM-1, sont utilisées pendant cette passe. Les réordonnements sont autorisés. Le treillis de mots cible est ensuite réévalué à l'aide des quatre sous-modèles du modèle IBM-4. Les meilleurs gains sur la tâche BTEC sont de l'ordre de 1 point BLEU, par rapport à un système de référence compétitif.

Dans [Saleem et al., 2004], traduire des treillis de mots apporte des gains d'amplitudes variables par rapport à la traduction de la meilleure hypothèse. Cependant, l'instabilité de ces gains est frappante : différents poids et densités de treillis sont considérés selon que la phrase est courte (moins de six mots), moyenne ou longue (plus de dix mots). Cela est attribué au fait que le score du modèle de langage source n'est pas incorporé au score global, car les treillis ne contiennent que les scores acoustiques. Ceci est significatif du compromis qu'il faut accepter pour traduire un treillis de mots : l'utilisation d'un modèle de langage source trigramme voire quadrigramme améliorerait les performances mais ferait exploser la taille du treillis de mots.

Enfin, Matusov et al. [2005c] obtiennent un appréciable gain de 2 points BLEU par rapport à la traduction de la meilleure hypothèse de reconnaissance en intégrant les scores acoustique et du modèle de langage source. La tâche considérée est la traduction de l'italien à l'anglais dans le cadre de BTEC ; la traduction repose sur un système par groupes de mots à l'état de l'art, même si les réordonnements de groupes de mots ne sont pas autorisés. En revanche, sur la tâche TC-STAR considérée dans cette thèse, les gains obtenus sur l'ensemble de développement ne se retrouvent pas sur les données d'évaluation. Ceci pourrait être du à l'important élagage des treillis de mots nécessaire avant la traduction.

Notons de plus que traduire le treillis de mots fait perdre la possibilité de déterminer l'hypothèse de reconnaissance obtenue par décodage consensus, dont le taux de mots erronés est légèrement meilleur que celui de l'hypothèse du treillis de probabilité a

posteriori maximale.

Intégration *via* le formalisme des automates à états finis

La traduction automatique peut être exprimée à l'aide d'un formalisme reposant sur des automates à états finis pondérés (*weighted finite state transducer*, WFST) [Vilar et al., 1996, Kumar et al., 2006]. Dans ce cadre, plusieurs automates caractérisant le processus de traduction sont composés. Par exemple, pour un modèle par groupes de mots, un transducteur peut accepter toutes les phrases dans la langue source et proposer leurs différentes segmentations en groupes de mots, un autre transducteur peut proposer plusieurs traductions pondérées pour chaque groupe de mots, ou encore un accepteur pondéré peut servir de modèle de langage.

Afin de calculer la traduction d'un treillis de mots, Mathias and Byrne [2006] proposent un algorithme pour transformer ce treillis en un treillis de groupes de mots, en incluant les scores acoustique et du modèle de langage source. Après cette étape, la traduction peut être effectuée en suivant la même procédure que pour une simple phrase. L'élégance de cette approche pour intégrer la reconnaissance et la traduction est indéniable, mais les gains rapportés sont pour l'instant très faibles. Avec une approche similaire, Zhou et al. [2007] parviennent à des gains importants, entre 1 et 2 points Bleu, sur une tâche de traduction de l'arabe vers l'anglais.

6.3.3 Importances relatives des différents modèles

Il existe d'autres aspects de l'interaction entre reconnaissance de la parole et traduction que l'on peut vouloir explorer. Par exemple, il serait intéressant de déterminer les importances relatives du modèle acoustique, du modèle de traduction, et des modèles de langage source et cible sur les performances globales, et quantifier l'impact des erreurs de reconnaissance sur la traduction.

À notre connaissance, nous avons été les premiers à évaluer séparément les impacts respectifs du modèle de langage source et du modèle acoustique ([Déchelotte et al., 2005], et chapitre 8). Il est apparu que, pour la traduction, le modèle de langage source est au moins aussi important que le modèle acoustique. Par ailleurs, au cours d'expériences de traduction de listes de n meilleures hypothèses de la reconnaissance, nous avons constaté que les scores acoutistique et du modèle de langage source étaient indispensables pour obtenir un gain, un résultat retrouvé simultanément par [Matusov et al., 2005c].

La nécessité du score acoustique est confirmée dans de nombreux articles. Reposer uniquement sur la topologie du treillis de mots, sans incorporer de score acoustique, dégrade les résultats [Saleem et al., 2004, Matusov et al., 2005c,a].

Plusieurs articles soulignent par ailleurs l'importance du modèle de langage source. Nous avons déjà cité à ce sujet [Matusov et al., 2005c] et [Déchelotte et al., 2005], et Saleem et al. [2004] attribuaient leurs résultats mitigés à l'absence de score linguistique. Plus généralement, une modélisation de la langue source est nécessaire, que ce soit directement *via* un modèle de langage, ou indirectement par un modèle n-gramme de

tuples bilingues [Matusov et al., 2005a].

Sarikaya et al. [2007] considèrent plusieurs systèmes de reconnaissance automatique qui diffèrent par leur modèle acoustique. Les taux de mots erronés de ces systèmes couvrent une large plage : de 0 à 40 % environ. Pour la traduction de l'anglais vers l'arabe iraquien, l'évaluation humaine et le score Bleu varient tout deux linéairement en fonction du taux de mots erronés des phrases traduites. Pour l'autre sens de traduction, de l'arabe vers l'anglais, la relation entre taux WER de la reconnaissance et qualité de la traduction est linéaire par morceaux : la traduction s'améliore rapidement lorsque le taux WER passe de 40 à 25 %, mais semble plafonner pour des taux inférieurs. Dans tous les cas, l'évolution du score Bleu reflète celle de l'évaluation humaine, même si cette dernière semble plus sensible aux erreurs introduites par la reconnaissance automatique que ne l'est Bleu.

6.3.4 Réglage de la reconnaissance automatique spécifiquement pour la traduction

Enfin, un dernier axe de recherche que nous évoquons est l'opportunité de modifier la façon dont fonctionne la reconnaissance automatique en vue d'optimiser les performances globales de la traduction de la parole. En effet, les systèmes de RAP sont attentivement développés de manière à diminuer le taux de mots erronés. Mais on peut envisager de « dérégler » ces systèmes, quitte à légèrement augmenter le taux de mots erronés, pour faire en sorte que leur sortie soit malgré tout mieux traduite. Voici les quelques articles que nous avons recensés sur ce sujet.

Dans [Gales et al., 2007], plusieurs stratégies de combinaison de systèmes de RAP sont comparées du point de vue de la traduction. Les tâches considérées sont la traduction de journaux télévisés et d'interviews de l'arabe vers l'anglais et du mandarin vers l'anglais. À chaque fois, deux systèmes de RAP sont combinés ou bien par ROVER, ou bien par *adaptation croisée*. L'adaptation croisée permet d'adapter les modèles acoustiques d'un système en lui présentant la sortie d'un autre système. Pour la traduction de l'arabe vers l'anglais des journaux télévisés, alors que la combinaison par ROVER obtient de meilleurs taux de mots erronés que la combinaison par adaptation croisée, la traduction du résultat de l'adaptation croisée est très légèrement meilleure que la traduction du ROVER. Les résultats dans les autres conditions sont plus contrastés, mais il en ressort que les gains en taux de mots erronés obtenus par le ROVER par rapport aux systèmes seuls ne se répercutent pas toujours en une amélioration de la traduction. Pour expliquer ce résultat, Gales et al. [2007] rappellent que la combinaison ROVER est déterminée en deux étapes : l'alignement des hypothèses de reconnaissance et le vote sélectionnant les mots. Ces deux étapes sont susceptibles de « casser » des groupes de mots, ce qui peut gêner la traduction.

Nous avons exploré une idée similaire à l'aide d'expériences où étaient traduites une combinaison de systèmes par ROVER, une sortie d'un système avec décodage consensus et une sortie sans décodage consensus. En effet, le décodage consensus comme le ROVER permettent d'abaisser le taux WER mais le font en contredisant les choix du modèle de langage source. Ces expériences sont décrites dans [Déchelotte et al., 2007a] et au chapitre 8.

Dans [Al-Onaizan and Mangu, 2007], plusieurs modifications sont apportées au système de RAP dans le but d'améliorer la traduction de sa sortie. Parmi ces modifications, le vocabulaire du système de RAP est étendu pour inclure celui du système de traduction. De plus, un modèle de langage est construit en incluant des mots composés *artificiels*, correspondant aux entrées de la table de traduction qui apparaissent au moins 20 fois dans les données d'entraînement. Malheureusement, l'article ne fournit pas les résultats de ces expériences. Par ailleurs, les poids du système de RAP sont modifiés de telle sorte qu'il ait plus tendance à supprimer des mots de la transcription. Cette tendance est mesurée par le ratio du nombre de suppressions par celui d'insertions. Deux points de fonctionnement sont comparés, entre lesquels le score Bleu est pratiquement inchangé mais le taux TER s'améliore légèrement.

Nous étudions plus en détail le lien entre les taux d'insertions et de suppressions de la reconnaissance et les performances de la traduction au chapitre 8.

Dans [Zhou et al., 2007], l'objectif est de faire préférer à la reconnaissance les groupes de mots que la traduction connaît et sait bien traduire. Pour cela, le système de RAP est modifié par le biais de son modèle de langage. Un nouveau modèle de langage est appris uniquement sur les groupes de mots source de la table de traduction, un poids plus important étant attribué aux groupes de mots source dont la traduction est peu ambiguë. Ce modèle de langage est alors interpolé avec le modèle utilisé normalement par le système de RAP. Le modèle de langage interpolé parvient à améliorer le taux WER de la reconnaissance automatique de près de 2 points sur les données de test, mais ses effets sur la traduction ne sont sensibles que sur le système de traduction entraîné sur peu de données, avec des gains entre 0,4 et 0,8 point Bleu.

Chapitre 7

Traduction d'un flux de mots

7.1 Introduction

Ce chapitre traite de plusieurs aspects spécifiques de la traduction de la parole et se place dans la situation suivante. Étant donné un flux audio, un système de reconnaissance de la parole produit un flux de mots dans une langue source, et ce flux doit être traduit en une série de phrases dans une langue cible. Dans ce chapitre, le système de RAP est considéré comme une « boîte noire » ; il peut s'agir d'un seul système ou d'une combinaison de systèmes développés sur plusieurs sites. Le système de traduction n'a accès qu'à la suite de mots, éventuellement pourvus d'indications temporelles, que le système de RAP considère être sa meilleure hypothèse. Par contraste, le chapitre suivant envisagera une interaction plus étroite avec le système de reconnaissance de la parole.

L'approche poursuivie dans ce chapitre est d'identifier et de corriger les divergences entre les données servant à entraîner le modèle de traduction et les données effectivement traduites pendant la phase d'évaluation. Plus précisément, voici les difficultés pour lesquelles nous proposons des solutions :

1. Les transcriptions de la parole peuvent contenir des *disfluences*, comme des hésitations, des répétitions, des mots d'appui, etc, qui apparaissent naturellement dans la parole. Le système de traduction doit reconnaître et supprimer ces phénomènes de la traduction[1].
2. La parole n'est pas *segmentée* explicitement en phrases, et retrouver cette segmentation automatiquement est une tâche difficile qui implique d'exploiter des informations linguistiques et acoustiques. De plus, les règles de placement des *ponctuations* faibles et moyennes (les virgules, par exemple) ne sont pas clairement définies et les systèmes de RAP sont souvent construits et développés sans comptabiliser les signes de ponctuation dans les taux de mots erronés.
3. Enfin, la *normalisation des mots* produits par la RAP peut être différente de celle attendue par le système de traduction, et aboutir à des performances sous-

[1] C'est du moins la consigne qui semble avoir été communiquée pour la traduction manuelle des données de développement.

optimales. Cet écueil est particulièrement sensible lorsque l'on traduit la sortie d'un système de RAP développé par un autre site, ou que l'on traduit le résultat d'une combinaison de systèmes.

7.2 Cadre expérimental

La tâche considérée dans ce chapitre est la traduction de transcriptions automatiques, de l'anglais vers l'espagnol, dans le cadre du projet TC-Star. Les résultats présentés portent sur l'ensemble de développement dev06 et l'ensemble de test eval07 (voir le tableau 2.1 pour les caractéristiques de ces données). Les expériences mettent en œuvre les deux systèmes de RAP et le système de traduction décrits ci-dessous.

7.2.1 Systèmes de reconnaissance de la parole

Les deux systèmes de RAP considérés sont :

1. le système de reconnaissance développé au LIMSI ;
2. et la combinaison par ROVER [Fiscus, 1997] des systèmes de reconnaissance de tous les partenaires du projet TC-Star.

Le système de RAP du LIMSI pour la transcription des discours parlementaires est décrit en détail dans [Lamel et al., 2007]. Parmi ses caractéristiques pertinentes aux expériences de ce chapitre, le système du LIMSI découpe automatiquement le signal audio en segments censés contenir un seul locuteur dans un seul environnement acoustique. La durée maximale de ces segments est de 30 secondes. Par ailleurs, le système du LIMSI produit un texte en vraie casse mais sans ponctuation.

La combinaison ROVER des systèmes du projet TC-Star est obtenue de la façon suivante. Les sorties des systèmes sont rassemblées et confiées à un partenaire du projet TC-Star. Avant d'être combinées, les hypothèses des systèmes sont converties en minuscules et les éventuels signes de ponctuation sont supprimés. Cette étape fait perdre de l'information mais permet en fait un meilleur alignement entre les hypothèses, car les différents systèmes risquent de ne pas utiliser les mêmes conventions quant à la casse et la ponctuation. La combinaison produite est donc entièrement en minuscules et ne contient aucune ponctuation. Elle est alors confiée à un autre partenaire afin que la casse et la ponctuation soient réintroduites. La casse est déterminée à l'aide d'un modèle de langage quadrigramme ; la ponctuation est insérée en prenant en compte la durée des pauses entre les mots et le score d'un modèle de langage [Stüker et al., 2006]. C'est le résultat de ces traitements que nous appelons par la suite la sortie ROVER.

7.2.2 Système de traduction

Le système de traduction utilisé dans ces expériences est le système par groupes de mots décrit au chapitre 4. La première passe emploie un modèle de langage trigramme et produit des listes de n meilleures hypothèses. À la seconde passe, ces listes sont réévaluées à l'aide d'un modèle de langage quadrigramme neuronal. Pour ces expériences,

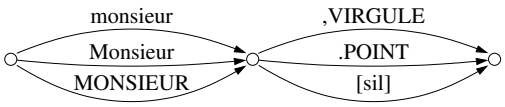


FIG. 7.1 – Treillis type généré pour chaque mot du fichier CTM à traduire. « [sil] » est une transition spontanée n'insérant aucun mot.

nous avons réutilisés les poids de première passe de la condition Verbatim, déterminés au chapitre 4. Les poids de la seconde passe ont en revanche été réoptimisés sur l'ensemble de développement dev06 avec la sortie ROVER.

7.3 Traitements proposés

Une série de traitements est ici proposée afin de rapprocher les données à traduire du format des données d'entraînement du système de traduction. La sortie du système de RAP, qu'il s'agisse du LIMSI ou du ROVER, est supposée être au format CTM (*time marked conversation*, parole annotée temporellement), qui fournit des marqueurs temporels et des indications de durée en plus des mots reconnus.

7.3.1 Casse et ponctuation

Pour commencer, les majuscules et les signes de ponctuation qui se trouvent dans le CTM à traduire sont supprimés. Nous en réinsérons ensuite en utilisant les mots et les marqueurs temporels contenus dans le CTM pour produire un treillis plat, ressemblant à un réseau de consensus. Plus précisément, chaque mot du CTM conduit à la création d'un nœud et de trois arcs, pour les trois variantes de ce mot : entièrement en minuscules, uniquement la première lettre en majuscule ou entièrement en majuscules. Entre les mots, des arcs sont ajoutés pour permettre l'insertion optionnelle d'une virgule ou d'un point. Un fragment d'un tel treillis est présenté à la figure 7.1. Il n'est pas possible d'introduire d'autres signes de ponctuation.

Le treillis ainsi constitué est réévalué par un modèle de langage quadrigramme spécialement conçu à cet effet. Ce modèle de langage résulte de l'interpolation de plusieurs modèles qui ont été entraînés sur les mêmes données mais en utilisant intentionnellement des choix de normalisation différents quant aux mots composés (avec les préfixes et les suffixes séparés ou non du mot) et aux acronymes (épelés ou en un mot). Ceci permet au modèle interpolé de traiter des treillis contenant des mots issus de systèmes dont les choix de normalisation ne sont pas connus.

Les fréquences respectives des virgules et des points ont été relevées sur les données d'entraînement et de développement. Il a été estimé qu'approximativement 3,5 % des « mots » étaient des points, et que 5 % étaient des virgules, soit environ un point tous les 29 mots et une virgules tous les 20 mots. Ce sont ces fréquences que notre algorithme va tenter de reproduire. Pour cela, deux quantités peuvent être ajustées :

1. le score présent sur les arcs contenant une ponctuation, sachant que les arcs « [sil] » ont un score nul (figure 7.1) ;
2. et la durée τ utilisée comme suit : un point est inconditionnellement inséré aux pauses plus longues que τ.

Ces deux quantités ont été estimées avec Condor en cherchant à minimiser la somme des erreurs quadratiques, où l'erreur correspond à la différence entre la fréquence cible et la fréquence obtenue. Un optimum local obtenant 3,8 % de virgules et 4,8 % de points a pu être atteint ; pour cet optimum, $\tau = 1{,}6$ secondes.

Une remarque importante concernant cette procédure est qu'elle ne nécessite pas de connaître les « vraies » règles grammaticales et orthographiques concernant la ponctuation et la casse. Son seul critère est la *quantité* de signes de ponctuation insérés. Sa « référence dorée » en matière de casse et de placement de la ponctuation correspond aux données d'entraînement du système de traduction, sur lesquelles le modèle de langage décrit plus haut est entraîné.

7.3.2 Suppression des disfluences et normalisations

Les hésitations et les mots d'appui se détectent facilement ; la liste complète des mots à supprimer est `eh`, `uh`, `uhm`, `huh` et `mmm`. Par ailleurs, les mots répétés sont supprimés, ne laissant que la première occurrence.

Le système de traduction peut attendre certains mots spécifiques sous une forme différente de celle produite par les système de RAP. C'est le cas d'un petit nombre de mots comme `Mrs.` ou `Mister`. De même, les acronymes qui ont été reconnus sous leur forme épelée (`C. N. R. S.`) doivent être rassemblés en un mot, comme ils apparaissent dans les données d'entraînement du système de traduction.

Enfin, la sortie d'un système de RAP peut contenir une fréquence relativement élevée de formes contractées, comme `it's` ou `can't`. En effet, il peut être profitable pour un système de RAP de produire, par exemple, un mot comme `it's` car lors du calcul du taux de mots erronés, un fichier GLM (*global map*) le fera correspondre à `it is` et `it has` si nécessaire. Les données d'entraînement EPPS, dont le niveau de langue est relativement élevé, contiennent des formes contractées mais à une fréquence bien plus faible que dans les transcriptions automatiques. C'est pourquoi les formes contractées sont toutes « étendues » avant la traduction. Dans ce travail, les formes contractées ambiguës sont étendues de façon déterministe en une forme arbitraire. Par exemple, `it's` a été systématiquement étendu en `it is` et `I'd` en `I would`.

7.3.3 Traitement des mots composés

Un des points de divergence entre un système de reconnaissance de la parole et un système de traduction peut être leur manière de gérer les mots composés. Certains préfixes, comme `pro-`, `anti-` et `trans-`, ainsi que certains suffixes comme `-like` en anglais, sont extrêmement génératifs, c'est-à-dire qu'ils sont susceptibles d'être associés à de nombreux mots pour en générer de nouveaux.

	Dev06	Eval07
ROVER	7,18	7,08
LIMSI	9,14	9,33

TAB. 7.1 – Taux de mots erronés (%) des deux systèmes de reconnaissance de la parole

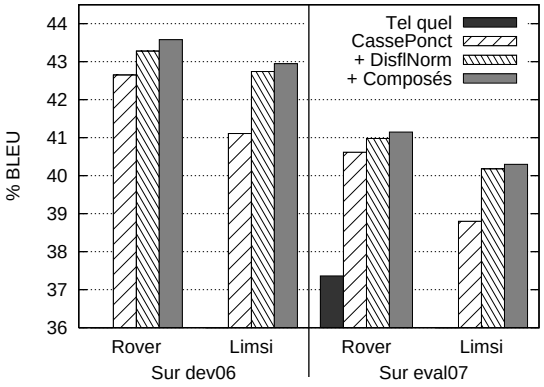
Puisque le LIMSI, et notamment le groupe TLP, est actif à la fois en reconnaissance de la parole et en traduction, nous avons mené plusieurs expériences dans le but d'utiliser la même normalisation de mots pour les deux systèmes. Toutefois, pour un système de RAP, intégrer tous les mots composés devient rapidement une charge de travail très importante, car des prononciations doivent être générées pour chaque mot supplémentaire. De plus, les modèles de langage risquent de ne pas disposer des données nécessaires pour estimer correctement les probabilités pour ces mots. Par ailleurs, l'outil évaluant les systèmes de reconnaissance coupe en général les mots aux tirets, de sorte que les systèmes ne sont pas pénalisés s'ils produisent `pro-Européen` en un mot ou `pro Européen` en deux mots. Tout cela incite à réduire la taille du vocabulaire des systèmes de reconnaissance en éclatant les mots composés. À l'inverse, les systèmes de traductions se doivent de produire les mots composés en un seul mot, et nos expériences ont montré qu'il était préférable de ne pas séparer les mots composés, dans la langue cible comme dans la langue source.

En conséquence, les deux systèmes de reconnaissance et de traduction ont chacun utilisé les règles de coupure des mots composés qui produisaient les meilleurs résultats en isolation, et un outil a été conçu pour recomposer les mots si nécessaire. Cet outil repose uniquement sur des comptes de n-grammes extraits du corpus d'entraînement du système de traduction. Ces comptes permettent de décider de produire, par exemple, le mot `pro-US` si cet unigramme est plus fréquent que le bigramme `pro US`. Des mots contenant jusqu'à trois tirets, comme `end-of-the-year`, peuvent être recomposés de cette façon.

7.4 Résultats

Les trois traitements décrits à la section précédente sont évalués systématiquement sur les données de développement dev06 et d'évaluation eval07. Parce que l'utilité de ces traitements dépend probablement du type de système de RAP, leurs effets sont comparés pour deux systèmes : celui du LIMSI et la combinaison ROVER des systèmes de TC-STAR. Les taux de mots erronés (WER) de ces systèmes sur les deux ensembles de données sont présentés au tableau 7.1. Ces taux sont calculés sans prendre en compte la casse et la ponctuation, comme il est d'usage dans la communauté de la RAP ; c'est aussi conforme au fait que notre premier traitement est, précisément, de recalculer la casse et la ponctuation.

		Tel quel	CassePonct	+ DisflNorm	+ Composés
Dev06	Rover	—	42,65	43,28	43,58
	Limsi	—	41,11	42,74	42,95
Eval07	Rover	37,36	40,62	40,98	41,15
	Limsi	—	38,80	40,18	40,30



TAB. 7.2 – Performances (% BLEU) des traitements décrits à la section 7.3. « Tel quel » : traduction directe du fichier fourni par le comité d'évaluation, disponible seulement pour la sortie ROVER sur eval07. « CassePonct » : avant la traduction, restauration de la casse et de la ponctuation. « DisflNorm » : suppression des disfluences et renormalisations. « Composés » : recomposition des mots composés.

7.4.1 Impact de la casse et de la ponctuation

Le script utilisé par le comité d'évaluation pour rétablir la casse et la ponctuation dans la sortie ROVER n'est pas public. Nous n'avons donc pas pu l'appliquer aux sorties Limsi et à la sortie ROVER dev06. C'est pourquoi l'utilité de notre procédure « CassePonct » ne peut être jugée qu'en la comparant à la sortie ROVER « telle quelle » sur l'ensemble eval07.

Avec une amélioration de plus de 3 points BLEU, de 37,4 % à 40,6 %, l'étape de restauration de la casse et la ponctuation joue clairement un rôle crucial dans la réduction de l'écart entre une sortie typique de RAP et le type de texte attendu par le système de traduction. Pour mieux situer ces scores BLEU, notons que notre système officiel, qui incluait des versions préliminaires des étapes « DisflNorm » et « Composés » mais pas « CassePonct », a obtenu un score de 37,6 %, et que le meilleur système officiel a obtenu un score de 39,2 %.

Afin d'expliquer le gain constaté, nous avons calculé les fréquences de virgules et de points sur les deux fichiers, « Tel quel » et après « CassePonct ». Le fichier CTM initial contenait 3,18 % de points et seulement 0,46 % de virgules, contre 4,09 % de points et 4,99 % de virgules dans le CTM après « CassePonct ». Le manque de virgules dans le premier CTM a deux effets. Dans un premier temps, le manque de virgules, donc de « mots » dans la phrase à traduire provoque un manque de mots dans la traduction, qui est sévèrement pénalisé par la pénalité de brièveté du score BLEU (voir la page 33

pour sa définition). Cette pénalité vaut en effet 0,934 dans le cas « Tel quel », alors qu'elle atteint 0,981 après « CassePonct ». Dans un second temps, même sans appliquer la pénalité de brièveté, la traduction après « CassePonct » obtient de meilleurs scores de précision et un BLEU « dépénalisé » de 41,4 %, alors qu'il n'est que de 40,0 % pour « Tel quel ». Ceci montre que non seulement la ponctuation permet d'éviter la pénalité de brièveté de BLEU, mais elle est aussi utile pour choisir les bons groupes de mots et, finalement, produire une traduction correcte. Cela suggère également qu'en n'imposant que la *quantité* de ponctuations, celles-ci ont été insérées aux *bons* endroits : ceux qui permettaient au modèle de traduction de retrouver des groupes de mots connus.

7.4.2 Impact de la suppression des disfluences et des normalisations

Le traitement « DisflNorm » a toujours un impact positif, même si son importance varie selon que l'on traduise la sortie LIMSI ou ROVER. Sur la sortie ROVER, « DisflNorm » apporte un gain de 0,6 point BLEU sur les données de développement et 0,4 point sur les données d'évaluation. Les gains sont plus importants en traduisant la sortie LIMSI : 1,6 points BLEU sur dev06 et 1,4 sur eval07.

Une inspection manuelle des données révèle la cause de ces différences. Sur la sortie LIMSI, la majorité des modifications apportés par « DisflNorm » consistent en l'expansion des très nombreuses formes contractées, suivies par la renormalisation des mots `Mr.` et `Mrs.`. Viennent ensuite la restauration des acronymes et la suppression de quelques mots d'appui. Sur les 1144 segments produits par le système LIMSI pour les données eval07, 508 sont ainsi modifiés. Sur la sortie ROVER, les modifications portent principalement sur les acronymes et les formes contractées, pour un total de seulement 192 segments modifiés sur 1159 à traduire[2]. Il semble qu'une partie du travail de renormalisation ait déjà été fait pour produire le ROVER.

7.4.3 Impact du traitement des mots composés

Le traitement « Composés » a modifié beaucoup moins de mots que « DisflNorm », et son impact est aussi plus faible, quoique toujours positifs. Les gains varient entre 0,1 et 0,3 point BLEU. Ce traitement a permis par exemple de traduire correctement des nombres comme `twenty two` en `veintidós` (vingt-deux) au lieu de `veinte dos` (vingt deux).

7.4.4 De l'intérêt de réoptimiser les poids de deuxième passe

Comme dit plus haut, les poids de la deuxième passe du système de traduction ont été réoptimisés pour la condition RAP. Par rapport aux poids pour la condition Verbatim déterminés au chapitre 4, ces nouveaux poids obtenaient des scores BLEU environ 0,4 point supérieur sur les données de développement dev06. Ces gains ont été confirmés sur les données eval06, même si l'amélioration n'était plus que de l'ordre de 0,2 point

[2]La différence de nombre de segments à traduire s'explique par le fait que les deux systèmes ont segmenté automatiquement, et différemment, le signal audio.

Bleu. Par conséquent, tous les résultats sur dev06 et eval07 décrits plus haut emploient les poids de seconde passe optimisés pour la condition RAP.

Toutefois, nous avons informellement relancé les expériences sur l'ensemble de test eval07 avec les poids optimisés pour Verbatim et avons constaté qu'ils obtenaient systématiquement de meilleurs résultats que les poids RAP, d'environ 0,2 point Bleu. Aucun de ces écarts ne n'est significatif statistiquement mais ils montrent l'instabilité des jeux de poids optimaux d'un ensemble d'évaluation à l'autre, et peut-être plus encore lorsqu'il s'agit de traduire une sortie aussi « imprévisible » que celle d'un système de RAP. De plus, on peut s'interroger du bien-fondé de l'optimisation spécifique pour la condition RAP, lorsque toutes les mesures décrites ci-dessus ont été prises pour faire ressembler la sortie de la RAP à la condition Verbatim.

7.5 Discussion

Ces expériences montrent que les performances de traduction dépendent de façon importante de l'écart entre les données d'entraînement et les données d'évaluation. Nous avons proposé plusieurs traitements qui visaient à rectifier cet écart et le gain cumulé sur l'ensemble de test est de presque 4 points Bleu. En particulier, le gain provenant de l'ajout des ponctuations était inattendu par son amplitude.

Parmi les améliorations futures possibles, citons la possibilité d'ajouter d'autres signes de ponctuation. Toutefois, un simple modèle de langage n-gramme ne permettra pas d'insérer de façon fiable un « ? » ou un « ! ». Pour ces signes de ponctuations, des indices prosodiques et des marqueurs linguistiques en début de phrase semblent être de meilleurs indicateurs. Compte tenu du manque de conventions concernant le placement des ponctuations faibles et moyenne, l'insertion de ponctuations comme « : » et « ; » demeure en revanche problématique.

Une autre amélioration possible a trait à l'expansion des formes contractées, qui pour l'instant est déterministe. Une manière de surmonter cette limitation serait de construire un treillis de mots proposant les différentes expansions possibles de chaque forme contractée.

Chapitre 8

Intégration avec la reconnaissance automatique de la parole

Au chapitre précédent, la reconnaissance de la parole était considérée comme une « boîte noire » dont il s'agissait de traduire le flux de mots produit. Ce chapitre « ouvre la boîte » et étudie plusieurs aspects de l'intégration de la reconnaissance automatique de la parole avec la traduction. Dans un premier temps, nous évaluons l'impact de la qualité des différents modèles utilisés par la reconnaissance automatique sur les performances de traduction. Nous essayons ensuite de limiter les effets des erreurs de reconnaissance en considérant non plus seulement la meilleure sortie du système de transcription, mais aussi les hypothèses proches, avec leurs scores de reconnaissance respectifs, ce que nous appelons la « sortie ambiguë » de la reconnaissance automatique. Enfin, nous tentons de modifier les caractéristiques du système de transcription de façon à maximiser les performances de traduction.

8.1 Importances relatives des différents modèles utilisés par la reconnaissance automatique

Pour concevoir un système de traduction de la parole, il paraît naturel, dans un premier temps, de concevoir le meilleur système de reconnaissance de la parole possible, et de traduire la sortie de ce système. Ce raisonnement repose sur l'hypothèse — raisonnable — que plus bas est le taux de mots erronés de la reconnaissance, meilleure sera la performance de la traduction.

Rappelons que la meilleure transcription $\mathbf{f}^*$ du signal acoustique $\mathbf{X}$ est obtenue formellement en résolvant l'équation 6.1, qui fait intervenir un modèle de langage (de la langue source) et un modèle acoustique. Supposons que les ressources pour améliorer le système de reconnaissance soient limitées : lequel de ces deux modèles aura le plus gros impact sur la performance globale (reconnaissance suivie de traduction) ?

Les expériences suivantes tentent d'apporter des éléments de réponse. Un traducteur statistique traduit la meilleure sortie de plusieurs systèmes de reconnaissance de la parole, qui diffèrent par la qualité de leurs modèles acoustiques et de langage. Le traducteur utilisé dans ces expériences est celui dit « à base de mots » décrit au chapitre 3. Les systèmes de reconnaissance de la parole sont décrits ci-dessous.

8.1.1 Système de reconnaissance de la parole

Le système de reconnaissance de la parole employé est similaire à celui employé au chapitre précédent [Gauvain et al., 2002, Lamel et al., 2007]. Nous ne décrivons ici que les caractéristiques relatives aux expériences menées dans cette section.

Modèles

La reconnaissance de la parole est effectuée en deux passes. Les mots reconnus à la première passe servent à adapter les modèles acoustiques de la seconde passe grâce à deux techniques d'adaptation non-supervisées, une MLLR[1] et une CMLLR[2]. Nos expériences utiliseront ces deux modèles acoustiques.

Plusieurs modèles de langage sont utilisés au cours des deux passes. La première passe emploie un modèle de langage trigramme (et le premier modèle acoustique) afin d'obtenir un treillis de mots. Ce treillis est ensuite réévalué à l'aide d'un modèle de langage quadrigramme afin d'extraire l'hypothèse « de première passe ». La seconde passe utilise un modèle de langage bigramme pour générer le treillis de mots, et un modèle de langage continu quadrigramme pour produire l'hypothèse finale. Les modèles de langages à repli appliquent le lissage de Kneser-Ney modifié ; le fonctionnement du modèle de langage continu, déjà utilisé dans cette thèse, est décrit dans [Schwenk, 2007].

Décodage par consensus

L'hypothèse obtenue en fin de seconde passe est généralement le fruit d'un décodage par consensus [Mangu et al., 1999]. Le tableau 8.1 donne les taux de mots erronés du système de reconnaissance de l'espagnol avec et sans décodage par consensus, avec deux modèles de langage. Le décodage par consensus améliore le taux de mots erronés obtenu avec le modèle quadrigramme à repli d'un demi point, et le gain est moindre avec le modèle neuronal.

Lors du décodage par consensus, le treillis de mots produit par le décodeur est modifié en fusionnant itérativement ses nœuds et en ajoutant des arcs jusqu'à obtenir un graphe linéaire. Ce processus ajoute d'une part des solutions qui n'étaient pas dans le treillis de mots initial, et fait perdre d'autre part la distinction entre scores acoustiques et de langage au profit de probabilités *a posteriori*. Dans les expériences suivantes, où le système de reconnaissance génère des listes de n meilleures hypothèses, le décodage

[1] Maximum likelihood linear regression, [Leggetter and Woodland, 1995]
[2] Constrained maximum likelihood linear regression, [Gales, 1998]

	Modèle 4g à repli	Modèle 4g neuronal
Meilleure hypothèse (MAP)	11,1 %	10,2 %
Décodage par consensus	10,6 %	10,0 %

TAB. 8.1 – Performance en taux de mots erronés du système de reconnaissance de la parole espagnole (modèle acoustique de 2$^{\text{nde}}$ passe et segmentation automatique).

Modèle de langage :	2-gramme	3-gramme	4-gramme neuronal
MA de 1$^{\text{ère}}$ passe	16,3 %	14,6 %	13,7 %
MA de 2$^{\text{nde}}$ passe	13,5 %	11,8 %	10,9 %

TAB. 8.2 – Taux de mots erronés avec les modèles acoustiques (MA) de première et seconde passe, et avec trois modèles de langage différents. La segmentation est imposée par les traductions de référence.

par consensus est désactivé de façon à conserver les scores acoustiques et de modèle de langage.

Segmentation automatique ou « manuelle »

Les résultats présentés au tableau 8.1 ont été obtenus en segmentant automatiquement le signal aux silences, donc *a priori* pas aux « vraies » frontières de phrases. Au moment où ces expériences ont été réalisées, l'évaluation des performances de la traduction nécessitait que les phrases à traduire soient segmentées de la même façon que les traductions de référence. L'organisme d'évaluation a donc manuellement établi une correspondance entre les traductions de référence et le signal audio, et les systèmes de reconnaissance étaient contraints de respecter cette segmentation.

Le tableau 8.2 rassemble les taux de mots erronés des six systèmes de reconnaissance utilisés par la suite. Tous respectent la segmentation manuelle. On observe qu'imposer la segmentation des traduction de référence à la reconnaissance de la parole provoque par exemple une augmentation de 0,7 point du taux de mots erronés, de 10,2 % (du tableau 8.1) à 10,9 %.

Si ces expériences devaient être reproduites, il serait maintenant possible de traduire la sortie du système de reconnaissance *tel qu'il l'a segmentée*. Un outil créé par Matusov et al. [2005b] et disponible librement permet en effet de resegmenter la traduction produite en l'alignant automatiquement aux traductions de référence.

8.1.2 Résultats

Les six systèmes de reconnaissance de la parole considérés sont construits en combinant :

– un modèle acoustique, parmi le modèle acoustique de première passe, avant adaptation, et le modèle acoustique de seconde passe, après adaptation ;

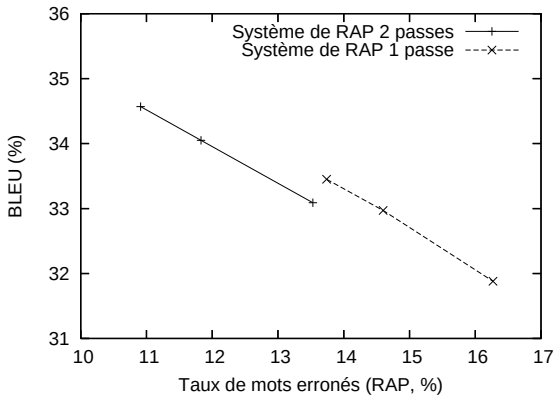


FIG. 8.1 – Scores BLEU obtenus en traduisant 6 sorties de RAP, obtenues en combinant deux modèles acoustiques et trois modèles de langage.

– et un modèle de langage, parmi un modèle bigramme à repli, un modèle trigramme à repli et un modèle quadrigramme neuronal.

La figure 8.1 présente les scores BLEU obtenus en traduisant la meilleure sortie de ces six systèmes de reconnaissance de la parole. Les trois points de droite (de taux de mots erronés plus élevés) correspondent au modèle acoustique de première passe, et les trois points de gauche au modèle acoustique de seconde passe. Pour chacun de ces deux triplets de points, le point en bas à droite correspond au modèle de langage bigramme, le point du milieu au modèle trigramme et le troisième point au modèle quadrigramme neuronal.

Les six points se répartissent sur deux droites presque parfaitement parallèles. Plus exactement, à modèle acoustique constant, le score BLEU augmente d'environ 1 point à chaque fois que le taux de mots erronés de la reconnaissance automatique diminue de deux points grâce à l'amélioration du modèle de langage source. L'augmentation du score BLEU associée à l'amélioration du modèle acoustique est légèrement plus faible, environ 1 point BLEU pour 2,5 points de taux de mots erronés de moins.

Le fait que le modèle de langue source soit aussi important sinon légèrement plus important que le modèle acoustique est un résultat surprenant. Notre intuition, en considérant l'équation 6.3, était que le modèle de langage $\Pr(\mathbf{f})$ serait moins important, car l'information qu'il apporte serait partiellement redondante avec celle fournie par le modèle de traduction $\Pr(\mathbf{f}|\mathbf{e})$. Les expériences de cette section semblent montrer que, lorsque le choix de la transcription est fait, son taux de mots erronés est un bon prédicteur du score BLEU de la traduction. Dans la section suivante, l'on se propose de retarder ce choix *après* la traduction, et ainsi de conserver l'ambiguïté de la sortie de la reconnaissance automatique.

8.2 Traduction de la sortie ambiguë de la reconnaissance automatique

8.2.1 Nécessité d'inclure le modèle de langage source

Réécrivons l'équation 6.3 de façon légèrement différente (un simple réordonnement de ses termes) :

$$\mathbf{e}^* \approx \underset{\mathbf{e}}{\operatorname{argmax}} \max_{\mathbf{f}} \Pr(\mathbf{X}|\mathbf{f}) \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \tag{8.1}$$

Le terme $\Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$ est ce que maximise, pour chaque $\mathbf{f}$ donné, le moteur de traduction (équation 2.2) et $\Pr(\mathbf{X}|\mathbf{f})$ représente le modèle acoustique utilisé par la reconnaissance de la parole. Le modèle de langage pour la langue source, $\Pr(\mathbf{f})$, est notoirement absent de l'équation 8.1, sa contribution étant incluse dans le modèle de traduction (*inverse*) $\Pr(\mathbf{f}|\mathbf{e})$. Cette observation a inspiré les expériences suivantes.

Pour chaque segment identifié par le moteur de reconnaissance de la parole, celui-ci produit non plus seulement sa meilleure hypothèse, mais ses n meilleures hypothèses, ainsi que leurs probabilités acoustiques et de langage respectives. Chacune de ces n meilleures hypothèses est traduite, produisant ainsi n traductions candidates, puis une traduction est sélectionnée parmi ces n. La décision sur la transcription du signal est ainsi retardée et prise en même temps que la décision sur sa traduction.

Dans un premier temps et pour appliquer strictement l'équation 8.1, la traduction sélectionnée parmi les n candidates est celle qui maximise une combinaison log-linéaire des scores des modèles acoustique, de traduction et de langage cible. L'expérience est répétée avec trois systèmes de reconnaissance de la parole : il s'agit des systèmes, déjà utilisés à la section précédente, qui utilisent le modèle acoustique dit « de seconde passe » et qui diffèrent par leur modèle de langage (bigramme à repli, trigramme à repli et quadrigramme neuronal). Les résultats de ces expériences sont rapportés à la figure 8.2 sous l'étiquette « MA + Trad ». On observe qu'ils sont moins bons que l'expérience de référence (étiquette « RAP (hypothèse) »), qui consiste à traduire la meilleure sortie de la reconnaissance automatique. De plus, la dégradation de performance est d'autant plus importante que le modèle de langage est bon. Alors qu'en traduisant la meilleure hypothèse de la reconnaissance, toute amélioration du taux de mots erronés se traduit quasi-linéairement en amélioration du score Bleu (section précédente), traduire des listes de n meilleures hypothèses aboutit à des résultats qui ne dépendent pratiquement pas du modèle de langage qui a servi à produire ces listes.

Au regard des équations théoriques, ce résultat est surprenant. Toutefois, supprimer le score du modèle de langage source fait intuitivement perdre de l'information, et le modèle de traduction $\Pr(\mathbf{f}|\mathbf{e})$ ne permet pas de le compenser. Au demeurant, c'est précisément parce que le modèle de traduction modélise insuffisamment la langue cible que le traducteur maximise la quantité $\Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$ plutôt que directement $\Pr(\mathbf{e}|\mathbf{f})$.

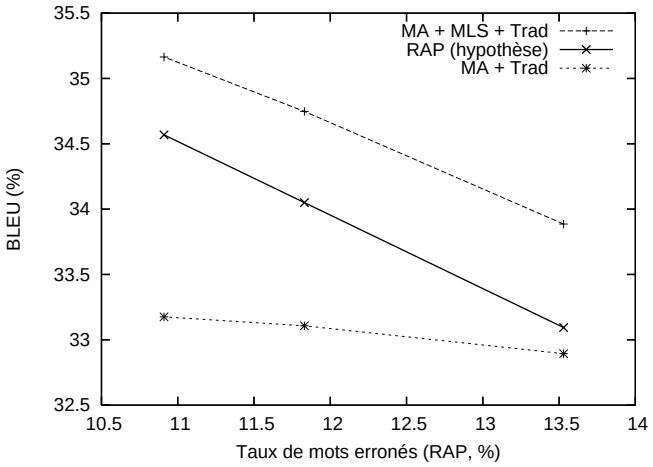


FIG. 8.2 – Traduction des n meilleures hypothèses de la reconnaissance automatique et sélection de la traduction finale en maximisant une combinaison des scores acoustique et de traduction (MA + Trad) ou des scores acoustique, de langage source et de traduction (MA + MLS + Trad). Pour comparaison, les scores obtenus en traduisant la meilleure hypothèse de la reconnaissance sont reproduits sous l'étiquette « RAP (hypothèse) » ; ils correspondent aux trois points de gauche de la figure 8.1.

8.2.2 Expériences incluant le modèle de langage source

Dans un second temps, le score du modèle de langage source est réintroduit dans la combinaison log-linéaire. En reprenant les pondérations utilisées par le système de reconnaissance pour les modèles acoustique et de langage source, et en assignant un poids nul aux scores de traduction, il toujours possible de reproduire l'expérience de référence (traduction de l'hypothèse de la reconnaissance automatique). Les résultats de cette expérience se doivent donc d'être au moins aussi bons que ceux de l'expérience de référence.

Comme on peut l'observer à la figure 8.2 sous l'étiquette « MA + MLS + Trad », les résultats sont effectivement meilleurs que ceux de l'expérience de référence. Cependant, les gains sont modestes, entre 0,8 point BLEU pour le système de reconnaissance utilisant un modèle bigramme et 0,6 point pour celui utilisant le modèle quadrigramme neuronal. Signalons que ces résultats ont été obtenus sur l'ensemble de développement qui a servi au réglage des poids : malheureusement, au moment où les expériences ont été réalisées, les expériences sur un ensemble de test n'ont pas été menées.

Pour tenter d'expliquer que les gains soient si faibles, on peut remarquer que les taux de mots erronés pour cette tâche sont bas : de l'ordre de 11 % au moment de ces expériences, et entre 7 et 9 % début 2007. Par conséquent, la dégradation des performances de la traduction due aux erreurs de reconnaissance est limitée, ce qui limite aussi ce que l'on peut espérer récupérer en traduisant plus que la meilleure hypothèse de la RAP. Remarquons que Matusov et al. [2005a] avaient constaté qu'à l'inverse, des taux de mots erronés de la RAP trop importants empêchaient aussi une bonne intégration de la reconnaissance automatique à la traduction.

Une autre direction de recherche en vue d'une meilleure intégration entre reconnaissance et traduction consiste à examiner dans quelles mesures le système de reconnaissance de la parole peut être modifié spécifiquement de sorte à améliorer les performances de la traduction. C'est l'objet de la section suivante.

8.3 Réglage de la reconnaissance automatique spécifiquement pour la traduction

8.3.1 Influence du décodage par consensus et du ROVER

Un système de traduction par groupes de mots considère les différentes segmentations possibles de la phrase source, extrait de la table de traduction les couples de groupes de mots susceptibles d'être utilisés, et finalement met en compétition les segmentations et les couples de groupes de mots pour produire la meilleure traduction. En pratique, l'extraction, ou filtrage, est réalisée par un outil avant le traduction proprement dite par Moses. Pour un système par groupes de mots, il est critique de pouvoir retrouver dans la phrase source des n-grammes observés dans l'ensemble d'entraînement.

Par ailleurs, les données d'entraînement du modèle de traduction servent aussi à entraîner le modèle de langage source employé par le moteur de reconnaissance de la parole. Dans ce cas précis, on peut presque parler de *synergie* entre la reconnaissance et la traduction : le modèle de langage source favorise les n-grammes fréquents dans les données d'entraînement, et ces n-grammes sont connus de la table de traduction (si l'alignement de mots a permis leur extraction).

Cependant, deux traitements peuvent assombrir ce tableau. Le premier est le décodage par consensus, déjà mentionné dans ce chapitre. Pour obtenir l'hypothèse par consensus, le treillis de mots de la reconnaissance est itérativement modifié et les scores du modèle de langage ne sont pas conservés. Finalement, l'hypothèse par consensus n'est pas celle prédite par les modèles acoustique et de langage. Elle peut même ne pas être présente dans le treillis de mots initial. Alors que l'hypothèse de probabilité *a posteriori* maximale minimise l'espérance du taux de *phrases* erronées, l'hypothèse par consensus minimise l'espérance du taux de mots erronés. Comme c'est ce taux qui sert de référence dans la communauté de la parole, tous les systèmes de reconnaissance mettent en œuvre le décodage par consensus.

L'autre traitement susceptible de casser les n-grammes est la combinaison de plusieurs systèmes de reconnaissance par ROVER. Les sorties des différents systèmes sont alignées de façon à obtenir un réseau de confusion, puis un vote pondéré détermine l'hypothèse ROVER. Ces deux étapes — alignement et vote — peuvent casser les n-grammes contenus dans les sorties de chaque système [Gales et al., 2007]. Toutefois, le ROVER permet souvent d'obtenir une sortie dont le taux WER est inférieur à tous ceux des systèmes combinés.

Dans cette section, d'une part, nous tentons de déterminer une mesure de la tendance des systèmes à casser les n-grammes, et d'autre part, nous en évaluons les conséquences sur la qualité de traduction. Pour le premier point, nous proposons de calculer le score

	Reconnaissance automatique			Traduction	
	Système	WER (%)	Bleu-RAP (%)	Nb. couples	Bleu (%)
Dev06	Rover	7,18	70,22	2231 k	43,58
	Limsi DC	9,14	63,98	2260 k	42,95
	Limsi MAP	9,53	63,92	2264 k	43,05
Eval07	Rover	7,08	67,92	2103 k	41,15
	Limsi DC	9,33	61,29	2123 k	40,30
	Limsi MAP	9,66	61,14	2130 k	40,19

Tab. 8.3 – Traduction de différents systèmes de RAP. Colonnes reconnaissance automatique : nom du système, son taux WER et son score Bleu par rapport à la transcription manuelle. Colonnes traduction : taille de la table de traduction filtrée et score Bleu.

Bleu de la sortie du système de RAP par rapport à la transcription de référence. Ce qui motive cette proposition est que le score Bleu prend en compte les n-grammes jusqu'à $n = 4$. Par ailleurs, nous donnons la taille de la table de traduction après filtrage sur la transcription à traduire, en nombre de couples de groupes de mots « recrutés » pour sa traduction.

Les données utilisées pour ces expériences sont l'ensemble de développement dev06 et l'ensemble d'évaluation eval07, de l'anglais vers l'espagnol. Les systèmes de reconnaissance ROVER et LIMSI sont décrits à la section 7.2.1. Pour le système LIMSI, nous pouvons activer le décodage par consensus (DC) ou extraire du treillis de mots l'hypothèse de probabilité *a posteriori* maximale (MAP, *maximum a posteriori*). Les trois traitements CassePonct, DisflNorm et Composés définis au chapitre 7 sont employés. Les résultats sont consignés dans le tableau 8.3.

Pour les deux ensembles dev06 et eval07, le système Rover obtient des taux WER largement inférieurs à ceux des autres systèmes. Sans surprise, sa traduction est aussi meilleure que celle des autres sorties.

La moitié supérieure du tableau (ensemble dev06) contient cependant une « inversion » comme nous espérions en trouver. Comme attendu, le taux WER de la sortie Limsi MAP est supérieur à celui de la sortie Limsi DC. Pourtant, la traduction de la première est très légèrement meilleure que celle de la seconde. Le score Bleu-RAP que nous proposions ne permet pas de l'expliquer, car il est toujours négativement corrélé avec le taux WER. Tout au plus, on constate que le nombre de couples de groupes de mots après filtrage est légèrement plus élevé pour Limsi MAP que pour Limsi DC et Rover. Ce n'est en aucun cas une indication que la traduction sera meilleure, mais cela tend à confirmer notre intuition que réaliser un décodage par consensus casse les n-grammes, et qu'effectuer une combinaison ROVER en casse plus encore. Toutefois, l'inversion entre les scores Bleu de Limsi MAP et Limsi DC ne se reproduit pas sur l'ensemble eval07. Ces expériences ne permettent donc pas de conclure, sinon pour confirmer que le taux WER de la RAP reste dans ce cas le meilleur prédicteur de la performance globale de la reconnaissance suivie de la traduction.

8.3.2 Influence des taux d'insertion et de suppression

Les résultats d'un système de reconnaissance de la parole dépendent bien entendu de ses modèles acoustique et de langage. Mais son comportement peut aussi être ajusté à l'aide de quelques pénalités, comme typiquement une pénalité de mot. Toutes choses égales par ailleurs, avec une pénalité de mot élevée, le système aura tendance à produire peu de mots, et inversement une pénalité de mot faible voire négative provoquera l'insertion de nombreux mots. Normalement, cette pénalité est ajustée de sorte à minimiser le taux de mots erronés sur un ensemble de développement. Nous proposons dans cette section de faire varier la pénalité de mot sur un large intervalle et d'observer les effets sur la qualité de la traduction.

Il s'agit ici d'expériences préliminaires. Leur objectif n'est pas tant d'améliorer un score que d'observer et comprendre les évolutions des scores. C'est pourquoi les expériences sont effectuées sur un seul ensemble, au lieu de deux habituellement. Par ailleurs, afin d'éviter d'éventuels biais dus à une trop bonne adaptation des systèmes de reconnaissance et de traduction à l'ensemble dev06, les expériences sont menées sur eval07. Le système de reconnaissance est celui utilisé à la section précédente, et tous les traitements définis au chapitre 7 sont à nouveau employés. Pour des raisons de temps, seule la première passe du système de traduction est ici utilisée. Aucun réglage n'est effectué au cours de ces expériences.

La figure 8.3 présente les taux de substitutions, d'insertions et de suppressions qui constituent le taux global de mots erronés de transcription automatique, en fonction de la pénalité de mot p. La ligne pointillée verticale à $p = 3$ indique le point de fonctionnement normal du système de reconnaissance. Le taux de mots erronés y atteint 9,14 %. L'écart avec le taux rapporté à la section précédente (9,33 %) s'explique par de légères différences dans les treillis de mots et les transcriptions de référence, qui ont été mises à jour. On constate que le système est réglé de façon quasi-optimale : le minimum absolu du taux WER vaut 9,13 % et est atteint pour $p = 0$. Le taux d'insertions décroît et celui de suppressions croît à mesure que la pénalité de mot augmente. Ils se croisent dans le voisinage de $p \approx 33$.

La figure 8.4 rassemble quatre graphiques montrant l'évolution des scores Bleu, NIST, WER et PER en fonction de la pénalité de mot. Nous attirons l'attention du lecteur sur l'*inversion* des axes des ordonnées de droite pour les scores Bleu et NIST. Dans les quatre cas, plus la courbe est basse, meilleure est la traduction.

Le score Bleu au point de fonctionnement habituel s'élève à 38,95 %. On observe que son évolution en deçà et au-delà de $p = 3$ est dissymétrique. Pour des valeurs de $p > 3$, le score Bleu perd jusqu'à 2 points, alors que le taux WER de la reconnaissance n'augmente que de 2,5 points. À l'inverse, lorsque $p < 3$, le taux WER se dégrade de 4,3 points mais le score Bleu ne perd que 1 point. La pénalité de brièveté, dont l'impact augmente a mesure que le nombre de mots diminue, explique cette asymétrie. Dans cette expérience, il se trouve que le score Bleu maximal est atteint précisément lorsque les taux d'insertions et de suppressions se croisent. À ce point, le taux WER de la reconnaissance est de 9,54 %, soit 0,40 point de plus que le point de fonctionnement normal, et le score Bleu atteint 39,13 %, une amélioration de 0,2 point sur le score de référence.

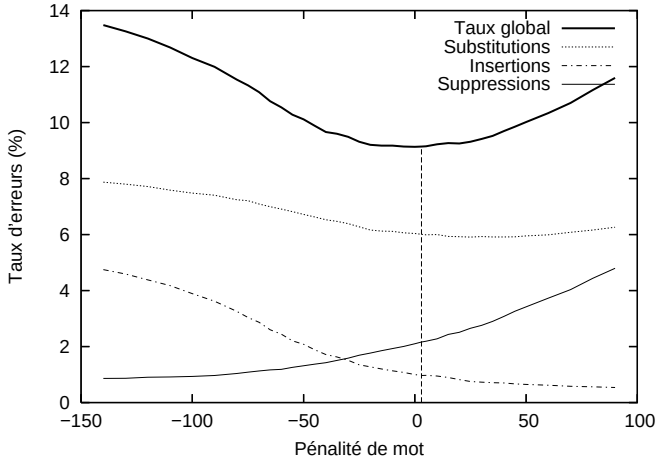


FIG. 8.3 – Évolutions du taux global de mots erronés, et des taux de substitutions, d'insertions et de suppressions en fonction de la pénalité de mot.

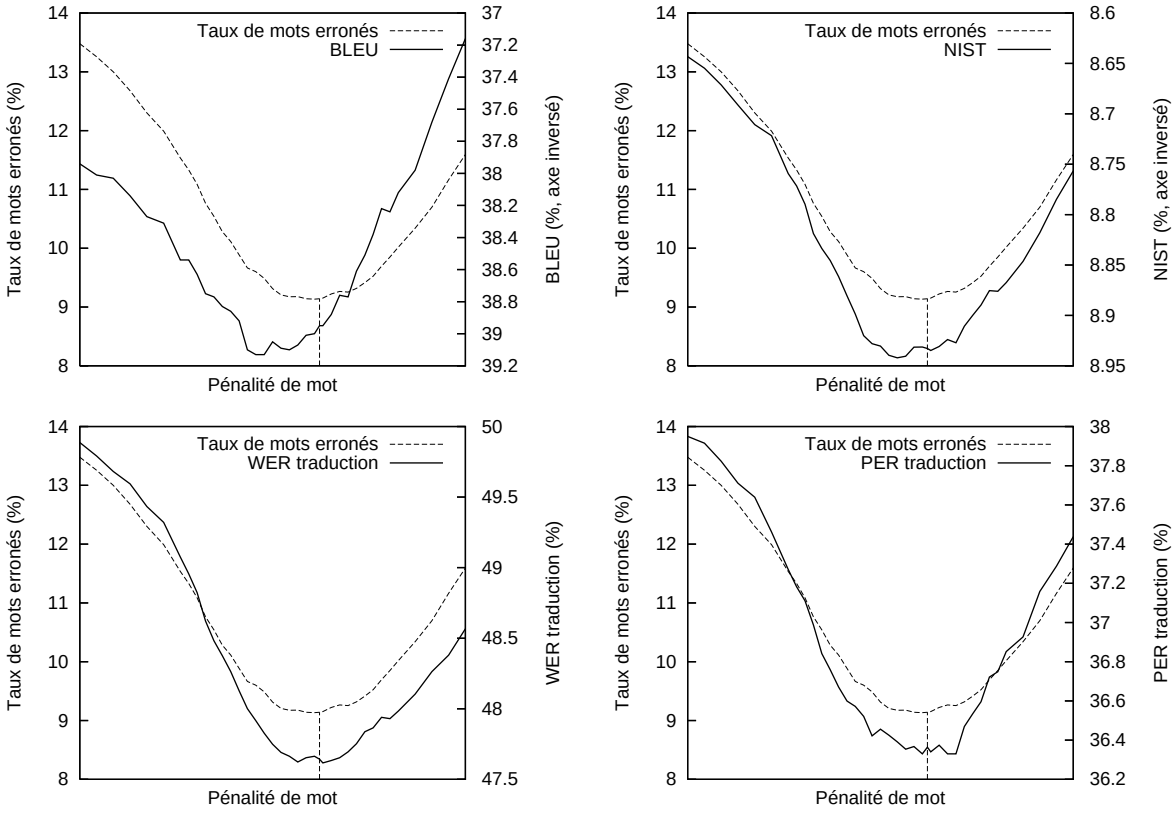


FIG. 8.4 – Évolutions de plusieurs mesures automatiques de la qualité de traduction en fonction de la pénalité de mot. La ligne verticale à $p = 3$ indique le point normal de fonctionnement du système de reconnaissance.

Comme le score Bleu, le score NIST est une moyenne de précisions affublée d'une pénalité de brièveté. Cependant, les variations du score NIST sont assez différentes de celles du score Bleu. Visuellement, elles suivent les variations du taux WER de la reconnaissance, et le meilleur score NIST, atteint pour $p = -15$, est pratiquement identique à celui pour $p = 3$ (8,942 contre 8,933).

Les taux de mots erronés de traduction suivent approximativement les variations du taux de mots erronés de la reconnaissance. Ce résultat peut s'expliquer intuitivement : les erreurs inhérentes à la traduction mises à part, les erreurs de reconnaissance ont tendance à générer des erreurs de traduction supplémentaires, et les scores WER et PER pénalisent symétriquement le manque ou le surnombre de mots cible. On remarque que les variations du score PER sont plus irrégulières que celles du score WER. Mais pour le WER comme pour le PER, le minimum absolu est obtenu très proche de la pénalité de référence, et la différence avec le taux d'erreur à $p = 3$ est insignifiante.

Pour conclure, ces expériences montrent qu'à nouveau, le taux WER de reconnaissance est un bon prédicteur de la qualité de traduction. Seule la pénalité de brièveté du score Bleu peut inciter à ajuster le système de reconnaissance de manière à insérer légèrement plus de mots. Il serait intéressant de confirmer ou d'infirmer ces observations avec d'autres systèmes de reconnaissance et d'autres mesures automatiques de la qualité de traduction.

Chapitre 9

Conclusion et perspectives

Les progrès de la recherche depuis un demi siècle ont permis à la reconnaissance de la parole et à la traduction automatique d'entrer dans nos vies quotidiennes. La reconnaissance automatique est maintenant en mesure de diriger automatiquement vers les services appropriés les clients contactant des centres d'appels. De son côté, la traduction automatique rend accessibles tous les jours des millions de pages web aux internautes du monde entier. La traduction de la parole est un thème de recherche plus récent, car elle combine ces deux problèmes scientifiques complexes. Mais on imagine sans mal les applications qu'elle rend possibles : systèmes de réservation multilingues, aide au tourisme, indexation cross-lingue de contenus multimédias, assistant pour l'échange d'informations et la négociation, etc.

Cette thèse a porté sur la traduction automatique et plus particulièrement sur la traduction de la parole reconnue automatiquement.

Conformément à l'état de l'art, les systèmes de traduction développés au cours de cette thèse reposent sur des modèles statistiques. Ces modèles sont constitués d'un grand nombre de paramètres — de l'ordre de plusieurs millions. Leur entraînement nécessite des textes *parallèles* : des milliers ou de préférence des millions de phrases traductions l'une de l'autre, grâce auxquelles les paramètres des modèles sont estimés. La tâche retenue tout au long de cette thèse est la traduction des discours des députés européens aux sessions plénières du parlement européen, entre l'anglais et l'espagnol.

Nos recherches ont débuté grâce à un décodeur pour le modèle « IBM-4 », un modèle à base de mots performant. Ce décodeur a été entièrement développé au cours de cette thèse en s'inspirant de la littérature à ce sujet et son fonctionnement a été décrit en détail. Il est notamment capable de produire des treillis de mots correspondant à l'espace de recherche exploré. Ceci a été mis à profit lors du développement d'un système de traduction à deux passes autour de ce décodeur. Nos expériences ont montré qu'un modèle quadrigramme à repli n'améliorait que faiblement les performances obtenues par un modèle trigramme : entre 0,2 et 0,6 point Bleu suivant le sens de traduction. En revanche, le modèle quadrigramme *neuronal*, développé originellement pour la reconnaissance de la parole, a obtenu de très bonnes performances. Il a permis d'améliorer le score Bleu de 1 à 1,5 points selon la direction de traduction, et d'améliorer significativement les autres mesures automatiques.

Nos recherches sur une meilleure interaction entre la reconnaissance et la traduction ont commencé avec ce système de traduction. À notre connaissance, nous avons été les premiers à mesurer l'impact du taux de mots erronés de la reconnaissance sur les performances de la traduction, et d'évaluer séparément les impacts respectifs du modèle de langage source et du modèle acoustique. Par ailleurs, au cours d'expériences de traduction de listes de n meilleures hypothèses de la reconnaissance, il est apparu qu'il était indispensable d'intégrer le score du modèle de langage source pour (légèrement) améliorer le résultat de référence, obtenu en traduisant la meilleure hypothèse de reconnaissance. Ceci semblait contredire l'analyse théorique effectuée par [Ney, 1999], mais s'explique par le fait que le modèle statistique de la quantité $\Pr(\mathbf{f}|\mathbf{e})$ est nettement moins performant pour prédire l'ordre des mots de la phrase $\mathbf{f}$ que le modèle de langage source $\Pr(\mathbf{f})$.

D'après notre propre expérience et les articles parus à ce sujet, on peut remarquer que la prise en compte de l'ambiguïté de la sortie de la reconnaissance automatique impose des compromis différents selon la méthode utilisée :

- En traduisant le treillis de mots produit par la reconnaissance, on perd la possibilité de calculer l'hypothèse de reconnaissance obtenue par décodage consensus. De plus, plusieurs études — dont la nôtre — ont montré l'importance du modèle de langage source. Or plus ce modèle de langage a une portée élevée, plus le treillis est gros et redondant.
- En traduisant un réseau de confusion, typiquement obtenu comme lors du décodage consensus, on perd cette fois l'accès aux scores linguistiques et acoustiques, au profit des seules probabilités *a posteriori*.
- En traduisant une liste de n meilleures hypothèses enfin, on ne peut que choisir entre les deux problèmes ci-dessus. Ou bien les n meilleures hypothèses sont extraites du treillis initial, et l'on perd le décodage consensus, ou bien elles sont extraites du réseau de confusion, et l'on perd les scores acoustiques et linguistiques. Dans les deux cas, cette solution aura un coût en temps de traitement proportionnel à n.
  Cette solution a toutefois l'avantage d'être la plus simple vis à vis traitements intermédiaires entre la reconnaissance et la traduction. Par exemple, la restauration de la casse et de la ponctuation, ainsi que le traitement des mots composés et autres renormalisations, dont nous avons vu l'importance pour la traduction, sont plus complexes à effectuer sur un treillis ou un réseau de confusion.

L'avènement du décodeur libre Moses au milieu de l'année 2006 eut un impact décisif sur cette thèse. Moses est un décodeur par groupes de mots à l'état de l'art. Il a pu être intégré au cadre expérimental mis en place pour le décodeur précédent et a immédiatement obtenu de très bonnes performances. Cela nous a donné l'opportunité de poursuivre nos recherches avec un autre modèle de traduction. Nous avons envisagé une collaboration entre les deux décodeurs, mais elle n'a malheureusement pas produit l'amélioration espérée.

Comme le premier décodeur, Moses est au centre d'une stratégie à deux passes, qui ne diffèrent que par le modèle de langage cible. D'après nos expériences, un modèle de langage quadrigramme à repli entraîne un gain compris entre 0,5 et 1 point Bleu, mais est susceptible de ne pas améliorer toutes les mesures automatiques de qualité de la traduction. Le modèle quadrigramme neuronal s'est à nouveau avéré très performant

dans les deux sens de traduction, avec des gains compris entre 1 et 2 points BLEU et des améliorations sensibles pour toutes les mesures automatiques (sauf dans un cas).

Les systèmes de traduction mis en œuvre dans cette thèse sont relativement simples en terme de nombre de fonctions caractéristiques. Ils ont « pourtant » été très compétitifs à la dernière évaluation TC-STAR, en février 2007. On peut supposer que multiplier les fonctions caractéristiques accroît les difficultés numériques pour l'optimisation de leurs poids et fait courir un risque de sur-apprentissage du corpus de développement.

Au cœur des systèmes de traduction par groupes de mots comme Moses, se trouve la table de traduction, sorte de « dictionnaire bilingue ». Les scores qu'elle contient sont le résultat de choix heuristiques. Nous avons proposé un algorithme inspiré de celui du Perceptron pour modifier de façon discriminante ces scores en observant les erreurs de traduction sur un ensemble de développement. Cet algorithme a procuré un gain de 1,2 points BLEU sur l'ensemble eval06, utilisé comme ensemble de contrôle, et de 0,9 point BLEU sur les données eval07. Toutefois, ces gains n'ont pas été confirmés sur une autre tâche. Nous pensons que ces résultats contrastés pourraient être dus par un défaut de cohérence entre les scores de la table de traduction. Un lissage des modifications apportées à ces scores pourrait permettre de bénéficier de la discrimination tout en préservant une bonne capacité à généraliser. Au cours de ces expériences, nous avons constaté que l'ajout, aux données d'entraînement, d'un petit nombre de traductions de bonne qualité et très proches de la tâche d'évaluation pouvait s'avérer bénéfique.

Concernant la traduction de la parole, nous nous sommes ensuite placés dans le cadre de la traduction d'un flux de mots produit par un système de reconnaissance « inconnu ». Plusieurs traitements spécifiques à la parole sont utiles, pour gérer les répétitions et autres disfluences. Nos expériences ont montré l'importance de transformer les données à traduire pour les faire ressembler aux données d'entraînement du système : ces traitements permettent de gagner entre 0,5 et 1,5 points BLEU selon les systèmes de reconnaissance. Mais de façon surprenante, nos expériences ont montré que la *ponctuation* était au moins aussi importante que les mots, en matière de divergence entre données d'entraînement et données de test. Notre algorithme d'insertion de ponctuations a pour seul critère la *quantité* de ponctuations insérées, et a procuré un gain de plus de trois points BLEU par rapport à la traduction de la sortie ROVER telle quelle.

Par ailleurs, certains traitements courants en reconnaissance de la parole, comme le décodage consensus et le ROVER, peuvent améliorer le taux de mots erronés mais « cassent » intuitivement les groupes de mots source. Cependant, nos recherches tendent à montrer que l'abaissement du taux WER est presque toujours bénéfique à la traduction.

Enfin, nous avons modifié le système de reconnaissance de manière à lui faire insérer ou supprimer plus de mots. Bien que le score BLEU puisse légèrement bénéficier d'un taux d'insertion plus élevé, le taux de mots erronés WER semble le bon critère à minimiser par la reconnaissance pour obtenir les meilleures performances de traduction de la parole.

Ce résultat est rassurant : il signifie que les chercheurs et chercheuses en reconnaissance de la parole peuvent continuer à abaisser les taux de mots reconnus erronés en étant confiants que cela bénéficiera aussi à la traduction. Mais nous avons vu que

l'interaction entre reconnaissance et traduction ne se limitait pas à ce taux de mots erronés, calculé *sans prendre en compte* la casse et la ponctuation. Sans revenir plus longuement sur l'importance de la casse et plus généralement de la normalisation des mots, la ponctuation et la segmentation en phrases ont un impact très important sur la qualité de la traduction, en tout cas telle qu'elle est évaluée par les mesures automatiques usuelles. Une piste prometteuse de recherche pourrait ainsi être de caractériser la ponctuation et la segmentation que doit produire la reconnaissance de la parole pour des performances de traduction optimales. Par ailleurs, la meilleure interface entre reconnaissance et traduction n'est pas encore déterminée, entre réseau de confusion et treillis de mots.

Plus généralement à propos de la traduction automatique, l'auteur souhaiterait s'intéresser à la réduction du nombre de paramètres libres dans la table de traduction. Pour ce faire, l'élagage est une piste, mais une autre pourrait être le *liage* de certains scores. Un objectif par exemple serait d'expliciter le lien entre les couples $\langle e_1, f_1 \rangle$, $\langle e_2, f_2 \rangle$ et $\langle e_1 e_2, f_1 f_2 \rangle$. Ceci aurait pour conséquence de lisser les scores et, peut-être, de permettre leur apprentissage discriminant.

Ce ne sont que quelques pistes de réflexion, le domaine du traitement de la langue et de la parole foisonne de défis passionnants à relever.

Bibliographie

Yasuhiro Akiba, Marcello Federico, Noriko Kando, Hiromi Nakaiwa, Michael Paul, and Jun'ichi Tsujii. Overview of the IWSLT04 evaluation campaign. In *Proc. of the Intl. Workshop on Spoken Language Translation*, pages 1–12, Kyoto, Japan, 2004.

Yaser Al-Onaizan and Lidia Mangu. Arabic ASR and MT integration for gale. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1285–1288, Honolulu, Hawaii, April 2007.

Yaser Al-Onaizan and Kishore Papineni. Distortion models for statistical machine translation. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 529–536, Sydney, Australia, 2006.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz J. Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical machine translation. Technical report, Johns Hopkins University, 1999.

Jan W. Amtrup, Hamid Mansouri Rad, Karine Megerdoomian, and Rémi Zajac. Persian-english machine translation : An overview of the shiraz project. Memoranda in computer and cognitive science, NMSU, CRL, 2000.

Abhishek Arun and Philipp Koehn. Online learning methods for discriminative training of phrase based statistical machine translation. In *Proc. of MT Summit*, pages 15–20, Copenhagen, Danemark, September 2007.

Bogdan Babych and Anthony Hartley. Modelling legitimate translation variation for automatic evaluation of MT quality. In *Proc. of the Intl. Conf. on Language Resources and Evaluation*, pages 833–836, Lisbon, Portugal, May 2004.

Nguyen Bach, Matthias Eck, Paisarn Charoenpornsawat, Thilo Köhler, Sebastian Stüker, ThuyLinh Nguyen, Roger Hsiao, Alex Waibel, Stephan Vogel, Tanja Schultz, and Alan W. Black. The CMU TransTac 2007 eyes-free and hands-free two-way speech-to-speech translation system. In *Proc. of the Intl. Workshop on Spoken Language Translation*, Trento, Italy, October 2007.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5 :179–190, March 1983.

Satanjeev Banerjee and Alon Lavie. METEOR : An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of the ACL*

Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization, Ann Arbor, Michigan, June 2005.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1) :164–171, 1970.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, John D. Lafferty, Robert L. Mercer, Harry Printz, and Luboš Ureš. The candide system for machine translation. In *Proc. of the Workshop on Human Language Technology*, pages 157–162, Plainsboro, NJ, 1994.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1) : 39–71, 1996. ISSN 0891-2017.

Frank Vanden Berghen and Hugues Bersini. CONDOR, a new parallel, constrained extension of Powell's UOBYQA algorithm : Experimental results and comparison with the DFO algorithm. *Journal of Computational and Applied Mathematics*, 181 : 157–175, 2005.

Nicola Bertoldi and Marcello Federico. A new decoder for spoken language translation based on confusion networks. In *Proc. of the Automatic Speech Recognition and Understanding Workshop (ASRU)*, Puertorico, USA, December 2005.

Nicola Bertoldi, Richard Zens, and Marcello Federico. Speech translation by confusion network decoding. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1297–1300, Honolulu, Hawaii, April 2007.

Laurent Besacier, Herve Blanchon, Yannick Fouquet, Jean-Philippe Guilbaud, Stephane Helme, Sylviane Mazenot, Daniel Moraru, and Dominique Vaufreydaz. Speech translation for French in the NESPOLE ! european project. In *Proc. of Eurospeech*, Aalborg, Danemark, September 2001.

Hélène Bonneau-Maynard, Alexandre Allauzen, Daniel Déchelotte, and Holger Schwenk. Combining morphosyntactic enriched representation with n-best reranking in statistical translation. In *Proc. of the NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*, pages 65–71, Rochester, New York, April 2007.

Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2) :79–85, 1990.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation : Parameter estimation. *Computational Linguistics*, 19(2) :263–311, 1993.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the role of Bleu in machine translation research. In *Proc. of EACL*, pages 249–256, Trento, Italy, April 2006.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (Meta-) evaluation of machine translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June 2007.

Roldano Cattoni, Nicola Bertoldi, and Marcello Federico. Punctuating confusion networks for speech translation. In *Proc. of InterSpeech*, pages 2453–2456, Antwerp, Belgium, August 2007.

Mauro Cettolo and Marcello Federico. Minimum error training of log-linear translation models. In *Proc. of the Intl. Workshop on Spoken Language Translation*, pages 103–106, Kyoto, Japan, 2004.

Mauro Cettolo, Marcello Federico, Nicola Bertoldi, Roldano Cattoni, and Boxing Chen. A look inside the ITC-irst SMT system. In *Proc. of MT Summit*, pages 451–457, Phuket, Thailand, September 2005.

Paisarn Charoenpornsawat, Virach Sornlertlamvanich, and Thatsanee Charoenporn. Improving translation quality of rule-based machine translation. In *Proc. of the COLING Workshop on Machine translation in Asia*, pages 1–6, Taipei, Taiwan, 2002.

Marta R. Costa-jussà, Josep M. Crego, Adrià de Gispert, Patrik Lambert, Maxim Khalilov, José B. Mariño, José A. R. Fonollosa, and Rafael Banchs. TALP phrase-based statistical translation system for European language pairs. In *Proc. of the Workshop on Statistical Machine Translation*, pages 142–145, New York City, June 2006.

Deborah Coughlin. Correlating automated and human assessments of machine translation quality. In *Proc. of MT Summit*, pages 63–70, New Orleans, LA, September 2003.

Josep M. Crego, José B. Mariño, and Adrià de Gispert. An ngram-based statistical machine translation decoder. In *Proc. of Interspeech*, pages 3185–3188, Lisbon, Portugal, September 2005.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5) :1470–1480, 1972.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4) :380–393, 1997.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. Why generative phrase models underperform surface heuristics. In *Proc. on the Workshop on Statistical Machine Translation*, pages 31–38, New York City, June 2006.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

George Doddington. Automatic evaluation of machine translation quality using n-gram cooccurrence statistics. In *Proc. of the Human Language Technology Conference*, pages 128–132, San Diego, CA, March 2002.

Loïc Dugast, Jean Senellart, and Philipp Koehn. Statistical post-editing on SYSTRAN's rule-based translation system. In *Proc. of the Workshop on Statistical Machine Translation*, pages 220–223, Prague, Czech Republic, June 2007.

Daniel Déchelotte, Holger Schwenk, Jean-Luc Gauvain, Olivier Galibert, and Lori Lamel. Investigating translation of parliament speeches. In *Proc. of the Workshop on Automatic Speech Recognition and Understanding*, pages 116–120, San Juan, Porto Rico, December 2005.

Daniel Déchelotte, Holger Schwenk, and Jean-Luc Gauvain. Transcription et traduction de débats parlementaires. In *Actes du congrès francophone Reconnaissance des Formes et Intelligence Artificielle*, Tours, France, January 2006a.

Daniel Déchelotte, Holger Schwenk, and Jean-Luc Gauvain. The 2006 LIMSI statistical machine translation system for TC-Star. In *Proc. of the TC-STAR Workshop on Speech-to-Speech Translation*, pages 25–30, Barcelona, Spain, June 2006b.

Daniel Déchelotte, Holger Schwenk, Gilles Adda, and Jean-Luc Gauvain. Improved machine translation of speech-to-text outputs. In *Proc. of the NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*, pages 2441–2444, Antwerp, Belgium, August 2007a.

Daniel Déchelotte, Holger Schwenk, Hélène Bonneau-Maynard, Alexandre Allauzen, and Gilles Adda. A state-of-the-art statistical machine translation system based on Moses. In *Proc. of MT Summit*, pages 127–133, Copenhagen, Danemark, September 2007b.

John G. Fiscus. A post-processing system to yield reduced word error rates : Recognizer output voting error reduction (ROVER). In *Proc. of the Workshop on Automatic Speech Recognition and Understanding*, pages 347–354, Santa Barbara, CA, December 1997.

George Foster and Roland Kuhn. Mixture-model adaptation for SMT. In *Proc. of the Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic, June 2007.

George Foster, Roland Kuhn, and Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 53–61, Sydney, Australia, July 2006.

Pascale Fung, Yi Liu, Yongsheng Yang, Yihai Shen, and Dekai Wu. A grammar-based Chinese to English speech translation system for portable devices. In *Proc. of the Intl. Conf. on Spoken Language Processing*, pages 1577–1580, Jeju Island, South Korea, October 2004.

Mark Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12(2) :75–98, April 1998.

Mark Gales, Xunying Liu, Rohit Sinha, Phil C. Woodland, Kai Yu, Spyros Matsoukas, Tim Ng, Kham Nguyen, Long Nguyen, Jean-Luc Gauvain, Lori Lamel, and Abdel Messaoudi. Speech recognition system combination for machine translation. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1277–1280, Honolulu, USA, 2007.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July 2006.

Yuqing Gao. Coupling vs. unifying : Modeling techniques for speech-to-speech. In *Proc. of EuroSpeech*, pages 365–368, Geneva, Switzerland, September 2003.

Yuqing Gao, Bowen Zhou, Ruhi Sarikaya, Mohamed Afify, Hong-Kwang Kuo, Weizhong Zhu, Yonggang Deng, Charles Prosser, Wei Zhang, and Laurent Besacier. IBM MASTOR system : Multilingual automatic speech-to-speech translator. In *Proc. of the Intl. Workshop on Medical Speech Translation*, pages 53–56, New York, NY, June 2006.

Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. The LIMSI broadcast news transcription system. *Speech Communication*, 37(2) :98–108, 2002.

Ulrich Germann. Greedy decoding for statistical machine translation in almost linear time. In *Proc. of the Conf. of NAACL on Human Language Technology*, pages 1–8, Edmonton, Canada, 2003.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optimal decoding for machine translation. In *Proc. of the Meeting of the Association for Computational Linguistics*, pages 228–235, Toulouse, France, 2001.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proc. of EAMT*, pages 133–142, Budapest, Hungary, May 2005.

Uchida Hiroshi and Zhu Meiying. Interlingua for multilingual machine translation. In *Proc. of MT Summit : International Cooperation for Global Communication*, pages 157–169, Kobe, Japan, July 1993.

Frederick Jelinek. Continuous speech recognition by statistical methods. *Proc. of IEEE*, 64(4) :532–556, April 1976.

Philipp Koehn. Pharaoh : a beam search decoder for phrase-based statistical machine translation models. In *AMTA 2004*, Washington, USA, 2004.

Philipp Koehn and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June 2007.

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. of the Human Language Technology Conference (HLT-NAACL)*, pages 127–133, Edmonton, Canada, May 2003.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses : Open source toolkit for statistical machine translation. In *ACL, demonstration session*, Prague, Czech Republic, June 2007.

Roland Kuhn, Denis Yuen, Michel Simard, Patrick Paul, George Foster, Eric Joanis, and Howard Johnson. Segment choice models : feature-rich models for global distortion in statistical machine translation. In *Proc. of the Conf. on Human Language Technology*, pages 25–32, New York, New York, 2006.

Shankar Kumar, Yonggang Deng, and William Byrne. A weighted finite state transducer translation template model for statistical machine translation. *Nat. Lang. Eng.*, 12(1) :35–75, 2006.

Akira Kurematsu and Tsuyoshi Morimoto. *Automatic speech translation : fundamental technology for future cross-language communication*. Gordon and Breach, Amsterdam, Pays-Bas, 1996.

Lori Lamel, Jean-Luc Gauvain, Gilles Adda, Claude Barras, Eric Bilinski, Olivier Galibert, Agusti Pujol, Holger Schwenk, and Xuan Zhu. The LIMSI 2006 TC-STAREPPS transcription systems. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 997–1000, Honolulu, USA, 2007.

John Lee and Stephanie Seneff. Interlingua-based translation for language learning systems. In *Proc. of the Workshop on Automatic Speech Recognition and Understanding*, pages 133–138, San Juan, Porto Rico, December 2005.

Chris J. Leggetter and Phil C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, 9(2) :171–185, 1995.

Lori Levin, Alon Lavie, Monika Woszczyna, Donna Gates, Marsal Gavaldá, Detlef Koll, and Alex Waibel. The janus-iii translation system : Speech-to-speech translation in multiple domains. *Machine Translation*, 15(1-2) :3–25, 2000.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 761–768, Sydney, Australia, July 2006.

Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding consensus among words : Lattice-based word error minimization. In *Proc. of Eurospeech*, pages 495–498, Budapest, Hungary, September 1999.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 133–139, Philadelphia, PA, July 2002.

Lambert Mathias and William Byrne. Statistical phrase-based speech translation. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 561–564, Toulouse, France, May 2006.

Spyros Matsoukas, Ivan Bulyko, Bing Xiang, Kham Nguyen, Richard Schwartz, and John Makhoul. Integrating speech recognition and machine translation. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1281–1284, Honolulu, Hawaii, April 2007.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. On the integration of speech recognition and statistical machine translation. In *Proc. of Interspeech*, pages 3177–3180, Lisbon, Portugal, September 2005a.

Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. Evaluating machine translation output with automatic sentence segmentation. In *Proc. of Intl. Workshop on Spoken Language Translation*, Pittsburgh, USA, October 2005b.

Evgeny Matusov, Hermann Ney, and Ralf Schlüter. Phrase-based translation of speech recognizer word lattices using loglinear model combination. In *Proc. of the Automatic Speech Recognition and Understanding Workshop (ASRU)*, Puertorico, USA, December 2005c.

Evgeny Matusov, Arne Mauser, and Hermann Ney. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proc. of the Intl. Workshop on Spoken Language Translation*, pages 158–165, Kyoto, Japan, 2006.

Arne Mauser, Richard Zens, Evgeny Matusov, Sasa Hasan, and Hermann Ney. The RWTH statistical machine translation system for the IWSLT 2006 evaluation. In *Proc. of the Intl. Workshop on Spoken Language Translation*, pages 103–110, Kyoto, Japan, November 2006.

Florian Metze, John McDonough, Hagen Soltau, Alex Waibel, Alon Lavie, Susanne Burger, Chad Langley, Lori Levin, Tanja Schultz, Fabio Pianesi, Roldano Cattoni, Gianni Lazzari, Nadia Mana, and Emanuele Pianta. The NESPOLE! speech-to-speech translation system. In *Proc. of the Intl. Conf. on Human Language Technology Research*, pages 378–383, San Diego, California, March 2002.

Teruko Mitamura, Eric H. Nyberg, and Jaime G. Carbonell. An efficient interlingua translation system for multi-lingual document production. In *Proc. of the Machine Translation Summit*, Washington DC, USA, July 1991.

Makoto Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. In *Proc. of the Intl. NATO symposium on Artificial and human intelligence*, pages 173–180, Lyon, France, 1984.

Toshiaki Nakazawa, Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. Example-based machine translation based on deeper NLP. In *Proc. of the Intl. Workshop on Spoken Language Translation : Evaluation Campaign on Spoken Language Translation*, pages 64–70, Kyoto, Japan, November 2006.

Hermann Ney. Speech translation : Coupling recognition and translation. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1149–1152, Pheonix, Arizona, May 1999.

Hermann Ney. One decade of statistical machine translation : 1996-2005. In *Proc. of MT Summit*, pages i–12–17, Phuket, Thailand, September 2005.

Franz J. Och. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan, July 2003.

Franz J. Och. Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. Master's thesis, Universität Erlangen-Nürnberg, Germany, 1995.

Franz J. Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL 2002*, pages 295–302, 2002.

Franz J. Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1) :19–51, 2003.

Franz J. Och and Hermann Ney. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4) :417–449, 2004.

Franz J. Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Copora*, pages 20–28, 1999.

Franz J. Och, Nicola Ueffing, and Hermann Ney. An efficient A* search algorithm for statistical machine translation. In *Data-Driven Machine Translation Workshop*, pages 55–62, Toulouse, France, July 2001.

Franz J. Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. A smorgasbord of features for statistical machine translation. In *Proc. of HLT-NAACL*, pages 161–168, Boston, MA, May 2004.

Kishore Papineni. Discriminative training via linear programming. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 561–564, Phoenix, AR, March 1999.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU : a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting of the ACL*, pages 311–318, University of Pennsylvania, 2002.

Michael Paul, Takao Doi, Youngsook Hwang, Kenji Imamura, Hideo Okuma, and Eiichiro Sumita. Nobody is perfect : ATR's hybrid approach to spoken language translation. In *Proc. of Intl. Workshop on Spoken Language Translation*, Pittsburgh, USA, October 2005.

Michael J.D. Powell. UOBYQA : Unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3) :555–582, 2002.

Vu H. Quan, Marcello Federico, and Mauro Cettolo. Integrated nbest re-ranking for spoken language translation. In *Proc. of Interspeech*, pages 3181–3184, Lisbon, Portugal, September 2005.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. of the Meeting of the Association for Computational Linguistics*, pages 47–54, Barcelona, Spain, July 2004.

Frank Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 :386–408, 1958.

Fatiha Sadat, Howard Johnson, Akakpo Agbago, George Foster, Roland Kuhn, Joel Martin, and Aaron Tikuisis. Portage : A phrase-based machine translation system. In *Proc. of the ACL Workshop on Building and Using Parallel Texts : Data-Driven Machine Translation and Beyond*, pages 133–136, Ann Arbor, MI, 2005 2005.

Shirin Saleem, Szu-Chen Jou, Stephan Vogel, and Tanja Schultz. Using word lattice information for a tighter coupling in speech translation systems. In *Proc. of the Intl. Conf. on Speech and Language Processing*, pages 41–44, Jeju Island, South Korea, October 2004.

Ruhi Sarikaya, Bowen Zhou, Daniel Povey, Mohamed Afify, and Yuqing Gao. The impact of ASR on speech-to-speech translation performance. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 1289–1292, Honolulu, Hawaii, April 2007.

Holger Schwenk. Continuous space language models. *Comput. Speech Lang.*, 21(3) : 492–518, 2007.

Holger Schwenk, Daniel Déchelotte, and Jean-Luc Gauvain. Continuous space language models for statistical machine translation. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 723–730, Sydney, Australia, July 2006.

Holger Schwenk, Daniel Déchelotte, Hélène Bonneau-Maynard, and Alexandre Allauzen. Modèles statistiques enrichis par la syntaxe pour la traduction automatique. In *Actes de la conf. sur le Traitement Automatique des Langues Naturelles*, pages 253–262, Toulouse, France, June 2007.

Jean Senellart, Péter Dienes, and Tamás Váradi. New generation SYSTRAN translation system. In *Proc. of MT Summit*, Santiago de Compostela, Spain, September 2001.

Dana Shapira and James A. Storer. Edit distance with move operations. *J. of Discrete Algorithms*, 5(2) :380–392, 2007.

Wade Shen, Richard Zens, Nicola Bertoldi, and Marcello Federico. The JHU workshop 2006 IWSLT system. In *Proc. of the Intl. Workshop on Spoken Language Translation*, pages 59–63, Kyoto, Japan, 2006.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. Rule-based translation with statistical phrase-based post-editing. In *Proc. of the Workshop on Statistical Machine Translation*, pages 203–206, Prague, Czech Republic, June 2007.

Harold Somers. Review article : Example-based machine translation. *Machine Translation*, 14(2) :113–157, June 1999.

Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *Proc. of the Intl. Conf. on Spoken Language Processing*, pages II : 901–904, Denver, CO, USA, 2002.

Volker Strom, Anja Elsner, Wolfgang Hess, Walter Kasper, Alexandra Klein, Hans U. Krieger, Jörg Spilker, Hans Weber, and Günther Görz. On the use of prosody in a speech-to-speech translator. In *Proc. of the European Conf. on Speech Communication and Technology*, Rhodes, 1997.

Sebastian Stüker, Christian Fügen, Roger Hsiao, Shajith Ikbal, Qin Jin, Florian Kraft, Matthias Paulik, Martin Raab, Yik-Cheung Tam, and Matthias Wölfel. The ISL TC-Star spring 2006 ASR evaluation systems. In *TC-STAR Workshop on Speech-to-Speech Translation*, pages 139–144, Barcelona, Spain, June 2006.

Sylvain Surcin, Elke Lange, and Jean Senellart. Rapid development of new language pairs at SYSTRAN. In *Proc. of MT Summit*, pages 443–449, Copenhagen, Danemark, September 2007.

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. Toward a broad-coverage bilingual corpus for speech translation of travel conversation in the real world. In *Proc. of the Intl. Conf. on Language Resources and Evaluation*, Las Palmas, Canary Islands, Spain, May 2002.

Tc-Star, 2006. Les technologies du langage humain pour l'Europe. Téléchargement : `http://www.tc-star.org/pubblicazioni/ITC_francese.pdf`, April 2006.

Christoph Tillmann. A projection extension algorithm for statistical machine translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 1–8, Morristown, NJ, USA, 2003.

Christoph Tillmann and Tong Zhang. A discriminative global training algorithm for statistical MT. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 721–728, Sydney, Australia, July 2006.

Ismael García Varea, Franz J. Och, Hermann Ney, and Francisco Casacuberta. Improving alignment quality in statistical machine translation using context-dependent maximum entropy models. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 1–7, Taipei, Taiwan, 2002.

Bernard Vauquois. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress (2)*, pages 1114–1122, 1968.

Juan M. Vilar, Victor M. Jiménez, Juan C. Amengual, Antonio Castellanos, David Llorens, and Enrique Vidal. Text and speech translation by means of subsequential transducers. *Nat. Lang. Eng.*, 2(4) :351–354, 1996.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proc. of the Conf. on Computational Linguistics*, pages 836–841, Morristown, NJ, USA, 1996.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. The CMU statistical machine translation system. In *Proc. of MT-Summit IX*, New Orleans, LA, USA, September 2003.

Wolfgang Wahlster. *Verbmobil : Foundations of Speech-to-Speech Translation*. Springer verlag, 2000.

Ye-Yi Wang and Alex Waibel. Decoding algorithm in statistical machine translation. In *Proc. of the Conf. on European chapter of the Association for Computational Linguistics*, pages 366–372, Madrid, Spain, 1997.

John S. White. Approaches to black box MT evaluation. In *Proc. of MT Summit*, Luxembourg, Luxembourg, July 1995.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*, 23(3) :377–403, 1997.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proc. of the Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, 2001.

Richard Zens, Franz J. Och, and Hermann Ney. Phrase-based statistical machine translation. *Lecture Notes in Artificial Intelligence*, 2479 :18–32, September 2002.

Ruiqiang Zhang and Genichiro Kikui. Integration of speech recognition and machine translation : Speech recognition word lattice translation. *Speech Communication*, 48 (3–4) :321–334, March 2006.

Bowen Zhou, Daniel Déchelotte, and Yuqing Gao. Two-way speech-to-speech translation on handheld devices. In *Proc. of the Intl. Conf. on Spoken Language Processing*, pages 1637–1640, Jeju Island, South Korea, October 2004.

Bowen Zhou, Laurent Besacier, and Yuqing Gao. On efficient coupling of ASR and SMT for speech translation. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 101–104, Honolulu, Hawaii, April 2007.

Publications de l'auteur

Daniel Déchelotte and Holger Schwenk and Jean-Luc Gauvain and Olivier Galibert and Lori Lamel. Investigating translation of parliament speeches. In *Proc. of the Workshop on Automatic Speech Recognition and Understanding*, pages 116–120, San Juan, Porto Rico, December 2005.

Daniel Déchelotte and Holger Schwenk and Jean-Luc Gauvain. Transcription et traduction de débats parlementaires. In *Actes du congrès francophone Reconnaissance des Formes et Intelligence Artificielle*, Tours, France, January 2006.

Daniel Déchelotte and Holger Schwenk and Jean-Luc Gauvain. The 2006 limsi statistical machine translation system for tc-star. In *Proc. of the TC-STAR Workshop on Speech-to-Speech Translation*, pages 25–30, Barcelona, Spain, June 2006.

Holger Schwenk and Daniel Déchelotte and Jean-Luc Gauvain. Continuous space language models for statistical machine translation. In *Proc. of the Intl. Conf. on Computational Linguistics*, pages 723–730, Sydney, Australia, July 2006.

Hélène Bonneau-Maynard and Alexandre Allauzen and Daniel Déchelotte and Holger Schwenk. Combining morphosyntactic enriched representation with n-best reranking in statistical translation. In *Proc. of the NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*, pages 65–71, Rochester, New York, April 2007.

Holger Schwenk and Daniel Déchelotte and Hélène Bonneau-Maynard and Alexandre Allauzen. Modèles statistiques enrichis par la syntaxe pour la traduction automatique. In *Actes de la conf. sur le Traitement Automatique des Langues Naturelles*, pages 253–262, Toulouse, France, June 2007.

Daniel Déchelotte and Holger Schwenk and Gilles Adda and Jean-Luc Gauvain. Improved machine translation of speech-to-text outputs. In *Proc. of Interspeech*, pages 2441–2444, Antwerp, Belgium, August 2007.

Daniel Déchelotte and Holger Schwenk and Hélène Bonneau-Maynard and Alexandre Allauzen and Gilles Adda. A state-of-the-art statistical machine translation system based on moses. In *Proc. of MT Summit*, pages 127–133, Copenhagen, Danemark, September 2007.

Evgeny Matusov and Gregor Leusch and Rafael Banchs and Nicola Bertoldi and Daniel Déchelotte and Marcello Federico and Muntsin Kolss and Young-Suk Lee and José Mariño and Matthias Paulik and Salim Roukos and Holger Schwenk and Hermann Ney. System combination for machine translation of spoken and written language. *IEEE Transactions on Audio, Speech and Language Processing*, to appear.