Université de Montréal

Mémoires de traduction sous-phrastiques

par

Michel Simard

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

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de
Philosophiae Doctor (Ph.D.)
en informatique

février, 2003

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée:

Mémoires de traduction sous-phrastiques

présentée par

Michel Simard

a été évaluée par un jury composé des personnes suivantes:

Yoshua Bengio
(Président-rapporteur)

Jian-Yun Nie
(Directeur de recherche)

Philippe Langlais
(Membre du jury)

Harold Somers
(Examinateur externe)

Alain Polguère
(Représentant du doyen)

Thèse acceptée le _____

RÉSUMÉ

Cette thèse présente un ensemble de travaux concernant la mise sur pied d'un système de *mémoire de traduction sous-phrastique*. Les *mémoires de traduction* constituent une classe d'outils d'aide à la traduction, dont l'objectif est de faciliter la réutilisation de traductions existantes. Ce type de système repose sur un archivage systématique de la production d'un ou de plusieurs traducteurs. Étant donné un nouveau texte à traduire, le système recherche dans ses archives des segments de texte qui ont possiblement déjà été traduits. Au moment opportun, la traduction des segments ainsi trouvés est proposée au traducteur, qui est alors libre de les réutiliser à son gré.

Les systèmes existants opèrent sur des unités de texte qui s'apparentent à des phrases, c'est-à-dire que les segments de texte qui sont archivés dans la mémoire de traduction et ultimement proposés pour réutilisation sont des phrases complètes. Ceci limite dans une mesure importante l'utilisabilité de tels systèmes, parce que la répétition de phrases complètes est un phénomène rare dans la langue en général.

Dans cette thèse, on explore donc la possibilité de mettre sur pied un système de mémoire de traduction capable de proposer des traductions pour des groupes de mots représentant des sous-parties d'une phrase.

Dans un premier temps, on examine la nature exacte des unités sur lesquelles serait idéalement basé ce genre de système. Cette étude repose notamment sur l'analyse de l'utilisation réelle du *concordancier bilingue TransSearch*, un système qui s'apparente à une mémoire de traduction, mais qui permet aux utilisateurs de rechercher de façon interactive la traduction de suites de mots arbitraires. De cette analyse, il ressort que la très grande majorité des segments de texte auxquels s'intéressent les utilisateurs

de ce système coïncident avec des suites de syntagmes simples (en anglais: *syntactic chunks*). Partant de cette observation, on propose un système qui permettrait la réutilisation de tels segments de texte.

La mise sur pied d'un tel système requiert l'utilisation de méthodes de *repérage de traductions*, c'est-à-dire de mécanismes capable de localiser la traduction d'un segment de texte spécifique dans une traduction, un problème notoirement difficile. Différentes méthodes sont proposées, reposant sur des modèles statistiques de la traduction. Une évaluation quantitative permet d'établir que les plus performantes de ces méthodes produisent des repérages corrects à près de 70% (*F-measure*).

Dans la pratique, les mécanismes présentés permettent d'extraire beaucoup plus de segments de texte réutilisables qu'il n'est réaliste d'en proposer à l'utilisateur. Il est donc nécessaire d'effectuer une sélection parmi les segments trouvés avant de les présenter à l'utilisateur. On propose différentes méthodes pour effectuer cette sélection, reposant sur une formalisation du problème en termes probabilistes, et sur l'utilisation de réseaux de neurones multicouches. Ces méthodes ont également fait l'objet d'une évaluation quantitative, dont on rapporte les résultats. Globalement, le système proposé permettrait la réutilisation de 15 à 30 fois plus de matériel déjà traduit qu'un système basé sur les phrases.

mots-clefs : outils d'aide à la traduction, traduction assistée par ordinateur, traduction automatique, modèles statistiques de traduction, traduction automatique basée sur les exemples, réseaux de neurones.

ABSTRACT

This thesis presents work on various aspects of the development of a *sub-sentential translation memory* system. A *translation memory* (TM) is a type of translation support tool, whose purpose is to facilitate the reuse of existing translations. It relies on the systematic archiving of the production of one or more translators. Given a new text to translate, the system will search this archive to find segments of text that may have been previously translated. The translation of these segments is then proposed to the translator, who may or may not use them as he or she sees fit.

Existing TM systems operate on text units that are generally sentences, i.e. the segments of text which the system archives and ultimately proposes to the user are complete sentences. Because the repetition of whole sentences is a rare phenomenon in general- purpose language, the usability of such systems is in fact quite limited.

This thesis therefore explores the possibility of developing a translation memory system capable of proposing translations for sequences of words that represent subparts of a sentence.

The exact nature of these sequences is first examined: What type of word sequences should such a system ideally consider? The analysis of this question is based in large part on a study of how real translators make use of the *TransSearch bilingual concordancer*, a tool that closely resembles translation memory systems but which allows users to interactively look up the translation of arbitrary sequences of words. This study reveals that the vast majority of multi-word user queries actually correspond to sequences of one or more *syntactic chunks* in the translation memory. Based on this observation, a system which would allow for the reuse of this sort of text segment is proposed.

The implementation of such a system relies on *translation spotting* techniques: mechanisms whose purpose is to locate the translation of a specific segment of text in a parallel text, a notoriously difficult problem. This thesis proposes different methods for this task, based on statistical translation models. An empirical evaluation shows that the best of these methods can attain 70% accuracy (*F-measure*).

In practice, the proposed mechanisms allow for the extraction of much more reusable material than the user can reasonably assimilate. A selection must therefore be made among the extracted segments of text before they can be presented to the user. A probabilistic formalization of the problem is proposed, and different selection methods are presented, based on multilayer neural networks. These methods are also evaluated on empirical data. The results of these experiments show that, compared to a sentence-based translation memory system, the proposed sub-sentential system would be capable of proposing 15 to 30 times more reusable material.

keywords : translation memory, translation support tools, computer assisted translation, statistical translation models, example-based machine translation, machine translation, neural networks..

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLES

LISTE DES ABRÉVIATIONS COURANTES

AAT	Arbre d'analyse de traduction
CES	*Corpus Encoding Standard*
EM	*Expectation Maximization*
HMM	*Hidden Markov Model*
ITG	*Inversion Transduction Grammar*
MT	Mémoire de traduction
RALI	Laboratoire de recherche appliquée en linguistique informatique
RNM	Réseau de neurones multicouches
RT	Repérage de traduction
SITG	*Statistical Inversion Transduction Grammar*
SMT	Système de mémoire de traduction
TA	Traduction automatique
TAP	*Think-aloud Protocol*
TAS	Traduction automatique statistique
TABE	Traduction automatique basée sur les exemples
TW	Translator's Workbench

REMERCIEMENTS

En premier lieu, je tiens à remercier sincèrement les membres du jury pour leur disponibilité et leurs précieux commentaires : les professeurs Yoshua Bengio, Harold Somers, Philippe Langlais, Alain Polguère, et Jian-Yun Nie.

Plusieurs personnes ont contribué de près et de loin à la réalisation des travaux qui sont rapportés ici, et je tiens à les remercier pour leur assistance: mon directeur, Jian-Yun Nie, pour son support et pour la confiance qu'il m'a accordée tout au long du projet; Philippe Langlais, pour sa constante collaboration et les multiples idées qui ont germé lors de nos échanges; Elliott Macklovitch, pour ses suggestions, commentaires et encouragements; enfin, George Foster, Guy Lapalme, Michael Carl, Graham Russell, et toute l'équipe du RALI, pour leur soutient technique et moral, les innombrables discussions constructives et toutes ces tasses de café. Je me dois également de remercier Pierre Isabelle, pour m'avoir encouragé pendant de longues années à me lancer dans cette entreprise. Ce qu'il y a de bon dans ces pages, c'est à eux tous que je le dois; le reste me revient entièrement.

Finalement, je souhaite exprimer toute ma gratitude envers ma famille: mon épouse Sophie et mes enfants, Jérôme, Mathilde et Éloïse. Sans votre soutient, votre patience et votre affection, rien de tout celà n'aurait été possible.

Merci à vous tous.

Chapitre 1

INTRODUCTION

La profession de traducteur, comme bien d'autres d'ailleurs, a subi d'importantes transformations au cours des 20 dernières années. D'abord, parmi les industries de services, la traduction est présentement l'une de celles qui présentent le plus fort taux de croissance, avec des prédictions de croissance annuelle qui vont jusqu'à 25% pour les secteurs technologiques [46]. En pratique, cette croissance est principalement le résultat de l'arrivée en masse des ordinateurs personnels dans les milieux de travail. En effet, on sait que la tendance actuelle à la mondialisation des marchés incite les industries à "internationaliser" leurs produits; en particulier, la diffusion à grande échelle de logiciels et autres produits informatiques a donné naissance à un nouveau secteur d'activité économique, la *localisation*, qui emploie un nombre croissant de traducteurs (au point qu'on entend certains parler de la traduction comme d'un "sous-secteur" de la localisation). Par ailleurs, l'explosion récente d'Internet, en permettant l'accès à une quantité d'information sans précédent, a également entraîné une demande croissante pour la traduction de cette information.

Mais dans le cas de la traduction, ce sont également les méthodes qui se transforment. Avec la bureautisation graduelle de la profession, on a assisté à la naissance d'une gamme d'outils d'aide informatiques conçus spécifiquement à l'intention des traducteurs. Plusieurs de ces outils se retrouvent maintenant dans des trousses qu'on désigne sous le terme générique de *poste de travail pour traducteurs*. Alors qu'il y a quelques années à peine, ce genre de poste de travail ne se retrouvait que dans quelques grandes organisations, on assiste maintenant à leur apparition dans de petits

cabinets et même chez des traducteurs indépendants.

La majorité de ces postes de travail sont organisés à l'entour d'un logiciel de traitement de texte, soit générique (on voit le plus souvent des intégrations au logiciel *MS-Word*), soit dédié. Dans ce dernier cas, le logiciel est capable de lire directement des documents qui ont été produits dans l'un ou l'autre des formats de texte les plus répandus (*MS-Word*, *WordPerfect*, RTF, HTML, etc.), et permet à l'utilisateur de produire sa traduction sans égard pour le format d'origine, qui est alors automatiquement transposé sur le texte traduit. Ceci libère dans une certaine mesure le traducteur des détails souvent fastidieux de la mise en page, et ainsi de concentrer son énergie et son temps sur la tâche de traduction à proprement parler.

À l'intérieur de ces logiciels de traitement de texte, l'utilisateur a alors accès à une gamme de fonctionnalités qui répondent spécifiquement aux besoins des traducteurs : accès en-ligne à des dictionnaires et lexiques spécialisés, consultation et gestion de fiches terminologiques, suivi de projet de traduction, etc. Mais la fonctionnalité la plus originale parmi celles-ci est sans doute la *mémoire de traduction*. En quelques mots, il s'agit d'un dispositif qui permet de détecter automatiquement si un segment du texte que le traducteur s'apprête à traduire a déjà été traduit, soit dans le document même, soit dans un document traduit antérieurement. Lorsqu'une telle éventualité se produit, le système est en mesure de récupérer la traduction existante de ce segment, et de la proposer au traducteur. Il s'agit en somme d'un mécanisme de *recyclage* ou de *récupération* des traductions.

Un tel dispositif repose sur un archivage systématique de la production d'un ou de plusieurs traducteurs, dans une base de données structurée de façon à permettre des récupérations de cette nature, ce qu'on appelle une *mémoire de traduction*. Conceptuellement, celle-ci peut être vue comme un ensemble de paires de segments de texte, traduction l'un de l'autre. Des mécanismes d'indexation plein-texte et de recherche adaptés permettent le repérage rapide d'un segment dans l'une ou l'autre des deux langues, et ainsi la récupération de sa traduction.

Bien que la nature des segments de texte qui constituent la mémoire de traduction varie, elle s'apparente habituellement à des phrases complètes. Par conséquent, la récupération de traduction s'effectue également sur la base de phrases complètes, c'est-à-dire que ces systèmes ne sont en mesure de proposer des traductions à l'utilisateur que dans le cas où une phrase complète du nouveau texte à traduire se trouve intégralement dans la mémoire de traduction. Il existe bien des dispositifs permettant de repérer des phrases qui ne coïncident que partiellement (on parle en anglais de *fuzzy matches*, terme qu'on pourrait traduire en français par *repérages flous*), et certains des systèmes existants sont même capables dans ces cas d'effectuer certains ajustements dans la traduction des segments ainsi repérés, dans le but de les adapter au contexte du nouveau texte. Mais ces mécanismes sont relativement limités, et dans tous les cas, les repérages, tant exacts que flous, se font toujours sur la base de phrases entières.

Ce genre de fonctionnalité répond aux besoins des traducteurs œuvrant dans certains secteurs d'activité bien ciblés, dans lesquels la répétition de phrases entières est un phénomène courant. Par exemple, on peut penser à la traduction de documents tels que des rapports financiers ou météorologiques, produits en grandes quantités, mais dont le contenu présente très peu de variété. La localisation de logiciel est un autre domaine dans lequel on observe ce genre de répétition. Mais l'utilisation la plus répandue de ces systèmes s'observe probablement dans le cadre de mises-à-jour de documents – dans ce cas, il n'est pas rare que la plus grande partie du texte d'un nouveau document se retrouve dans la mémoire de traduction. La récupération automatique de la traduction des segments inchangés permet alors des économies substantielles, puisqu'il ne reste en définitive au traducteur qu'à traduire les différences entre la nouvelle et l'ancienne version du document.

D'une façon générale, toutefois, il est clair que la répétition de phrases entières est un phénomène extrêmement rare, et pour la très grande majorité des traductions, les systèmes de mémoire de traduction existants s'avèrent d'une utilité très limitée.

Cet état de fait est d'autant plus désolant que la langue est faite de répétitions : collocations, expressions idiomatiques, formules toutes-faites en sont autant de manifestations. D'une certaine façon, les récents succès des méthodes de modélisation statistique des langues naturelles, telles que les modèles de Markov, reposent précisément sur ces phénomènes de répétition. Intuitivement, on serait en droit d'espérer en tirer un profit similaire pour la traduction.

Le problème des mémoires de traduction existantes provient principalement du fait qu'elles abordent les phénomènes de répétitions à un niveau de résolution où ils sont quasiment inexistants, à savoir celui des phrases. Pour s'en convaincre, on n'a qu'à imaginer une mémoire de traduction qui opérerait à un niveau de résolution beaucoup plus fin, par exemple celui des mots. Pour peu que la taille et le contenu de la mémoire de traduction soient adéquats, on se convaincra facilement qu'un tel système serait en mesure de proposer des traductions pour la majorité des mots d'un nouveau texte.

Le principal facteur qui limite à la phrase le niveau de résolution des actuels systèmes de mémoire de traduction est la difficulté d'établir automatiquement les liens entre les mots d'une phrase et ceux de sa traduction. En fait, à quelques exceptions près (notamment le système *Trados*), les systèmes existants ne comprennent même pas de méthode automatique pour la mise en correspondance des phrases, puisque pour la constitution des mémoires de traduction, on préconise plutôt une méthode de collecte semi-automatique des paires de phrases, à même l'environnement de traitement de texte dédié.

D'autre part, il est clair qu'un système de mémoire de traduction opérant au niveau des mots plutôt qu'au niveau des phrases ne serait probablement d'aucune utilité pour les traducteurs. D'abord, pour plusieurs des mots, la mémoire contiendrait de multiples traductions. Toutes les proposer à l'utilisateur reviendrait souvent à submerger ce dernier d'information. Afin de contrôler la quantité de matériel à proposer à l'utilisateur, il serait donc impératif de mettre en place des mécanismes

de sélection, capables d'identifier quelle traduction parmi toutes celles trouvées pour un mot, est la plus susceptible d'être utile au traducteur dans le contexte donné.

Ensuite, même en supposant qu'on arrive à limiter les propositions émanant de la mémoire à une seule par mot du texte-source, en présentant les propositions dans le même ordre que les mots dont ils sont la traduction dans le texte-source, on se trouverait essentiellement à proposer une traduction automatique mot-à-mot. Or on sait que de telles traductions sont généralement inacceptables, de telle sorte que le traducteur se verrait systématiquement dans l'obligation de réordonner correctement les mots proposés, d'éliminer les mots superflus et d'ajouter les mots manquants pour satisfaire aux contraintes inhérentes à la *langue-cible*. Un tel exercice est extrêmement fastidieux en pratique, et de nombreuses expériences de jumelage homme-machine de ce genre ont démontré qu'il est contre-productif. Qui plus est, les traducteurs se montrent généralement très réticents à travailler avec ce genre "d'assistance".

Entre la phrase et le mot, pourtant, il doit exister un niveau de résolution qui permette de tirer meilleur parti du contenu des mémoires de traduction existantes. Dans cette thèse, nous examinons la possibilité de mettre sur pied un tel *système de mémoire de traduction sous-phrastique*, c'est-à-dire qui opère sur des unités de traduction dont la taille est inférieure à celle d'une phrase, mais supérieure à celle du simple mot. Nous nous penchons principalement sur trois questions, soit celle de la nature de l'unité de traduction appropriée, celle des méthodes de repérage de traduction, et celle de la sélection des propositions.

Différentes études semblent indiquer que les traducteurs humains, dans l'exercice de leur profession, opèrent naturellement sur des unités qui dépassent le simple mot (voir, par exemple, Bernardini [9] pour une revue de ces études). Il a été suggéré que les systèmes de mémoire de traduction devraient faire de même. Dans le chapitre 2, nous abordons cette question en présentant les résultats d'une étude sur l'utilisation réelle par des traducteurs humains de *TransSearch*, un "concordancier bilingue", dont les fonctionnalités s'apparentent de très près à celles des mémoires de traduction.

Partant des résultats de cette étude, nous proposons un système dont l'unité de base est la séquence de tronçons syntaxiques (*syntactic chunks* en anglais), et nous exposons comment pourrait être effectuée l'extraction des segments pertinents dans la mémoire de traduction d'un tel système.

Le repérage de traduction est la tâche qui consiste à identifier, dans une paire de segments de texte traduction l'un de l'autre, l'ensemble des mots en langue-cible qui constituent la traduction d'un ensemble donné de mots en langue-source. Cette tâche s'apparente donc à l'alignement des mots dans une traduction, tâche qui a le plus souvent été attaquée via des approches statistiques. Dans le chapitre 3, nous présentons ces approches, et examinons comment elles peuvent être modifiées, notamment en y intégrant des contraintes de *contiguïté* et de *compositionnalité*, afin d'élaborer des méthodes de repérage de traduction, spécifiquement adaptées au contexte d'un système de mémoire de traduction tel que celui proposé ici. Nous présentons également les résultats de différentes expériences visant à mesurer le niveau de performance qu'on peut attendre de ces méthodes.

En combinant les méthodes d'extraction présentées au chapitre 2 et les méthodes de repérage de traduction du chapitre 3, on arrive à produire un ensemble de segments de texte en langue-cible, susceptibles d'être utiles pour la traduction d'un texte en langue-source donné. Toutefois, la taille de cet ensemble est souvent prohibitive, de telle sorte qu'il est impraticable de le proposer intégralement au traducteur. Il faut donc effectuer une sélection parmi les *segments-cible* obtenus. Au chapitre 4, nous abordons cette question d'un point de vue probabiliste : il s'agit donc d'identifier, parmi l'ensemble de tous les segments-cible extraits de la mémoire de traduction, le sous-ensemble qu'il est le plus probable que le traducteur souhaite réutiliser. Nous proposons différentes façons de calculer ces sous-ensembles, reposant sur différentes caractérisations des segments-cible, de même que sur différentes façons de combiner ces caractérisations, notamment au moyen de réseaux de neurones. Ici encore, nous présentons les résultats de plusieurs expériences visant à mesurer la performance des

méthodes proposées.

Ensemble, les composantes proposées dans cette thèse constituent le noyau d'un système de mémoire de traduction sous-phrastique. Pour en arriver à une implantation complète d'un tel système, deux questions additionnelles devraient toutefois être abordées, soit celle de la collecte et de la constitution des mémoires de traduction, et celle de l'interface-utilisateur. En fait, les approches et méthodes que nous proposons ici reposent sur une mémoire de traduction "standard", c'est-à-dire analogue à celles que l'on retrouve dans les systèmes existants, c'est pourquoi nous avons choisi de ne pas nous attarder sur cette question. Quant à la question de l'interface-utilisateur, elle est complexe, et pourrait sans doute constituer à elle seule un projet de doctorat. On peut toutefois présager que les interfaces des systèmes existants, moyennant quelques adaptations mineures, pourraient accommoder des propositions de traductions sous-phrastiques, comme on le propose ici.

On peut donc résumer ainsi les contributions scientifiques qu'on retrouve dans cette thèse :

- Une analyse originale de l'unité de traduction adéquate pour les systèmes de mémoires de traduction, reposant sur une étude de l'utilisation par des traducteurs d'un concordancier bilingue;

- Des méthodes nouvelles de repérage de traduction, spécifiquement adaptées au contexte des mémoires de traduction;

- La présentation d'un problème nouveau, soit celui de la sélection des segments-cible dans un système de mémoire de traduction;

- Des méthodes originales pour résoudre ce problème, notamment au moyen de modèles statistiques de traduction et de réseaux neuronaux.

Chapitre 2

SYSTÈMES DE MÉMOIRE DE TRADUCTION SOUS-PHRASTIQUE

2.1 Introduction

Le terme *système de mémoire de traduction* (SMT) désigne une gamme d'outils informatiques d'aide aux traducteurs, dont l'objectif est d'éviter à son utilisateur de traduire des segments de texte qui ont déjà été traduits par le passé. Brièvement, ce genre de système mémorise un ensemble de traductions existantes; lorsqu'il rencontre dans un nouveau texte à traduire une portion de texte dont la traduction est déjà connue, il extrait celle-ci de sa mémoire, et la propose au traducteur. Ces systèmes sont typiquement intégrés dans un *poste de travail pour traducteurs*, c'est-à-dire un environnement informatique regroupant une gamme d'outils d'aide à l'intention des traducteurs professionnels. Différents produits de cette nature sont présentement disponibles sur le marché.

Les unités de traduction sur lesquelles opèrent les SMT existants sont normalement identifiées sur la base de critères de nature typographique : ponctuation, marqueurs de fins de paragraphes, entêtes de chapitres, etc. Ces unités s'apparentent donc majoritairement à des phrases, c'est-à-dire que le contenu de la mémoire consiste principalement en paires de phrases traduction l'une de l'autre (ce que nous appelons des *couples*), que le système compare avec les phrases du nouveau texte à traduire. De ce fait, ces systèmes ne récupèrent que la traduction de segments de texte qui coïncident dans leur intégralité. La plupart des systèmes permettent des coïncidences approximatives (*fuzzy matches*), mais toujours sur la base de phrases entières.

Un tel niveau de résolution limite dans une mesure importante l'utilité des SMT. En effet, s'il existe des domaines d'application pour lesquels la répétition de phrases entières est courante (notamment la mise-à-jour de documents techniques et certains aspects de la localisation de logiciels), il reste que ce genre de répétition est extrêmement rare en général. De ce fait, pour la grande majorité des textes que traitent quotidiennement les millions de traducteurs sur la planète, les SMT actuels sont d'une utilité très limitée.

Dans leur article intitulé *What's been Forgotten in Translation Memory* (en français : *Qu'a-t-on oublié dans la mémoire de traduction*), Macklovitch et Russell [63] font état de la quantité de matériel potentiellement réutilisable laissée de côté par les SMT existants. D'après les auteurs, ces *silences* sont dus en partie au manque de souplesse des mécanismes de recherche utilisés pour fouiller la mémoire de traduction, mais également à la notion-même d'*unité de traduction* sur laquelle ils reposent majoritairement, à savoir la phrase.

Suivant cette logique, un SMT qui serait capable de "plonger" sous le niveau de la phrase, tant dans le nouveau segment à traduire que dans le contenu de la mémoire de traduction, serait théoriquement en mesure de proposer beaucoup plus de matériel utile au traducteur.

Pour s'en convaincre, et supposant l'existence de méthodes fiables permettant d'établir entre deux textes traduction l'un de l'autre des équivalences au niveau des mots, on n'a qu'à imaginer un SMT qui rechercherait dans sa mémoire la traduction de mots individuels du nouveau texte à traduire : un tel système arriverait sans doute à proposer de multiples traductions pour la plupart des mots du texte-source.

En revanche, la grande majorité de ces propositions serait probablement inutilisables par le traducteur. À défaut de mécanismes plus sophistiqués de sélection et d'adaptation des propositions, un tel SMT agirait essentiellement comme un dictionnaire bilingue automatique : il proposerait pour chaque mot du texte un ensemble de traductions possibles, sans considération pour le contexte dans lequel ces mots appa-

raissent. Il y a fort à parier que la majorité des traducteurs jugerait un tel système inutile et encombrant.

Les comportements de ces deux genres de SMT – appelons-les systèmes *phrase-à-phrase* et *mot-à-mot* – suggèrent un parallèle avec le domaine de la recherche d'information : si on considère que les phrases du nouveau texte à traduire sont des *requêtes* et que la mémoire de traduction est une *collection de documents*, on constate que les SMT phrase-à-phrase favorisent une stratégie de précision maximale (les propositions faites par le système sont presque toujours bonnes), au détriment du rappel (la majorité des requêtes ne trouve aucune "réponse"). À l'opposé, un SMT mot-à-mot tel que décrit ci-dessus favorise un rappel maximal au prix d'une dégradation radicale de la précision. Dans cette analogie, c'est le niveau de résolution du système, tel que déterminé par les unités de traduction qu'il considère, qui joue le rôle de contrôle du rappel.

Dans les pages qui suivent, nous examinons de plus près cette question : entre la phrase et le mot, existe-t-il un niveau de résolution qui permette à un SMT d'atteindre un compromis plus satisfaisant entre rappel et précision? Après un aperçu de l'origine des SMT et une description un peu plus détaillée des systèmes existants à la section 2.2, nous présentons à la section 2.3 les résultats d'une étude sur l'utilisation réelle d'une mémoire de traduction interrogeable en-ligne, le système TransSearch. De cette étude, nous tirons différentes conclusions sur la nature de l'unité de traduction "idéale" pour les SMT, dont nous discutons à la section 2.4. Partant de là, nous traçons un schéma général du mode de fonctionnement d'un système opérant à ce niveau de résolution, à la section 2.5. Finalement, nous présentons à la section 2.6 les résultats de quelques expériences, visant à établir le degré de couverture d'un nouveau texte à traduire qu'on pourrait espérer atteindre avec un tel système.

2.2 Systèmes de mémoire de traduction (SMT)

2.2.1 Définitions et origines

D'une façon générale, le terme *mémoire de traduction* (MT) désigne un type de dispositif informatique à l'intention des traducteurs humains, visant à promouvoir la réutilisation des traductions existantes. D'après Macklovitch et Russell [63], deux définitions de ce terme sont admises, que nous appelons informellement *définition pratique* et *définition théorique*.

La définition pratique se lit comme suit :

> ...une mémoire de traduction [...] est un type d'outil d'aide à la traduction qui maintient une base de données de paires de phrases en langues source et cible, et qui extrait automatiquement la traduction des phrases d'un nouveau texte qui apparaissent dans cette base de données.

La définition théorique est, quant à elle, beaucoup plus englobante :

> ...une archive de traductions existantes, conçue de façon à faciliter la réutilisation des traductions.

La définition théorique est certes plus attrayante que la première. Malgré tout, il faut comprendre que celle-ci (la définition pratique) découle directement d'une réalité indéniable : elle décrit en fait beaucoup plus adéquatement la nature des SMT existants à l'heure actuelle, et les "théoriciens" ont en réalité eu très peu à voir avec l'émergence de ces systèmes. Par ailleurs, on peut voir que la définition théorique ne fait référence qu'à la base de données contenant les traductions, c'est-à-dire la *mémoire de traduction* (MT) à proprement parler, alors que la définition pratique englobe également l'environnement permettant au traducteur d'exploiter effectivement cette base de données, ce que nous appelons ici le *système de mémoire de traduction*.

Dans un article sur les origines du *poste de travail du traducteur*, John Hutchins dément la croyance populaire suivant laquelle les systèmes de mémoires de traduction sont une invention des années 1990 [44]. D'après l'auteur, les premiers systèmes informatiques d'aide aux traducteurs remontent aux années 1960. Il mentionne notamment le dispositif mis en place dès 1965 par les services de traduction de l'armée allemande : ce système, appelé *LEXIS*, consistait essentiellement en un ensemble de glossaires et dictionnaires terminologiques multilingues, que les traducteurs et lexicographes du service pouvaient consulter et modifier en-ligne. Bien qu'une mémoire de traduction n'en fasse pas partie, le concepteur du système, Friedrich Krollmann, envisageait dès 1971 d'y intégrer une *archive de traduction* :

> ... au moyen de descripteurs ou de mots-clefs, on pourrait rechercher des passages spécifiques dans de grandes quantités de texte, qui pourraient alors être affichés sur écran vidéo afin d'aider le traducteur. Dans le cas de nouvelles éditions révisées de textes déjà traduits, seuls les passages modifiés nécessiteraient une resaisie. L'insertion des changements et des corrections dans l'ancienne version du texte seraient faite automatiquement par l'ordinateur [56].

L'idée d'une mémoire de traduction informatisée n'est donc pas nouvelle. Par ailleurs, l'idée d'intégrer ce genre de mécanisme à même un environnement de traitement de texte n'est pas nouvelle non plus : Dennett [31] reproduit par exemple cette citation datant de 1980, de P.J. Arthern, du Secrétariat du Conseil Européen :

> Il m'est toutefois apparu que le Secrétariat du Conseil (Européen), comme peut-être d'autres institutions et grandes organisations, et peut-être même certains "petits utilisateurs", pourraient tirer un immense avantage d'un système intégré de traitement de texte, capable de produire des traductions par "recherche documentaire" plutôt qu'au moyen

> de programmes de traduction automatique. Ce serait une simple extension logique du principe “une fois suffit”... Il doit en fait être possible de mettre sur pied un programme permettant à un logiciel de traitement de texte de “se rappeler” si quelque partie d'un nouveau texte a déjà été traduite, d'aller chercher cette partie avec sa traduction existante, et de l'afficher à l'écran ou de l'imprimer, tout celà de façon automatique.[6]

Il a malgré tout fallu attendre l'arrivée des premiers ordinateurs personnels et leur entrée massive dans les milieux de travail pour que le poste de travail du traducteur prenne réellement son envol. Dans les faits, les premiers systèmes de mémoire de traduction (SMT) sont apparus au cours des années 1980, sous la forme de dispositifs pratiques conçus pour les traducteurs, souvent même par des traducteurs. L'émergence de ces systèmes coïncide bien plus avec la bureautisation graduelle de la profession de traducteur qu'avec d'éventuels transferts technologiques en provenance des cercles de la recherche en linguistique informatique. Dans la réalité, les premiers systèmes sont le fruit d'efforts indépendants dans certains cabinets de traduction, soucieux de mettre leurs archives grandissantes au service de la rentabilité [97].

Le premier produit commercial correspondant réellement à la définition pratique de mémoire de traduction fut sans doute le système mis au point par la société ALPS (*Automated Language Processing Systems*), qui tournait sur un ordinateur de type IBM-AT [44]. Ce système proposait un éditeur de texte, dans lequel les textes source et cible étaient présentés côte-à-côte. Les codes de formattages du texte-source étaient transposés directement dans le texte-cible, et l'utilisateur pouvait facilement copier et coller des segments entiers, par exemple des tableaux et des figures. Ce programme permettait en outre de consulter directement un dictionnaire central, dont étaient automatiquement extraits les termes pertinents du texte à traduire.

De façon plus importante, le système de ALPS comprenait un mécanisme de *traitement des répétitions* : à mesure que l'utilisateur rédigeait ses traductions avec

ce programme, les paires de phrases traduites pouvaient être stockées automatiquement dans un *fichier de répétitions*; avant de traduire un nouveau texte, le traducteur pouvait consulter ce fichier de répétitions, de façon à en extraire les phrases déjà traduites pour inclusion directe dans la nouvelle traduction. Le mécanisme de traitement des répétitions du système ALPS permettait en outre de repérer des segments déjà traduits qui ne différait du nouveau texte qu'en regard d'expressions numériques; il s'agissait en somme d'une forme restreinte de *repérages flous*.

ALPS cessa la vente de son environnement pour traducteurs vers 1990; d'une certaine façon, leur produit était arrivé trop tôt sur le marché, alors que beaucoup de traducteurs n'étaient pas encore prêt à faire le "grand saut". Mais plusieurs produits allaient s'en inspirer pour prendre la relève.

2.2.2 SMT commerciaux

Différents SMT commerciaux sont apparus concurremment vers 1990, notamment les *Translation Manager 2* (ou *TM2*, depuis peu retiré du marché) de IBM, *Translator's Workbench* (ou *TW*) de Trados et le système *Transit* de *Star AG*. D'autres joueurs se sont ajoutés plus récemment, tels que *ATRIL* avec le système *Déjà Vu*, *Cypresoft* avec le système *TransSuite 2000* et la firme *SDL* avec *SDLX Translation Suite*.

Tous ces systèmes se présentent en fait sous la forme d'un environnement de travail pour traducteur, dont la mémoire de traduction n'est qu'une des fonctionnalités. Par exemple, *Transit* et *Déjà Vu* proposent un ensemble de fonctionnalités reliées à la gestion du flot de l'information dans le processus de traduction : gestion de projets de traduction, répartition des tâches, annotation des versions, etc. La majorité des systèmes intègre également des fonctionnalités de gestion terminologique et de *lexiques de projet*. Tous les systèmes fournissent également un éditeur de texte dédié, qui permet au traducteur de faire abstraction du format d'origine du texte à traduire : ces éditeurs acceptent des textes dans une variété de formats d'encodage (*MS-Word*, RTF, *WordPerfect*, *FrameMaker*, HTML, etc.), et transposent automatiquement la

mise en page du texte-source sur le texte-cible que produit le traducteur. En théorie, le traducteur n'a donc plus à se soucier des laborieuses questions de mise en page. Par ailleurs, *TW* propose également l'alternative d'une intégration de son application à l'éditeur *MS-Word*.

Au sein de ces environnements de traduction, on préconise habituellement une méthode de travail dans laquelle le traducteur traduit les phrases du texte une par une, en écrasant le texte-source : la phrase courante est mise en surbrillance dans l'éditeur, et le traducteur rédige la traduction "par dessus" le texte original. Il semble que cette façon de faire corresponde assez bien aux méthodes de travail habituelles de nombreux traducteurs.

Tous les systèmes mentionnés ici reposent sur des mémoires de traduction qui peuvent essentiellement être vues comme des collections de paires de phrases. Dans tous les cas, la construction de ces mémoires de traduction est normalement assurée par les traducteurs eux-mêmes : à mesure que ceux-ci produisent de nouvelles traductions au sein de l'environnement, le système stocke les paires de phrases résultantes dans la MT. À notre connaissance, seul TW propose un système d'alignement automatique des phrases (le programme *WinAlign*), alors que *TransSuite 2000* inclut un environnement d'alignement de phrases "semi-automatique".

Sous le mode d'opération "normal", lorsque le système trouve dans la mémoire une traduction pour la phrase courante, il la propose au traducteur : celle-ci apparaît alors dans une zone dédiée de l'interface-utilisateur, ou dans une fenêtre transitoire. S'il le désire, l'utilisateur peut alors récupérer cette proposition par une simple manoeuvre au clavier ou au moyen de la souris. La traduction proposée apparaît alors en lieu et place de la phrase-source originale, et l'utilisateur est libre de modifier celle-ci au besoin ou de passer à la phrase suivante.

Des systèmes comme *TW* et *Déjà Vu* proposent également un mode d'opération par *prétraduction*. Dans ce mode, le système examine d'abord chacune des phrases du nouveau texte à traduire, et remplace automatiquement par sa traduction toutes

celles qui se retrouvent intégralement dans la MT. Le traducteur se retrouve donc d'emblée avec un texte partiellement traduit, dans lequel il ne lui reste qu'à vérifier les phrases ainsi *prétraduites* et à traduire les phrases non-trouvées dans la MT. Dans un esprit similaire, *Déjà Vu* propose également un mécanisme de *propagation*, dans lequel le système propage automatiquement la traduction des phrases qui se répètent au sein même du texte à traduire, à mesure que l'utilisateur en traduit la première occurrence.

Le mécanisme de repérage des traductions dans la MT est un élément central de ces systèmes. Tous les systèmes sont capables d'effectuer des repérages exacts, c'est-à-dire de retrouver dans la MT les couples dont le texte en langue-source coïncide mot pour mot avec une des phrases du texte à traduire. Tous les systèmes présentent également la capacité d'effectuer des *repérages flous* (en anglais : *fuzzy matches*), c'est-à-dire de repérer pour une phrase-source donnée des couples dont la partie source ne coïncide que partiellement. Ainsi, un système comme *TW* permet à l'utilisateur de régler le "degré de coïncidence minimal" des repérages flous, par exemple : extraire de la MT les couples dont la partie-source coïncide au moins à 70% avec la phrase à traduire. La signification exacte de cette mesure varie d'un système à l'autre, et les méthodes utilisées pour la calculer constituent des secrets commerciaux; néanmoins, comme le suggèrent Planas et Furuse [83], tout porte à croire qu'il s'agit de variations sur la mesure de distance entre deux chaînes, à la façon de Wagner et Fischer [113]. On peut soupçonner que les comparaisons sont faites en termes de mots (plutôt qu'en termes de caractères), et que différentes classes de mots font possiblement l'objet de pondérations différentes dans le calcul (par exemple, les expressions numériques pèsent possiblement moins lourd dans le calcul de la similitude que les mots véritables). Mais le calcul de similitude des repérages, qu'ils soient exacts ou flous, se fait toujours sur la base de phrases complètes.

La construction et l'exploitation de la MT dans ces systèmes reposent de façon cruciale sur une segmentation en phrases du texte-source, qui constitue normalement

la première étape du traitement d'un nouveau texte à traduire : c'est sur la base de cette segmentation que, d'une part, le système recherche les traductions antérieures dans la MT et, d'autre part, que les nouvelles paires de traductions sont ajoutées à la MT. Tous les systèmes s'appuient à cet égard sur des heuristiques, qui tiennent compte des détails de la mise en page (titres, paragraphes, etc.) et de l'usage habituel des signes de ponctuation (point final, d'interrogation, d'exclamation, etc.). Certains systèmes, notamment *Déjà Vu*, permettent à l'utilisateur de modifier dans une certaine mesure les règles de segmentation, en spécifiant par exemple des marqueurs de fins de phrases alternatifs (point-virgule, deux-points, etc.).

Certains systèmes proposent des utilitaires pour éditer le contenu de la MT, notamment *Transit* et *TransSuite 2000*. Ce type de fonctionnalité est intéressant dans certaines applications telles que la localisation de logiciel, dans lesquelles l'uniformité des traductions et la cohérence terminologique revêtent une importance particulière. Dans ce secteur d'activité, il n'est pas rare que la MT devienne le principal répertoire de cette terminologie, et le principal outil de contrôle de l'uniformité, d'où l'importance de pouvoir en vérifier et modifier le contenu au besoin.

Mentionnons finalement que le système *Déjà Vu* intègre certains mécanismes d'adaptation des repérages flous : par exemple, lorsque la phrase à traduire et le segment extrait de la MT ne diffèrent que par une expression numérique ou une entité nommée (nom de lieu ou de personne), ce système est capable de modifier dynamiquement la traduction pour l'adapter au nouveau contexte.

Un important développement des dernières années dans le domaine des SMT concerne la mise sur pied d'un standard pour l'encodage du contenu des mémoires de traduction [76]. Le standard *TMX* (*Translation Memory eXchange*) est basé sur le format d'annotation XML et permet de transférer une mémoire de traduction d'un SMT à un autre, ou, plus simplement, d'exploiter une même mémoire de traduction sous plusieurs systèmes différents. À ce jour, la majorité des systèmes décrits ci-dessus permet d'exporter et d'importer des mémoires de traduction dans ce format.

2.2.3 Travaux de recherche apparentés

Malgré leurs origines commerciales, il est clair que les caractéristiques et fonctionnalités des SMT existants appellent des parallèles avec certaines approches mises de l'avant par la communauté scientifique en traduction. On pense notamment aux paradigmes de traduction automatique (TA) basée sur les corpus (en anglais : *data-driven machine translation*), tels que la TA statistique et la TA basée sur les exemples. À la limite, on peut voir les SMT comme une forme très simplifiée de traduction basée sur les exemples.

Par ailleurs, l'apparition des SMT coïncide également avec l'émergence d'une certaine rhétorique au sein même de la communauté des chercheurs en TA, au sujet de "la juste place de l'homme et de la machine dans l'exercice de la traduction" : en réaction à un certain désenchantement vis-à-vis des insuccès de la TA, certains chercheurs ont commencé à s'intéresser à l'élaboration d'outils d'aide à la traduction. Dans un article datant du début des annéees 80, et récemment repris dans la revue *Machine Translation* [53], Martin Kay remettait en question la traduction automatique *per se*, et se faisait l'avocat d'une approche plus modeste à l'automatisation de la traduction (*the translator's amanuensis*). Kay argumente qu'il est futile de tenter de concocter des solutions mécaniques à un problème dont les fondements théoriques sont aussi mal compris que le sont ceux de la langue : de telles tentatives ne nous apprennent rien sur les langues naturelles (donc, sont stériles d'un point de vue scientifique) et sont condamnées à produire de mauvais résultats de façon systématique.

Kay avance donc l'idée d'une station de travail pour les traducteurs, dans laquelle ne sont automatisées que les tâches que l'ordinateur sait présentement faire de façon satisfaisante : *"Little steps for little feet"*[1]. Dans cette philosophie, la TA comme telle n'apparaitrait qu'en fin de course, et seulement à la demande du traducteur.

Alan Melby, de son côté, s'étonnait qu'on puisse même songer à automatiser une

[1] Dans cet article, cette liste est très courte, et les tâches énumérées sont pour le moins modestes!

tâche aussi complexe que la traduction [75]. Il proposait quant à lui un *concordancier bilingue* pour les paires de textes qui sont des traductions l'un de l'autre. À l'instar du concordancier habituel, un tel système présente ensemble toutes les occurrences d'un mot dans un texte, accompagnées de leur contexte immédiat et de leur traduction dans ce contexte. Bien que de telles concordances impliquent un appariement d'un texte avec sa traduction, Melby n'envisageait de toute évidence pas encore que cette opération puisse être effectuée automatiquement, et suggérait en fait qu'un opérateur humain se charge de cette étape. Quant aux applications de cet outil, elles étaient principalement académiques (l'étude de la traduction), mais Melby démontrait quand même comment il pourrait servir aux traducteurs, par exemple pour faciliter la détection de certaines erreurs de traduction.

L'idée de mémoire de traduction (dans sa définition "théorique") apparaît réellement dans la littérature scientifique avec Brian Harris, qui pousse plus loin l'idée d'Alan Melby, et propose que les traducteurs archivent leur production sous la forme d'*hyper-bitexte* [43]; ainsi, un traducteur recherchant une traduction pour un mot donné peut soumettre une requête au système d'archivage, qui lui affichera alors la concordance bilingue pour ce mot sur la totalité de l'archive (plutôt que sur un document unique, comme le proposait Melby). Un tel système constitue en somme un dictionnaire bilingue vivant, et comporte deux avantages notoires sur un dictionnaire bilingue traditionnel : les mots cherchés sont présentés dans de véritables contextes d'utilisation, et le système s'enrichit lui-même, à même les nouvelles traductions de l'utilisateur.

Peu avant 1990, Pierre Isabelle lance au CITI (Centre d'innovation en technologies de l'information, un établissement de recherche du gouvernement fédéral canadien) un programme de recherche axé sur l'élaboration d'un *poste de travail pour traducteurs* [48]. Outre diverses fonctionnalités assez générales (traitement de texte, vérification orthographique, gestion de fiches terminologiques, accès à des dictionnaires en-ligne, etc.), ce poste de travail devait comprendre des outils entièrement

originaux, spécifiquement conçu à l'intention des traducteurs :

TransCheck : un outil de vérification automatique des traductions.

En jumelant une liste de traductions proscrites (ou *anti-dictionnaire*) à un programme d'alignement de traduction, on est en mesure de détecter certains types d'erreurs de traduction, tel que l'usage de faux-amis (*physicien* VS *physician*), de calques (*à la journée longue*), etc. [61]. Le même genre de mécanisme, assorti d'un glossaire terminologique, peut être utilisé pour assurer la cohérence terminologique d'une traduction [62].

TransTalk : un système de dictée pour traducteurs.

Un système de dictée automatique convertit le signal provenant de la voix d'un locuteur en un texte écrit. Dans le contexte de la traduction, on peut théoriquement améliorer la performance d'un tel système, en exploitant le texte-source comme une source d'information additionnelle [13]. Ainsi, devant une hésitation entre les mots *cheveux* et *chevaux*, un système de dictée pour traducteur pourra facilement faire un choix, sachant que la phrase dont on dicte la traduction contient le mot anglais *horse*.

TransSearch : un concordancier bilingue.

Partant du principe que

> ...les traductions existantes renferment infiniment plus de solutions à plus de problèmes de traduction que tout autre outil de référence [47]...

on archive dans une base de données textuelle la production d'un ou plusieurs traducteurs, sous la forme de paires de phrases appariées; on peut alors soumettre des requêtes à cette base de données, ce qui permet visualiser en contexte

l'emploi de termes ou expressions, de même que leur traduction. Il s'agit en somme d'une implantation de l'idée proposée par Harris, qu'on a assorti d'un mécanisme de recherche sophistiqué, permettant en outre de rechercher des expressions complexes [64, 103].

Comme on le voit, tous ces travaux, bien que reliés à divers degré à l'idée de mémoire de traduction, ne s'attaquent qu'indirectement aux problématiques techniques des SMT. Deux exceptions notoires concernent les travaux de Langé *et al.* [57] et de McTait et al. [68–70].

Langé *et al.* proposent une approche dans laquelle un SMT pourrait éventuellement *composer* une proposition de traduction à l'intention du traducteur, à partir de segments disparates récoltés dans la MT. Bien que s'inspirant largement de la traduction automatique basée sur les exemples, l'approche proposée va beaucoup moins loin que cette dernière, et ce à deux égards. D'abord, contrairement aux systèmes de TA habituels, on ne s'attend pas à ce que les propositions du système couvrent la totalité du texte-source : dans un contexte d'aide à la traduction humaine, il est tout à fait acceptable que le système n'offre qu'une couverture partielle. Ensuite, alors que la littérature en TABE met de l'avant toute une gamme de mécanismes sophistiqués pour l'analyse du texte-source, de la base d'exemples, la génération de la traduction, etc., le mécanisme proposé par les auteurs reposerait essentiellement sur la composition de deux types d'unité de traduction : des *briques* et des *squelettes*.

Les *briques* sont des paires de séquences de mots contigus $\langle s, t \rangle$ extraites de la MT. Suivant le modèle proposé, ces briques peuvent essentiellement être considérées comme des entrées lexicales dans un dictionnaire bilingue, de telle sorte que la séquence t peut être proposée intégralement au traducteur lorsque la séquence s est observée dans la phrase à traduire. Les briques peuvent ainsi être de simples paires de mots, par exemple :

brique 1 : *Esc/Échap*

(il s'agit du nom d'une touche sur un clavier d'ordinateur) ou des séquences de mots arbitraires, par exemple :

brique 2 : *Proceed with / Passez à l'étape de*

brique 3 : *installation verification / vérification de l'installation*

Les *squelettes* sont également des paires de séquences de mots, mais dans lesquelles apparaissent des *variables*, susceptibles d'être instanciées au moyen soit de termes extraits automatiquement d'un glossaire terminologique, soit de briques, soit d'autres squelettes, ou éventuellement par le traducteur lui-même. Par exemple :

squelette 1 : *Press the X_s key to continue / Appuyez sur X_t pour continuer*

Confronté à une nouvelle phrase

Press the Esc key to continue, and proceed to installation checking.

le système localiserait les squelettes et briques pertinents au moyen d'une simple comparaison caractère par caractère; en substituant dans le squelette 1 ci-dessus la brique 1 à la paire de variables X_s/X_t, le système produirait la proposition :

Appuyez sur Échap pour continuer

et en juxtaposant les briques 2 et 3, il produirait :

Passez à l'étape de vérification de l'installation

Sans élaborer à fond sur la nature exacte de ce qui pourrait constituer une brique ou un squelette dans un tel système, les auteurs suggèrent que la terminologie pourrait faire partie des briques, et que les termes eux-mêmes pourraient être identifiés de façon automatique ou semi-automatique par des méthodes telles que celles mises de l'avant par les auteurs eux-mêmes [29, 39].

McTait et Trujillo [70] élaborent une approche similaire : ils proposent une méthode de repérage automatique dans une mémoire de traduction de *patrons de traduction* (essentiellement, des squelettes). Cette méthode repose sur une analyse des co-occurrences d'ensembles de mots dans les couples de la MT. Par exemple, si on examine les deux couples suivants :

> *The commission gave the plan up / La commission a abandonné le plan*
>
> *The government gave all laws up / Le gouvernement a abandonné toutes les lois*

On observe que les mots anglais *gave* et *up* y co-occurrent de même que la séquence *a abandonné*, ce qui nous mène à formuler l'hypothèse qu'il existe une correspondance entre *gave...up* et *a abandonné*. Pour en arriver à un patron de traduction, on doit alors aligner le *complément*, c'est-à-dire le reste des phrases. Ainsi, on établit (en se basant notamment sur d'autres exemples) que *The commission* correspond à *La commission* et que *the plan* correspond à *le plan*, et similairement pour le deuxième couple. Par déduction, on arrive à un patron se voulant une généralisation de ces deux phrases, à savoir :

> X_s *gave* Y_s *up* / X_t *a abandonné* Y_t

Notons que des méthodes similaires pour l'identification de ce genre de patrons ont été proposées par Güvernir et Cicekli [42] et Carl [22]. D'une façon plus générale, la notion même de squelettes ou de patrons de traduction s'apparente très clairement avec le concept d'*exemple généralisé* en TABE (une description de ce concept est donnée dans [106], alors que Brown en donne un exemple typique dans [19]).

2.3 Une étude sur les requêtes soumises au système TransSearch

Nous avons brièvement évoqué à la section 2.2.3 le système TransSearch, qui permet de consulter en-ligne des collections de documents en français et en anglais.

Cet outil correspond en fait assez bien à la définition théorique des mémoires de traduction proposée par Macklovitch et Russell: une “archive de traductions existantes, conçue de façon à faciliter la réutilisation des traductions” [63]. En pratique, TransSearch repose sur des données qui sont essentiellement identiques à celles qu'on retrouve dans les SMT décrits à la section 2.2.2, à savoir un ensemble de paires de phrases traductions l'une de l'autre. C'est principalement le mode d'interrogation qui distingue les deux genres d'application : alors que dans les SMT, on recherche automatiquement des segments de nature fixe (des phrases) d'un texte à traduire, dans TransSearch, c'est l'utilisateur qui prend l'initiative de soumettre des requêtes, qui peuvent correspondre à des segments de nature variable.

Depuis 1997, le prototype élaboré par l'équipe du CITI a été repris en main par le laboratoire RALI de l'Université de Montréal, et depuis le mois de mai 2001, TransSearch opère sur le mode d'un service Internet payant[2]. Chaque semaine les utilisateurs de TransSearch soumettent des dizaines de milliers de requêtes au système. Toutes ces requêtes sont systématiquement inscrite dans un *registre des requêtes*. Ce fichier est utilisé principalement pour le calcul de différentes statistiques à l'intention des gestionnaires du site. Outre le texte d'une requête, on peut extraire de ce registre l'information suivante :

- la date et l'heure à laquelle la requête a été soumise;
- le nom de l'utilisateur qui a soumis la requête;
- l'adresse Internet d'où provenait la requête;
- la base de données interrogée;
- le nombre de résultats (couples) extraits de la base de données;

[2] `http://www.tsrali.com`

- le temps requis pour traiter la requête.

Il nous est apparu qu'il y avait possiblement beaucoup à apprendre du contenu de ce registre. Bien que le RALI ne dispose pas de statistiques précises sur le statut professionnel de sa clientèle, il est assez clair que celle-ci est constituée à peu près exclusivement de traducteurs professionnels (on y compte notamment le Bureau de la traduction du gouvernement canadien et plusieurs des plus importants cabinets de traduction au Canada). Certains utilisateurs soumettent des dizaines de requêtes par jour, parfois des centaines. Tout porte à croire que ces requêtes sont soumises ponctuellement par les traducteurs eux-mêmes, à mesure que les problèmes se présentent. Sans prétendre que le contenu du registre est un reflet transparent du processus mental sous-jacent à l'exercice de traduction, il est raisonnable de penser qu'il offre une perspective nouvelle et éclairante sur la nature des problèmes auxquels sont confrontés les traducteurs et sur leur façon de les aborder ou de les formuler en pratique. Compte tenu des similitudes évidentes entre le système TransSearch et le type d'application qui nous intéresse ici, on peut penser que la nature des informations que les traducteurs vont chercher dans TransSearch est possiblement une bonne indication du genre d'information qu'ils attendent des mémoires de traduction en général.

Nous décrivons donc ici une étude que nous avons menée, basée sur le contenu des *registre de requêtes* du service TransSearch. Après un rapide portrait du système, nous présentons les résultats de notre étude.

2.3.1 TransSearch

TransSearch permet d'accéder à deux imposantes mémoires de traduction en anglais et en français, soit une collection de documents du Hansard canadien (1986-2002) et une collection de documents de nature juridique. En date de janvier 2003, la première de ces collections (*Hansard*) contenait 2010 paires de documents, totalisant près de

100 millions de mots dans chaque langue, alors que la seconde (*Juridique*) contenait 4486 paires de documents, soit environ 5 millions de mots dans chaque langue.

Le processus de construction de ces bases de données procède en trois étapes :

1. **Conversion en format CES** : Les documents, qui sont habituellement obtenus en format HTML, sont convertis vers un format appelé CES (*Corpus Encoding Standard* [45]); il s'agit d'un format XML, permettant des annotations de nature linguistique diverses, notamment une segmentation en paragraphes et en phrases.

2. **Alignement des phrases** : On procède ensuite à un alignement des phrases, au moyen du programme *SFIAL*, dont il est question à la section 2.5.3. Cette opération transforme chaque paire de documents en une séquence de couples. En très grande majorité, ces couples consistent en une simple paire de phrases traduction l'une de l'autre; on retrouve néanmoins une certaine proportion de couples contenant plus d'une phrase d'un côté ou de l'autre, de même que des couples dont une des parties est vide.

3. **Indexation** : Chaque couple est alors inséré dans une base de données textuelles, qui effectue une indexation de tous les mots (TransSearch utilise le système *MG* – voir la section 2.5.2); cette indexation permettra ultérieurement des recherches rapides sur la base du contenu textuel des couples.

TransSearch permet à ses utilisateurs de rechercher des séquences de mots arbitraires dans ces bases de données, via une interface HTML. On peut résumer la syntaxe des requêtes de TransSearch par quelques exemples, tirés du manuel en-ligne du système :

- "`poussières`" : recherche toutes les occurrences de la forme plurielle du mot *poussière*.

- "`mordre la poussière`" : recherche textuellement cette suite de mots.

- "`mordre+ la poussière`" : l'opérateur "+" permet de rechercher toutes les formes fléchies du verbe *mordre* (*mord*, *mords*, *mordons*, *mordu*, etc.); cette requête produirait donc des couples contenant des formes de l'expression comme *mordant la poussière*, *mordu la poussière*, etc.

- "`mordre...poussière`" : l'opérateur "..." permet de rechercher des suites non-contiguës; ici, *mordre* éventuellement suivi dans la même phrase de *poussière*, par exemple *il va* ***mordre*** *une deuxième fois la* ***poussière***.

- "`mordre..poussière`" : l'opérateur ".." est une forme restreinte de l'ellipse ci-dessus, limitant sa portée à 25 caractères.

TransSearch propose en fait deux interfaces pour soumettre des requêtes : une interface dite *simple* et une autre dite *bilingue*. Dans l'interface simple, l'utilisateur soumet une requête sans en spécifier la langue; le système recherche alors l'expression demandée sans égard pour la langue, et affiche tous les couples dans lesquelles celle-ci apparaît, que ce soit dans la partie anglaise ou dans la partie française. Dans l'interface bilingue en revanche, l'utilisateur peut spécifier deux requêtes distinctes, l'une en français et l'autre en anglais, qui sont alors soumises conjointement : le système n'affiche alors que les couples qui satisfont aux deux requêtes. On pourra ainsi rechercher des traductions spécifiques, par exemple pour établir dans quel contexte le mot anglais *commitment* se traduit en français par *attachement*.

Par ailleurs, dans cette même interface, il est possible de laisser vide l'un des deux champs; cette fonctionnalité est particulièrement commode lorsqu'on a affaire à des formes qui sont homographes dans l'une et l'autre langue et qui de ce fait généreraient un bruit excessif dans les résultats d'une requête simple (par exemple : *file*, *rein*, *stage*, *pour*, etc.).

Les résultats d'une requête apparaissent sous la forme d'une suite de couples, dans lesquels l'expression recherchée est mise en évidence. Chaque couple est assorti d'un bouton qui permet de visualiser celui-ci dans un contexte plus large. Normalement, le système n'affiche que les 10 premiers couples trouvés; un bouton au bas de la page permet toutefois de récupérer les résultats suivants, 10 à la fois.

2.3.2 Résultats de l'étude

Pour les fins de cette étude, nous avons concentré notre attention sur les requêtes soumises dans la semaine allant du 3 au 10 novembre 2002. Le tableau 2.1 fournit quelques données statistiques de base sur le contenu du registre couvrant cette période.

Comme on peut le voir, la plupart des requêtes sont dirigées vers le *Hansard*. Cette base de données est d'ailleurs beaucoup plus fournie que l'autre, et présente un niveau de langue beaucoup plus général. Par ailleurs, la très grande majorité (95,5%) des requêtes est soumise dans l'interface simple[3].

Le résultat le plus intéressant pour notre propos est certainement le contenu même des requêtes : près de 85% des requêtes soumises concernent des suites de 2 mots et plus. La figure 2.1 illustre la distribution des requêtes en fonction du nombre de mots qu'elles contiennent. Les requêtes portant sur deux mots sont de loin les plus fréquentes, suivies des requêtes de 3 mots, de 1 mot et de 4 mots. Ceci peut être interprété de deux façons : soit que les utilisateurs de TransSearch sont rarement confrontés à des problèmes d'ordre lexical (c'est-à-dire concernant un mot isolé), soit qu'ils ont plutôt tendance à se tourner vers d'autres ressources dans ces situations (dictionnaires, glossaires, thésaurus, banques terminologiques, etc.). Ce qui est clair toutefois, c'est que les problèmes de traduction prenant la forme de séquences de

[3] Il faut également noter que c'est la base de données qui est interrogée par défaut; l'utilisateur qui souhaite plutôt consulter la base de données *Juridique* doit le mentionner explicitement lorsqu'il soumet une requête.

Nombre total de requêtes		32 671	
	Hansard	27 986	(85,66%)
	Juridique	4 685	(14,34%)
contenu de la requête :			
	mot isolé	5 165	(15,81%)
	séquence contiguë	25 748	(78,81%)
	séquence avec ellipse (...)	1 758	(5,38%)
expansion morphologique (+)		1 095	(3,35%)
interface :	simple	31 201	(95,50%)
	bilingue		
	français et anglais	238	(0,73%)
	français seulement	216	(0,66%)
	anglais seulement	954	(2,92%)
requêtes productives		21 259	(65,07 %)
résultats par requête		5,68	

Table 2.1. Statistiques sur les requêtes soumises à TransSearch (semaine du 3 novembre 2002)

plusieurs mots sont la principale motivation pour utiliser le système.

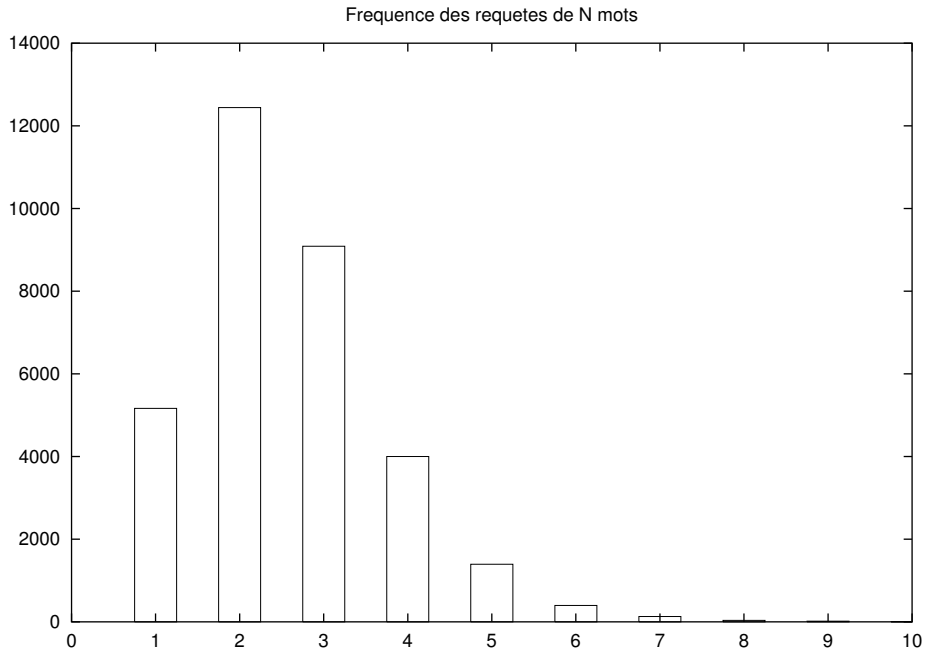

Figure 2.1. Distribution des requêtes en fonction du nombre de mots qu'elles contiennent

On constate également que les mécanismes de recherche plus élaborés (ellipses, expansion morphologique, requêtes bilingues) sont très peu utilisés. En fait, on peut voir les opérateurs "...", ".." et "+" comme des mécanismes de généralisation, c'est-à-dire qu'ils permettent à l'utilisateur d'exprimer un problème dans des termes moins spécifiques que ceux dans lesquelles il est apparu à l'origine. Ainsi, un traducteur qui bûte sur un texte contenant l'expression *porter atteinte*, sous la forme *...ne porteront certainement pas atteinte aux...* a tout à gagner à soumettre une requête plus générale que "`porteront certainement pas atteinte`", par exemple "`porter+..atteinte`". Or, ils le font très rarement.

Ici encore, plusieurs facteurs peuvent expliquer cet état de fait. L'explication la plus simple est que la grande majorité des problèmes rencontrés par les traducteurs

concerne des séquences de mots contigus, sur lesquelles la morphologie flexionnelle n'aurait pas d'impact. Par ailleurs, on peut également penser que les requêtes sont souvent soumises très spontanément, dans des situations de pression, et que les utilisateurs n'ont tout simplement pas le loisir d'effectuer l'effort nécessaire à un exercice de généralisation; les requêtes sont alors soumises dans leur forme la plus simple et la plus directe, peut-être même par une opération de copier-coller. Finalement, on peut soupçonner que les mécanismes d'ellipse et d'expansion sont mal compris de certains utilisateurs, ou alors qu'ils ne les connaissent tout simplement pas. Ce facteur n'est sans doute pas à négliger.

Nous nous sommes intéressé au statut linguistique des requêtes de plus d'un mot. En effet, certaines études psycholinguistiques tendent à indiquer que les traducteurs opèrent naturellement sur des unités qui s'apparentent le plus souvent à des constituants, au sens syntaxique du terme (voir par exemple [9]). Sans aller aussi loin, nous avons voulu savoir dans quelle mesure les requêtes soumises à TransSearch pouvaient s'apparenter à des *tronçons syntaxiques* (notre traduction libre de l'anglais *syntactic chunk*), une notion introduite par Steven Abney [3]. Pour reprendre (à un détail près) l'exemple de Abney :

> *[I begin] [with an intuition]: [when I read] [a sentence], [I read it] [a single chunk] [at a time]*

Cette notion s'inspire en fait de celle de *structure de performance* de Gee et Grosjean [40]. Ces structures correspondent à des groupements de mots au sein de la phrase, dont l'existence est révélée, par exemple, en mesurant expérimentalement la durée des pauses lors d'une lecture à voix haute. Les tronçons de Abney se veulent une formalisation de cette idée en des termes linguistiques. Très grossièrement :

- Une *tête lexicale* est un mot-plein[4];

[4] Un *mot-plein* fait référence à un objet, un concept ou une notion: ceci regroupe habituellement les

- Toute tête lexicale est dite *majeure* si elle ne sépare pas un mot-outil f du mot-plein que f sélectionne;

- Un *tronçon* est formé d'une *tête lexicale majeure* et de ses dépendants immédiats.

Dans l'exemple ci-dessus, le tronçon *a single chunk* illustre le rôle de la notion de tête majeure : *single* et *chunk* sont tous deux des mots-pleins, mais *single* ne peut être considéré comme une tête majeure, parce qu'il sépare le mot-outil *a* du mot-plein *chunk* qu'il sélectionne. Il en résulte donc un tronçon unique *a single chunk*.

Toujours d'après Abney, les tronçons jouent un rôle important en ce qu'ils établissent un pont entre la perspective syntaxique classique (en constituants) et les phénomènes de prosodie [4]. Plus généralement, ils constituent des éléments de forte cohésion interne dans le discours. Si les locuteurs d'une langue en général leur confèrent un tel statut, il n'y a pas a priori de raison de croire que les traducteurs procèdent de façon différente.

Évidemment, la notion de tronçon s'inscrit dans une analyse linguistique; il n'est donc pas possible d'établir a priori si une requête soumise à TransSearch constitue ou non un tronçon ou une suite de tronçons : ce n'est que dans un contexte d'utilisation précis qu'on peut le vérifier. Par exemple, la requête "`le port d'armes`" peut ressembler à un groupe nominal simple, mais dans un contexte comme

> *... les agents portuaires ont effectué une saisie dans* ***le port d'armes*** *automatiques prohibées ...*

la même suite de mots se trouve en fait à chevaucher deux tronçons distincts.

---

noms, les verbes, les adjectifs et certains adverbes. Ils s'opposent aux *mots-outils*, c'est-à-dire les mots n'ayant pas de référence concrète ou notionnelle dans la réalité, et dont le rôle est d'assurer la liaison ou la cohérence entre les principales parties d'un texte (par exemple, les prépositions, déterminants, conjonctions, signes de ponctuation, etc.).

Évidemment, la seule façon d'établir avec certitude la nature des requêtes d'un point de vue linguistique serait d'en vérifier le statut dans le contexte précis du texte d'où ils sont tirés par les utilisateurs. Malheureusement, nous n'avons pas accès à ces données (si même elles existent – après tout rien n'empêche un utilisateur "d'inventer" des requêtes à loisir). En revanche, nous pouvons examiner le contexte dans lequel ces mêmes requêtes apparaissent dans les mémoires de traduction de TransSearch. Nous avons donc mené une expérience en ce sens : nous avons resoumis une partie des requêtes trouvées dans le registre des requêtes pour la période mentionnée ci-dessus, et avons examiné les contextes dans lesquels elles apparaissaient, pour voir si les séquences correspondant aux requêtes coïncidaient avec des suites de tronçons. Pour cette expérience, nous nous en sommes tenu à la partie anglaise des bases de données. Voici comment nous avons procédé :

- Nous avons considéré les 20 000 premières requêtes du registre; ce nombre nous semblait constituer un échantillon suffisant[5].

- Nous avons alors écarté les requêtes ne contenant qu'un seul mot.

- Les requêtes restantes ont été resoumises à TransSearch, chacune sur la même base de donnée que celle choisie par l'utilisateur (*Hansard* ou *Juridique*).

- Ce processus retournait pour chaque requête un ensemble de couples, parmi lesquels nous ne retenions que ceux dans lesquels la requête coïncidait avec un segment de texte dans la partie anglaise; nous appelons *repérage* ce segment de texte. Nous n'avons alors conservé que les 10 premiers couples satisfaisant à cette contrainte (ce nombre correspond à la quantité de couples que TransSearch affiche par défaut lorsqu'on soumet une requête).

[5] En fait, des expériences sur des échantillons beaucoup plus petits ont produit des résultats tout à fait comparables à ceux rapportés ici.

- Nous avons ensuite effectué un tronçonnage de la partie anglaise de chacun des couples retenus. Vue la masse de données à analyser, il était hors de question d'effectuer ces tronçonnages manuellement. Nous nous sommes donc tourné vers une méthode automatique : la tronçonneuse *Texas*, que nous avons utilisée pour cette expérience, est décrite à la section 2.5.1.

- Enfin, nous avons comparé le résultat du tronçonnage avec le repérage, et avons recueilli différentes statistiques sur cette comparaison.

Les principaux résultats de cette expérience apparaissent au tableau 2.2.

Requêtes :		
examinées	20 000	
soumises (2 mots et +)	16 230	(81,15%)
couples produits	63 352	(3,9 par requête)
Taille moyenne des repérages :		
nombre de mots	2,82	
nombre de tronçons	2,00	
Coïncidence des frontières :		
à gauche	34 685	(54,75%)
à droite	53 399	(84,29%)
les deux	28 838	(45,52%)

Table 2.2. Résultats des expériences de tronçonnage sur les couples extraits de TransSearch

Les principaux résultats d'intérêt dans cette table concernent les *coïncidences de frontières*, c'est-à-dire la proportion de couples extraits dans lesquels l'une et/ou

l'autre frontière du repérage coïncide avec une frontière entre deux tronçons[6]. La figure 2.2 donne quelques exemples de telles coïncidences.

requête	repérage tronçonné
responsible and accountable	$[_{NP}$ Exempt] $[_{NP}$ it] $[_{PP}$ from] $[_{NP}$ the things] $[_{NP}$ that] $[_{VP}$ are appropriate and leave] $[_{NP}$ it] $[_{ADJP}$ *responsible and accountable*] $[_{PP}$ for] $[_{NP}$ the rest]
full marks	$[_{NP}$ *Full marks*] $[_{PP}$ for] $[_{NP}$ that lady] and $[_{NP}$ *full marks*] $[_{PP}$ for] $[_{NP}$ the business]
satisfy the condition	$[_{NP}$ Each new member] $[_{VP}$ must] $[_{VP}$ *satisfy*] $[_{NP}$ *the condition*] $[_{PP}$ of] $[_{VP}$ carrying] $[_{PP}$ on] $[_{NP}$ business] $[_{PP}$ in] $[_{NP}$ common] $[_{PP}$ with] $[_{NP}$ a view] $[_{PP}$ to] $[_{NP}$ profit]
there is evidence of	$[_{NP}$ *There*] $[_{VP}$ *is*] $[_{NP}$ *evidence*] $[_{PP}$ *of*] $[_{NP}$ substantial implementation failure]

Figure 2.2. Exemples de requêtes coïncidant avec des frontières de tronçons

L'aspect le plus intriguant de ces résultats est certainement l'apparente asymétrie des coïncidences de frontières : alors que la fin du repérage coïncide presque toujours avec une frontière de tronçon (près de 85%), pour le début du repérage, celà ne se produit que dans 55% des cas.

Cette différence est d'autant plus surprenante quand on prend en compte la longueur moyenne des tronçons, qui est d'environ 1,55 mots. Dans ces conditions, la probabilité que l'une des extrémités d'un segment de texte arbitraire coïncide avec une

[6] la tronçonneuse peut identifier certains mots d'une phrase comme n'appartenant à aucun tronçon (étiquette "O"); pour les fins de cette expérience, nous avons considéré que toute suite de tels mots "O" constituait un tronçon unique, de type inconnu ("UNK").

frontière de tronçon est d'environ 1/1,55 (0,64). En somme, les résultats obtenus tendent à indiquer que le début des requêtes soumises par les utilisateurs de TransSearch ont moins tendance à coïncider avec des frontières de tronçons que des séquences arbitraires.

Un examen sommaire des résultats de requêtes permet toutefois de formuler une explication relativement simple à cette énigme. Lorsque le début d'un repérage coïncide avec celui d'un tronçon, ce dernier est le plus souvent un groupe nominal (NP), comme on peut le constater dans le tableau 2.5, qui donne la fréquence d'apparition des différents types de tronçons au début des repérages, lorsque les frontières des deux coïncident (la table 2.6 présente des statistiques analogues pour les fins de repérages, alors que le tableau 2.7 affiche les séquences de tronçons auxquelles correspondent le plus souvent les repérages). Or en anglais, la tête lexicale de ce type de tronçon a tendance à se trouver à la fin du tronçon, les modifieurs se trouvant devant. On peut donc soupçonner que beaucoup de requêtes débutent en fait par la tête d'un groupe nominal, optionnellement précédée de quelques modifieurs. En pratique, le début de ces requêtes coïncide rarement avec la frontière du tronçon, parce que d'autres modifieurs apparaissent normalement au début de celui-ci, le plus fréquent étant un déterminant.

Par exemple, la toute première requête dans la partie du registre que nous avons examinée est "`civil courts`" (en français : *cours civiles*). Dans les 10 couples extraits par TransSearch pour cette requête, le repérage apparaît dans un tronçon de type NP; mais dans 6 de ces couples, il est précédé d'au moins un prémodifieur (le déterminant *the* 5 fois sur 6). La figure 2.3 en donne quelques exemples (le tableau 2.3 donne la signification des étiquettes apparaissant au début de chaque tronçon [10]).

L'exemple 4 dans cette figure constitue le seul cas où la frontière droite du repérage ne coïncide pas avec la fin du tronçon; on constate en fait qu'il s'agit d'une erreur de tronçonnage (possiblement attribuable à l'absence d'une virgule après *civil courts*).

1. [$_{NP}$ Those matters] [$_{VP}$ are best left] [$_{PP}$ to] [$_{NP}$ the **civil courts**] .
2. [$_{NP}$ It] [$_{VP}$ mirrors] [$_{NP}$ the legislation] [$_{NP}$ that] [$_{VP}$ was introduced] [$_{PP}$ in] [$_{NP}$ the House] [$_{VP}$ dealing] [$_{PP}$ with] [$_{NP}$ the DNA identification data bank] [$_{PP}$ in] [$_{NP}$ the **civil courts**] .
3. [$_{NP}$ If] [$_{ADVP}$ only] [$_{NP}$ they] [$_{VP}$ were being tried] [$_{PP}$ in] [$_{NP}$ American **civil courts**] .
4. [$_{PP}$ In] [$_{NP}$ **civil courts** crown liability] [$_{VP}$ exists] [$_{PP}$ by] [$_{NP}$ virtue] [$_{PP}$ of] [$_{NP}$ the Crown Liability and Proceedings Act] .

Figure 2.3. Exemples de repérages pour la requête *civil courts*

étiquette du tronçon	signification
ADJP	groupe adjectival
ADVP	groupe adverbial
CONJP	groupe conjonctif
INTJ	interjection
NP	groupe nominal
PP	groupe prépositionnel
PRT	particule
SBAR	relative ou subordonnée
UCP	groupe coordonné dissemblable
UNK	inconnu
VP	groupe verbal

Table 2.3. Signification des étiquettes de tronçons

Pour vérifier l'hypothèse suivant laquelle le début de plusieurs requêtes correspond à un "tronçon nominal abrégé", nous avons comptabilisé les situations où le début du repérage se trouvait en fait à coïncider avec le deuxième mot d'un tronçon NP, le premier mot étant l'un des déterminants *the*, *a* ou *an*. Le tableau 2.4 reprend les statistiques de coïncidences des frontières, cette fois en tenant compte de cette éventualité.

Coïncidence des frontières :		
à gauche	43 821	(69,17%)
à droite	53 399	(84,29%)
les deux	36 655	(57,86%)

Table 2.4. Coïncidences entre les repérages et les frontières de tronçon, tenant compte d'un déterminant précédant le début du repérage.

Comme on le voit, le simple fait d'admettre la présence d'un déterminant devant le repérage permet de faire monter la proportion de coïncidences à gauche de plus de 25%, au dessus de la "barre du hasard". Clairement, l'admission d'autres types de modifieurs (adjectifs, noms communs, etc.) mènerait à des constatations similaires. On peut également soupçonner que des phénomènes similaires se produisent avec certains groupes verbaux (omission d'un auxilliaire ou d'un verbe modal), adjectivaux, etc.

Si on suppose que les utilisateurs de TransSearch formulent mentalement leurs problèmes de traduction sous la forme de séquences de tronçons, comment expliquer cette tendance à tronquer ainsi le début du tronçon initial? Différents facteurs y concurrent probablement. D'abord, on peut penser qu'ils agissent ainsi par souci de concision ou d'efficacité : pourquoi insérer un déterminant initial qui n'ajoute rien au sens de l'expression? À cet égard, on peut également soupçonner que les utilisateurs de TransSearch "importent" certains réflexes, développés avec l'utilisation de moteurs

de recherche sur Internet (élimination des mots-outils). Par ailleurs, l'escamotage des modifieurs initiaux trahit peut-être également un mécanisme, conscient ou non, de généralisation de la requête : en omettant des modifieurs qui ne contribuent pas effectivement au problème, l'utilisateur augmente ses chances de trouver des solutions récupérables. L'expérience joue sans doute un rôle important dans ces mécanismes : il est possible que les utilisateurs du système acquièrent après un certain temps une intuition du niveau de spécificité au-delà duquel une requête n'est plus susceptible de fournir des résultats intéressants.

premier tronçon	fréquence relative
NP-	0,3711
VP-	0,2789
PP-	0,2204
ADVP-	0,0542
ADJP-	0,0384
SBAR-	0,0171
UNK-	0,0064
PRT-	0,0048
CONJP-	0,0015
INTJ-	0,0008
UCP-	0,0007

Table 2.5. Distribution des types de tronçons coïncidant avec le début d'une requête TransSearch

En résumé, nous retenons de cette étude que

- les utilisateurs de TransSearch soumettent principalement des requêtes correspondant à des suites de deux mots contigus ou plus; la très grande majorité de ces requêtes vise à retrouver des séquences verbatim (peu d'utilisation des

dernier tronçon	fréquence relative
-NP	0,5152
-PP	0,2008
-VP	0,1499
-ADJP	0,0451
-ADVP	0,0362
-SBAR	0,0302
-PRT	0,0122
-UNK	0,0039
-INTJ	0,0007
-CONJP	0,0001
-UCP	0,0000

Table 2.6. Distribution des types de tronçons coïncidant avec la fin d'une requête TransSearch

mécanismes de généralisation).

- ces séquences correspondent majoritairement à des séquences de tronçons syntaxiques, le début du tronçon initial étant occasionnellement tronqué, ce qui a pour effet d'éliminer certains prémodifieurs, tout en conservant la tête lexicale;

- la composition des séquences est extrêmement variée, les séquences les plus fréquemment observées étant constituées de 3 tronçons ou moins, majoritairement des groupes nominaux, verbaux et prépositionnels.

2.4 Unités de traduction pour les SMT

Nous examinons ici la question de l'*unité de traduction* la plus adéquate pour un SMT. L'objectif visé par les SMT, rappelons-le, est d'augmenter la productivité des

séquence de tronçons	fréquence relative	séquence de tronçons	fréquence relative
NP	0,1988	VP-NP-PP	0,0099
PP-NP	0,1451	NP-VP-NP	0,0099
VP	0,0629	PP-NP-PP-NP	0,0086
VP-NP	0,0585	ADJP-PP	0,0086
VP-PP	0,0577	VP-SBAR	0,0085
NP-PP	0,0414	NP-VP-SBAR	0,0077
NP-VP	0,0381	ADVP-PP	0,0064
PP-NP-PP	0,0380	SBAR-NP-VP	0,0063
NP-PP-NP	0,0194	NP-VP-ADJP	0,0059
ADJP	0,0172	PP-VP	0,0053
VP-PP-NP	0,0169	NP-VP-PP	0,0053
VP-ADVP	0,0114	ADJP-PP-NP	0,0049
VP-ADJP	0,0113	ADVP-PP-NP	0,0047
ADVP-VP	0,0105	VP-ADJP-PP	0,0045
VP-PRT	0,0101	PP-NP-VP	0,0045
ADVP	0,0100	NP-VP-NP-PP	0,0045
		autres	0,1472

Table 2.7. Distribution des séquences de tronçons coïncidant le plus fréquemment avec une requête TransSearch

traducteurs, en leur évitant de refaire du travail qui a déjà été fait. En pratique, cette assistance peut prendre deux formes :

1. **Cognitive** : Suggérer une formulation qui ne serait peut-être pas venue à l'idée de l'utilisateur pour la traduction d'un segment donné.

2. **Ergonomique** : Accélérer la production de la traduction, en limitant les manipulations requises pour sa saisie.

La première forme d'assistance suggère une situation où le traducteur ne sait pas d'emblée comment traduire un certain segment, alors que la deuxième sous-entend plutôt que le traducteur a fixé son choix sur une formulation, et qu'il ne reste en définitive qu'à la mettre en forme. Malgré tout, ces deux éventualités ne sont pas mutuellement exclusives, de telle sorte que les deux formes d'assistance peuvent entrer simultanément en jeu pour un seul et même segment.

Pour qu'une suggestion émanant d'un SMT soit jugée utile par un traducteur, elle doit donc répondre de façon satisfaisante à trois critères distincts :

1. **Pertinence** : la proposition doit de toute évidence constituer une traduction valable pour une partie ou la totalité du segment de texte à traduire.

2. **Adaptabilité** : par ailleurs, il faut également que cette proposition puisse s'insérer de façon cohérente dans l'ensemble de la traduction que produit l'utilisateur. Ce critère s'appuiera non seulement sur les aspects syntaxiques et morphologiques, mais aussi sur des considérations d'ordre stylistique.

3. **Rentabilité** : les manipulations requises pour insérer la proposition dans la traduction en devenir ou, alternativement, pour construire une traduction à l'entour de cette proposition doivent entraîner une économie réelle dans la mise en forme du produit fini, ou du moins la perception d'une économie.

L'importance de ce critère est bien entendu fonction de facteurs ergonomiques liés à la nature de l'interface-utilisateur, mais également aux compétences de l'utilisateur en cette matière (vitesse de frappe, préférences quant à l'utilisation du clavier et du pointeur, etc.).

Ces deux derniers critères suggèrent déjà certaines contraintes concernant la nature des unités de traduction pour un SMT. Considérons d'abord la simple *rentabilité* : il est assez clair qu'à moins d'une révolution dans le design des interfaces-utilisateur des SMT, l'emploi d'une proposition émanant de la MT entraînera un coût de base non-négligeable en terme de manipulations, que ce soit une opération de copier-coller, de glisser-déposer, un clic de souris, une séquence de touches au clavier, ou même le simple fait de déplacer le curseur par-dessus la proposition dans une prétraduction partielle. Considérant que la plupart des traducteurs sont maintenant des dactylos aguerris, on peut présager que lorsque les propositions du SMT sont courtes, le traducteur préférera souvent les saisir directement au clavier plutôt que d'entreprendre une manipulation de récupération. Abstraction faite de l'aspect cognitif, on peut donc considérer que les propositions "trop courtes" seront perçues comme inutiles par le traducteur, voire irritantes.

Du point de vue de l'*adaptabilité*, on sait que les problèmes à cet égard se présentent principalement (mais pas toujours) aux frontières de propositions; la littérature en TABE évoque d'ailleurs abondamment ces questions de *friction aux frontières* (en anglais : *boundary friction* [106]). Sous l'hypothèse que les propositions formulées par le SMT affichent une cohérence interne, une longue proposition présente naturellement moins de frontières que plusieurs petites propositions, et donc potentiellement moins de problèmes de cet ordre. À la limite, une proposition qui couvre une grande partie de la phrase d'origine (disons, plus de la moitié) a de plus fortes chances d'être employée par l'utilisateur comme point de départ pour toute sa traduction, ce qui limite appréciablement les problèmes de friction mentionnés ci-dessus.

Notons d'autre part que ce même argument milite en faveur de propositions qui prennent la forme de *suites de mots contigus*, par opposition à des "propositions à trous" que le traducteur doit remplir manuellement (des squelettes partiellement instanciés, à la façon de Langé *et al.* [57], par exemple). On remarque également que l'instanciation automatique des variables dans ces mêmes squelettes soulève des problèmes aigus de cohérence interne, nécessitant l'implantation de mécanismes complexes d'adaptation linguistique [69].

En définitive, il revient probablement à l'utilisateur de décider de la longueur minimale des propositions du SMT selon ses préférences. Mais a priori, on peut déjà considérer, à l'instar de Langé et al. [57], que les segments de traduction proposés au traducteur ont d'autant plus de chance d'être utiles à celui-ci qu'ils sont longs.

D'autre part, McTait *et al.* [69] proposent un type de SMT opérant sur des unités s'apparentant à celles sur lesquelles opèrent naturellement les humains. Ce choix est motivé par un ensemble d'études psycholinguistiques, visant à expliciter comment les humains procèdent lorsque confrontés à une tâche de traduction, et plus précisément quelle part du traitement est effectuée de façon "automatique" et quelle part de façon "rationnelle".

De ces études, McTait *et al.* tirent essentiellement trois conclusions relatives à l'implantation des SMT :

1. Les traducteurs humains ne procèdent habituellement pas de façon linéaire dans un texte à traduire, c'est-à-dire du début vers la fin, mais plutôt en suivant un schéma *retrospectif-prospectif*, c'est-à-dire en effectuant de fréquents retours en arrière (*backtracking*) et incursions non-linéaires vers l'avant. Ceci supporterait l'idée d'un SMT capable d'appliquer et de réappliquer ses connaissances de façon dynamique.

2. Les traducteurs humains opèrent naturellement sur des ensembles de mots de taille variable, allant du mot isolé à la proposition complexe, mais rarement

jusqu'à la phrase complète. La taille moyenne de ces ensembles varie principalement en fonction de l'expérience du traducteur, les traducteurs plus expérimentés maniant plus aisément de longues propositions que les traducteurs débutants. On peut voir ce processus comme une forme d'automatisation de la traduction, qui se développe graduellement avec l'expérience. Ceci supporte le principe même sur lequel se base tout le concept de SMT, mais souligne également la nécessité d'opérer "sous" le niveau de la phrase si l'on aspire à une certaine efficacité.

3. La nature elle-même des unités de traduction sur lesquelles opèrent naturellement les traducteurs humains varie, mais elle s'apparente le plus souvent à celle de groupements de mots "linguistiquement motivés", c'est-à-dire des constituants au sens syntaxique du mot. Partant du principe que les traducteurs humains seront plus enclins à réutiliser des segments de traduction qui coïncident avec leur propre "découpage mental" du texte-source, on conclut qu'un SMT devrait également procéder sur ce genre d'unités.

Clairement, les résultats de notre étude sur les requêtes soumises au système TransSearch (section 2.3) vont dans la même direction que cette dernière observation. Qui plus est, ils semblent confirmer que c'est précisément de l'information de cette nature que les traducteurs attendent des mémoires de traduction.

À l'instar de Planas [82], nous suggérons donc que les séquences de tronçons constituent une unité de traduction particulièrement appropriée pour les SMT :

- Les tronçons constituent des groupements de mots sous-phrastiques de taille minimale, correspondant à une réalité relativement intuitive pour les traducteurs.

- Quoique les tronçons soient typiquement limités dans leur taille à quelques mots seulement, le fait de considérer des séquences de tronçons permet de récupérer

des séquences de taille variable, potentiellement des phrases complètes.

- Comme les séquences de tronçons coïncident avec des suites de mots contigus dans les phrases de la MT, on évite en principe les problèmes de cohérence interne qu'on rencontre avec des éléments composites tels que les squelettes ou patrons de traduction présentés à la section 2.2.3.

- Par ailleurs, les tronçons se prêtent particulièrement bien à des méthodes de détection automatique (voir la section 2.5.1).

Mais sur quelles séquences faut-il précisément qu'un SMT s'attarde? Les résultats de la section 2.3, et plus spécifiquement la distribution observée des séquences de tronçons auxquelles coïncident les requêtes (tableau 2.7) démontrent la grande variété des types de constituants auxquels s'intéressent les utilisateurs de TransSearch. Ceci suggère qu'on ne peut pas déterminer a priori quelles séquences sont susceptibles d'être utiles, et qu'il est donc souhaitable que toutes les séquences existantes dans la MT soient éventuellement disponibles.

2.5 Implantation d'un SMT sous-phrastique

Dans un SMT opérant sur des séquences de tronçons syntaxiques, il n'est plus absolument nécessaire que la MT soit constituée de paires de phrases : conceptuellement, on peut simplement la voir comme une collection $\mathcal{M}$ de paires de documents $\langle S, T \rangle$ traduction l'un de l'autre, S étant un document en langue-source et T sa traduction en langue-cible. Devant un nouveau document à traduire D, le système procède comme suit :

1. **tronçonnage du nouveau texte** : Le nouveau texte à traduire D est tronçonné automatiquement, produisant une séquence de tronçons $D = d_1...d_n$.

2. **recherche des séquences** : On recherche alors dans les textes en langue-source de la mémoire de traduction $\mathcal{M}$ toutes les sous-séquences $d = d_i...d_j$ de D; pour chaque séquence d, on obtient donc un ensemble $\mathcal{M}_d$ (possiblement vide) de paires de documents $\langle S, T \rangle$, telles que la séquence de tronçons d apparaît dans le document S.

3. **repérage de traduction** : Dans chaque paire $\langle S, T \rangle$, on identifie la portion t_d du document T qui constitue la traduction de la séquence d.

4. **sélection des candidats** : En dernier lieu, on doit effectuer une sélection parmi l'ensemble des traductions t_d trouvées pour les séquences d du document D. Cette sélection constituera l'ensemble des propositions faites par le SMT au traducteur.

Nous examinons plus en détail chacune de ces étapes ci-dessous.

2.5.1 Tronçonnage

Les méthodes de tronçonnage automatique ont reçu beaucoup d'attention récemment, particulièrement auprès de la communauté des chercheurs en apprentissage-machine des langues naturelles, qui leur ont consacré une part importante de la conférence CoNNL-2000 [21]. Dans le cadre de cet événement, on avait organisé une compétition, visant à comparer la performance de différentes approches d'apprentissage automatique sur cette tâche. Les participants avaient accès à un ensemble de données d'entraînement commun, extraites du *Penn Treebank* [66], un corpus de texte enrichi d'annotations syntaxiques, à partir desquelles un tronçonnage de référence avait été déduit de façon automatique [20].

La performance de tous les systèmes ainsi développés étaient mesurée sur un même échantillon, extrait du même corpus, et mesurée en termes de précision (proportion des tronçons de la référence identifiés par le système), de rappel (proportion des

tronçons identifiés automatiquement qui coïncident avec ceux de la référence) et de *F-measure* [107] : $(precision \times rappel)/(precision + rappel)$. La majorité des participants a obtenu des résultats excédant 0,91 dans toutes ces métriques [100].

Presque toutes les méthodes proposées considèrent le tronçonnage comme une tâche d'*étiquetage* : il s'agit d'associer à chaque mot du texte une étiquette, qui prend l'une des formes "B-X" (le mot marque le début d'un tronçon de type X), "I-X" (le mot fait partie d'un tronçon de type X, mais n'en est pas le premier mot) ou "O" (le mot ne fait pas partie d'un tronçon). Nous désignons ce genre d'étiquette par le terme *étiquette IOB*.

Nous décrivons ici la méthode proposée par Osborne [80], qui effectue cet étiquetage au moyen des étiqueteurs statistiques à entropie maximale de Ratnaparkhi [96]. Osborne utilise en fait trois étiqueteurs distincts, chacun agissant sur la sortie des précédents. Étant donnée une suite de mots $w_1 ... w_n$, son système procède de la façon suivante :

1. Le premier étiqueteur associe à chaque mot w_i une étiquette syntaxique t_i, par exemple NN (un nom commun), DT (déterminant), etc. En somme, c'est un étiqueteur morpho-syntaxique classique.

2. Le second étiqueteur prend en entrée une concaténation du mot w_i et de l'étiquette syntaxique t_i qui lui a été associée par le premier étiqueteur, et associe à ce symbole $w_i \cdot t_i$ une étiquette IOB c_i.

3. Le troisième étiqueteur vise à faire intervenir un contexte plus large dans la prise de décision : son entrée consiste en une concaténation des étiquettes t_i et c_i associées au mot en position i, tels que produites par les deux étiqueteurs précédents, à laquelle on ajoute également les étiquettes t_{i+1} et c_{i+1} du mot suivant (position $i + 1$). À ce symbole $t_i \cdot c_i \cdot t_{i+1} \cdot c_{i+1}$, le troisième étiqueteur associe une nouvelle étiquette IOB c'_i. C'est cette dernière qui constitue la sortie

du processus[7].

La figure 2.4 illustre ce processus sur un exemple simple.

étiquetteur 1		
the	→	DT
black	→	AJ
cat	→	NN
eats	→	VB
the	→	DT
white	→	AJ
mouse	→	NN
.	→	PT

étiquetteur 2		
the·DT	→	B-NP
black·AJ	→	I-NP
cat·NN	→	I-NP
eats·VB	→	B-VP
the·DT	→	B-NP
white·AJ	→	I-NP
mouse·NN	→	I-NP
.·PT	→	O

étiquetteur 3		
DT·B-NP·AJ·I-NP	→	B-NP
AJ·I-NP·NN·I-NP	→	I-NP
NN·I-NP·VB·B-VP	→	I-NP
VB·B-VP·DT·B-NP	→	B-VP
DT·B-NP·AJ·I-NP	→	B-NP
AJ·I-NP·NN·I-NP	→	I-NP
NN·I-NP·PT·O	→	I-NP
PT·O·–·–	→	O

Figure 2.4. Tronçonnage à la façon de Osborne

Nous avons produit notre propre implantation de la tronçonneuse de l'anglais de Osborne, que nous avons baptisée *Texas*. La principale différence entre l'implantation de Osborne et la notre réside dans notre utilisation d'étiqueteurs basés sur des modèles de Markov cachés (*HMM*), à la façon de Church [24] (nous utilisons l'implantation de George Foster [34]). Dans la tâche commune de la conférence CoNNL-2000, la tronçonneuse de Osborne affichait une performance en *F-measure* de près de 0,92. Bien que nous ayons entraînée *Texas* sur les mêmes données, un examen informel de nos tronçonnages révèle que notre performance est appréciablement inférieure (aux alentours de 85%). Au moins deux facteurs expliquent cette différence :

- L'implantation d'Osborne n'impliquait en fait que deux étiqueteurs, puisque les

[7] La configuration optimale proposée par Osborne est en fait un peu plus complexe, faisant également intervenir dans le symbole d'entrée du troisième étiqueteur des sous-chaînes du mot w_i; mais en définitive, ces subtilités jouent un rôle mineur dans la performance obtenue et sont de peu d'intérêt dans le présent exposé.

étiquettes morpho-syntaxiques étaient fournies dans les corpus d'entraînement et de test. Osborne (comme tous les participants à cette compétition) pouvait donc compter sur un étiquetage morpho-syntaxique "parfait", ce qui n'est pas notre cas.

- Les évaluations de la conférence CoNNL-2000 étaient faites sur un échantillon de test issu du même corpus que le matériel d'entraînement. Nos propres évaluations ont été faite sur le Hansard, dont le contenu diffère substantiellement de celui du Penn Treebank (qui provient du *Wall Street Journal*).

Malgré tout, nous obtenons des tronçonnages de l'anglais qui sont en général acceptables. La figure 2.5 en montre un exemple.

[$_{NP}$ The government] [$_{VP}$ is putting] [$_{NP}$ a $2,2 billion tax] [$_{PP}$ on] [$_{NP}$ Canada] [$_{NP}$'s most vulnerable industry] , [$_{NP}$ the airline industry] .

Figure 2.5. Exemple d'un tronçonnage obtenu avec *Texas*

Ce type de tronçonneuse peut traiter efficacement de grandes quantités de données : pour un jeu d'étiquettes de taille N, un modèle de Markov caché permet d'obtenir la séquence d'étiquette la plus probable d'un segment de texte de longueur T en un temps dans $\mathcal{O}(N^2T)$ (algorithme *Viterbi* [95]). Dans notre application, la taille des jeux d'étiquettes est fixe, de telle sorte qu'on peut effectivement considérer que la tronçonneuse opère en temps linéaire.

Tel que mentionné plus haut, le tronçonnage identifie parfois certains mots comme n'appartenant à aucun tronçon (étiquette "O"). Pour les fins de notre application, il est commode de considérer toute séquence de mots ainsi étiquettés comme appartenant à un même tronçon. Par convention, nous attachons une étiquette "UNK"

(inconnu) à ces tronçons.

Notons finalement que le processus de tronçonnage présuppose un découpage préalable du texte en mots. Pour la majorité des langues européennes, ce processus est relativement simple. Pour notre part, nous avons mis au point un système de segmentation en mots, appelé *maskot* (acronyme "récursif" pour *Maskot: A Simple Kind Of Tokenizer*), dont le fonctionnement repose sur une série de transformations du texte au moyen d'expression régulières du langage *Perl*. Ce programme segmente de façon acceptable des textes en différentes langues, telles que l'anglais, le français, l'espagnol, l'italien et l'allemand. Il va sans dire que pour des langues dont la morphologie est très riche, comme le finnois ou le turc, ou dont le système d'écriture ne marque pas explicitement les frontières entre les mots, tel que le chinois, l'utilisation d'un tel système de segmentation est hors de question, et il faut alors recourir à des mécanismes plus complexes.

2.5.2 Recherche des séquences

L'étape de recherche des séquences (étape 2 ci-dessus) semble supposer qu'il faut également procéder à un tronçonnage des documents en langue-source de la MT, puisque ce sont des séquences de tronçons que nous recherchons. En fait, ce n'est pas nécessairement le cas : à chaque sous-séquence de tronçons d correspond une suite de mots $w = w_k...w_l$ dans le texte D, et on peut se contenter de rechercher cette suite. Ceci nous permet d'avoir recours à des méthodes de recherche de chaîne standard pour calculer les ensembles de documents $\mathcal{M}_d$ correspondant à la séquence d.

Toutefois, comme on l'a mentionné un peu plus haut, la nature exacte du découpage en tronçons d'une suite de mots dépend en fait du contexte dans lequel elle s'insère. On ne peut donc pas présumer a priori que l'occurrence d'une suite de mots w dans un document S de la MT se prête exactement au même tronçonnage $(d_i...d_j)$ que dans le document D. Pour s'en assurer, il est donc nécessaire d'effectuer le tronçonnage de S, ou tout du moins de la phrase dans laquelle w s'insère dans S.

On peut toutefois se demander si une identité exacte entre les tronçonnages de w dans D et dans les documents de la MT est absolument essentielle dans ce contexte d'utilisation. Les résultats de notre étude de la section 2.3 laissent à penser qu'une même séquence de mots peut apparaître dans des tronçonnages différents (par exemple, précédée de prémodifieurs différents), sans pour autant que ces différences marquent des usages différents, et de là des traductions différentes ou moins utilisables. C'est pourquoi nous suggérons en fait que la stricte identité des tronçonnages n'est pas essentielle pour un SMT opérant au niveau de séquences de tronçons, et donc que le tronçonnage de la MT n'est pas nécessaire. On pourra néanmoins tenir compte de ce facteur lors de la sélection finale des propositions, que nous traitons plus loin.

La recherche de toutes les sous-séquences de tronçons dans D soulève évidemment des problèmes de temps de calcul, et il convient de mettre en place des mécanismes appropriés à cette fin, notamment une forme quelconque d'indexation du contenu de la MT. Des méthodes d'indexation classiques, telle que l'utilisation d'indexes inversés, sont possiblement adéquates, quoique pas nécessairement optimales pour ce genre de tâches. Des structures telles que des tableaux de suffixes, arbres *Patricia*, graphes de mots acycliques dirigés, etc. sont probablement plus appropriées [7]. Quelle que soit la méthode d'indexation employée, on peut accélérer le processus en recherchant, à partir d'un point de départ donné, des séquences de longueurs croissantes; dès qu'une recherche s'avère infructueuse, on passe au point de départ suivant, et ainsi de suite. Il peut également s'avérer utile de mémoriser les résultats des recherches pour les séquences plus fréquentes (utilisation d'une *cache*).

Par ailleurs, dans la mesure où l'on considère que les séquences "courtes" ne sont pas susceptibles de mener à des propositions intéressantes ou réutilisables pour l'utilisateur, on voudra possiblement les ignorer lors de cette étape de recherche. Sachant qu'il existe une forte corrélation entre la longueur des séquences source et cible qui sont traduction l'une de l'autre, on peut fixer d'emblée une longueur mini-

male (mesurée en nombre de mots) pour les séquences de tronçons à rechercher dans la MT, en fonction des préférences de l'utilisateur quant à la taille minimale des propositions faites par le système. Par exemple, si l'utilisateur spécifie qu'il n'est pas intéressé à se faire proposer des séquences (en langue-cible) de moins de 5 mots, il n'y a sans doute pas d'intérêt à rechercher dans la MT des séquences (en langue-source) de moins de 3 mots.

Dans tous les cas, on voudra éviter les "séquences" de tronçons qui ne recouvrent qu'un seul mot de la langue-source. La recherche de telles séquences serait inutilement coûteuse en temps et en mémoire, et intuitivement, elle serait contre-productive. Ceci nous ramènerait en effet à un système *mot-à-mot* tel qu'évoqué à la section 2.1. Or, tout porte à croire que pour des mots isolés, les traducteurs préfèrent se tourner vers d'autres ressources que les MT. Cette intuition semble d'ailleurs confirmée par nos observations de la section 2.3.2 sur l'utilisation du système TransSearch.

Dans le cadre de nos expériences, nous nous sommes contenté d'une solution sous-optimale au problème de la recherche des séquences de tronçons, qui repose sur l'utilisation d'un système de gestion documentaire issu du domaine public, le système *MG* (acronyme de *Managing Gigabytes* [115]). Ce système permet de stocker de grandes masses de données textuelles, puis d'en extraire efficacement des segments au moyen de requêtes de différents types. Pour retrouver une séquence de mots $w = w_1...w_m$, on effectue d'abord une requête booléenne sur l'ensemble des mots de la séquence : "w_1 ET w_2 ET ... w_m". Pour éliminer les documents ainsi produits dans lesquels ces mots n'apparaissent pas dans le bon ordre, il suffit alors d'effectuer une comparaison de chaîne. Notons que c'est essentiellement sur ce mécanisme que repose le moteur de recherche du système TransSearch. Avec ce système, la recherche de toutes les occurrences d'une séquence donnée dans le corpus *Hansard* s'effectue normalement en quelques dixièmes de secondes. Un tel délai serait inacceptable pour l'implantation d'un réel SMT, puisque le traitement d'un document de quelques pages peut entraîner la recherche de plusieurs milliers de séquences. Néanmoins, cette

solution est acceptable dans le cadre de nos expériences.

La figure 2.6 illustre toutes les séquences de tronçons trouvées (sans vérification de coïncidence des tronçonnages) dans le corpus *Hansard* du système TransSearch pour la phrase de la figure 2.5 (la mention "1000+" dans la colonne du nombre d'occurrences indique une séquence retrouvée plus de 1000 fois). La figure 2.7 donne quelques exemples de couples extraits de cette même MT pour la dernière séquence de cette phrase.

Séquence de tronçons	**nombre d'occurrences**
The government	1000+
The government is putting	188
is putting	1000+
a $2.2 billion tax	3
a $2.2 billion tax on	2
a $2.2 billion tax on Canada	1
Canada 's most vulnerable industry	2
's most vulnerable industry	2
, the airline industry	22
the airline industry	764

Figure 2.6. Séquences de tronçons trouvées dans la MT pour la phrase de la figure 2.5

2.5.3 Repérage des traductions

Les méthodes de repérage de traductions font l'objet du chapitre 3 de la présente thèse. La figure 2.8 présente des exemples de repérages de traductions possibles sur les couples de la figure 2.7.

In closing , I will say that I am sad for workers in **the airline industry**	↔	En terminant , je dirai que c' est triste pour les travailleurs et les travailleuses du secteur de l' aviation .
My colleague spoke about **the airline industry** .	↔	Mon collègue a parlé de l' industrie du transport aérien .
People in **the airline industry** have become unemployed .	↔	Des gens de l' industrie aérienne sont devenus chômeurs .
This tax will cripple some of the small companies in **the airline industry** .	↔	Cette surtaxe va nuire aux petits transporteurs aériens .

Figure 2.7. Exemples de couples extraits de la MT *Hansard* **pour le tronçon** *the airline industry*

In closing , I will say that I am sad for workers in **the airline industry**	↔	En terminant , je dirai que c' est triste pour les travailleurs et les travailleuses **du secteur de l' aviation** .
My colleague spoke about **the airline industry** .	↔	Mon collègue a parlé de **l' industrie du transport aérien** .
People in **the airline industry** have become unemployed .	↔	Des gens de **l' industrie aérienne** sont devenus chômeurs .
This tax will cripple some of the small companies in **the airline industry** .	↔	Cette surtaxe va nuire aux petits **transporteurs aériens** .

Figure 2.8. Repérages de traductions pour les couples de la figure 2.7

Sans même entrer dans le détail des méthodes utilisées pour cette étape, on peut déjà se demander dans quelle mesure cette opération peut ou doit être précalculée dans la MT. Par exemple, dans le cadre d'un système de traduction automatique combinant des méthodes de TA statistique et basée sur les exemples, Marcu [65] propose d'utiliser une mémoire de traduction constituée de paires de séquences de mots contigus. Cette MT est construite automatiquement à partir d'un corpus de paires de phrases appariées (des couples) : Marcu utilise d'abord ces données pour acquérir les paramètres d'un modèle de traduction statistique; il utilise ensuite ce modèle pour calculer des alignements entre les mots des couples du corpus; il extrait finalement toutes les paires de séquences de mots contigus contenues dans ces alignements (voir l'exemple à la figure 2.9). Dans la MT résultante, les repérages de traduction sont donc entièrement précalculés.

Couple :		
Aucun syndicat particulier ne est en cause	↔	There is no one union involved
Alignement :		
Aucun ... ne	↔	no
syndicat	↔	union
particulier	↔	one
est	↔	is
en cause	↔	involved
Paires de séquences contiguës :		
syndicat particulier	↔	one union
aucun syndicat particulier ne	↔	no one union
aucun syndicat particulier ne est	↔	is no one union
aucun syndicat particulier ne est	↔	there is no one union
aucun syndicat particulier ne est en cause	↔	is no one union involved
aucun syndicat particulier ne est en cause	↔	there is no one union involved

Figure 2.9. Précalcul des repérages dans la MT proposée par Marcu.

Nous verrons au chapitre 3, où nous traitons en détail de la question du repérage des traductions, qu'il existe des avantages à effectuer cette opération "en temps réel", c'est-à-dire au moment même de traiter un nouveau document à traduire. Rien n'empêche toutefois d'effectuer un prétraitement partiel, par exemple en effectuant un alignement entre les phrases des paires de documents $\langle S, T \rangle$ de la MT au moment de la construction de celle-ci, à l'aide de l'une des nombreuses méthodes d'alignement de phrases proposées dans la littérature (voir [110] pour une revue de ces méthodes). Cette façon de faire réduit l'espace de recherche pour le repérage des traductions à des paires de segments dont l'ordre de grandeur est essentiellement celui d'une phrase. En somme, le processus décrit ici n'est en rien incompatible avec les MT existantes, constituées de paires de phrases appariées. Il serait d'ailleurs raisonnable de limiter la recherche des séquences de tronçons (l'étape 2) aux seules séquences qui sont entièrement comprises à l'intérieur d'une seule et même phrase.

Par ailleurs, il est intéressant de noter que le processus décrit ici n'est pas nécessairement dépendant d'une segmentation en phrases du texte à traduire et de la MT. Ceci signifie qu'il pourrait accommoder des méthodes d'alignement de traduction telles que celles proposées par Church [25] ou Simard et Plamondon [105]; ce genre de méthodes, plutôt que d'apparier deux à deux des segments de texte délimités (par exemple des phrases), établissent plutôt des correspondances ponctuelles entre les deux textes, sous la forme d'un *mappage*. Dagan et Church [27] utilisent d'ailleurs ce genre de mappage dans une application de concordancier bilingue s'apparentant au système TransSearch.

Dans les diverses expériences rapportées dans ces pages, nous avons utilisé comme mémoires de traduction les bases de données du système TransSearch (*Hansard* et *Juridique*), dans lesquelles les paires de documents sont toutes alignées au niveau des phrases. Ces alignements ont été effectués de façon automatique au moyen du programme *SFIAL*, une implantation d'une version quelque peu modifiée de la méthode proposée par Simard, Foster et Isabelle [102]. Étant donnée une paire de textes

segmentés en phrases, ce programme effectue une *segmentation parallèle* des deux textes, telle que le $n^{\text{ième}}$ segment d'un texte corresponde au $n^{\text{ième}}$ de l'autre texte. Cette segmentation repose sur une modélisation statistique rudimentaire des traductions, qui tient essentiellement compte des longueurs relatives des segments mis en correspondance, de même que du nombre de *mots apparentés* (en anglais : *cognates*) qu'ils contiennent (les mots apparentés sont des paires de mots de langues différentes qui, de par une origine commune, affiche des graphies ressemblantes, par exemple *soupe*/*soup*, *raison*/*reason*, *simple*/*simple*, etc.).

La figure 2.10 illustre par un exemple le genre d'alignement de phrases que produit ce programme (le symbole ¶ désigne une frontière entre deux phrases).

2.5.4 Sélection des propositions

Pour chaque sous-séquence d du document D, l'étape de repérage des traductions produit un ensemble de traductions possibles t_d, que nous appelons l'*ensemble des candidats* C_d pour la séquence d. Alors que certains de ces ensembles seront vides, d'autres pourront contenir plusieurs traductions, chacune pouvant éventuellement être proposée au traducteur pour réutilisation. La figure 2.11 montre les ensembles de candidats obtenus pour les séquences de la figure 2.6. En pratique, il arrivera souvent que ces ensembles soient trop volumineux pour qu'il soit opportun des les présenter intégralement à l'utilisateur.

Par ailleurs, comme on peut le voir dans les exemples précédents, les étapes de recherche des séquences et de repérage des traductions peuvent fort bien produire des ensembles de candidats C_d non-vides pour des segments du document D qui se chevauchent. Tout porte à croire qu'il serait contre-productif de suggérer au traducteur des propositions de traduction pour toute ces séquences. Il faudra donc effectuer une sélection, tant parmi les séquences de tronçons trouvées dans la MT que parmi les traductions repérées pour ces séquences. La figure 2.12 illustre une sélection possible parmi les résultats de la figure 2.11. Cette question est par ailleurs l'objet du

La vraie question posée par cette controverse est la suivante : ¶ qu'est ce que la pensée? ¶	Behind this debate lies the question, What does it mean to think? ¶
Elle mystifie l'humanité (seule, apparemment, à pouvoir penser) depuis des millénaires. ¶	The issue has intrigued people (the only entities known to think) for millennia. ¶
Des ordinateurs qui ne pensent pas ont cependant réorienté la question et éliminé diverses réponses. ¶	Computers that so far do not think have given the question a new slant and struck down many candidate answers. ¶
La vraie réponse reste cependant inconnue. ¶	A definitive one remains to be found. ¶
L'esprit est-il un programme d'ordinateur? ¶	Is the brain's Mind a Computer Program? ¶
JOHN SEARLE ¶ Non : ¶ un programme manipule seulement des symboles, mais le cerveau leur donne un sens. ¶	No. ¶ A program merely manipulates symbols, whereas a brain attaches meaning to them. ¶ by John R. Searle ¶

Figure 2.10. Un alignement de phrases, tel que produit par le programme *SFIAL*

chapitre 4.

2.6 Expériences

On peut se demander comment un SMT opérant sur des séquences de tronçons syntaxiques, tel que suggéré ici, se comparerait aux systèmes traditionnels, basés sur des phrases. La question qui nous intéresse particulièrement ici concerne la *couverture* du texte-source, c'est-à-dire la proportion d'un nouveau texte à traduire pour laquelle on peut espérer proposer des traductions avec un tel SMT.

Pour répondre à cette question, nous avons effectué quelques expériences sur des textes réels, issus du Journal des débats de la Chambre des communes (*Hansard*). Nous avons d'abord constitué une mémoire de traduction à partir de tous les documents publiés dans ce corpus, durant la période allant d'avril 1986 à janvier 2002. Pour ce faire, nous avons utilisé directement les documents préparés et alignés pour le système TransSearch (section 2.3.1), dont les couples ont alors été versés dans une base de données textuelle, tel que suggéré à la section 2.5.2.

Nous avons par ailleurs retenu une publication du Hansard distincte du contenu de la MT, soit un document paru en mars 2002, que nous appelons le *document-test*. La version anglaise de ce document a d'abord été segmentée en phrases, puis tronçonnée avec le programme *Texas* (section 2.5.1). Le tableau 2.8 résume les principales caractéristiques de ce document.

phrases :	1683
mots :	32 771
tronçons :	21 124

Table 2.8. Statistiques sur le document-test

Nous avons recherché dans la MT toutes les séquences de tronçons de ce document qui contenaient 2 mots ou plus (section 2.5.2). Nous avons alors mesuré le nombre

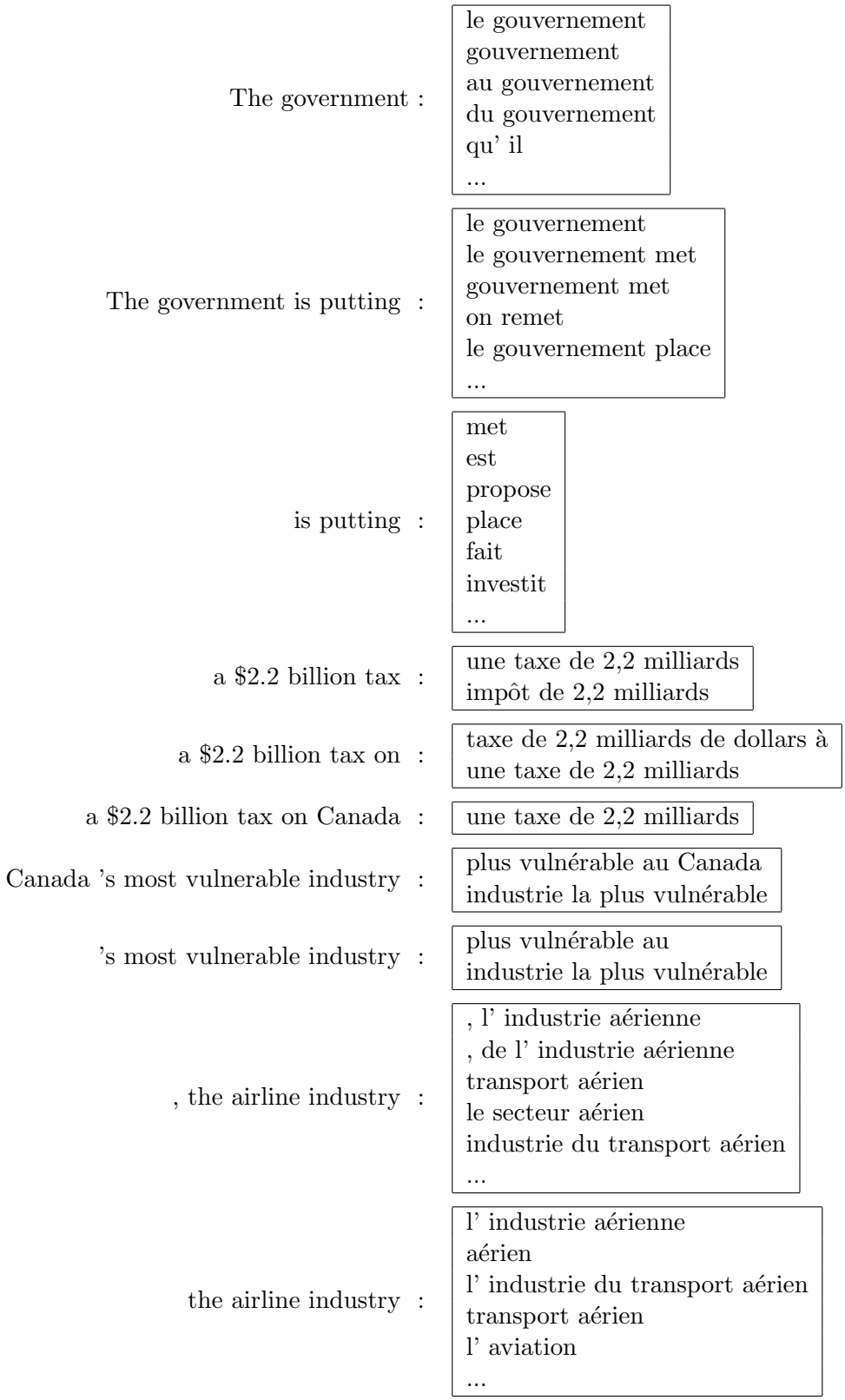

Figure 2.11. Ensembles de candidats pour les séquences de la figure 2.6

The government is putting	→	le gouvernement met
a $2.2 billion tax on	→	taxe de 2,2 milliards de dollars à
Canada 's most vulnerable industry	→	industrie la plus vulnérable
, the airline industry	→	, l' industrie aérienne

Figure 2.12. Un exemple de sélection effectuée parmi les séquences et les ensembles de candidats de la figure 2.11

de mots du document-test qui pouvait ainsi être *couverts* avec des séquences de mots provenant de la MT. Pour fins de comparaison, nous avons également recherché les occurrences de phrases complètes du document-test. Nous nous sommes limité pour cette étude à des recherches verbatim (pas de *repérages flous*). Les résultats de cette expérience sont rapportés au tableau 2.9.

séquences de tronçons		
extraites	29 982	(109 092 mots)
distinctes	23 793	
taille moyenne	3,64 mots	
couverture du texte-source	26 500 mots	80,86%
phrases complètes		
extraites	52	(156 mots)
distinctes	45	
taille moyenne	3,00 mots	
couverture du texte-source	156 mots	0,48%

Table 2.9. Statistiques sur les séquences extraites pour le document-test (langue-source : anglais)

Comme on peut le constater, le document-test choisi serait un assez mauvais

candidat pour un SMT phrase-à-phrase: seules 52 des 1683 phrases du document se retrouvent intégralement dans la MT, pour un total de 156 mots, soit moins de 0,5% de l'ensemble du texte. À l'examen, on constate que ces phrases sont toutes de courts titres (3 mots en moyenne), introduisant le plus souvent le sujet d'une partie du débat. En supposant que la traduction extraite de la MT pour chacune de ces phrases soit directement réutilisable (ce qui n'est pas nécessairement le cas), la récupération de ces phrases permettrait au mieux de produire 193 des 37 546 mots du texte-cible, soit 0,51%.

En revanche, avec des séquences de tronçons, on arrive aisément à couvrir environ 80% du texte-source. On remarque que le nombre total de mots dans les séquences extraites (109 092) excède de beaucoup le nombre total de mots dans le texte-source lui-même (32 771). Ce phénomène s'explique par le simple fait que les séquences trouvées dans la MT se chevauchent fortement. En pratique, chaque phrase du texte-source donne lieu en moyenne à 17,82 séquences distinctes. Chacune de ces séquences se retrouve elle-même dans 56,29 couples de la MT en moyenne, pour un total de plus de 1000 couples de la MT par phrase. À lui seul, ce nombre explique la nécessité d'effectuer une sélection parmi les propositions, tel que suggéré à la section 2.5.4.

On retrouve parfois des séquences de tronçons relativement longues, allant dans certains cas jusqu'à 25 mots. Il s'agit habituellement de longs segments issus de phrases de nature "protocolaire", dans lesquelles seuls quelques détails changent (par exemple: l'heure ou la date). On peut penser que de telles phrases auraient pu être récupérées par les mécanismes de repérage flous des SMT commerciaux décrits à la section 2.2.2. Toutefois, les séquences trouvées sont en majorité relativement courtes (3,64 mots en moyenne). La figure 2.13 montre la distributions des séquences trouvées en fonction de leur longueur, mesurée en nombre de mots.

Évidemment, notre évaluation de la couverture du document-test avec des phrases complètes de la MT n'est pas directement comparable avec celle qu'on pourrait attendre avec les SMT commerciaux, puisque nous n'avons pas tenu compte des repérages

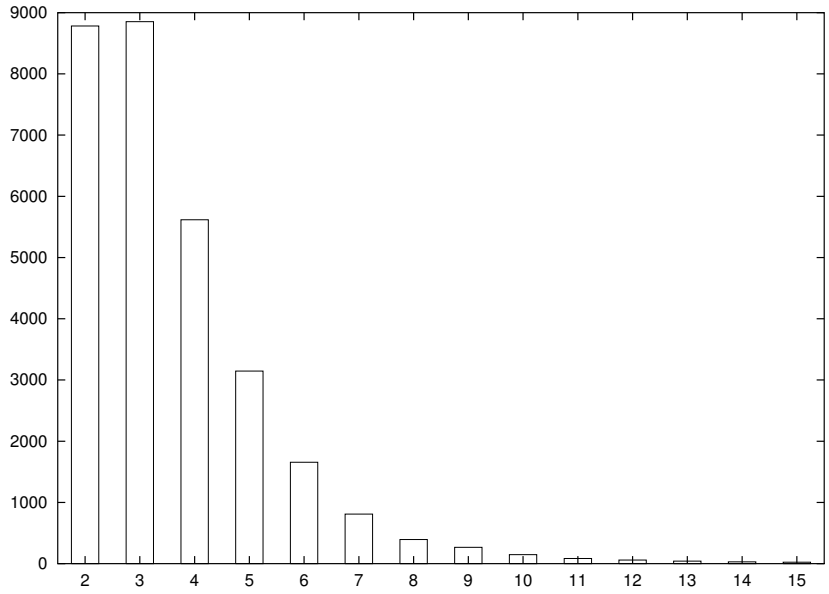

Figure 2.13. Distribution des séquences extraites en fonction de leur longueur (en mots)

flous. De la même façon, tous les couples extraits de la MT pour des séquences de tronçons du document-test ne donneront pas nécessairement lieu à des propositions utilisables (nous verrons plus en détail de quoi il en retourne au chapitre 4). Les résultats de ces quelques expériences laissent néanmoins présager de l'ampleur des gains qu'on peut espérer en passant de la phrase à la séquence de tronçons comme unité de traduction.

2.7 Conclusions

Dans ce chapitre, nous nous sommes intéressé à un type de système de mémoire de traduction capable d'opérer sous le niveau de la phrase, c'est-à-dire de suggérer à un traducteur des traductions pour des segments de texte plus petits qu'une phrase. Un examen des requêtes soumises au système TransSearch, une mémoire de traduction interrogeable en-ligne, nous a permis de conclure que les séquences de tronçons

syntaxiques constituaient un niveau de résolution approprié pour ce genre de système.

Nous avons donc mené quelques expériences, visant à déterminer le niveau de couverture d'un nouveau texte à traduire qu'on pouvait espérer avec un tel système de mémoire de traduction sous-phrastique. Les résultats de notre étude suggèrent que pour un texte qui, bien qu'apparenté au contenu de la mémoire de traduction, ne contient presqu'aucune phrase en commun avec celle-ci, on arrive à couvrir 80% du texte-source avec des séquences de tronçons.

Nous avons donc proposé un schéma de traitement général pour ce genre de système, procédant essentiellement en quatre étapes : tronçonnage du texte à traduire, recherche des séquences de tronçons dans la mémoire de traduction, repérages des traductions pour les séquences trouvées, et sélection parmi les traductions repérées des propositions finales. Alors que les deux premières de ces étapes représentent des problèmes relativement bien étudiés, les deux suivants soulèvent de nombreuses difficultés. Nous en traitons en détail dans les chapitres suivants.

Chapitre 3

REPÉRAGE DE TRADUCTION

3.1 Introduction

Le terme *repérage de traduction* (RT) se veut une traduction libre du terme anglais *translation spotting*, introduit par Jean Véronis et Philippe Langlais [112]. Ce terme désigne l'action consistant à identifier, dans une paire de textes traduction l'un de l'autre, le sous-ensemble des occurrences de mots du texte en langue-cible qui constituent la traduction d'un sous-ensemble donné des occurrences de mots du texte en langue-source. La figure 3.1 présente quelques exemples de tels RT. Dans ces exemples, les mots en italique constituent la séquence recherchée en langue-source, alors que les mots en gras constituent le résultat du repérage.

Comme on le voit dans ces exemples, le repérage prend pour données un *couple*, c'est-à-dire une paire de segments de texte en langues source et cible, traduction l'un de l'autre, et un *point de mire*, c'est-à-dire un sous-ensemble des mots du segment-source du couple, sur lequel l'attention va être centrée. Le résultat du repérage consiste en fait en deux ensembles de mots, soit un pour la source et un pour la cible : ce sont les mots en caractères gras dans la figure 3.1. Nous désignons ces ensembles de mots *repérage-source* et *repérage-cible*.

Le deuxième exemple dans cette figure illustre comment un point de mire coïncidant avec une séquence de mots contigus dans la source peut donner lieu à des mots qui ne sont pas contigus dans la cible; on parle alors de *repérage-cible non contigu*. Par ailleurs, rien n'empêche que le point de mire soit lui-même constitué de mots non-contigus, comme on le voit dans le troisième exemple. Cette situation de *repérage-source non contigu* peut bien entendu donner lieu à un repérage-cible

	point de mire	couple	
1.	*and a growing gap* :	Is this our model of the future, regional disparity ***and a growing gap*** between rich and poor?	Est ce là le modèle que nous visons, soit la disparité régionale **et un fossé de plus en plus large** entre les riches et les pauvres?
2.	*the government's commitment* :	***The government's commitment*** was laid out in the 1994 white paper.	**Le gouvernement** a exposé **ses engagements** dans le livre blanc de 1994.
3.	*close to* [...] *years* :	I have been fortunate to have been travelling for ***close to*** 40 ***years***.	J'ai eu la chance de voyager pendant **près de** 40 **ans** .
4.	*it would make* :	That is why the government is not doing it. ***It would make sense***.	Voilà donc pourquoi le gouvernement ne fait rien, parce que **ça aurait du sens**.
5.	*to the extent that* :	*To the extent that* the Canadian government could be open, it has been so.	Le gouvernement canadien a été aussi ouvert qu'il le pouvait.

Figure 3.1. Exemples de repérages de traduction

contigu ou non. Cette question de contiguïté revêt une importance particulière dans l'application qui nous intéresse; nous en discutons un peu plus loin.

Le quatrième exemple illustre une situation particulière, quoique relativement fréquente : il est impossible d'identifier les mots qui constituent la traduction du point de mire sans élargir le contexte de celui-ci dans le segment-source; dans ce cas-ci, on ne peut "expliquer" *it would make* sans tenir compte du fait que ces mots apparaissent dans la séquence *it would make sense*, dont la traduction est alors *ça aurait du sens*. On parlera alors d'*élargissement* du point de mire.

Finalement, il arrive que la traduction escamote en quelque sorte certains mots, de telle façon qu'il est impossible d'identifier des mots dans le segment cible qui rendent compte du point de mire de façon satisfaisante. C'est ce qu'illustre le cinquième exemple. On parle alors de *repérage-cible vide*. Celui-ci résulte généralement d'une reformulation de la phrase au complet (Véronis parle de *traduction divergente* [109]) ou simplement d'une *omission*, qu'elle soit volontaire ou non.

Le RT connaît différentes applications. La plus directe apparaît dans le contexte d'un concordancier bilingue, comme le système *TransSearch* [64], qui permet la consultation en ligne d'une base de données de traductions, alignées au niveau des phrases (voir la section 2.3.1). Un utilisateur soumet un mot ou une expression au système, qui affiche alors toutes les phrases de la base de données dans lesquelles apparait cette expression avec, en regard, la traduction de chaque phrase. L'utilisateur doit alors localiser visuellement le ou les segments qui constituent la traduction de l'expression cherchée. Il est clair qu'un tel système, s'il était doté d'un mécanisme de repérage automatique, épargnerait cette tâche fastidieuse à l'utilisateur.

Une autre application est la traduction automatique basée sur les exemples. Un système qui construit la traduction d'un texte en combinant des segments en langue-cible extraits d'une base d'exemples de traduction doit forcément, à un moment ou à un autre, mettre en relation les segments source et cible de sa base d'exemples. Sans entrer dans les détails, on voit assez bien comment cette opération pourrait prendre

la forme d'un RT (voir, par exemple, [18]).

Toutefois, dans la suite de ce chapitre, nous nous intéressons tout particulièrement au problème du RT dans le cadre d'une application de mémoire de traduction sous-phrastique, c'est-à-dire un système d'aide à la traduction qui, étant donnée une nouvelle phrase à traduire, recherche dans une base de données constituée de paires de segments traduction l'un de l'autre (la *mémoire de traduction*) des portions de la nouvelle phrase, dans le but de proposer au traducteur des éléments susceptibles de lui venir en aide dans la production de sa traduction. Dans le cadre qui nous intéresse, et tel que proposé au chapitre 2, les portions de la phrase-source qui nous intéressent sont des séquences de mots contigus dans la phrase-source, et les propositions faites par le système prendront également la forme de séquences de mots en langue-cible.

Nous supposons donc :

- que notre système a analysé la phrase P à traduire,
- qu'il a identifié au sein de celle-ci un ensemble $\Phi_P = \{p | p = p_i ... p_j, 1 \leq i \leq j \leq |P|\}$ de sous-séquences d'intérêt,
- qu'il a recherché chacune de ces sous-séquences p dans une mémoire de traduction $\mathcal{M}$,
- qu'il a extrait pour chacune un ensemble C_p (possiblement vide) de couples $\langle S, T \rangle$, dans lequel S est un segment de texte en langue-source, T est un segment en langue-cible, S et T sont traduction l'un de l'autre, et p apparaît dans le segment S.

Notre objectif à ce stade est d'identifier le sous-ensemble t_p des mots de T qui constitue la traduction de p dans le couple $\langle S, T \rangle$. Du point de vue du RT, on considérera donc p comme le point de mire, qui sera toujours contigu; par ailleurs, comme on l'a

mentionné plus haut, le RT identifiera en fait une paire de sous-ensembles $\langle s_p, t_p \rangle$, les mots qui constituent la séquence p se retrouvant alors dans s_p.

Il existe une parenté évidente entre le RT et le problème d'alignement de traduction au niveau des mots, ou ce que nous appelons d'une façon plus générale l'*analyse de traduction*, sujet qui a suscité beaucoup d'intérêt au fil des dernières années. En fait, on peut voir le RT comme un sous-produit de l'analyse de traduction. Dans la suite de ce chapitre, nous commençons donc à la section 3.2 par discuter de cette notion, et passons en revue dans la section 3.3 les principales approches proposées.

Il y a toutefois certains avantages à considérer le RT comme un problème à part entière, pour lequel il est approprié de proposer des méthodes de résolution spécifiques. Nous discutons de ces motivations à la section 3.4, et proposons différentes méthodes à la section 3.5. Nous décrivons ensuite notre implantation informatique de ces méthodes à la section 3.6. Nous présentons enfin une évaluation de leur performance respective à la section 3.7.

3.2 Analyse de traduction

Nous désignons par *analyse de traduction* une description explicite des liens qui unissent un texte et sa traduction[1]. Cette définition est délibérément très vague, et couvre ainsi un large éventail de méthodes visant à retracer les liens implicites entre deux textes qui sont traduction l'un de l'autre. Le concept d'analyse de traduction recouvre en outre les notions d'*alignement*, d'*appariement*, de *correspondance*, etc. qu'on retrouve abondamment dans la littérature.

Depuis près de vingt ans, ces analyses ont trouvé de multiples applications, tant dans le domaine de la traduction automatique (traduction automatique statistique, par analogie) que dans ceux de la traduction assistée par ordinateur (mémoires de traduction phrastiques, vérification automatique de traduction, dictée pour traduc-

[1] Alternativement, ce terme peut désigner l'action visant à produire cette description.

teurs), de la terminologie et de la lexicographie.

Le terme *analyse de traduction* a d'abord été introduit par Pierre Isabelle. On reprend ici les grandes lignes de l'argumentation de Isabelle *et al.* [48].

D'une façon formelle, il est assez courant de voir la traduction comme une relation $T_{L_1L_2}$ entre les textes d'une langue L_1 et ceux d'une langue L_2. On admet généralement que $T_{L_1L_2}$ est définie récursivement, c'est-à-dire que deux textes S et T appartiendront à la relation $T_{L_1L_2}$ si et seulement si il existe une certaine décomposition de S et T en *éléments primitifs* $S = s_1, ..., s_M$ et $T = t_1, ..., t_M$, telle que les paires $(s_1, t_1)...(s_M, t_M)$ sont également dans la relation $T_{L_1L_2}$:

$$T_{L_1L_2}(S,T) \iff T_{L_1L_2}(s_1,t_1)...T_{L_1L_2}(s_M,t_M)$$

Cette façon de voir confère un caractère *compositionnel* à la relation de traduction.

Dans cette perspective, tout système de traduction automatique peut être vu comme une procédure qui, étant donné un certain texte S en langue L_1, tente de produire un ou plusieurs textes T en langue L_2, tels que $T_{L_1L_2}(S,T)$. Dans la plupart des systèmes décrits dans la littérature, cette procédure dépend pour son fonctionnement d'une certaine formalisation des relations entre éléments primitifs (les $T_{L_1L_2}(s_i,t_j)$), de même que des règles qui régissent la composition de celles-ci.

Les systèmes de TA abordent donc la relation de traduction d'un point de vue *génératif*. Toutefois, rien n'empêche de considérer cette même relation d'un point de vue *reconnaissance*, tel que proposé par Fathi Debili [30] : la question n'est alors plus de produire un texte-cible pour un texte-source donné, mais plutôt de déterminer si une paire de textes donnée appartient ou non à cette relation. Un programme qui effectue cette tâche serait donc un *reconnaisseur* de traduction. Si de plus un tel programme est en mesure, lorsqu'il reconnaît une paire de textes comme traduction, d'expliciter récursivement les relations qui unissent les sous-parties de ces textes jusqu'au niveau de leurs éléments primitifs, alors on peut parler d'un *analyseur de traduction*, et la description qu'il produit prend la forme d'un *arbre d'analyse de*

traduction (AAT).

On peut tracer un parallèle entre ce concept d'analyse de traduction et celui, mieux connu, d'analyse syntaxique : dans ce dernier cas, l'objectif est d'expliciter la structure sous-jacente d'une unité linguistique quelconque (par exemple, d'une phrase), en regard d'un certain modèle linguistique. De façon analogue, une analyse de traduction représentera "ce qui se passe entre" un texte-source et sa traduction. Par exemple, la figure 3.2 représente un AAT pour une paire de phrases, en regard d'un modèle de la traduction quelconque (source : Isabelle *et al.* 1993 [48]).

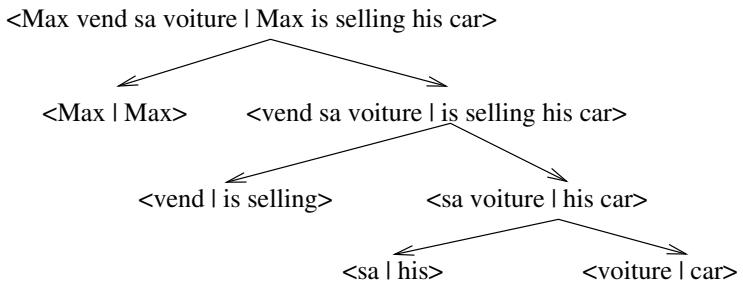

Figure 3.2. Une analyse de traduction.

En réalité, la très grande majorité des approches qui ont été jusqu'à maintenant proposées produisent des analyses plates, c'est-à-dire dans lesquelles tous les éléments mis en correspondance sont au même niveau. Par exemple, la figure 3.3 présente une analyse de ce genre pour la même paire de phrases qu'à la figure 3.2. On présente ces différentes approches à la section 3.3.

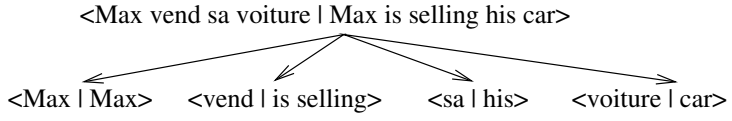

Figure 3.3. Une analyse de traduction *plate*, au niveau des mots.

3.3 Méthodes d'analyse de traduction

Nous passons ici en revue les principales méthodes existantes d'analyse de traduction. Nous distinguons en fait les analyses qui établissent des alignements directs entre les mots (section 3.3.1), celles qui mettent plutôt en correspondance des séquences de mots (section 3.3.2) et celles qui considèrent les relations entre des éléments structurels des textes (section 3.3.3).

3.3.1 Alignements au niveau des mots

L'apparition dans la littérature scientifique des méthodes automatiques d'analyse de traduction coïncide à peu près avec les premiers travaux portant sur la traduction automatique statistique. Aux alentours de 1990, une importante équipe de recherche chez IBM s'est penchée sur une approche purement statistique à la traduction automatique [15] : on cherchait ici à démontrer la faisabilité d'un système de TA dans lequel toute connaissance linguistique était acquise de façon entièrement automatique à partir de grandes quantités de données brutes, en l'occurrence de corpus de textes bilingues. Cette connaissance était stockée dans un *modèle de traduction* purement statistique. L'équipe d'IBM a proposé en fait une série de modèles de complexité croissante [17], s'inspirant directement des modèles de *canal bruité* (*noisy channel*) proposé par Shannon [101] et employé avec succès en reconnaissance de la parole.

Par exemple, dans l'élaboration d'un système de TA du français vers l'anglais, on suppose que toute phrase française S est en fait une phrase anglaise T "bruitée". La traduction est alors vue comme le processus consistant à retrouver la phrase anglaise T à l'origine de S. On considère à cet égard que toutes les phrases anglaises sont des traductions possibles de S, mais que certaines sont plus susceptibles que d'autres d'être produites par un traducteur; on peut ainsi associer à toute paire de phrases $\langle S, T\rangle$ une probabilité $\Pr(T|S)$ qu'un traducteur produise la phrase T lorsqu'on lui présente la phrase S. La stratégie en TAS est de rechercher, pour une phrases-source

S donnée, la phrase-cible $\hat{T}$ qui maximise $\Pr(T|S)$. Suivant le théorème de Bayes, on écrit :

$$\Pr(T|S) = \frac{\Pr(T)\Pr(S|T)}{\Pr(S)}$$

et comme le dénominateur ne dépend pas de T, la recherche de $\hat{T}$ revient à trouver la phrase T qui maximise le produit $\Pr(T)\Pr(S|T)$. Dans cette équation, la distribution $\Pr(T)$ joue ainsi le rôle d'un *modèle de langue* de l'anglais (l'équivalent d'une grammaire de la langue-cible dans un système de TA traditionnel), alors que $\Pr(S|T)$ est le *modèle de traduction* (l'équivalent d'une composante de transfert).

Dans l'approche d'IBM, les paramètres numériques qui constituent ces deux distributions sont acquis de façon entièrement automatique à partir de grandes masses de textes bilingues. Ici, on s'attaque de front à la question de l'analyse de traduction, puisque la méthode proposée suppose que les phrases du corpus d'apprentissage sont *alignées*, c'est-à-dire que les phrases qui sont des traductions l'une de l'autre sont explicitement mises en correspondance. L'équipe d'IBM a donc été parmi les premières à proposer une méthode d'analyse de traduction, sous la forme d'un système d'*alignement de phrases*[14].

Dans l'élaboration de leurs modèles statistiques de traduction, Brown et al. introduisent une formalisation de la notion d'alignement de mots dans laquelle chaque mot s_i du texte-source $S = s_1...s_M$ est connecté à un mot t_j du texte-cible $T = t_1...t_N$. On ajoute conventionnellement à T un *mot-nul* t_0, auquel il est toujours possible de connecter les mots de S qui ne "génèrent" pas de mots dans T. On exclut par ailleurs d'emblée la possibilité qu'un même mot-source soit connecté à plus d'un mot-cible (mais pas l'inverse). La figure 3.4 présente un exemple de ce genre d'alignement (la langue-source est le français). Cette définition permet de décrire l'alignement entre les textes t_1^N et s_1^M comme une séquence a_1^M, telle que $a_i = j$ si le mot s_i est connecté à t_j, ou à aucun mot si $j = 0$.

Les modèles de traduction de IBM visent à estimer $\Pr(S|T)$, la probabilité d'une

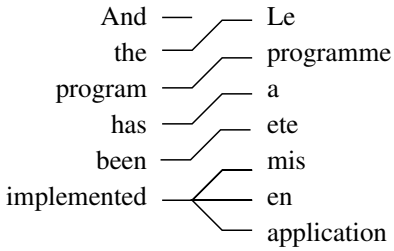

Figure 3.4. Un alignement de mots, à la façon IBM

phrase-source S étant donnée sa traduction T. Avec cette notation pour les alignements de mots, on peut réécrire $\Pr(S|T)$ comme :

$$\Pr(S|T) = \sum_{a \in \mathcal{A}(T,S)} \Pr(S, a|T), \tag{3.1}$$

où $\mathcal{A}(T, S)$ est l'ensemble des alignements de mots possibles entre T et S, tels que définis ci-dessus. Sans perdre de généralité, le terme $\Pr(S, a|T)$ peut lui-même se réécrire

$$\Pr(S, a|T) = \Pr(M|T) \prod_{i=1}^{M} \Pr(a_i|a_1^{i-1}, s_1^{i-1}, M, T) \Pr(s_i|a_1^i, s_1^{i-1}, M, T), \tag{3.2}$$

où M est la longueur de la phrase-source S, a_i est l'alignement du mot s_i dans la phrase T, s_1^i désigne la séquence de mots $s_1 ... s_i$ et a_1^i la séquence d'alignements correspondants dans la phrase T. Notons qu'il ne s'agit pas d'une approximation, mais bien d'une expression exacte de $\Pr(S, a|T)$, simplement une des nombreuses décompositions possibles de cette quantité en probabilités conditionnelles.

Brown *et al.* construisent une succession de modèles statistiques (les modèles 1 à 5), chacun servant de base à l'élaboration du suivant. Les modèles 1 et 2 sont basés sur la décomposition de $\Pr(S, a|T)$ ci-dessus, dans laquelle on introduit certaines simplifications. Dans le modèle 1, on suppose que

- $\Pr(M|T)$ (la probabilité que la phrase-source S à l'origine de la traduction T comporte M mots) est un terme constant;

- pour une longueur de phrase-cible donnée N, les $\Pr(a_i|a_1^{i-1}, s_1^{i-1}, M, T)$ sont uniformément distribuées;

- $\Pr(s_i|a_1^i, s_1^{i-1}, M, T)$ ne dépend en fait que de s_i et t_{a_i}, c'est-à-dire que s_i ne dépend que du mot auquel il est connecté dans l'alignement a.

Ainsi, ce modèle extrêmement simplifié repose uniquement sur un ensemble de paramètres $t(s_i|t_{a_i})$ qui sont une approximation des $\Pr(s_i|a_1^i, s_1^{i-1}, M, T)$.

Dans le modèle 2, on lève l'hypothèse d'uniformité des alignements : on suppose que $\Pr(a_i|a_1^{i-1}, s_1^{i-1}, M, T)$ ne dépend que de a_i, de i, de M et de N. On introduit donc un deuxième ensemble de paramètres $a(j|i, M, N)$, qui doit estimer la probabilité que le mot en position i dans une phrase de longueur M se voit traduit par un mot en position j dans la cible, lorsque celle-ci a longueur N.

De façon similaire, on lève graduellement certaines de ces simplifications dans les modèles subséquents, de façon à se rapprocher de l'expression exacte de $\Pr(S, a|T)$. Nous n'approfondissons pas davantage la description de ces modèles ici. Notons toutefois que les modèles 3, 4 et 5 sont basés sur une décomposition différente de celle présentée dans l'équation 3.2 ci-dessus.

Dans tous les modèles d'IBM, les valeurs individuelles des paramètres sont apprises automatiquement à partir d'une base d'exemples, constituée de paires de phrases qui sont traduction l'une de l'autre. On cherche alors l'ensemble des paramètres (par exemple, les $t(s_i|t_j)$ pour le modèle 1) qui maximise la vraisemblance de la base d'exemples. On utilise pour ce faire l'algorithme *EM* [8] qui, partant d'un ensemble de paramètres quelconque, recherche itérativement les paramètres optimaux. En général, l'algorithme EM garantit de trouver un maximum, mais ce dernier peut être local. La qualité du modèle obtenu dépendra donc théoriquement du choix des paramètres de départ. Toutefois, il se trouve que les propriétés mathématiques du modèle 1 permettent de garantir la convergence de EM vers un maximum global. L'idée de Brown *et al.* est donc d'utiliser les paramètres optimaux du modèle 1

comme paramètres de départ pour l'estimation des paramètres du modèle 2, et ainsi de suite jusqu'au modèle 5.

Dans les paires de phrases de la base d'exemples, les alignements de mots corrects ne sont pas connus. Ceci n'est théoriquement pas un problème pour l'apprentissage, parce que la recherche des paramètres optimaux suppose en fait qu'on examine tous les alignements possibles pour chaque paire de phrases de la base d'exemples. En pratique, toutefois, le nombre d'alignements possibles s'élève à 2^{MN} pour une paire de phrases de longueurs N et M respectivement, de telle sorte qu'un examen exhaustif de tous les alignements est hors de question. Les propriétés mathématiques des modèles 1 et 2 permettent de contourner cette difficulté, et d'obtenir un calcul exact de la vraisemblance en temps polynomial. Malheureusement, ces propriétés ne s'étendent pas aux modèles 3, 4 et 5. Pour l'estimation des paramètres de ces modèles, Brown *et al.* ont donc recours à une heuristique, qui consiste à estimer la vraisemblance d'une paire de phrases à partir d'un échantillon des alignements de mots possibles, choisi parmi les alignements les plus probables.

On va donc introduire la notion d'*alignement Viterbi*, c'est-à-dire l'alignement a entre les mots de T et S pour lequel $\Pr(a|S,T)$ est maximal. Quoiqu'il n'existe pas de méthode générale et efficace pour calculer l'alignement Viterbi, les propriétés mathématiques des modèles 1 et 2 de IBM vont encore une fois s'avérer utiles, puisqu'il se trouve que pour ces modèles, $\Pr(a|S,T)$ s'exprime ainsi :

$$\Pr(a|S,T) = \prod_{i=1}^{M} \Pr(a_i|i,M,N) \tag{3.3}$$

dans lequel

$$\Pr(j|i,M,N) = \frac{\gamma(j,i,M,N)}{\sum_{J=0}^{N} \gamma(J,i,M,N)}, \tag{3.4}$$

et

$$\gamma(j,i,M,N) = t(s_i|t_j)a(j,i,M,N)$$

(Pour le modèle 1, les probabilités d'alignement sont distribuées uniformément,

de telle sorte que $a(j, i, M, N) = (M + 1)^{-1}$.) L'alignement Viterbi se calcule alors simplement en choisissant pour chaque position i de S l'alignement a_i qui maximise $\gamma(a_i, i, M, N)$.

Pour les modèles 3, 4 et 5, toutefois, on est réduit à examiner tous les alignements possibles si on veut trouver celui dont la probabilité est maximale. En pratique, on contourne cette difficulté par une heuristique : on part de l'alignement Viterbi du modèle 2, dans lequel on introduit de petites modifications (ajout ou suppression d'un lien, inversion de deux liens) et on réévalue la probabilité de l'alignement en regard des paramètres du modèle considéré (3, 4 ou 5). On examine ainsi toutes les variantes obtenues en effectuant une modification à l'alignement Viterbi-2, et on conserve celle qui produit la plus grande amélioration de $\Pr(a|T, S)$. On répète ce processus jusqu'à convergence, c'est-à-dire jusqu'à ce qu'aucune modification n'améliore l'alignement.

Cette approche produit des résultats intéressants, mais limités par la définition même des alignements de mots considérée. Malgré tout, on retrouve dans la littérature différentes variantes de ce genre d'approche. Par exemple, Gale et Church [36] proposent une méthode d'alignement de mots basée sur un modèle de traduction extrêmement simplifié, n'impliquant en fin de compte que les positions relatives des mots considérés. Pour qu'une telle approche soit envisageable, il faut évidemment restreindre d'emblée les correspondances possibles : pour ce faire, Gale et Church (tout comme Brown *et al.*) prennent comme point de départ un alignement de phrases, et utilisent une table de correspondances lexicales permises.

Cette table est elle-même produite automatiquement à partir des textes à apparier et de leur alignement de phrases de départ : on calcule pour chaque paire de mots (t, s) une mesure d'association ϕ^2 à partir de la *table de contingence* de t et s :

	t	$\neg t$
s	a	b
$\neg s$	c	d

Dans cette table, a est le nombre de paires de phrases appariées dans lesquelles les

mots t et s apparaissent tous deux, b est le nombre de paires de phrases où s apparaît mais non t, etc. La mesure d'association ϕ^2 se calcule alors comme suit :

$$\phi^2 = \frac{(ad - bc)^2}{(a + b)(a + c)(b + d)(c + d)}$$

On prétend que cette mesure se comporte de façon plus robuste que les paramètres $t(s_i|t_j)$ estimés par EM de Brown *et al.*, surtout lorsque la quantité de données est limitée.

Gale et Church soulèvent par ailleurs l'existence d'un problème computationnel : l'application de l'algorithme EM pour l'apprentissage d'un modèle de traduction nécessite la gestion d'une table de taille $V_T \times V_S$, où V_T est la taille du vocabulaire cible et V_S la taille du vocabulaire source. Même en limitant ces vocabulaires, la taille de cette table peut devenir problématique. Pour contourner ce problème, Gale et Church procèdent itérativement, sur des sous-ensembles du corpus d'apprentissage de taille croissante. À chaque itération, on ne considère que les paires rencontrées, et on ne préserve que les paires dont le ϕ^2 excède un certain seuil. À l'itération suivante, on ignore toutes les paires déjà présentes dans la table, et ainsi de suite[2].

Le résultat obtenu est un alignement de type un-à-un, couvrant environs 60% des mots du texte, avec une précision de 95%. Dans la même veine, Melamed [74] propose quant à lui une approche très similaire à celle de Gale et Church, se basant plutôt sur une statistique G^2, proposée par Dunning [32] :

$$G^2 = -2 \log \frac{B(a|a + b, p_1)B(c|c + d, p_2)}{B(a|a + b, p)B(c|c + d, p)}$$

où $B(k|n, p) = \begin{pmatrix} n \\ k \end{pmatrix} p^k(1-p)^{n-k}$ est la probabilité d'une binômiale, et $p_1 = a/(a+b)$, $p_2 = c/(c + d)$ et $p = (a + b)/(a + b + c + d)$. Melamed prétend obtenir des résultats légèrement supérieurs avec ce G^2 qu'avec le ϕ^2 de Gale et Church.

[2] Brown *et al.* contournent quant à eux le problème au moyen de techniques de *matrice creuse* (*sparse matrix*); notons par ailleurs que, si ce problème était criant à l'époque où ces résultats ont été publiés, il est clair qu'il l'est beaucoup moins maintenant.

Dagan *et al.* [28] proposent également une méthode d'alignement de mots inspirée du modèle IBM-2. Pour pallier certains problèmes de robustesse, plutôt que de partir d'un alignement de phrases, Dagan *et al.* utilisent quant à eux la sortie du système `char_align` de Church [25], lequel produit un mappage approximatif entre deux textes : pour chaque mot du texte-source, `char_align` prédit une position correspondante approximative dans le texte-cible.

Pour adapter le modèle IBM-2 à ce nouveau contexte, certaines reformulations sont nécessaires. Notamment, les paramètres $t(s_i|t_j)$ représentent plutôt la probabilité que s_i se retrouve dans une fenêtre de taille fixe à l'entour du point de mappage prédit par `char_align` pour t_j, alors que les $a(a_i|i, M, N)$ sont remplacés par des paramètres plus simples $o(k)$, représentant l'écart attendu entre la position de s_i et le point de mappage de t_j. Le système `word_align` de Dagan *et al.* prend donc comme point de départ une paire de textes et le mappage entre ceux-ci produit par `char_align`; il utilise d'abord ce mappage pour estimer les paramètres de son modèle de traduction, et calcule ensuite par programmation dynamique l'alignement de mots dont les correspondances présentent la probabilité conjointe maximale.

Le résultat obtenu est un alignement de mots de bonne précision, mais encore une fois limité par la définition même des alignements de Brown *et al.* De plus, pour éviter différents problèmes de sous-représentation des données (*data sparseness*), attribuables à la procédure d'entraînement (on entraîne individuellement pour chaque paire de textes à apparier), Dagan et al. ignorent d'emblée beaucoup des mots du texte (trop ou pas assez fréquents), de même que les paires de mots dont le $t(s_i|t_j)$ est trop petit. Par conséquent, tout comme dans le cas de Gale et Church, les alignements de mots obtenus par Dagan *et al.* sont partiels.

Finalement, il faut dire que beaucoup des travaux sur la modélisation statistique de la traduction doivent éventuellement avoir des impacts indirects sur l'alignement des mots. On peut souligner à cet égard les efforts déployés dans le cadre du projet VerbMobil [79, 99, 114]. Notamment, Vogel *et al.* [108] proposent de remplacer le

paramètre $a(a_i|i, M, N)$ du modèle IBM-2 par un modèle markovien d'ordre 1, qui capture mieux l'effet de "localisation" qu'on observe dans la traduction : les alignements entre un texte et sa traduction ne sont pas répartis uniformément, mais tendent plutôt à se regrouper. Ceci suggère un modèle où l'alignement d'un mot dans la traduction dépend de l'alignement du mot précédent : $\Pr(a_i|a_{i-1}, N)$[3]. Comme le modèle IBM-2, ce modèle peut être entraîné par l'algorithme EM, à partir du modèle IBM-1. Expérimentalement, le modèle avec HMM affiche une meilleure perplexité que le modèle IBM-2 (c'est-à-dire une plus grande vraisemblance du corpus d'entraînement), mais un examen informel des alignements révèle des lacunes dans le modèle, en particulier en ce qui a trait à certains mouvements de "grande envergure", par exemple quand la traduction d'un petit groupe de mots qui apparaissait au début d'une phrase-source se trouve relocalisée à la fin de la phrase-cible.

3.3.2 Alignements sous-phrastiques

Avant même l'émergence des premières méthodes automatiques d'analyse de traduction, dans un des premiers articles traitant de l'analyse de traduction, Alan Melby souligne les difficultés que soulève l'objectif visant à retracer la traduction de chaque mot [75]. Il établit quant à lui que ses *concordances bilingues* doivent mettre en relation des unités suffisament grandes pour éviter qu'il soit nécessaire de "prendre des décisions difficiles". En pratique, ses analyses (qui sont en fait produites par un opérateur humain) vont réunir des sous-segments de phrases de taille variable, qui correspondent plus ou moins à des propositions ou des groupes de propositions.

C'est également au niveau des propositions que Meunier [77] et Boutsis et Piperidis [12] vont chercher les correspondances. Par exemple, pour mettre en correspondance des propositions dans des paires de textes anglais-grec, ces derniers proposent une méthode en quatre étapes : dans un premier temps, les textes sont appariés au niveau

[3] En fait, le modèle s'intéresse plutôt à la distance entre a_i et a_{i-1}

des phrases avec la méthode de Gale et Church [35]; ensuite, on segmente le texte en mots, et on effectue un étiquetage morpho-syntaxique de ceux-ci (on utilise un modèle de Markov caché pour l'anglais, des règles manuscrites de désambiguïsation lexicale pour le grec); on identifie ensuite des propositions au moyen d'automates à états finis, qui recherchent les plus longues occurrences de schémas typiques; l'étape finale est l'appariement de ces propositions. On examine pour ce faire des combinaisons impliquant 0, 1 ou 2 propositions de chaque langue, dont la pertinence est évaluée en regard d'un modèle statistique de la traduction dérivé du modèle IBM-1. Selon le nombre de propositions impliquées, on utilisera soit une méthode de recherche optimale (par programmation dynamique) ou sous-optimale (par recuit simulé [1]).

Gaussier [37], quant à lui, s'intéresse aux correspondances entre des syntagmes nominaux de petite taille, principalement dans le but de constituer automatiquement des lexiques terminologiques bilingues à partir de corpus alignés; en fait, on suppose que les textes sont déjà alignés au niveau des phrases ou mieux, au niveau des propositions (tel que proposé par Meunier [77]). Partant d'un étiquetage morpho-syntaxique, Gaussier commence par identifier automatiquement des syntagmes nominaux simples au moyen de patrons pré-établis (*nom commun-nom commun*; *adjectif-nom commun*, etc.). Ces syntagmes sont alors considérés au même titre que les mots simples dans l'entraînement d'un modèle de traduction stochastique (différentes variations sur le modèle IBM-1 sont considérées), qu'on peut alors utiliser pour calculer un alignement *Viterbi*, à la façon de Brown *et al.*

L'approche de Gaussier pour trouver l'alignement optimal est particulièrement intéressante, puisqu'il présente cette opération comme un problème de flot dans un graphe [38] : par exemple, pour l'alignement de mots simples, on construit un graphe avec un sommet pour chaque mot-source et chaque mot-cible (incluant deux mots-vides), et deux sommets additionnels (*source* et *cible*); la *source* est reliée par une arête à chaque mot-source, la *cible* à chaque mot-cible, et toutes les paires de mots source-cible sont reliées entre elles. Des contraintes de flot sur les arêtes servent à

s'assurer que le flot passant par un mot-source ira à un et un seul mot-cible. On associe à chaque arête un coût, qui correspond en fait à la probabilité de l'alignement correspondant, tel qu'estimé par le modèle de traduction statistique. On utilise alors un algorithme standard pour trouver le flot réalisable dont le coût est optimal. La figure 3.5 illustre le graphe représentant un tel alignement de mots (dans cette figure, seules les arêtes par lesquelles "passe le flot" sont dessinées – cette figure provient de [38]).

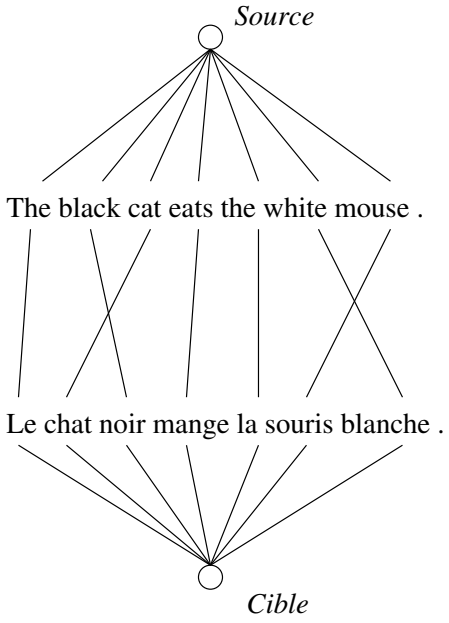

Figure 3.5. L'alignement de mots comme un problème de flot

Pour passer de l'alignement de mots à l'alignement de termes dans ce cadre, on doit relâcher la contrainte de un-pour-un. Pour ce faire, Gaussier introduit de nouveaux sommets dans le graphe, correspondant à des séquences de mots contigus, de longueur variable, et des arêtes reliant ces nouveaux sommets. Cette façon de faire augmente substantiellement la complexité, et soulève différents problèmes d'estimation, mais on réussit ainsi à établir des correspondances plusieurs-à-plusieurs.

3.3.3 Alignements structurels

Certains chercheurs, notamment dans le domaine de la traduction automatique basée sur les exemples, ont mis de l'avant des méthodes nécessitant des analyses structurelles des textes, et des alignements mettant en correspondance ces éléments de structure.

Kaji *et al.* [51], Matsumoto *et al.* [67] et Grishman [41] proposent des méthodes d'analyse de traduction structurelle qui présentent plusieurs similarités entre elles[4]. Ces méthodes utilisent des textes préalablement alignés au niveau des phrases, et effectuent essentiellement un traitement en deux étapes : d'abord, une analyse monolingue, puis une mise en correspondance des structures.

Plus précisément, l'analyse monolingue doit produire pour chaque paire de phrases une paire de structures arborescentes; chez Grishman et Matsumoto *et al.*, il s'agit d'arbres de dépendances de style LFG (*Lexical functional grammars*, [52]), alors que chez Kaji *et al.*, on utilise des grammaires de constituants.

Une fois ces analyses effectuées, on cherche à mettre en correspondance les sous-parties des structures obtenues. On utilise pour ce faire des dictionnaires bilingues; typiquement, on tentera de faire correspondre des structures dont le contenu lexical est aussi similaire que possible en regard de ces dictionnaires (on ignore généralement les mots-outils, pour se concentrer sur les mots pleins). D'autres critères peuvent également entrer en compte : dans Kaji *et al.*, on tente d'apparier des structures de tailles comparables; Matsumoto *et al.* utilisent également un thésaurus (Roget), pour mesurer la "distance" entre des mots qui, sans être traduction l'un de l'autre, sont susceptibles d'exprimer des notions apparentées.

Pour trouver le meilleur alignement de structures, ces différents critères de ressemblance sont quantifiés et combinés de façon *ad hoc* en un score de similitude. On fouille alors l'ensemble des possibilités d'alignement, pour trouver celui qui maxi-

[4] D'ailleurs, toutes trois ont été développées dans le contexte de travaux sur la TA basée sur les exemples entre le japonais et l'anglais.

mise ce score. Matsumoto et al. commencent par aligner les racines des structures, et examinent ensuite récursivement les alignements de sous-structures de manière descendante; on utilise une stratégie de *branch-and-bound* pour limiter l'espace de recherche. Kaji *et al.* et Grishman, quant à eux, commencent par mettre en correspondance des mots, pour ensuite aligner les structures de manière ascendante; pour limiter l'espace de recherche, Grishman effectue une fouille en faisceau (*beam-search*).

On remarque que ces trois approches dépendent de l'existence de ressources linguistiques non négligeables : grammaires monolingues à large couverture, dictionnaires bilingues, thésaurus, etc. Grishman évoque la possibilité de partir d'un dictionnaire minimal pour effectuer un premier alignement, à partir duquel on pourrait enrichir le dictionnaire, pour ensuite réaligner de façon itérative, jusqu'à convergence du processus (on retrouve des idées analogues à différents endroits, notamment dans [36, 54, 71]). Toutefois, cette stratégie ne semble pas avoir été mise à l'essai. Matsumoto *et al.*, quant à eux, suggèrent qu'il pourrait être souhaitable de remplacer le dictionnaire bilingue par un modèle de traduction statistique similaire à ceux proposés par l'équipe IBM.

Un aspect intéressant de tous ces travaux d'analyse de traduction concerne en fait plutôt l'analyse syntaxique : en effet, dans les trois cas présentés ci-dessus, on utilise les correspondances obtenues pour lever les ambiguïtés dans les analyses syntaxiques. Par exemple, dans la méthode présentée par Grishman, les analyseurs syntaxiques produisent toutes les analyses possibles; pendant l'étape d'alignement, on essaie de mettre en correspondance toutes les paires possibles, et on ne conserve à la fin que les analyses qui produisent l'alignement de score maximal. En fait, c'est suivant ce critère que la méthode de Grishman est évaluée : un alignement est considéré comme bon s'il sélectionne la bonne analyse.

Dans tous ces travaux, on a adopté une approche que Dekai Wu qualifie de *parse-parse-match* : on commence par effectuer une analyse syntaxique de chacun des deux textes, pour ensuite examiner la mise en correspondance des structures obtenues de

part et d'autre. Ces approches sont à contraster avec celle mise de l'avant par Wu lui-même, principalement dans [118] : l'auteur propose un formalisme linguistique appelé *Inversion Transduction Grammars* (ITG; "grammaires de transduction avec inversions"), servant à décrire des grammaires bilingues.

Ces ITGs sont essentiellement des *grammaires hors-contexte parallèles*, qui génèrent simultanément des paires de chaînes dans deux langues L_1 et L_2. Les symboles non-terminaux d'une ITG sont comme ceux d'une grammaire hors-contexte normale, alors que les terminaux sont tous de la forme x/y, où $x \in L_1$ et $y \in L_2$. L'un ou l'autre de x ou y peut être ϵ (le mot-nul). De plus, le membre droit des règles de la grammaire est annoté par une *direction* (marquée par des crochets "[ ]" ou des chevrons "⟨ ⟩") : celle-ci détermine si la dérivation de cette règle se fait de façon parallèle ou *inversée*. Ainsi, la règle $X \rightarrow [X_1 X_2 X_3]$ (avec des crochets) dérive parallèlement la chaîne $X_1 X_2 X_3$ dans les deux langues L_1 et L_2, alors que $X \rightarrow \langle X_1 X_2 X_3 \rangle$ (avec des chevrons) dérive simultanément la chaîne $X_1 X_2 X_3$ en langue L_1 et la chaîne $X_3 X_2 X_1$ en langue L_2.

Par exemple, supposons les règles suivantes, décrivant certains syntagmes nominaux simples dans une ITG anglaise-française :

$$
\begin{aligned}
SN &\rightarrow [Det\ NN] \\
NN &\rightarrow [Adj\ N] && (3.5) \\
NN &\rightarrow \langle Adj\ N \rangle && (3.6) \\
Det &\rightarrow \text{the/le} \\
N &\rightarrow \text{dog/chien} \\
Adj &\rightarrow \text{little/petit} \\
Adj &\rightarrow \text{brown/brun}
\end{aligned}
$$

La règle (3.5) permet de dériver des paires de syntagmes où l'ordre des constituants

est le même dans les deux langues :

$$((\text{the})_{Det}((\text{little})_{Adj}(\text{dog})_{N})_{NN})_{SN}$$
$$((\text{le})_{Det}((\text{petit})_{Adj}(\text{chien})_{N})_{NN})_{SN}$$

À l'opposé, les crochets dans la règle (3.6) indiquent une inversion : cette règle produira donc la séquence *Adj N* en anglais et *N Adj* en français, ce qui permet de rendre compte des situations où l'ordre de ces constituants est inversé :

$$((\text{the})_{Det}((\text{brown})_{Adj}(\text{dog})_{N})_{NN})_{SN}$$
$$((\text{le})_{Det}((\text{chien})_{N}(\text{brun})_{Adj})_{NN})_{SN}$$

On remarque qu'une telle grammaire permettrait également de produire des paires illégales, telles que

** the brown dog / le brun chien*;

pour contourner ce genre de situation, il faudrait distinguer en français les deux sortes d'adjectifs, antéposé et postposé, et tenir compte de différents cas d'exceptions. Mais ceci dépasse notre propos.

Wu soutient que ce formalisme peut être utilisé pour décrire des grammaires bilingues, lesquelles peuvent alors être utilisées pour analyser des paires de phrases. De toute évidence, la structure arborescente résultant d'une tele analyse décrit explicitement des équivalences de traduction au niveau structurel. Par exemple, on peut utiliser les crochets "[ ]" et chevrons "⟨ ⟩" pour réécrire la deuxième analyse ci-dessus de façon parallèle :

$$[[\text{the/le}]_{Det}\langle[\text{brown/brun}]_{Adj}[\text{dog/chien}]_{N}\rangle_{NN}]_{SN}$$

Différentes variantes des ITGs sont proposées. Wu décrit notamment une version stochastique (SITG), dans laquelle on associe une probabilité à chaque règle de réécriture. Les paramètres de ce modèle de traduction statistique peuvent être acquis

automatiquement via la procédure EM [116], et un algorithme existe, permettant d'analyser une paire de phrases avec une SITG en temps polynomial ($\mathcal{O}(K^3M^3N^3)$, où K est le nombre de non-terminaux de la grammaire, M et N la longueur des deux phrases à aligner).

D'après Wu, même avec une grammaire minimale, on peut effectuer de nombreuses tâches d'analyse de corpus, dont la segmentation en mots (l'auteur s'intéresse particulièrement à cette question relativement au chinois, dont le système d'écriture ne marque pas explicitement les frontières entre les mots), le parenthésage structurel (*bracketing*), de même que l'alignement au niveau des mots et des syntagmes (*phrasal alignment*). Par exemple, dans [117], l'auteur s'intéresse aux *Bracketing Transduction Grammars* (BTG), une autre variante des ITGs n'admettant qu'un seul symbole non-terminal (outre le symbole de départ S) et où toute règle de dérivation est obligatoirement accompagnée de son pendant *inversé* (par exemple, $A \rightarrow [AA]$ est systématiquement accompagnée de $A \rightarrow \langle AA \rangle$). En adjoignant à une telle grammaire un dictionnaire probabiliste, et en ayant recours à différents pré- et post-traitements, l'auteur arrive à obtenir des correspondances entre structures, dont il évalue que plus de 70% sont bonnes.

3.4 Repérage de traduction pour une mémoire de traduction sous-phrastique

L'application qui nous intéresse, nous le rappelons, est un système d'aide à la traduction capable de récupérer, dans une mémoire de traduction, la traduction de segments sous-phrastiques d'un nouveau texte à traduire. Nous examinons ici les contraintes et besoins spécifiques d'une procédure de repérage de traduction dans ce contexte d'application particulier.

Avant d'aller plus loin, toutefois, il convient d'examiner d'un peu plus près le problème de l'analyse de traduction, ne serait-ce que pour se convaincre du niveau

de difficulté qu'il présente. Pour ce faire, il suffit en fait de se prêter au petit jeu consistant à relier les mots correspondants entre un texte et sa traduction de la figure 3.6 : il s'agit de trouver, pour chaque mot dans la traduction (à droite), le ou les mots du texte-source (à gauche) qui en est à l'origine. (Ces phrases proviennent du *Rapport du Secrétaire-Général sur l'Activité de l'Organisation des Nations-Unies*, 1993.)

Cet exemple fait ressortir un aspect important de la problématique de l'analyse de traduction : alors qu'il existe certaines unités linguistiques entre lesquelles les correspondances sautent aux yeux (*paix/peace*, *développement/development*, *démocratie/democracy*), il en est d'autres pour lesquelles les liens sont beaucoup plus complexes. Par exemple, dans la figure 3.6, la présence du mot français *ensemble* ne peut s'expliquer qu'en prenant en considération l'unité plus englobante *réalité d'ensemble*, qu'on peut alors relier à *comprehensive reality*; de la même façon, l'adverbe *indissolublement* ne trouve pas d'équivalent direct dans l'anglais, et il faut plutôt regarder le groupe *indissolublement liés*, qui correspond alors à toute la phrase *They are interlocking and mutually reinforcing.* (Ou bien, est-ce seulement *interlocking*, le reste de la deuxième phrase anglaise ne jouant alors qu'un rôle de "colle syntaxique" dans la traduction?).

La première leçon que l'on peut tirer de cet exemple bien réel, c'est que les ressources sur lesquelles reposera toute méthode d'analyse de traduction devront être très flexibles. Un dictionnaire bilingue établira probablement des liens explicites entre des mots comme *paix* et *peace*, ou encore *démocracie* et *democracy*, mais possiblement pas entre *liés* et *interlocking*, et encore moins entre *apparaître* et *reveal*, ou encore *ensemble* et *comprehensive*. Des thésaurus ou autres ressources lexicales apparentées pourraient certainement servir à combler ces lacunes, mais bien peu de telles ressources sont disponibles dans un format adapté au type de traitement automatisé que requiert notre contexte d'application.

À l'opposé, dans la mesure où ils sont exposés au matériel adéquat, les modèles statistiques de traduction sont théoriquement en mesure de capter des associations

The	Ce
comprehensive	sont
reality	trois
is	objectifs
most	indissolublement
clearly	liés
revealed	–
through	paix
three	,
objectives	développement
:	et
peace	démocratie
,	–
development	qui
and	font
democracy	le
.	mieux
They	apparaître
are	la
interlocking	réalité
and	d’
mutually	ensemble
reinforcing	.
.	

Figure 3.6. Un exercice d’alignement de mots

plus ou moins lointaines, et d'exprimer celles-ci quantitativement. En outre, considérant d'une part la disponibilité croissante de grandes quantités de texte bilingue en format électronique (voir par exemple les récents efforts d'extraction automatique de corpus bilingues du Web [23, 98]), d'autre part l'augmentation de la capacité des ordinateurs, ces modèles deviennent de plus en plus faciles et de moins en moins coûteux à développer. En somme, nous suggérons que les modèles statistiques sont mieux adaptés au genre d'application qui nous intéresse que les ressources "traditionnelles".

Depuis les travaux innovateurs de l'équipe IBM, la modélisation statistique de la traduction constitue un champ fertile d'activité scientifique, et de nombreux modèles ont été proposés qui tentent de combler l'une ou l'autre des faiblesses des modèles originaux. Nous en avons décrit brièvement quelques-uns à la section 3.3. Malgré tout, les modèles IBM demeurent encore la norme par rapport à laquelle tous les nouveaux modèles se comparent. Ceux-ci pourraient donc servir avantageusement de point de départ pour le développement de nos méthodes de repérage de traduction.

L'exemple de la figure 3.6 illustre un autre grand problème, que soulignait déjà Alan Melby il y a plus de 20 ans [75] : il semble qu'on ne puisse pas fixer *a priori* le niveau de résolution idéal de l'analyse de traduction. Dans certains cas, il est possible de trouver des correspondances au niveau des mots (on parle alors de *traduction mot-à-mot*); dans d'autres cas, il n'y a de bonne correspondance qu'au niveau de la phrase (on parle parfois de *paraphrase*); et entre ces deux pôles, tout semble possible *a priori*.

Ce problème découle en fait de la nature capricieuse du principe de compositionnalité de la traduction, telle qu'énoncé à la section 3.2. Il semble en effet que si ce principe s'applique à peu près systématiquement jusqu'au niveau des phrases, il tend à s'effriter graduellement à mesure qu'on approche du niveau du mot. En somme, pour en arriver à des analyses de traduction valables, on souhaiterait disposer de mécanismes qui soient capables de détecter le *point de rupture* au-delà duquel la compositionnalité ne s'applique plus. Ce problème est fondamental, et dans une

certaine mesure, on peut dire que c'est présentement la pierre d'achoppement de la modélisation statistique de la traduction.

Malgré tout, lorsque l'objectif visé par l'analyse de traduction est le repérage d'un segment spécifique, il semble qu'on puisse atténuer la portée de ce problème, du moins dans une certaine mesure. D'une part, on peut considérer que la séquence de mots du texte-source que constitue le point de mire du repérage fixe le niveau de résolution maximal de façon absolue : en principe, il n'y a pas lieu que l'analyse de traduction aille "plus profond" que ce niveau, c'est-à-dire qu'elle entre dans le point de mire. En d'autres mots, on peut considérer que les problèmes de compositionnalité (ou de non-compositionnalité) qui surgissent à l'intérieur du point de mire ne nous concernent pas.

Qu'en est-il au niveau même du point de mire et à l'extérieur de celui-ci? Lorsque le RT est utilisé dans un contexte de mémoire de traduction, nous prétendons qu'on peut formuler certaines hypothèses sur la compositionnalité. Le concept même de mémoire de traduction s'appuie sur deux suppositions :

1. on suppose qu'il existe dans les langues naturelles des suites de mots qui opèrent de façon autonome, c'est-à-dire de façon *relativement* indépendante du contexte dans lequel elles s'insèrent, et que de ce fait, ces suites de mots sont réutilisables dans d'autres contextes pour exprimer un concept similaire;

2. on suppose qu'il en va de même pour la traduction de ces suites de mots.

En somme, on fait l'hypothèse que de telles séquences opèrent essentiellement comme des unités lexicales dans l'une et l'autre langue.

De ce fait, pour une paire de textes S et T contenant une telle paire de séquences s et t, on peut également supposer qu'il existe un analyse de traduction strictement compositionnelle qui va au moins jusqu'au niveau de s et t. À la limite, cette analyse se limitera à dire que chacune des parties du couple $\langle S, T \rangle$ se décompose en trois parties :

- $S = s_1\ s\ s_2$, c'est-à-dire le segment s, la partie de S qui le précède et la partie qui le suit;

- $T = t_1\ t\ t_2$, c'est-à-dire un découpage analogue de T.

et que l'équivalence entre S et T peut être obtenue par la composition de $\langle s, t\rangle$ avec $\langle s_1\ s_2, t_1\ t_2\rangle$.

Ces suppositions n'ont rien de très osé : en fait, elles se vérifient pour la plupart des mots qu'on trouve dans un dictionnaire. Par exemple, si on revient à l'exemple de Isabelle *et al.* (figure 3.2), l'analyse de la figure 3.7, basée sur l'équivalence entre *sa* en français et *his* en anglais, serait tout à fait valable (à défaut d'être très utile). Par ailleurs, les systèmes de mémoires de traduction existants supposent que c'est également vrai pour des phrases complètes. L'existence de telles séquences entre le niveau de la phrase et celui du mot ne fait certainement pas de doute; le problème, c'est de savoir les reconnaître.

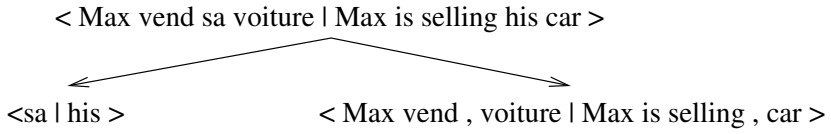

Figure 3.7. Une analyse compositionnelle simpliste

Au chapitre 2, nous avons formulé l'hypothèse qu'en choisissant dans le texte à traduire des séquences de mots dont les frontières coïncident avec celles de constituants syntaxiques (spécifiquement : de tronçons), on augmentait les chances de tomber sur de telles séquences "autonomes". La nature même de l'application de mémoire de traduction nous permet d'aller plus loin : nous suggérons que le simple fait qu'une séquence donnée se retrouve dans plus d'un contexte renforce d'autant la probabilité que cette séquence opère de façon autonome dans le texte, et donc que les suppositions ci-dessus concernant la compositionnalité soient vérifiées.

En résumé, le contexte dans lequel nous utilisons ici le RT nous permet de formuler une hypothèse de compositionnalité stricte, au moins jusqu'au niveau du point de mire. (Et au-delà de ce niveau, la question ne nous concerne plus.)

Il y aura évidemment des situations où cette hypothèse s'effondrera, peut-être même en sera-t-il ainsi dans la majorité des cas. Le quatrième couple de la figure 3.1 en est d'ailleurs un bel exemple : le point de mire *it would make* se trouve en fait ici impliqué dans une séquence non-compositionnelle qui en dépasse les frontières, plus spécifiquement l'expression idiomatique *to make sense* (en français, *avoir du sens*). Dans ce cas, une application stricte du principe de compositionnalité risque de nous mener au repérage-cible erroné *ça aurait*. Comme on l'a vu, ce genre de phénomène appelle généralement un *élargissement* du repérage-source à l'entour du point de mire, menant ici au repérage correct *it would make sense*/*ça aurait du sens*.

Évidemment, dans le contexte d'application qui nous intéresse, on peut argumenter que le repérage correct ne serait d'aucune utilité pour le traducteur, puisqu'en élargissant le contexte du point de mire, on a perdu la stricte équivalence avec celui-ci. En pratique, ce n'est pas nécessairement le cas, et on aurait peut-être tort de vouloir éliminer d'emblée ces repérages. Par exemple, il se trouve tout à fait par hasard que le point de mire *it would make* dans notre exemple provenait initialement de la phrase de départ suivante :

> Reducing further capital gains tax with the ultimate goal of eliminating *it would make* a great deal of sense.

Le verbe *to make* dans les deux phrases se retrouve donc en fait impliqué dans la même expression idiomatique *to make sense*, sauf que dans la phrase de départ, *make* et *sense* se trouvent séparés par une locution adverbiale. Autre problème : le rattachement du pronom *it* dans cette phrase est complètement différent de celui du même pronom dans la phrase issue de la mémoire de traduction. Malgré tout,

la traduction ultimement proposée par le traducteur pour la phrase de départ est la suivante :

> *Il serait tout à fait sensé* d'abaisser davantage cet impôt, l'objectif ultime étant de l'éliminer complètement.

Dans ce cas spécifique, on aurait très bien pu utiliser le repérage-cible *ça aurait du sens* à la place de la formulation choisie ici :

> *Ça aurait du sens* d'abaisser davantage cet impôt, l'objectif ultime étant de l'éliminer complètement.

Idéalement, un processus de repérage de traduction devrait fonder la décision de procéder ou non à ce genre d'élargissement du point de mire sur la base de connaissances fines sur la compositionnalité de la traduction : par exemple, un mécanisme de repérage du *point de rupture* tel que suggéré plus haut. À défaut de telles connaissances toutefois, il semble que certaines notions plus facilement accessibles puissent s'avérer utiles. Par exemple, si on "sait" dans cet exemple spécifique que le mot anglais *sense* est lié syntaxiquement de façon forte au verbe *make* qui le précède, on peut interdire une coupure entre les deux. De ce fait, on va se trouver à élargir le point de mire, augmentant ainsi substantiellement nos chances d'aboutir au repérage correct.

D'une façon plus générale, ceci suggère qu'un minimum de connaissances sur les liens qui unissent les mots à l'intérieur d'une phrase peut être bénéfique au RT. Faut-il aller aussi loin dans cette direction que Wu avec ses grammaires d'inversion? Probablement pas. D'ailleurs, il faut remarquer que dans son utilisation des ITGs, Wu ne fait aucune tentative sérieuse pour élaborer des grammaires bilingues linguistiquement motivées. En pratique, il se contente plutôt de grammaires triviales (dans les expériences décrites dans [118], l'ITG la plus complexe comporte quatre

symboles non-terminaux et 13 règles – à part le symbole de départ, aucun des non-terminaux ne représente un consituant linguistique au sens habituel du terme). Ces grammaires sont extrêmement permissives, en ce sens qu'elles n'imposent quasiment aucune contrainte sur la nature des structures arborescentes produites; dans la plupart des expériences qu'il décrit, Wu se fie entièrement au modèle de traduction (typiquement, l'équivalent d'un modèle IBM-1) pour décider de la nature des structures. Pour combler les lacunes d'une telle approche, plutôt que d'élaborer des grammaires bilingues plus complexes, Wu préfère adjoindre au système un ensemble de mécanismes de pré- et de post-traitement, qui vont par exemple se charger de rattacher les déterminants et les prépositions au groupe suivant plutôt qu'au précédent, etc. Par de tels mécanismes, Wu se rapproche donc en fait graduellement des approches *parse-parse-match* décrites à la section 3.3.3. Par ailleurs, ces traitements périphériques s'apparentent aux "ajustements linguistiques" proposés par Brown *et al.* pour contourner les faiblesses de leurs modèles de traduction statistiques [16].

C'est donc plutôt vers ce genre d'approche que nous nous proposons d'aller pour le RT : l'intégration de connaissances linguistiques rudimentaires, par exemple des tronçonnages syntaxiques, qui permettront d'établir les frontières entre les constituants de base des textes source et cible.

Finalement, nous avons mentionné en introduction que dans notre contexte d'application, les points de mire forment naturellement des séquences contiguës du texte source, mais que rien n'oblige les repérages-cible à être également contigus. Toutefois, comme on l'a souligné à la section 2.4, des repérages-cible non contigus risquent d'être difficilement utilisables par un traducteur humain. C'est pourquoi nous suggérons que, toujours dans le contexte spécifique du RT pour une mémoire de traduction, il sera généralement préférable que les repérages produits soient contigus.

3.5 Méthodes de repérage de traduction

Dans cette section, nous présentons une série de méthodes de repérage de traduction, adaptées au contexte des mémoires de traduction sous-phrastique, dans lesquelles nous intégrons graduellement les différents éléments dont nous avons discuté à la section précédente.

Nous commençons par proposer une méthode reposant directement sur les techniques d'alignement de mots proposées par Brown et al. [17] dans le cadre de la traduction automatique statistique. La présentation de cette première méthode a pour but de mettre la table pour les méthodes suivantes, et d'établir un point de comparaison.

Partant de cette méthode, nous montrons comment on peut y intégrer d'une part des connaissances linguistiques rudimentaires, et d'autre part des contraintes de compositionnalité et de contiguïté du repérage-cible.

3.5.1 Repérage de traduction statistique

Brown *et al.* [17] proposent différentes méthodes statistiques d'alignement des mots, que nous avons décrites à la section 3.3. Ces méthodes, dites *Viterbi*, recherchent l'alignement le plus probable, sous la contrainte que chaque mot de la source est connecté à au plus un seul mot de la cible (rien n'empêche qu'un mot de la cible soit lié à plus d'un mot dans la source toutefois). Ainsi, pour une phrase-source $S = s_1...s_M$ et sa traduction $T = t_1...t_N$, un tel alignement prend la forme d'une séquence $a = a_1...a_M$, qui établit pour chaque mot s_i de la source sa "destination" la plus probable t_{a_i} dans la cible (admettant la possibilité que $a_i = 0$, ce qui dénote le cas où s_i ne génère rien dans la traduction).

Le calcul de ces alignements se fait au moyen des paramètres des différents modèles (on parlera d'un alignement *Viterbi-1* si on utilise les paramètres d'un modèle IBM-1, *Viterbi-2* pour les modèles IBM-2, etc.). Rappelons toutefois qu'à partir du modèle

3, il n'est plus possible de trouver l'alignement optimal en temps polynomial, et on a alors recours à des heuristiques pour approcher le maximum.

On peut évidemment concevoir une méthode de RT qui utilise directement un tel alignement : si le point de mire p (contigu) coïncide avec la séquence $s_p = s_{i_1}...s_{i_2}$ de S, le repérage-cible sera l'ensemble $t_p = \{t_{a_i} | i_1 \leq i \leq i_2\}$.

Si on dispose plutôt d'un modèle inverse, c'est-à-dire que l'alignement a fait référence à des positions dans la source plutôt que la cible, on dira alors que $t_p = \{t_j | i_1 \leq a_j \leq i_2\}$. *A priori*, il n'y a pas de raisons particulières de préférer une méthode à l'autre (mais nous y reviendrons plus tard), et nous nous en tenons donc pour l'instant au modèle standard ici. Dans la suite de l'exposé, nous désignons les RT ainsi obtenues sous le terme *RT Viterbi* ou, plus succintement, *RT V*. Un exemple de ce genre d'alignement apparaît à la figure 3.8.

Comme on le sait, s'il est basé sur un modèle IBM-1 ou IBM-2, l'alignement Viterbi peut être calculé de façon exacte et efficace, en identifiant, pour chaque mot s_i de la source, le mot t_j de la cible pour lequel $t(s_i|t_j)a(j|i,m,n)$ est maximal. Dans la mesure où l'on a accès à ces paramètres en temps constant (ce qui est possible s'ils sont stockés dans un tableau ou une table de hachage, par exemple), l'alignement requiert un nombre d'opérations dans $\mathcal{O}(MN)$, M étant la longueur de la phrase source S et N celle de la phrase cible T. Pour la tâche de repérage, on peut en fait se contenter de ne calculer que la partie de l'alignement qui concerne le point de mire. Si celui-ci comporte I mots, le nombre d'opérations requises pour calculer le RT V est dans $\mathcal{O}(IN)$.

3.5.2 Connaissances linguistiques

Comme on l'a suggéré plus haut, tout porte à croire que les méthodes d'alignement de mots, statistiques ou autres, pourraient profiter de certaines connaissances linguistiques, même rudimentaires. Par ailleurs, dans le cadre de notre application, lorsque les séquences de mots dont on cherche la traduction coïncident avec des séquences de

point de mire : *the government 's commitment*

couple :

- Let us see where *the government's commitment* is really at in terms of the farm community.
- Voyons quel est le véritable engagement du gouvernement envers la communauté agricole.

alignement (partiel) :

the	→	le
government	→	gouvernement
's	→	du
commitment	→	engagement

repérage-cible t_p :

- Voyons quel est **le** véritable **engagement du gouvernement** envers la communauté agricole.

Figure 3.8. Exemple de RT Viterbi.

tronçons syntaxiques dans la phrase à traduire P, on peut penser qu'il serait bénéfique d'en tirer parti dans le RT, par exemple en basant le repérage sur un alignement de tronçons plutôt que sur un alignement de mots.

On peut envisager une adaptation directe de la méthode de RT Viterbi proposée ci-dessus, qui incorpore ce genre de connaissance, dans la mesure où l'on dispose d'une part de tronçonnages pour la langue-source et la langue-cible, et d'autre part d'un modèle statistique de traduction basé sur les tronçons plutôt que sur les mots. Nous développons cette idée ci-dessous.

Un modèle statistique de traduction basé sur des tronçons syntaxiques pourrait s'inspirer directement des modèles de *canal bruité* proposés par Brown *et al.* [17]. Dans cette perspective, on considère les textes source S et cible T non plus comme des séquences de mots, mais bien comme des séquences de tronçons, que l'on dénote

$$S = c_{S_1}...c_{S_m}, T = c_{T_1}...c_{T_n}$$

Ici, c_{S_i} (respectivement, c_{T_j}) dénote le i-ème tronçon de S (respectivement, T); la phrase S comporte donc m tronçons, alors que T en comporte n. On suppose par ailleurs que tous les mots de S et T appartiennent à un tronçon (en pratique, les séquences de mots contigus laissées de côté par le tronçonnage peuvent être soit regroupées en tronçons de type *inconnu*, soit considérées individuellement comme des tronçons autonomes). Avec cette nouvelle notation, on peut réécrire ainsi l'équation 3.2 :

$$\Pr(S, a|T) = \Pr(m|T) \prod_{i=1}^{m} \Pr(a_i|a_1^{i-1}, c_{S_1}...c_{S_{i-1}}, m, T)\Pr(c_{S_i}|a_1^i, c_{S_1}...c_{S_{i-1}}, m, T). \tag{3.7}$$

Notons qu'ici, a dénote un alignement entre des tronçons plutôt qu'entre des mots.

D'un point de vue génératif, ce modèle suggère que la production d'une phrase-source S et d'un alignement a, étant donnée une phrase en langue-cible T, procède de la façon suivante : on décide d'abord du nombre de tronçons m dans S, en fonction

de T; ensuite, connaissant T et m, on peut décider du tronçon $c_{T_{a_1}}$ de T qui va donner lieu au premier tronçon de S; enfin, à partir de toutes ces connaissances, on génère le premier tronçon de S : c_{S_1}; on répète ensuite ce processus pour les tronçons subséquents de S, en prenant compte en plus des tronçons déjà générés et des tronçons de T auxquels ils correspondent.

Suivant la piste tracée par Brown *et al.*, on peut formuler certaines hypothèses simplificatrices sur les distributions apparaissant dans l'équation 3.7, et ainsi élaborer différents modèles visant à produire une approximation de $\Pr(S|T)$. Nous nous en tiendrons ici à un modèle analogue au modèle 2 de Brown *et al.*, principalement pour des raisons de complexité que nous détaillons plus loin.

Dans ce modèle, que nous appelons *modèle-tronçon 2*, on supposera donc que

- $\Pr(m|T)$ suit une distribution uniforme (on emploie en pratique une constante ϵ);
- $\Pr(a_i|a_1^{i-1}, c_{S_1}...c_{S_{i-1}}, m, T)$ ne dépend que de a_i, i, m et n;
- $\Pr(c_{S_i}|a_1^i, c_{S_1}...c_{S_{i-1}}, m, T)$ ne dépend que de c_{S_i} et de $c_{T_{a_i}}$;

Le modèle dépendra donc des paramètres suivants :

- ϵ : la probabilité du nombre de tronçons m;
- $t_c(c_{S_i}|c_{T_j})$: les probabilités de traduction entre tronçons;
- $a_c(j|i, m, n)$: les probabilités d'alignement entre tronçons.

Bien qu'un tel modèle soit parfaitement bien fondé du point de vue mathématique, l'acquisition de ces paramètres bute en pratique sur un obstacle majeur. Comme on le sait, ces paramètres sont normalement appris automatiquement sur un corpus d'exemples constitué de paires de phrases, au moyen de la procédure EM. Dans le

cas des modèles-tronçon, cet apprentissage soulève de façon critique le problème de sous-représentation des données. Ce problème se pose tout particulièrement pour la distribution t_c, puisqu'elle doit couvrir toutes les paires de tronçons possibles. Considérant que la sous-représentation des données est déjà problématique pour les modèles de mots standard, un entraînement direct sur des paires de tronçons apparaît irréaliste.

L'alternative que nous proposons consiste à produire une approximation des distributions du modèle-tronçon au moyen d'un modèle de traduction sur les mots. Cette approche se fonde bien entendu sur l'intuition qu'il existe un lien compositionnel entre les paires de tronçons d'un alignement et les paires de séquences de mots qui les composent. Ainsi, si on dispose des paramètres t et a d'un modèle IBM-2 standard (c'est-à-dire, un modèle se basant sur les mots), on peut obtenir une approximation de la valeur de t_c pour une paire de tronçons $c_{S_i} = s_{i_1}...s_{i_2}$ et $c_{T_j} = t_{j_1}...t_{j_2}$, apparaissant dans des phrases S et T de longueur M et N respectivement. Pour ce faire, nous adaptons directement la formule proposée par Brown *et al.* [17] pour $\Pr(s|t)$:

$$t_c(c_{S_i}|c_{T_j}) \approx \epsilon \prod_{I=i_1}^{i_2} \left[t(s_I|t_0)a(0|I,M,N) + \sum_{J=j_1}^{j_2} t(s_I|t_J)a(J|I,M,N) \right] \tag{3.8}$$

Cette formule est obtenue en considérant la probabilité d'observer les mots qui constituent le tronçon c_{S_i}, étant donnés les mots du tronçon c_{T_j}, en regard de l'ensemble des alignements possibles entre tous ces mots. Rappelons qu'ici, la position $J = 0$ correspond au mot-nul, conventionnellement placé en position zéro de la cible, et qui sert à expliquer les mots de la source qui ne génèrent rien dans la cible. Le terme $t(s_I|t_0)a(0|I,M,N)$ dans le produit correspond donc à la contribution de ce mot-nul.

Un calcul analogue nous permet d'obtenir une approximation des paramètres a_c à partir des paramètres a :

$$a_c(j|i,m,n) \approx \prod_{I=i_1}^{i_2} \left[a(0|I,M,N) + \sum_{J=j_1}^{j_2} a(J|I,M,N) \right] \tag{3.9}$$

Il est important de noter que l'une et l'autre de ces approximations ne sont valables que pour des phrases de M et N mots : pour être plus exact, nous devrions écrire ci-dessus $t_c(c_{S_i}|c_{T_j}, M, N)$ et $a_c(j|i, m, n, M, N)$. Une approximation plus générale de t_c et a_c devrait prendre en compte toutes les longueurs M et N possibles, et faire intervenir un estimé de la distribution de $\Pr(M, N|m, n)$. Nous allons toutefois nous en tenir à ces formules (et à cette notation) dans la suite, parce qu'en pratique, l'une et l'autre de ces approximations peuvent être calculées en temps réel, pour des paires de phrases pour lesquelles les longueurs M et N sont connues.

Évidemment, rien n'empêche de baser notre approximation de t_c sur les paramètres d'un modèle de mots tout autre, par exemple le modèle 3 ou 4, ou tout autre modèle du même genre, tels que ceux décrits à la section 3.3. Toutefois, avec le modèle 2, on peut calculer l'approximation de l'équation (3.8) dans un temps proportionnel au produit de la longueur des tronçons, ce qui n'est pas le cas avec les modèles d'ordre supérieur. C'est pourquoi nous nous en tenons à ce modèle.

À l'instar de Brown *et al.*, on peut alors définir une notion d'alignement *Viterbi* pour le modèle-tronçon 2 : c'est l'alignement a^* entre les tronçons qui maximise $\Pr(a|T, S)$. Tout comme c'était le cas pour le modèle IBM-2 standard, cet alignement se trouve en recherchant, pour chaque tronçon de la source c_{S_i}, le tronçon-cible j qui maximise $\gamma_c(j, i, m, n) = t_c(c_{S_i}|c_{T_j})\ a_c(j|i, m, n)$.

À partir de ce point, on peut formuler une méthode de RT, que nous désignons *RT Viterbi-tronçons* (ou *RT VT*), dans laquelle on utilise directement l'alignement a^* entre les tronçons pour repérer la traduction du point de mire. Cette méthode est en tous points analogue au RT Viterbi proposé plus haut, à une différence près : rien ne garantit *a priori* que le point de mire du RT coïncide nécessairement avec une séquence de tronçons dans S. En effet, des contextes différents peuvent très bien entraîner des tronçonnages différents, de telle sorte qu'une suite de mots qui constitue un tronçon dans la phrase de départ P peut se trouver englobée dans un tronçon plus large dans S, ou encore chevaucher deux tronçons de S.

On peut envisager différentes stratégies pour contourner cette difficulté, celle que nous proposons consiste simplement à ne retenir, dans l'alignement a^*, que les paires de tronçons qui couvrent en tout ou en partie le point de mire. En somme, on étendra le point de mire aux frontières de tronçons les plus proches dans S.

La figure 3.9 illustre un exemple de RT Viterbi-tronçons.

Du point de vue de son implantation informatique, le RT Viterbi-tronçon (VT) s'effectue de la même façon que le Viterbi normal, à la différence qu'on doit également calculer les approximations des paramètres t_c et a_c à partir des paramètres t et a d'un modèle de mots, suivant les formules des équations (3.8) et (3.9). Si, pour simplifier, on fait l'hypothèse que les tronçons source et cible comptent en moyenne C mots, un point de mire de I mots comportera donc I/C tronçons, alors que la phrase-cible T en comportera n/C. La recherche du repérage requerra donc In/C^2 comparaisons.

Par ailleurs, le calcul de chaque paramètre t_c et a_c suivant les équations (3.8) et (3.9) exige en moyenne un nombre d'opérations dans $\mathcal{O}(C^2)$. En somme, le nombre total d'opérations pour le RT VT est dans $\mathcal{O}(InC^2/C^2) = \mathcal{O}(In)$, c'est-à-dire une complexité identique à celle de la méthode RT V.

3.5.3 Contiguïté du repérage

De par la nature de notre procédure de recherche des couples $\langle S, T\rangle$ dans la mémoire de traduction $\mathcal{M}$, le repérage-source s_p dénote toujours une séquence contiguë de S. Toutefois, avec les méthodes de RT statistiques proposées ci-dessus, rien ne garantit que ce sera le cas du repérage-cible t_p. Dans les alignements IBM, un mot de la cible peut avoir pour origine plusieurs mots de la source, parfois disséminés çà et là dans la phrase, parfois même plusieurs occurrences d'un même mot. En pratique, cette façon de choisir l'alignement d'un mot de la source sans égard pour les autres connexions mène souvent à des aberrations. Par ailleurs, et comme on l'a suggéré à la section 3.4, la contiguïté de t_p apparaît avantageuse dans notre application.

Pour imposer une telle contrainte, une approche possible consiste à effectuer un

point de mire : *the government 's commitment*

couple :

- $[_{VP}$ Let] $[_{NP}$ us] $[_{VP}$ see] $[_{ADVP}$ where]

  $[_{NP}$ the government] $[_{NP}$'s commitment] $[_{VP}$ is]

  $[_{ADVP}$ really] $[_{PP}$ at] $[_{PP}$ in terms of]

  $[_{NP}$ the farm community]

- $[_{VP}$ Voyons] $[_{VP}$ quel est] $[_{NP}$ le véritable engagement]

  $[_{PP}$ du] $[_{NP}$ gouvernement] $[_{PP}$ envers]

  $[_{NP}$ la communauté agricole]

alignement (partiel) :

$[_{NP}$ the government]	$\rightarrow$	$[_{NP}$ gouvernement]
$[_{NP}$'s commitment]	$\rightarrow$	$[_{NP}$ le véritable engagement]

repérage-cible t_p :

- Voyons quel est **le véritable engagement** du **gouvernement** envers la communauté agricole.

Figure 3.9. Exemple de RT Viterbi-tronçons.

post-traitement sur l'ensemble de mots t_p résultant de l'alignement, de façon à produire un nouveau repérage t'_p ayant la propriété souhaitée. Voici quelques stratégies possibles :

- *tolérance-zéro* : C'est la stratégie radicale, qui consiste à stipuler que tout repérage t_p non contigu est forcément inutilisable pour le traducteur. Partant de ce principe, on produira $t'_p = t_p$ si t_p est contigu, et $t'_p = \emptyset$ sinon.

- *séquence couvrante* : C'est, d'une certaine façon, la stratégie inverse : on fait l'hypothèse qu'il est plus utile de fournir une séquence qui contient la traduction de p plutôt que rien du tout. Ainsi, on obtiendra un repérage contigu t'_p en déterminant les *mots limites* de t_p, c'est-à-dire le mot t_i qui est le plus à gauche dans T et t_j qui est le plus à droite, et on produira la plus petite séquence contiguë t'_p qui recouvre tous les mots de t_p.

$$t'_p = \{t_k | i \leq k \leq j\}$$

- *meilleure séquence contiguë* : C'est une stratégie mitoyenne : il s'agit en fait de ne conserver dans t'_p que le sous-ensemble des mots de t_p qui y constitue la plus longue séquence contiguë.

Évidemment, on peut imaginer d'autre stratégies plus élaborées, tenant compte par exemple des probabilités d'alignement, ou encore de la distance entre les segments contigus de t_p. Mais nous nous en tenons pour l'instant à ces approches simples. La figure 3.10 illustre l'effet de ces trois stratégies sur le repérage Viterbi de la figure 3.8.

On se convaincra que ces différents post-traitements peuvent aisément être implantés de façon à être effectués en un temps proportionnel à la taille du repérage-cible, lui-même proportionnel à la taille de la phrase-cible ($\mathcal{O}(n)$).

Alternativement, on peut chercher à imposer la contrainte de contiguïté à même la procédure d'alignement. Par exemple, on peut formuler une variante de la méthode

repérage :

- Let us see where **the government's commitment** is really at in terms of the farm community.
- Voyons quel est **le** véritable **engagement du gouvernement** envers la communauté agricole.

post-traitements :

tolérance-zéro :	$t'_p = \emptyset$
séquence couvrante :	$t'_p =$ le véritable engagement du gouvernement
meilleure séquence :	$t'_p =$ engagement du gouvernement

Figure 3.10. Post-traitements sur le RT Viterbi visant à produire un repérage-cible contigu.

de RT Viterbi, qui cherche l'alignement qui maximise $p(a|S,T)$, sous la contrainte que les mots de la cible qui sont alignés au point de mire doivent être contigus. Considérons une procédure qui recherche un segment $t_{J_1}...t_{J_2}$ de T, tel que :

$$\langle J_1, J_2 \rangle = \text{argmax}_{1 \leq j_1 \leq j_2 \leq N} \Pr(a_r | s_{i_1}^{i_2}, t_{j_1}^{j_2}) \Pr(a_{\bar{r}} | s_1^{i_1-1} s_{i_2+1}^{M}, t_1^{j_1-1} t_{j_2+1}^{N})$$

Une telle procédure produit en fait deux alignements partiels entre S et T : un alignement a_r, qui associe les mots du point de mire (la séquence $s_{i_1}^{i_2}$) avec une séquence de mots contigus de T (la séquence $t_{j_1}^{j_2}$), et un alignement $a_{\bar{r}}$, qui associe le reste de la phrase S (c'est-à-dire : tous les mots à l'extérieur du point de mire) avec le reste de la phrase T. Ensemble, ces deux alignements constituent l'alignement $a = a_r \cup a_{\bar{r}}$, dont la probabilité est maximale, sous la double contrainte que :

1. les mots du point de mire (la séquence $s_{i_1}^{i_2}$) ne peuvent être la source que de mots dans une séquence contiguë de T (les séquences $t_{j_1}^{j_2}$));

2. les mots qui se trouvent à l'extérieur du point de mire (dans l'une des deux séquences $s_1^{i_1-1}$ et $s_{i_2+1}^{M}$) ne peuvent être la source que de mots qui se trouvent à l'extérieur de $t_{j_1}^{j_2}$.

À partir d'un tel alignement, il est trivial de concevoir une méthode de RT, qui prendra pour repérage-cible la séquence $t_{J_1}^{J_2}$. (Notons que la procédure ci-dessus ne prend pas en compte la possibilité d'un repérage-cible vide, mais on peut aisément le considérer séparément; c'est seulement pour alléger la notation que nous l'avons omis.) Nous appelons cette méthode *RT Viterbi-contigu* (ou *RT VC*).

En fait, la contrainte 1 ci-dessus serait suffisante pour assurer la contiguïté du repérage-cible : on admettrait alors que certains mots de $t_{j_1}^{j_2}$ peuvent avoir pour source un mot à l'extérieur du point de mire. Le rôle de la contrainte 2 est d'obliger la procédure à produire un repérage de taille minimale, en interdisant les liens entre celui-ci et des mots à l'extérieur du point de mire. En d'autres mots, on fait

l'hypothèse que le point de mire agit de façon autonome dans la phrase-source, et de ce fait génère dans T une séquence contiguë, indépendamment du reste de la phrase. Sans aller aussi loin que les modèles un-à-un de Melamed [74] ou les ITG de Wu [118], le RT Viterbi-contigu soumet en somme l'alignement à une forme minimale de compositionnalité, ne s'appliquant qu'au seul point de mire, tel que proposé à la section 3.4.

Le RT Viterbi-contigu peut se calculer de façon directe, en trouvant les alignements a_r et $a_{\bar{r}}$ pour chaque paire de positions $\langle j_1, j_2 \rangle$ dans la cible. On calcule alors la probabilité conjointe de ces deux alignements, suivant la formule :

$$\Pr(a_r | s_{i_1} ... s_{i_2}) = \prod_{i=i_1}^{i_2} \Pr(a_{ri} | i, M, N)$$

où $\Pr(j|i, M, N)$ se calcule comme dans l'équation 3.4. On conserve finalement la paire $\langle J_1, J_2 \rangle$ qui maximise cette probabilité. La figure 3.11 illustre l'alignement Viterbi-contigu obtenu sur l'exemple de la figure 3.8.

Une implantation directe du RT Viterbi-contigu (VC) requiert qu'on examine toutes les séquences contiguës $t_{j_1} ... t_{j_2}$ de T, et qu'on effectue 2 alignments Viterbi pour chacune. Plus précisément :

- Si le point de mire comporte I mots et que le repérage-cible sous examen est de longueur $j_2 - j_1 + 1 = J$, l'alignement a_r requiert IJ comparaisons, alors que l'alignement $a_{\bar{r}}$ en requiert $(m - I)(n - J)$;

- Le calcul de la probabilité de a_r et $a_{\bar{r}}$ peut se faire au fil de ces comparaisons, sans ajouter à la complexité de l'opération;

- On doit examiner n séquences de longueur 1 (donc J ci-dessus prendra n fois cette valeur), $n - 1$ de longueur 2, etc.

En somme, le nombre d'opérations requises pour un point de mire de I mots sera :

$$\sum_{J=1}^{n} (n - J)(IJ + (m - I)(n - J))$$

	Let	→	Voyons
	us	→	Voyons
	see	→	Voyons
	where	→	quel
point de mire :	the	→	engagement
	government	→	gouvernement
	's	→	du
	commitment	→	engagement
	is	→	est
	really	→	véritable
	at	→	la
	in	→	envers
	terms	→	envers
	of	→	envers
	the	→	la
	farm	→	agricole
	community	→	communauté
	.	→	.

Figure 3.11. Alignement Viterbi sous la contrainte de contiguïté

Sans entrer dans le développement fastidieux de cette somme, on peut constater que la complexité du RT VC sera dans $\mathcal{O}(mn^3)$.

On peut réduire substantiellement ce temps de calcul en ayant recours à une astuce et à une heuristique. D'abord l'heuristique : on constate en pratique qu'un point de mire de longueur I a de fortes chances de donner lieu à un repérage-cible de longueur comparable. Donc, en pratique, on peut se limiter à n'examiner que les séquences dont la longueur J est proche de I, par exemple $I/2 \leq J \leq 2I$, ou encore $I - K \leq J \leq I + K$. Sachant qu'en pratique les points de mire ne font jamais plus d'une dizaine de mots, et typiquement 3 ou 4, ceci réduit effectivement la complexité moyenne du calcul d'un ordre de grandeur, donc dans $\mathcal{O}(mn^2)$.

Ensuite, on peut accélérer substantiellement le temps moyen de calcul en précalculant partiellement les alignements Viterbi. Pour ce faire, il suffit de construire, pour chaque position i dans la source, une liste des positions j dans la cible, triée en ordre décroissant de $t(s_i|t_j)a(j,i,m,n)$. Lors du calcul effectif des alignements a_r et $a_{\bar{r}}$, on consultera pour chaque i la liste triée correspondante, dans laquelle on pigera la première position j qui satisfasse à la contrainte de contiguïté. Dans le cas où le segment-cible examiné représente moins de la moitié de la phrase T, la plupart du temps, l'alignement pour une position-source i sera le premier j dans la liste. Or, c'est précisément ce qui se produit si cette astuce est employée conjointement avec l'heuristique ci-dessus, dans laquelle les repérages sont typiquement beaucoup plus courts que les phrases dans lesquelles ils s'insèrent. Sans changer la complexité de la procédure, cette astuce nous permettra donc de nous rapprocher d'un nombre d'opérations proportionnel à mn, surtout lorsque les phrases considérées sont longues. Par ailleurs, l'investissement initial pour le tri des listes peut se faire dans $\mathcal{O}(mn \log n)$, en utilisant par exemple un monceau (*heap*).

3.5.4 Contiguïté et connaissances linguistiques

Nous avons vu d'une part à la section 3.5.2 comment on pouvait incorporer certaines connaissances linguistiques dans le processus de RT statistique, et d'autre part à la section 3.5.3 comment on pouvait imposer une contrainte de contiguïté à ce même processus. Évidemment, rien n'empêche de faire les deux.

Pour ce faire, il suffit en fait d'appliquer la méthode de RT Viterbi-contigu avec un modèle de tronçons, tel que décrit à la section 3.5.2. Ceci donne lieu à une méthode *Viterbi-tronçon-contigu* (RT VTC), dans laquelle le repérage-cible consistera obligatoirement en une séquence de tronçons.

Alternativement, on peut utiliser directement un modèle de mots pour calculer les alignements Viterbi, et se contenter de contraindre la recherche aux seules paires $\langle j_1, j_2 \rangle$ qui coïncident avec des frontières de tronçons. Nous désignons cette variante sous le terme *RT Viterbi-contigu-contraint* (RT VCC). *A priori*, rien ne porte à croire que cette façon de faire diffère substantiellement du RT Viterbi-tronçon-contigu, mais nous verrons ce qu'il en est en pratique à la section 3.7.

Dans cette procédure, le temps de calcul des paires d'alignement Viterbi est le même que pour la méthode RT VC. Toutefois, puisque les repérages-cible sont contraints de coïncider avec des frontières de tronçon, le nombre d'alignements à considérer se trouve divisé par un facteur de C^2 par rapport à la méthode RT VC, où C est la longueur moyenne des tronçons. Le repérage se fera ainsi en un nombre d'opérations dans $\mathcal{O}(mn^2/C^2)$.

Il en va de même pour le RT Viterbi-tronçon-contigu (RT VTC), sauf qu'en plus, si on précalcule les alignements Viterbi comme dans le RT VC, les approximations de t_c et a_c ne sont pas à recalculer pour chaque repérage-cible considéré. Indépendemment du gain enregistré grâce au tri, ceci réduit effectivement le temps requis pour calculer chaque paire d'alignements Viterbi à $\mathcal{O}(mn/C^2)$. En somme, la complexité moyenne de la méthode RT VTC est dans $\mathcal{O}(mn^2/C^4)$.

3.5.5 RT compositionnel

Les alignements de type IBM autorisent qu'un mot de la cible soit lié à plusieurs mots de la source, ce qui mène parfois à des aberrations, comme on l'a souligné à la section 3.5.3. À l'opposé, dans des modèles tels que ceux de Melamed [74] et Wu [118], on impose une contrainte d'unicité, c'est-à-dire que les mots tant de la cible que de la source ne peuvent être liés au plus qu'à un seul mot. Ceci revient à une certaine formulation du principe de compositionnalité de la traduction, dans laquelle on suggère que chaque mot de la source opère de façon indépendante dans la phrase pour générer au plus un mot de la cible, qui ne dépend alors d'aucun autre mot de la source.

On a vu à la section 3.5.3 comment une contrainte de contiguïté du repérage permettait de faire intervenir le principe de compositionnalité dans l'alignement, mais d'une façon moins contraignante. En fait, on peut pousser un peu plus loin l'application de ce principe. Considérons une procédure qui cherche à découper la paire de phrases $\langle S, T \rangle$ en deux parties indépendantes, de façon à maximiser la probabilité de l'alignement Viterbi résultant :

$$\langle I, J, D \rangle = \text{argmax}_{\langle i,j,d \rangle} \begin{cases} d = 1 \ : & \Pr(a_1 | s_1^i, t_1^j) \Pr(a_2 | s_{i+1}^m, t_{j+1}^n) \\ d = -1 \ : & \Pr(a_1 | s_1^i, t_{j+1}^n) \Pr(a_2 | s_{i+1}^m, t_1^j) \end{cases} \tag{3.10}$$

Dans le triplet $\langle i, j, d \rangle$ ci-dessus, i représente un *point de coupe* dans la phrase-source S, j en est l'analogue dans la phrase-cible T, et d est une *direction de correspondance* : $d = 1$ désigne une correspondance *parallèle*, c'est-à-dire que $s_1 ... s_i$ correspond à $t_1 ... t_j$ et $s_{i+1} ... s_m$ correspond à $t_{j+1} ... t_n$, alors que $d = -1$ désigne une correspondance *croisée*, c'est-à-dire que $s_1 ... s_i$ correspond à $t_{j+1} ... t_n$ et $s_{i+1} ... s_m$ correspond à $t_1 ... t_j$.

Une telle procédure trouvera un alignement de probabilité maximale entre S et T, sous l'hypothèse que l'une et l'autre phrases sont constituées de deux parties

indépendantes ($s_1...s_I$ et $s_{I+1}...s_m$ d'une part, $t_1...t_J$ et $t_{J+1}...t_n$ d'autre part), qui se correspondent deux-à-deux, suivant la direction D.

On peut répéter ce processus de façon récursive sur chacune des paires de segments ainsi identifiées, jusqu'au point où l'un ou l'autre des sous-segments ne peut plus être redécoupé[5]. Ceci donne lieu à une procédure d'alignement que nous désignons sous le terme d'*alignement compositionnel*.

Il est aisé de voir que les paires de segments $\langle s_{i_1}...s_{i_2}, t_{j_1}...t_{j_2}\rangle$ mises en correspondance par une telle procédure peuvent être organisées en arbre. On obtient alors une analyse de traduction arborescente, du genre présenté à la figure 3.2, et similaire à celles produites par les ITG, à la différence que les découpages de la procédure ci-dessus n'ont aucune prétention syntaxique.

Bien entendu, laissée à elle-même, cette procédure est condamnée à se heurter de plein fouet aux problèmes dont on a discuté à la section 3.4 : en poussant la logique compositionnelle jusqu'au mot, on bute sur les phénomènes de non-compositionnalité, et les alignements produits souffrent alors d'aberrations aussi graves que celles engendrées par l'alignement IBM classique.

Le cadre du repérage de traduction nous permet toutefois d'éluder cette difficulté, du moins dans la mesure où l'on fait la supposition que la compositionnalité s'applique au moins jusqu'au niveau du point de mire. Sous cette hypothèse, on peut adapter la procédure d'alignement compositionnel au problème du RT. Soit une paire de phrases $\langle S, T\rangle$ et un point de mire $s_{i_1}...s_{i_2}$, cette adaptation procède suivant les modifications suivantes :

1. On interdit les découpages à l'intérieur du point de mire, c'est-à-dire avec $i_1 \leq i \leq i_2$;

[5] En pratique, avec un modèle IBM, on peut admettre des découpages donnant lieu à un segment vide dans la cible, c'est-à-dire une valeur de $J < 1$ ou $J > n$: ceci force alors des alignements nuls pour les mots du segment-source correspondant; on ne peut toutefois pas admettre de segments vides dans la source, parce que les modèles IBM ne nous permettent pas d'estimer $\Pr(\epsilon|t)$.

2. à chaque niveau de l'alignement, on ne considère que la paire de segments mise en correspondance par le niveau précédent qui inclut le point de mire;

3. la procédure termine lorsqu'il n'est plus possible de découper le segment contenant le point de mire; la paire de segments en question constitue alors le repérage final.

Ceci donne donc lieu à une méthode que nous appelons *RT compositionnel*, ou *RT C*. La figure 3.12 donne un exemple d'un tel repérage (les mots en italique représentent le point de mire).

Comme c'était le cas pour la méthode de RT Viterbi-contigu, on peut introduire des connaissances linguistiques dans le processus de deux façons simples :

- *RT compositionnel-contraint* (*RT CC*) : On force les points de coupe i et j à coïncider avec des frontières entre tronçons.

- *RT compositionnel-tronçons* (*RT CT*) : On base le calcul des alignements sur un modèle de tronçons, ce qui implique ici aussi que les points de coupe i et j coïncident avec des frontières de tronçon.

Si on examine la complexité des méthodes compositionnelles, on constate qu'au premier niveau d'analyse, le calcul des points de coupe optimaux suivant l'équation (3.10) nécessite l'examen de $(m-1)n$ paires $\langle i, j\rangle$: i peut prendre les valeurs $1...m-1$, alors que j peut prendre les valeurs $0...n-1$, puisqu'on permet un découpage engendrant un segment-cible vide ($j = 0$ par convention). Chaque découpage entraîne lui-même le calcul de quatre alignements Viterbi, soient deux pour $d = 1$ et deux pour $d = -1$. Ensemble, chaque paire d'alignement couvre la totalité de la paire $\langle S, T\rangle$. On peut donc considérer que le premier niveau d'analyse implique un nombre d'opérations dans $\mathcal{O}(m^2n^2)$.

niveau 1	Let us see	$\longleftrightarrow$	Voyons
	where *the government 's commitment* is really at in terms of the farm community	$\longleftrightarrow$	quel est le véritable engagement du gouvernement envers la communauté agricole
niveau 2	where *the government 's commitment* is really at	$\longleftrightarrow$	quel est le véritable engagement du gouvernement
	in terms of the farm community	$\longleftrightarrow$	envers la communauté agricole
niveau 3	where	$\longleftrightarrow$	quel
	the government 's commitment is really at	$\longleftrightarrow$	est le véritable engagement du gouvernement
niveau 4	*the government 's commitment* is really	$\longleftrightarrow$	est le véritable engagement du gouvernement
	at	$\longleftrightarrow$	
niveau 5	*the government 's commitment*	$\longleftrightarrow$	engagement du gouvernement
	is really	$\longleftrightarrow$	est le véritable

Figure 3.12. Exemple du processus de RT compositionnel

Pour les niveaux suivants, il faut considérer le pire cas et le cas moyen. Dans le pire des cas, les découpages optimaux mettent en correspondance la totalité du segment-cible T avec des séquences de S dont la longueur diminue d'un seul mot à chaque niveau. Donc, le premier découpage se fait dans $\mathcal{O}(m^2n^2)$, le suivant dans $\mathcal{O}((m-1)^2n^2)$, et ainsi de suite jusqu'au point où la séquence-source est réduite au point de mire, de taille I. Globalement, et sous l'hypothèse que la longueur du point de mire est négligeable par rapport à celle de la phrase S, l'opération se fera dans $\mathcal{O}(m^3n^2)$.

Dans le cas moyen, on s'attend plutôt à ce que la taille des segments considérés diminue de moitié à chaque niveau. Pour simplifier, considérons que $m = n + 1$ et que n est une puissance de 2. Au premier niveau, on aura donc $(m-1)n = n^2$ paires $\langle i, j\rangle$ à considérer, au second $n^2/4$, au troisième $n^2/16$, et ainsi de suite. La recherche s'arrêtera à une profondeur $\log(n)$. Donc, le nombre total d'alignements Viterbi à effectuer sera :

$$\begin{aligned} 4(n^2 + \frac{n^2}{4} + \frac{n^2}{16} + ... + \frac{n^2}{n^2}) &= 4n^2 \sum_{k=0}^{\log(n)} (\frac{1}{4})^k \\ &= 4n^2 \frac{(1 - \frac{1}{4}^{\log(n)})}{(1 - \frac{1}{4})} \\ &= \frac{16}{3}(n^2 - 1) \in \mathcal{O}(n^2) \end{aligned}$$

En somme, on s'attend à une complexité moyenne dans $\mathcal{O}(m^2n^2)$ pour la méthode RT C.

Comme c'était le cas pour les méthodes contiguës (RT VC, RT VTC, RT VCC), on peut améliorer cette situation en ne considérant que des segmentations qui donnent lieu à des paires de segments de longueurs comparables. Par exemple, pour un point de coupe source i donné, on peut se limiter aux points de coupe cible j variant entre $i - K$ et $i + K$ d'une part, et $(n - i) - K$ et $(n - i) + K$ d'autre part, pour une constante K donnée. Ceci permet de limiter le nombre de paires $\langle i, j\rangle$ à examiner à

un facteur constant de m, ce qui réduit d'un ordre de grandeur la complexité de la méthode.

Le précalcul des alignements Viterbi permet ici aussi de gagner un peu de temps, quoique ce gain soit possiblement moins marqué que pour les méthodes dites contiguës. Ceci est dû au fait que chaque paire d'alignements Viterbi dans l'équation (3.10) concerne en moyenne des paires de segments de tailles à peu près égales; de ce fait, dans la moitié des cas en moyenne, l'alignement optimal pour un point de la source ne sera pas le même que pour l'alignement Viterbi normal. Malgré tout, sans changer la complexité de l'algorithme, on peut s'attendre à ce que le précalcul des alignements Viterbi permette de diminuer de moitié les temps d'exécution du RT C.

Pour la méthode RT CC, on n'examine que les points de coupe i et j qui coïncident avec des frontières entre tronçons. Si on compare à la méthode RT C, ceci divise le nombre d'alignements à calculer à chaque niveau par un facteur de C^2, ce qui mène à une complexité moyenne dans $\mathcal{O}(m^2n^2/C^2)$.

Il en va de même pour la méthode RT CT, sauf si on précalcule les alignements Viterbi, et de ce fait les valeurs des paramètres t_c et a_c pertinents (un investissement initial dans $\mathcal{O}(mn)$) : le coût des alignements à chaque niveau est alors réduit lui aussi par un facteur de C^2, pour une complexité totale dans $\mathcal{O}(m^2n^2/C^4)$.

3.6 Implantation des méthodes de RT

Nous avons proposé à la section 3.5 huit méthodes de repérage de traduction, basées sur la notion d'alignement Viterbi dans les modèles de traduction statistiques IBM :

- RT Viterbi (RT V)
- RT Viterbi-tronçons (RT VT)
- RT Viterbi-contigu (RT VC)

- RT Viterbi-tronçons-contigu (RT VTC)
- RT Viterbi-contigu-contraint (RT VCC)
- RT compositionnel (RT C)
- RT compositionnel-tronçons (RT CT)
- RT compositionnel-contraint (RT CC)

Les méthodes RT V et RT VT produisent occasionnellement des repérages non contigus, ce qui nous a mené à proposer également différents post-traitements visant à identifier une séquence contiguë à partir du repérage proposé :

- Tolérance-zéro (Z)
- Séquence couvrante (SC)
- Meilleure séquence contiguë (M)

Le tableau 3.1 ci-dessous résume les caractéristiques de ces différentes procédures en terme de complexité (m et n sont les tailles des phrases source et cible, I est la taille du point de mire, et C est la taille moyenne d'un tronçon).

Ces méthodes de RT et post-traitements ont fait l'objet d'une implantation, dont l'élément central est un programme appelé *tsalign*, qui prend en entrée des paires de phrases source et cible $\langle S, T \rangle$, dans lesquelles une sous-séquence $s = s_{i_1}..s_{i_2}$ de S est identifiée comme point de mire; pour chaque paire de phrases, le programme produit en sortie la sous-séquence de $t = t_{j_1}..t_{j_2}$ de T qu'il juge est la "meilleure traduction" de s. Pour ce faire, *tsalign* peut utiliser l'une ou l'autre des méthodes de repérage de la section 3.5.

méthode	complexité moyenne
RT V	$\mathcal{O}(In)$
RT VT	$\mathcal{O}(In)$
RT VC	$\mathcal{O}(mn^2)$
RT VCC	$\mathcal{O}(mn^2/C^2)$
RT VTC	$\mathcal{O}(mn^2/C^4)$
RT C	$\mathcal{O}(m^2n^2)$
RT CC	$\mathcal{O}(m^2n^2/C^2)$
RT CT	$\mathcal{O}(m^2n^2/C^4)$

Table 3.1. Complexité moyenne des méthodes de RT (impliquant les heuristiques le cas échéant)

L'implantation de *tsalign* est faite en C++. Bien qu'on ait tenu compte des aspects de complexité présentés à la section précédente, elle n'a pas été faite avec un grand souci d'efficacité. En particulier, plutôt que de charger en mémoire vive les paramètres des modèles de traduction, nous avons choisi une méthode de stockage externe (tables de hachage et *B-trees* de la librarie *Berkeley DB*). Cette façon de faire, bien que relativement coûteuse en accès-disque à l'exécution, nous permet notamment de faire fonctionner les programmes même sur des ordinateurs dont la capacité en mémoire vive est limitée. Elle permet également à d'autres programmes, écrits en différents langages (C, C++, Perl, etc.) d'utiliser les mêmes données de façon relativement directe.

Les méthodes RT VT et RT VTC reposent sur l'existence d'un tronçonnage des textes source et cible. Les tronçonneuses utilisées reposent sur la tronçonneuse *Texas*, décrite à la section 2.5.1. Ici encore, nous avons employé pour l'anglais une tronçonneuse superposant trois modèles de Markov cachés, entraînés sur un extrait du *Wall Street Journal* annoté à la main [100].

Nous avons par ailleurs dû développer une tronçonneuse pour le français, essentiellement similaire à celle de l'anglais, si ce n'est que son entraînement a été effectué sur un extrait du corpus *Corfrans* [2], soit environs 2000 phrases du journal *Le Monde*, enrichies d'étiquettes morpho-syntaxiques et d'arbres d'analyse de constituant, à la façon du *Penn Treebank*.

Afin de tirer un tronçonnage de ce corpus, nous avons adapté la méthode du programme *chunklink* [20], qui effectuait cette tâche pour le *Penn Treebank*. Essentiellement, ce programme prend en entrée un arbre de constituants, élimine certains types de noeuds (notamment les groupes adjectivaux AP qui se trouvent à l'intérieur de groupes nominaux NP, et les groupes verbaux VN à l'intérieur de groupes infinitifs $VPinf$); pour ensuite "aplatir" la structure restante, en juxtaposant les noeuds fils à leur père. En dernière étape, on élimine tous les noeuds qui ne sont pas pertinents pour le tronçonnage. Dans notre cas, nous n'avons conservé que les constituants de type NP (groupe nominal), VN (groupe verbal), $VPinf$ (groupe infinitif) et PP (groupe prépositionnel). La figure 3.13 donne un exemple de ce processus.

Pour ce qui est des modèles de traduction statistiques, nous avons utilisé la trousse du domaine public $EGYPT$ [5]. Cette trousse, développée lors d'un atelier sur la traduction automatique tenu à Johns Hopkins University à l'été 1999, se veut une implantation des procédures d'entraînement des modèles de traduction IBM 1-4. Le programme *GIZA*, qui se charge de l'entraînement des modèles, produit les paramètres obtenus dans des fichiers en format-texte. De simples scripts Perl ont alors été déployés pour verser le contenu de ces fichiers dans des bases de données *BerkeleyDB*, tel que requis par nos programmes de RT.

Ces modèles ont été entraînés sur un corpus de textes bilingues (français-anglais) extraits du Hansard, c'est-à-dire la transcription des débats de la Chambre des communes, à Ottawa. Notre corpus est constitué de toutes les parutions du Hansard, d'avril 1986 à janvier 2002 (il s'agit du même corpus utilisé pour les expériences décrites à la section 2.6). Ce corpus a été segmenté en phrases et aligné automatique-

Entrée :

$[_S$ $[_{VN}$ nous avons]
$[_{NP}$ les meilleurs etudiants
$[_{COORD}$ et
$[_{NP}$ le
$[_{AP}$ plus beau]
campus
$[_{PP}$ du
$[_{NP}$ monde]]]]] .]

Phase 1 : Élimination de *AP* sous *NP* et de *VN* sous *VPinf.*

$[_S$ $[_{VN}$ nous avons]
$[_{NP}$ les meilleurs etudiants
$[_{COORD}$ et
$[_{NP}$ le *plus beau* campus
$[_{PP}$ du
$[_{NP}$ monde]]]]] .]

Phase 2 : Aplatissage sous *S*

$[_S$ $[_{VN}$ nous avons]
$[_{NP}$ les meilleurs etudiants]
$[_{COORD}$ et]
$[_{NP}$ le plus beau campus]
$[_{PP}$ du]
$[_{NP}$ monde]
.]

Phase 3 : Élimination de tous les parenthésages autres que *NP*, *PP*, *VN* et *VPinf* :

$[_{VN}$ nous avons]
$[_{NP}$ les meilleurs etudiants]
et
$[_{NP}$ le plus beau campus]
$[_{PP}$ du]
$[_{NP}$ monde]
.

Figure 3.13. Transformation d'un arbre de constituant en tronçonnage

ment au niveau des phrases, en utilisant le programme *SFIAL*, une version quelque peu améliorée de la méthode décrite dans [102]. Le corpus d'entraînement des modèles statistiques de traduction est constitué de 1 200 000 paires de phrases, choisies au hasard parmi l'ensemble du corpus (soit un peu plus de 20% du corpus). Bien que les techniques de RT décrites à la section 3.5 reposent sur le modèle IBM-2, nous avons en fait utilisé les paramètres obtenus d'un modèle IBM-3, obtenu après 10 itérations de la procédure EM sur chacun des modèles 1, 2 et 3.

3.7 Évaluation

Afin d'évaluer et de comparer entre elles les différentes méthodes de RT proposées ci-dessus, nous avons mis en place un protocole d'évaluation faisant intervenir des textes dans lesquels des repérages ont préalablement été effectués à la main. Nous décrivons dans cette section ce corpus de test, la nature des expériences que nous avons effectuées, la méthode d'évaluation et les résultats obtenus.

3.7.1 Corpus de test

Nous avons mis sur pied un corpus de test pour les méthodes de RT. Il s'agit d'un ensemble de couples $\langle S, T \rangle$; dans chacun de ces couples, un point de mire est spécifié dans la partie source, et le repérage correspondant est identifié dans la partie-cible.

Les couples du corpus de test ont tous été extraits du même corpus qui a servi pour l'entraînement des modèles de traduction IBM, soit une mémoire de traduction regroupant l'ensemble des éditions du Hansard parues entre avril 1986 et janvier 2002, alignées au niveau des phrases[6]. Notons que du point de vue méthodologique, et contrairement à ce qui se produit dans un contexte de traduction automatique, il n'y a pas d'objection à tester les méthodes de RT sur les mêmes données qui ont

[6] Rappelons toutefois que les modèles de traduction n'ont pas été entraînés sur la totalité de ces données; voir la section 3.6

servi à entraîner les modèles de traduction : au contraire, entraîner les modèles de traduction à même le contenu de la mémoire de traduction constitue une utilisation tout à fait normale de données qui sont disponibles pour le système. Évidemment, rien n'empêche d'utiliser un ensemble de données distinct pour l'entraînement, mais *a priori*, il n'y a pas d'avantage à le faire, à moins que la taille de la mémoire de traduction soit insuffisante pour assurer une évaluation fiable des paramètres (auquel cas, on voudrait sans doute combiner les données de la MT au corpus d'entraînement).

Par contre, pour l'identification des points de mire, nous avons eu recours à une paire de documents distincte du corpus d'entraînement, soit une parution du Hansard plus récente (mars 2002 – il s'agit du même document-test qui a été utilisé dans les expériences décrites à la section 2.6). Suivant la méthode décrite à la section 2.5, nous avons soumis la partie anglaise (source) de ce document à un tronçonnage, et avons recherché dans la mémoire de traduction toutes les séquences de tronçons d'au moins 3 mots. Parmi toutes les séquences retrouvées dans la MT, nous en avons choisi 100 au hasard. La figure 3.14 donne quelques exemples de ces points de mire.

Pour chacune de ces séquences, nous n'avons retenu que les 100 premières occurrences trouvées dans la MT. Chacune de ces occurrences prenait donc la forme d'un couple, le plus souvent une paire de phrases, dans laquelle le point de mire était identifié au moyen de balises. Dans chacun de ces couples, nous avons alors localisé manuellement le repérage-cible, c'est-à-dire l'ensemble de mots de la cible (partie française) qui rendait le mieux la traduction du point de mire. Cette opération a été faite suivant les normes proposées par Véronis pour l'annotation manuelle des repérages [109], normes elles-mêmes inspirées d'un guide similaire produit dans le cadre du projet Blinker [73]. (voir la figure 3.15).

Le corpus de test regroupe en tout 4100 couples. Le tableau 3.2 résume les principales caractéristiques de celui-ci.

a brief overview of
all those in
and a growing gap
and a more
and other interested
and the second highest corporate taxes in
as I mentioned at the beginning of
asked for information about
BQ) : Mr. Speaker , is it
Byrne , Hon. Gerry
close to 40 years
commend the government and
complaints about the committee report
creating more jobs

Figure 3.14. Exemples de points de mire du corpus de test

```
<couple>
 <part lang="en">
  <s>
   Let us see where <match> the government 's commitment </match>
   is really at in terms of the farm community .
  </s>
 </part>
 <part lang="fr">
  <s>
   Voyons quel est <match> le </match> véritable <match>
   engagement du gouvernement </match> envers la communauté
   agricole .
  </s>
 </part>
</couple>
```

Figure 3.15. Exemple de couple du corpus de test

couples:	4100	
mots :		
source :	120 731	
cible :	132 565	
points de mire:	100	
long. min. :	3	
long. max. :	13	
long. moyenne :	3,85	
repérages-cible:	4100	
long. min. :	0	
long. max. :	13	
long. moyenne :	3,50	
contigus :	3741	(91,24%)
vides :	235	(5,73%)

Table 3.2. Caractéristiques du corpus de test

3.7.2 Métriques d'évaluation

Une fois munis d'un corpus de test annoté à la main, il était possible de soumettre celui-ci à notre programme de RT, de façon à en évaluer objectivement la performance sous différentes conditions. Pour une telle évaluation, on peut évidemment calculer la proportion des repérages produits qui sont identiques aux repérages de référence. Ceci donne lieu à une statistique que nous appelons *exactitude*.

Toutefois, dans une application comme la nôtre, même des repérages qui ne sont pas "parfaits" peuvent s'avérer utiles, dans la mesure de leur ressemblance avec les repérages de référence. C'est pourquoi nous avons également retenu l'ensemble des métriques proposées pour le projet ARCADE [112], suivant une proposition initiale de Isabelle et Simard [49].

Dans ce cadre, les repérages obtenus sont comparés aux repérages de référence en termes de *rappel* et *précision*. Soient une paire de phrases $\langle S, T \rangle$, le repérage de référence $\langle s_r, t_r \rangle$ et le repérage $\langle s, t \rangle$ obtenu, on calcule :

$$\text{rappel} = \frac{|t \cap t_r|}{|t_r|} \tag{3.11}$$

$$\text{précision} = \frac{|t \cap t_r|}{|t|} \tag{3.12}$$

L'une et l'autre de ces métriques sont donc calculées en termes de mots. On produit également une métrique combinant ces deux mesures, soit la *F-measure* [107] :

$$\begin{aligned} F &= 2\,\frac{\text{précision} \times \text{rappel}}{\text{précision} + \text{rappel}} \\ &= 2\,\frac{|t \cap t_r|}{|t| + |t_r|} \end{aligned} \tag{3.13}$$

Pour les fins de ces calculs, on considère qu'un repérage vide contient en fait un mot-nul fictif. D'une part, ceci s'accorde bien avec les modèles IBM utilisés, qui font

la même supposition; d'autre part, ça permet de traiter adéquatement le problème des divisions par zéro.

Toutes ces mesures sont calculées individuellement sur chaque couple $\langle S, T \rangle$, et les résultats rapportés ci-dessous représentent des mesures moyennes. Cette façon de faire permet de faire abstraction de la taille des repérages. Elle tient toutefois explicitement compte des fréquences absolues des points de mire, c'est-à-dire qu'un point de mire plus "productif" (qui extrait un plus grand nombre de couples de la mémoire de traduction) aura un poids d'autant plus élévé dans les mesures globales.

3.7.3 Expériences

Nous avons soumis le corpus de test décrit ci-dessus au différentes méthodes de RT proposées à la section 3.5. Les résultats de ces expériences apparaissent au tableau 3.3. Dans ce tableau, comme dans les suivants, la méthode RT V est notée simplement "V", et similairement pour les autres méthodes; les résultats en gras indiquent par ailleurs le meilleur résultat obtenu selon chaque métrique.

	Méthode							
métrique	V	VT	VC	VCC	VTC	C	CC	CT
exactitude	0,17	0,16	0,36	0,35	0,32	**0,40**	0,33	0,15
précision	0,60	0,54	**0,75**	0,68	0,69	0,72	0,67	0,64
rappel	0,57	0,61	0,66	0,69	0,64	0,70	**0,72**	0,54
F	0,57	0,55	0,68	0,66	0,64	**0,69**	0,67	0,55
temps d'exécution (sec.)	300	319	303	368	320	1539	803	519

Table 3.3. Résultats des expériences de RT

Alors que les méthodes *contiguës* et *compositionnelles* intègrent naturellement une contrainte de contiguïté sur le repérage-cible, les méthodes RT V et RT VT

produisent à l'occasion des repérages non contigus. Pour chacune de ces méthodes, nous avons également mesuré l'effet des post-traitements *tolérance-zéro* (Z), *séquence couvrante* (SC) et *meilleure séquence contiguë* (M), visant à produire des repérages contigus dans ces cas. Le tableau 3.4 présente les résultats de ces expériences (ici, *V+Z* désigne "la méthode de RT V, suivie du post-traitement Z", et ainsi de suite).

	Méthode					
métrique	V+Z	V+SC	V+M	VT+Z	VT+SC	VT+M
exactitude	0,20	**0,26**	0,03	0,19	0,21	0,08
précision	0,28	0,51	0,63	0,34	0,51	**0,67**
rappel	0,28	0,71	0,20	0,36	**0,72**	0,43
F	0,28	0,55	0,29	0,34	**0,56**	0,50

Table 3.4. Résultats des post-traitements

Le post-traitement *Z* réduit à un repérage-cible vide tout repérage non contigu. On a tenu compte de ces repérages-cible vides dans le calcul des différentes métriques dans le tableau 3.4; c'est ce qui explique notamment les faibles performances observées avec ce post-traitement. En pratique, toutefois, l'intention visée est de filtrer les repérages non contigus, sous l'hypothèse qu'ils ne sont pas réellement susceptibles d'être utiles à un traducteur. Pour cette raison, nous avons jugé qu'il était pertinent d'évaluer la performance de ces méthodes sur les seuls couples où elles produisaient un repérage non vide. Les résultats de ces calculs apparaissent au tableau 3.5. On y mentionne également le pourcentage des couples du corpus de test ayant mené à des repérages non vides, ce qui donne une idée de la perte engendrée par cette stratégie. Évidemment, il arrive aussi à l'occasion que les autres méthodes présentées ici produisent des repérages vides; mais en pratique, c'est très rare : exclusion faite de RT V et RT VT, toutes les méthodes présentées ici produisent des repérages non vides dans plus de 98,7% des cas.

	Méthode	
métrique	V+Z	VT+Z
% du corpus	28,20%	43,27%
exactitude	**0,56**	0,35
précision	**0,83**	0,70
rappel	**0,82**	0,76
F	**0,81**	0,70

Table 3.5. Performance des méthodes de RT Viterbi, excluant les repérages-cible vides

3.7.4 Discussion

Globalement, si on prend pour critère d'évaluation la seule exactitude des repérages, c'est la méthode de RT C qui présente les meilleurs résultats, avec 40% de repérages corrects, ce qui représente un gain de plus de 135% par rapport au repérage Viterbi (tableau 3.3); en terme de *F-measure* le gain est de plus de 20%. De tels gains sont impressionants lorsqu'on considère que les deux méthodes (RT V et RT C) utilisent exactement les mêmes données.

Du point de vue du rappel, la méthode RT CC surclasse quelque peu le RT C : ceci peut s'expliquer par le fait que le RT CC tend naturellement à produire des repérages plus grands, tant sur la source que sur la cible. Ce gain se fait toutefois au prix d'une perte assez importante (5%) en précision.

La méthode RT VC est de loin la plus performante en terme de précision (0,75). Une comparaison avec la méthode RT V (précision $= 0,60$) démontre bien l'impact de la seule contrainte de contiguïté (ou l'absence d'une telle contrainte) pour la tâche de repérage de traduction. En fait, la performance globale de cette méthode est très proche de celle de la méthode RT C ($F = 0,68$), et dans le contexte de notre

application, il est fort probable qu'on souhaite favoriser la précision plutôt que le rappel dans les repérages, ce qui en ferait un bon choix.

D'une façon générale, l'intégration de connaissances linguistiques dans le processus de RT donne des résultats plutôt décevants, du moins de la façon dont elle a été effectuée ici. Globalement, notre implantation d'un modèle de traduction basé sur les tronçons tend à produire de moins bons repérages que le modèle de mots sur lequel il est basé. Pour les méthodes Viterbi (RT V et RT VT) la perte n'est pas spectaculaire, et on observe même un léger gain du point de vue du rappel. Mais pour les méthodes *contiguës*, il y a perte à tous les niveaux, et pour les méthodes *compositionnelles*, la perte est carrément catastrophique, puisque les repérages produits sont globalement moins bon que le Viterbi simple.

On constate par contre que l'exploitation des frontières entre tronçons pour limiter l'espace de recherche (méthodes RT VCC et RT CC), en plus d'un impact souvent non négligeable sur les temps de calcul, permet généralement des gains significatifs en rappel (2-3%). Dans un contexte où le rappel prime sur la précision, l'emploi de ce genre de mécanisme pourrait se justifier.

Lorsqu'on considère l'ensemble des couples du corpus de test, l'emploi des différents post-traitements proposés pour obtenir des repérages-cible contigus avec les méthodes RT V et RT VT donne des résultats plutôt mitigés (tableau 3.4). Il est intéressant de noter qu'en général, la méthode RT VT bénéficie plus de l'effet de ces post-traitements que la méthode RT V, qu'elle surclasse alors en rappel et précision. Le calcul de la plus petite séquence couvrante (SC) est particulièrement efficace, et permet au RT VT de rejoindre la meilleure méthode au titre du rappel (RT CC), pour un coût beaucoup moindre en temps de calcul. Mais en général, force est de constater qu'il est beaucoup plus productif d'incorporer la contrainte de contiguïté à même la procédure de RT que de l'imposer après coup.

Toutefois, la stratégie de *tolérance-zéro*, lorsqu'on l'utilise comme un filtre plutôt que comme un simple post-traitement, s'avère particulièrement efficace, surtout con-

jointement avec la méthode RT V, qui surclasse alors de loin toutes les autres méthodes proposées ($F = 0,81$, 56% de repérages corrects; tableau 3.5). Mais cette amélioration ne se fait qu'au prix de l'élimination d'une grande quantité de couples contenant des repérages potentiellement utilisables. À preuve, seuls 28% des repérages produits par la méthode RT V passent le test de la contiguïté, alors que la proportion réelle dans la référence excède 90% (tableau 3.2).

Pour comprendre cet écart, il faut revenir à la définition d'alignement dans les modèles IBM : celle-ci spécifie que chaque mot de la source est connecté au plus à un mot de la cible. Ceci a une conséquence très directe sur le résultat du RT V : le repérage-cible d'un point de mire de I mots comportera lui-même au maximum I mots. De ce fait, cette méthode aura systématiquement des problèmes si le repérage-cible réel est plus long que le repérage-source. Or, cette situation est très fréquente quand on aligne de l'anglais vers le français, comme c'est le cas ici. Considérons par exemple l'anglais *airport security*, qui est le plus souvent rendu en français par *sécurité dans les aéroports*. L'alignement Viterbi produira normalement des liens *airport* → *aéroport* et *security* → *sécurité*, et la séquence *dans les* sera forcément laissée de côté (ou récupérée accidentellement par des liens erronés partant d'autres mots de la source), laissant ainsi un repérage non contigu. Notons par ailleurs qu'en utilisant un modèle "inverse" pour faire le repérage, tel qu'évoqué brièvement à la section 3.5.1, on aura exactement le problème inverse, c'est-à-dire que les repérages-cible obtenus seront systématiquement plus grands ou égaux en longueur au point de mire. En moyenne, cette façon de faire serait peut-être avantageuse dans le cas de l'anglais et du français, mais ça ne constitue certainement pas une solution générale au problème.

Le fait d'effectuer l'alignement entre des tronçons plutôt qu'entre des mots permet de récupérer certaines de ces situations, comme en témoignent les résultats en terme de rappel du RT VT dans les tableaux 3.3 et 3.5. Mais ça ne règle pas le problème de façon entièrement satisfaisante, parce qu'il ne semble pas y avoir de bonne corres-

pondance entre les tronçons du français et de l'anglais. Par exemple, dans l'exemple du paragraphe présent, les tronçonnages obtenus sont

$[_{NP}$ airport security]

et

$[_{NP}$ sécurité] $[_{PP}$ dans] $[_{NP}$ les aéroports]

Par la force des choses, deux des tronçons du français seront laissés de côté dans le repérage-cible.

Le post-traitement SC, qui trouve la plus petite séquence contiguë du texte-cible recouvrant l'ensemble du repérage Viterbi, s'avère une solution efficace à ce problème (tableau 3.4), du moins dans les cas les plus simples, tels que l'exemple ci-dessus. Toutefois, l'intégration de contraintes à même la procédure de recherche, telle que proposée dans les méthodes dites contiguës et compositionnelles, s'avère beaucoup plus efficace, sans pour autant éliminer complètement le problème. Ceci s'explique en partie par le fait que ces méthodes reposent également sur des alignements de type IBM; lorsque les mots "surnuméraires" dans le repérage-cible se trouvent aux frontières de celui-ci, ces mots ne jouent aucun rôle dans le calcul de la probabilité de l'alignement, de telle sorte qu'il n'y a pas de raison particulière de favoriser leur inclusion dans le repérage. Considérons le couple suivant :

- These companies **indicated their support** for the government 's decision.

- Ces compagnies ont déclaré qu'elles appuyait la décision du gouvernement.

Dans la recherche d'un repérage pour la séquence *indicated their support*, il est fort probable qu'on favorise un alignement établissant des liens *indicated* → *déclaré* et *support* → *appuyaient*. De par la contrainte de contiguïté, la séquence *qu'elles* dans la cible va naturellement se trouver englobée dans le repérage, forçant possiblement

un lien *their* → *elles*. Toutefois, le seul mot de la source susceptible d'être lié à *ont* dans la source est le verbe *indicated*, mais celui-ci est déjà accaparé par *déclaré*. De ce fait, *ont* sera laissé de côté dans l'alignement final, et ainsi ne participera pas au calcul de la probabilité (équation (3.3)

Mentionnons finalement que du point de vue des temps d'exécution (tableau 3.3), les méthodes Viterbi (RT V et RT VT) et contiguës (RT VC, RT VCC et RT VTC) procèdent dans des temps à peu près similaires, soit un peu plus d'une dizaine de repérages à la seconde. En regard des aspects de complexité discutés à la section 3.5, ceci suggère que le temps consacré strictement à calculer les repérages dans ces méthodes est probablement négligeable par rapport au temps nécessaire pour accéder aux données des modèles de traduction, qui sont stockées sur support externe dans notre implantation, comme on l'a expliqué à la section 3.6. En revanche, la complexité des méthodes compositionnelles (RT C, RT CC et RT CT) est supérieure par au moins un ordre de grandeur, ce qui se traduit dans les faits par des temps de calcul appréciablement plus élevés.

En définitive, une analyse plus poussée serait sans doute nécessaire pour identifier plus précisément les aspects susceptibles d'améliorer les temps d'exécution.

3.8 Conclusions

Le problème du repérage de traduction consiste essentiellement à localiser dans une paire de segments traduction l'un de l'autre la traduction d'une suite de mots donnée. La résolution de ce sous-problème de l'analyse de traduction est fondamental pour la mise sur pied d'un système de mémoire de traduction capable d'opérer sous le niveau de la phrase. Toutefois, ce problème est rendu particulièrement difficile par la variété des procédés employés par les traducteurs humains pour rendre le sens d'un texte d'une langue à une autre.

Nous avons proposé différentes méthodes de RT, toutes basées sur l'utilisation de

modèles statistiques de traduction. D'après nos expériences, les meilleures de celles-ci peuvent produire des repérages dont la performance (mesurée en terme de *F-measure*) atteint environ 70%.

En pratique, certaines caractéristiques propres au contexte d'utilisation qui nous intéresse permettent d'imposer sur la tâche de repérage des contraintes qui s'avèrent bénéfiques d'une façon générale. L'introduction d'une contrainte de contiguïté du repérage-cible s'avère particulièrement profitable. Dans son incarnation la plus simple (élimination systématique des repérages non contigus), elle permet d'améliorer de plus de 40% la qualité des repérages (*F-measure*), quoiqu'au prix de l'élimination de plus de 70% des candidats. Une intégration plus fine de cette contrainte au processus de repérage permet néanmoins des gains excédant 20% sur l'ensemble des couples considérés.

L'ajout de connaissances linguistiques rudimentaires, sous la forme de tronçonnages des textes, ne s'est pas avéré très fructueux, sinon du point de vue de la complexité calculatoire des méthodes concernées. Au mieux, notre façon de faire permet occasionnellement de meilleurs alignements pour certaines catégories de mots (articles, pronoms et autres mots-outils), en forçant un rattachement au bon constituant. Mais en pratique, les gains observés en rappel sont souvent contrebalancés par des pertes comparables en précision. La qualité des tronçonnages est possiblement en cause ici : en particulier, notre tronçonneuse pour le français n'a pu être entraînée que sur une très petite quantité de données, et les découpages qu'elle produit sont parfois douteux. Par ailleurs, bien que l'idée d'un modèle de traduction basé sur les tronçons demeure justifiée, l'approximation des paramètres d'un tel modèle au moyen d'un modèle de mots ne semble pas permettre de gains notables.

Ceci ne veut pas dire que des connaissances sur les liens de traduction au niveau des séquences de mots ne seraient pas bénéfiques pour un modèle de traduction, au contraire. Toutefois, la nature exacte de ces séquences reste à déterminer, et les méthodes de RT proposées ici pourraient certainement bénéficier des résultats

de travaux dans ce domaine, comme par exemple l'identification d'équivalences entre des "suites non-compositionnelles" [72]. Dans la même veine, l'ajout de connaissances terminologiques pourrait également jouer un rôle [58].

Il est intéressant de constater que, outre des connaissances linguistiques dont l'apport est somme toute contestable, toutes les méthodes proposées ici reposent sur le même ensemble de connaissances, à savoir un modèle de traduction statistique IBM-2. Les écarts observés entre les repérages de probabilité maximale (*Viterbi*) et ceux obtenus au moyen de méthodes intégrant une certaine forme de compositionnalité (sous la forme de contraintes de contiguïté) mettent certainement en relief les lacunes de ces modèles, et l'importance de la notion de compositionnalité.

On sait toutefois que les modèles 2 ne sont pas le *nec plus ultra* de la traduction statistique, et on peut se demander comment les méthodes proposées se compareraient à des alignements obtenus avec des modèles 3, 4, 5, ou tout autre modèle du même genre. En effet, la plupart des modèles proposés depuis intègrent, sous une forme ou une autre, des concepts qui s'apparentent à la contiguïté telle que formulée ici. On n'a qu'à penser aux paramètres de *fertilité* des modèles 3 en montant, ou à la modélisation markovienne des probabilités d'alignement de Vogel *et al.* [108].

À cet égard, on peut quand même remarquer que les méthodes proposées ici sont toutes basées sur la recherche d'alignements dont la probabilité est maximale, sous différentes contraintes. Si on fait abstraction des questions de complexité et d'implantation, ces méthodes sont toutes indépendantes du modèle utilisé, en autant qu'on soit en mesure d'en extraire un estimé d'une telle probabilité. Par exemple, nos méthodes auraient tout aussi bien pu avoir recours aux paramètres d'un modèle 3, 4 ou 5, pour peu qu'on soit en mesure de calculer des alignements optimaux avec ces modèles. C'est principalement pour des raisons de complexité que nous nous en sommes tenus au modèle 2 dans nos expériences. Par ailleurs, considérant l'importance des gains obtenus dans ces conditions, il n'est pas déraisonnable de penser qu'on pourrait également obtenir des gains avec des modèles plus sophistiqués.

D'autre part, il est intéressant de noter que, dans la modélisation statistique de la traduction, l'alignement de probabilité maximale n'est pas le mot de la fin. Chaque alignement ne représente en fait qu'un dénouement possible parmi tant d'autres dans l'histoire d'une traduction. Ultimement, la probabilité d'une traduction se calcule sur l'ensemble des alignements possibles. Ceci suggère que, pour arriver à de meilleurs repérages, il serait peut-être utile de considérer plus d'un alignement, de façon à y identifier certaines tendances. C'est une piste à explorer.

Chapitre 4

SÉLECTION DES SEGMENTS-CIBLE PROPOSÉS PAR UNE MÉMOIRE DE TRADUCTION

4.1 Introduction

Dans ce chapitre, nous nous intéressons au problème de la sélection des segments de texte en langue-cible qui seront proposés à l'utilisateur d'un système de mémoire de traduction (SMT). Ce problème se pose de façon particulièrement critique dans le cas d'un SMT sous-phrastique tel que proposé au chapitre 2, parce que pour une phrase donnée en langue-source, un tel système peut repérer dans sa mémoire de traduction un très grand nombre de segments en langue-cible susceptibles d'être utiles au traducteur. Les lui présenter tous reviendrait à le submerger d'information. Il faut donc effectuer un choix parmi les segments repérés.

Pour résumer la situation, on suppose à ce stade qu'on dispose d'un mécanisme qui, étant donnée une nouvelle phrase-source à traduire, repère dans une mémoire de traduction des segments en langue-cible, qui constituent potentiellement des traductions réutilisables à des portions de la phrase-source, ce que nous appelons des *candidats-cible*. Ces candidats peuvent être très nombreux: même en se limitant à rechercher des traductions pour des segments d'au minimum 3 ou 4 mots de la phrase-source, il n'est pas rare de se retrouver avec plusieurs centaines de candidats par phrase. Par exemple, on retrouve à la figure 4.1 une phrase de 19 mots, pour laquelle 334 repérages distincts ont été extraits d'une mémoire de traduction.

Dans le contexte d'un système d'aide à la traduction tel que proposé ici, il est impensable que le traducteur examine toutes ces propositions avant de produire sa traduction. Si on souhaite que le système soit d'une quelconque utilité, on doit obli-

Phrase-source: The government is putting a $2.2 billion tax on Canada 's most vulnerable industry , the airline industry .

Ensembles de candidats:		(nombre)
The government:	le gouvernement gouvernement au gouvernement ...	48
The government is putting :	le gouvernement le gouvernement met gouvernement met ...	78
is putting :	met est propose ...	83
a $2.2 billion tax :	une taxe de 2,2 milliards impôt de 2,2 milliards	3
a $2.2 billion tax on :	taxe de 2,2 milliards de dollars à une taxe de 2,2 milliards	2
a $2.2 billion tax on Canada :	une taxe de 2,2 milliards	1
Canada 's most vulnerable industry :	plus vulnérable au Canada industrie la plus vulnérable	2
's most vulnerable industry :	plus vulnérable au industrie la plus vulnérable	2
, the airline industry :	, l' industrie aérienne , de l' industrie aérienne transport aérien ...	19
the airline industry :	l' industrie aérienne aérien l' industrie du transport aérien ...	43
	Total:	**334**

Figure 4.1. Exemples d'ensembles de candidats-cible

gatoirement limiter la quantité d'information présentée à l'utilisateur. Bien entendu, on voudra préserver les candidats qui sont le plus susceptibles d'être utiles au traducteur. En pratique, le nombre de propositions devrait être réglable par l'utilisateur lui-même, mais d'une façon générale, il semble raisonnable de supposer que la taille totale de la sélection (mesurée en nombre de mots) ne devrait pas excéder de beaucoup la taille du texte à traduire. (Elle pourrait, par exemple, s'exprimer comme un facteur de celle-ci.)

Dans la suite de ce chapitre, nous formalisons ce problème, à la section 4.2. Sur la base de cette formalisation, nous proposons à la section 4.3 différentes procédures de recherche adaptées au problème, reposant sur l'utilisation d'un modèle statistique de traduction. Partant des résultats d'une évaluation quantitative présentées en section 4.4, nous identifions certaines lacunes dans les méthodes proposées, et suggérons à la section 4.5 des alternatives à l'utilisation d'un modèle de traduction pour évaluer le potentiel des candidats. Dans la section 4.6 nous explorons comment combiner l'information provenant de différentes sources. Nous terminons avec une discussion sur les futures avenues possibles à la section 4.8.

4.2 Formalisation du problème

Soit une mémoire de traduction $\mathcal{M}$, constituée de paires de segments de texte $\langle S, T \rangle$ qui sont traduction l'un de l'autre. Tel que suggéré au chapitre 2, nous pouvons supposer que S et T sont chacun constitué d'une ou plusieurs phrases. Le segment S est en langue-source, alors que T est en langue-cible.

On suppose également qu'on dispose d'un mécanisme qui, étant donnée une nouvelle phrase en langue-source P que l'on souhaite traduire, recherche dans $\mathcal{M}$ les couples dont la partie source S contient une séquence de mots $w_i...w_j$ de P, par exemple, le genre de mécanisme décrit au chapitre 2, section 2.5. Dans la suite de l'exposé, nous notons p_{ij} cette séquence $w_i...w_j$ de P, ou simplement p lorsqu'il n'y a

pas de confusion possible.

On suppose enfin l'existence d'une procédure capable, pour la sous-séquence p, d'identifier dans le couple $\langle S, T \rangle$ une paire de sous-séquences $c_p = \langle s_p, t_p \rangle$, telle que:

- s_p est une sous-séquence de S;
- p est une sous-séquence de s_p;
- t_p est une sous-séquence de T;
- s_p et t_p sont traductions l'un de l'autre;
- s_p et t_p sont aussi petites que possible.

En somme, t_p est la sous-séquence de T qui rend le mieux la traduction de p dans le couple $\langle S, T \rangle$. Nous appelons un tel couple c_p un *repérage de traduction* pour le segment p ou, plus simplement, un *repérage*; s_p est le *repérage-source*, alors que t_p est le *repérage-cible*. Les différentes méthodes de repérage de traduction décrites au chapitre 3, section 3.5, produisent toutes ce genre de résultat.

Chacun des segments t_p ainsi identifié pour P constitue donc un candidat potentiel à proposer au traducteur, et nous appelons C_p l'ensemble de repérages $c_p = \langle s_p, t_p \rangle$ extraits de $\mathcal{M}$ pour le segment p de la phrase P. L'ensemble $C_P = \cup C_p$ constitue l'ensemble de tous les repérages extraits de $\mathcal{M}$ pour les sous-séquences p de la phrase P. Si on revient à la figure 4.1, P est la *phrase-source* au haut de la figure, les segments en langue-source dans la colonne de gauche sont les p, les "boîtes" de la colonne du centre contiennent les segments en langue-cible t_p des ensembles de repérages C_p, dont l'union constitue l'ensemble C_P.

Le problème de sélection des candidats consiste donc à identifier un sous-ensemble de C_P susceptible d'être utile au traducteur, mais dont la taille est limitée. Cette contrainte de taille est fondamentale dans la formulation de problème, et nous allons la spécifier davantage.

D'une certaine façon, le problème de la sélection est apparenté à celui de la génération dans les systèmes de traduction automatique. Il s'en distingue néanmoins sous deux aspects:

- Il n'est pas nécessaire de combiner les repérages-cible sélectionnés, afin de formuler une traduction cohérente de P, puisque cette tâche est laissée à l'utilisateur du système. En pratique, ça veut dire que nous n'avons pas à nous soucier de l'ordre dans lequel doivent être proposés les candidats, et surtout que nous n'avons pas à considérer explicitement les problèmes de grammaticalité, ou de *friction aux frontières* entre les segments t_p proposés. Bien entendu, on souhaiterait être en mesure de proposer des candidats qui s'harmonisent bien ensemble, mais ce n'est pas une indispensable.

- Il n'est pas nécessaire que les repérages c_p sélectionnés contiennent des traductions pour la totalité de la phrase P. Évidemment, on souhaite que les propositions du système en recouvrent une aussi grande proportion que possible: on voudrait éviter de sélectionner un sous-ensemble de C_P qui concentre toute son attention sur une zone spécifique de la phrase. Mais dépendemment du contenu de la mémoire de traduction et des contraintes ergonomiques liées à l'application, une couverture totale ne sera en général pas possible.

Ce dernier aspect suggère que la sélection finale devrait viser à couvrir une aussi grande proportion que possible de la phrase P. C'est ce que nous appelons, dans ce qui suit, la contrainte de *couverture-source optimale*. L'hypothèse sous-jacente est, bien entendu, qu'en maximisant la couverture du texte-source, on devrait se trouver à maximiser également celle du texte-cible.

Par ailleurs, et tel que discuté à la section 2.5.4, il semble qu'il ne soit pas souhaitable de proposer des segments de texte en langue-cible qui constituent des traductions de segments de la source qui se chevauchent. Si on revient à l'exemple

de la figure 4.1, on préférerait éviter de proposer à la fois des traductions pour les séquences "*a $2.2 billion tax on Canada*" et "*Canada 's most vulnerable industry*". Intuitivement, une telle sélection serait contre-productive, puisque l'utilisateur serait alors forcé soit de faire un choix entre les deux, soit d'avoir recours à des manipulations plus ou moins complexes visant à les combiner.

Par conséquent, nous proposons de formuler l'objectif de la sélection en terme de *couverture du texte-source*. On définira ainsi une couverture R_P pour la phrase P:

definition : Soit R_P, un sous-ensemble des candidats C_P extraits pour la phrase P; R_P est une *couverture* de C_P si et seulement si pour toutes les paires de candidats c_{p_1} et c_{p_2} de R_P, les segments p_1 et p_2 sont disjoints dans P.

On remarque que dans cette définition, la notion de couverture pour les repérages c_p concerne le segment p (ce qu'on appelle le *point de mire* au chapitre 3) plutôt que le repérage-source correspondant s_p. On a vu que ces deux segments ne sont pas nécessairement identiques puisque, dépendant de la procédure de repérage de traduction employée et du contexte du couple $\langle S, T \rangle$, il est parfois nécessaire d'étendre le repérage-source pour arriver à un repérage-cible cohérent (voir la discussion sur les *élargissements du repérage* à la section 3.1). A priori toutefois, il n'y a pas de raison de croire que le segment t_p d'un repérage pour lequel $s_p \neq p$ est de ce fait inutile pour le traducteur. C'est pourquoi nous exprimons la couverture en fonction de p plutôt que de s_p.

La figure 4.2 illustre un exemple de couverture sur les candidats de l'exemple de la figure 4.1. Comme on peut le voir dans cet exemple, en cherchant à effectuer une sélection qui consiste en une couverture de la phrase-source, nous allons naturellement nous trouver à imposer une limite sur la taille de la sélection, puisque chaque mot (plus précisément, chaque occurrence de mot) de la phrase P ne pourra intervenir dans plus d'un repérage de la sélection.

R_P:			
c_1	=	⟨ “The government is putting”,	“le gouvernement met” ⟩
c_2	=	⟨ “a $2.2 billion tax”,	“une taxe de 2,2 milliards” ⟩
c_3	=	⟨ “Canada ’s most vulnerable industry”,	“industrie la plus vulnérable” ⟩
c_4	=	⟨ “, the airline industry”,	“, l’ industrie aérienne” ⟩

Figure 4.2. Exemple de couverture

Dans la suite de ce chapitre, nous allons donc examiner des méthodes de sélection des candidats qui recherchent effectivement des couvertures de la phrase à traduire, satisfaisant à certaines contraintes additionnelles.

4.3 Procédures de sélection

Malgré les différences qui existent entre le problème de la sélection des candidats et celui de la génération en traduction automatique, le parallèle entre les deux suggère qu’on peut s’inspirer des méthodes développées pour l’une afin d’aborder l’autre. Dans cette perspective, les approches probabilistes en TA sont particulièrement intéressantes, puisqu’elles visent à trouver la séquence de mots la plus probable en langue-cible, étant donnée une séquence de mots en langue-source. En pratique dans ces systèmes de TA, une recherche exhaustive de l’espace des solutions n’est pas possible: théoriquement, on devrait examiner une infinité de combinaisons de mots. Même en limitant la longueur des phrases en langue-cible, un examen exhaustif des traductions possibles est impensable, et ne serait même pas souhaitable, parce que la capacité des modèles statistiques existants à départager les bonnes traductions des mauvaises est encore très limitée.

Pour toutes ces raisons, la plupart des *décodeurs* existants (procédure de génération des systèmes de TA statistique) vont avoir recours à diverses heuristiques: par exemple, ils vont limiter le vocabulaire-cible actif, afin de se concentrer sur les

mots les plus susceptibles d'apparaître dans la traduction; la recherche de la solution globale procèdera le plus souvent par combinaison de solutions locales, ce qui permet l'utilisation de méthodes de recherche efficaces (programmation dynamique, recherche vorace, etc.); souvent, l'ordre des mots en langue-cible sera calqué sur celui de la langue-source; etc.

Le problème de la sélection des candidats peut également être formulé en termes probabilistes: l'objectif revient alors à trouver le sous-ensemble des repérages-candidats qui maximise la probabilité que les segments-cible qu'il contient soient réutilisés par le traducteur. Si, comme on l'a suggéré à la section précédente, nous concentrons cette recherche sur les sous-ensembles qui constituent une couverture de la phrase-source, une telle approche probabiliste reposerait donc sur une distribution statistique des couvertures R_P, étant donnés différents facteurs, au nombre desquels on compterait, par exemple:

- P: la phrase-source à traduire;
- D: l'ensemble du texte d'où est issue P, de façon à tenir compte de certaines caractéristiques propres à celui-ci;
- $T_{P'}$: la traduction des autres phrases P' de D, déjà effectuées par le traducteur, de façon à tenir compte de certains choix lexicaux ou terminologiques, etc.;
- H: le traducteur, afin de tenir compte de certaines préférences qui lui sont propres.

Ainsi la probabilité que l'utilisateur préfère une couverture de candidats-cible s'écrirait:

$$\Pr(R_P|P, D, T_{P'}, H)$$

D'un point de vue pratique, une telle modélisation du problème n'est pas plus réaliste pour le problème de la sélection qu'elle ne l'est pour la traduction automatique. On aura donc recours au même genre de simplifications qui ont cours en TA statistique, notamment des hypothèses d'indépendance entre R_P, D, H et les $T_{P'}$. De la même façon, on voudrait supposer que les candidats-cible c_p qui constituent une couverture R_P sont indépendants les uns des autres. On aurait ainsi:

$$\Pr(R_P|P) = \prod_{c_p \in R_P} \Pr(c_p|P)$$

Par ailleurs, on aimerait également supposer que la probabilité qu'un candidat c_p soit utile au traducteur dépend principalement de la relation qui existe entre le segment en langue-cible t_p et la séquence de mots p de P dont elle est possiblement la traduction, alors que le repérage-source s_p et le reste de la phrase P ne jouent aucun rôle mesurable:

$$\Pr(c_p|P) \approx \Pr(t_p|p)$$

On constate que cette formulation nous ramène à une distribution essentiellement identique à celles qu'on retrouve dans les sytèmes de TA statistiques basés sur un modèle dit *de canal bruité*, par exemple les modèles IBM de Brown *et al.* [15] dont il a été question au chapitre 3. Intuitivement d'ailleurs, rien ne permet de penser que la probabilité qu'un segment de phrase soit utile à un traducteur humain soit substantiellement différente de la probabilité que ce même segment apparaisse dans une traduction existante. Nous proposons donc d'utiliser de tels modèles pour estimer cette distribution.

Rappelons que Brown *et al.* proposent 5 modèles de complexité croissante; dans tous ces modèles, la probabilité d'observer une séquence $f = f_1...f_m$ en langue-cible, étant donnée une séquence $e = e_1...e_l$ en langue-source, s'écrit[1]:

[1] Dans la présentation de Brown *et al.* f désigne en fait le segment en langue-source (en français)

$$\Pr(f|e) = \sum_{a} \Pr(f, a|e) \tag{4.1}$$

où a est un alignement entre les mots de f et e, c'est-à-dire une suite d'entiers $a = a_1...a_m$ qui dénote, pour chaque mot f_j du segment en langue-cible, le mot e_{a_j} en langue-source qui en est à l'origine dans la traduction. On admet conventionnellement l'existence d'un mot *nul* e_0 dans la source, qui permet d'expliquer les mots de la cible qui apparaissent spontanément dans la cible; ces mots de la cible auront alors une composante d'alignement $a_j = 0$. Le calcul de $\Pr(f|e)$ se fait donc en examinant tous les alignements possibles entre les mots de f et de e.

Dans les modèles IBM, les segments de texte auquels les variables f et e font référence dans l'équation (4.1) sont des phrases. Les différents modèles proposés formulent différentes hypothèses d'indépendance, qui permettent une décomposition du terme $\Pr(f, a|e)$ en termes plus maniables. Typiquement, cette décomposition est faite de façon à considérer séparément deux types de distributions:

- une distribution *lexicale*, c'est-à-dire qui s'intéresse à la probabilité d'observer un mot f_j de la langue-cible étant donné le mot qui est à son origine e_{a_j} dans la langue-source; dans les modèles IBM, cette distribution est estimée via un ensemble de paramètres $t(e|f)$, qui représente une approximation de $\Pr(f|e)$.

- une distribution sur les alignements, qui s'intéresse aux mouvements des mots dans le passage d'une langue à l'autre. Dépendant des modèles, celle-ci peut prendre différentes formes, mais dans son incarnation la plus simple (modèle 2), elle est rendue par un ensemble de paramètres $a(a_j|j, l, m)$ qui approximent la probabilité qu'un mot en position j dans une phrase de l mots donne lieu à un mot en position a_j dans une phrase de m mots.

---

et e le segment en langue-cible (en anglais); mais cette subtilité est sans importance pour notre exposé.

Si on compte utiliser un tel modèle pour l'estimation de $\Pr(t_p|p)$ dans les procédures de sélection, les distributions d'alignement posent un problème: en effet, dans la grande majorité des cas, t_p et p ne sont pas des phrases complètes; qui plus est, elle proviennent de phrases qui ne sont pas des traductions l'une de l'autre. Il n'y a donc pas lieu de penser que leurs positions respectives dans les phrases T et P puissent avoir une signification particulière. C'est pourquoi on aimerait pouvoir faire abstraction de cet aspect, du moins dans une certaine mesure.

Le modèle 1 de IBM nous offre à cet égard une porte de sortie, parce qu'il considère en fait que tous les alignements a entre la source et la cible sont équiprobables. Dans ce modèle, la probabilité d'observer une séquence f étant donnée une séquence e s'écrit:

$$\begin{aligned} \Pr(f|e) &= \frac{\epsilon(m|l)}{(l+1)^m} \sum_{a_1=0}^{l} \ldots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(f_j|e_{a_j}) \\ &= \frac{\epsilon(m|l)}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i) \end{aligned} \tag{4.2}$$

où $\epsilon(m|l)$ représente la probabilité d'observer une traduction de longueur m pour une phrase de longueur l; dans le modèle 1, on suppose essentiellement que cette distribution est uniforme. Quant au terme $(l+1)^{-m}$, il représente la probabilité (également uniforme) d'un alignement individuel a.

En outre, ce modèle est particulièrement attrayant pour notre application, parce que le calcul de l'équation (4.2) requiert un nombre d'opérations arithmétiques dans $\mathcal{O}(lm)$.

En somme, il semble qu'on pourrait développer une méthode de sélection des candidats qui repose directement sur le modèle IBM-1. Seule la procédure de recherche serait alors différente, de façon à accommoder les besoins spécifiques de l'application.

Dans un cadre probabiliste tel que suggéré ci-dessus, la sélection des candidats viserait à trouver la couverture R_P^* de C_P qui maximise la probabilité que les segments-

cible t_p soient utiles au traducteur pour traduire la phrase P. Formellement:

$$R_P^* = \text{argmax}_{R_P} \Pr(R_P|P)$$

et, partant des hypothèses d'indépendance énoncées plus haut:

$$\Pr(R_P|P) = \prod_{c_p \in R_P} \Pr(t_p|p) \tag{4.3}$$

Étant donnée la définition de couverture donnée précédemment, il est clair que le principe d'optimalité s'applique dans la recherche de la couverture optimale de P, c'est-à-dire qu'il existe toujours une certaine décomposition de la phrase P en deux segments disjoints P_1 et P_2 telle que la couverture optimale pour P est obtenue en combinant les couvertures optimales de P_1 et P_2. Ceci suggère l'utilisation de méthodes de programmation dynamique. La procédure **max-prob** ci-dessous (algorithme 1) calcule ainsi la couverture de P pour laquelle la probabilité jointe que les t_p soient utiles est maximale:

En somme, cette procédure calcule la couverture optimale pour des sous-séquences de taille croissante de P, sur la base des couvertures optimales obtenues pour les sous-séquences plus petites. (La procédure part de la fin de P; elle aurait tout aussi bien pu partir du début.)

Dans cette procédure, le score $-\log(\Pr(t_{ijk}|p_{ij}))$ attribué à chaque repérage peut être calculé directement avec un modèle IBM-1, suivant la formule de l'équation (4.2). Le score final, obtenu dans *Score*[1] représente donc $-\log(\Pr(R_P^*|P))$. Un détail important: pour les fins de la minimisation, on doit introduire une constante C_0, qui fait figure de coût associé à laisser découvert un mot de P. En termes plus formels, on peut considérer que chaque ensemble C_{ij} contient un candidat *vide* $c_0 = \langle \epsilon, \epsilon \rangle$, et que $C_0 = -\log(\Pr(c_0|p_{ij}))$.

On peut récupérer les candidats $\langle s_{ijk}, t_{ijk} \rangle$ de cette couverture à l'aide de la procédure **get-cover** (algorithme 2).

Algorithme 1 (**max-prob**): calcule la couverture R_P^* de P qui maximise $\Pr(R_P|P)$

Soit la phrase $P = w_1...w_n$ et, pour chaque sous-séquence $p_{ij} = w_i...w_j$ de P, l'ensemble (possiblement vide) des repérages correspondants $C_{ij} = \{c_{ij1}, ..., c_{ijm}\}$:

$Score[n+1] \leftarrow 0$

pour $i \leftarrow n, ..., 1$ **faire**

  $Score[i] \leftarrow C_0$

  $Cover[i] \leftarrow 0$

  pour tout $c_{ijk} = \langle s_{ijk}, t_{ijk} \rangle \in C_{ij}$, $j \geq i$ **faire**

    $x \leftarrow -\log(\Pr(t_{ijk}|p_{ij})) + Score[j+1]$

    si $x < Score[i]$ **alors**

      $Score[i] \leftarrow x$

      $Cover[i] \leftarrow c_{ijk}$

    fin si

  fin pour

fin pour

Algorithme 2 (**get-cover**): récupère la couverture optimale calculée par l'algorithme 1

$R_P^* \leftarrow \{\}$

$i \leftarrow 1$

tant que $i \leq n$ **faire**

  si $Cover[i] = 0$ **alors**

    $i \leftarrow i + 1$

  sinon

    $R_P^* \leftarrow R_P^* \cup \{Cover[i]\}$

    $i \leftarrow i + |Cover[i]|$

  fin si

fin tant que

Bien que nous n'ayons pas fixé les détails de l'ergonomie de notre système d'aide à la traduction, il apparaît assez clairement que le système devrait prioriser les segments-source les plus longs, de façon à minimiser l'impact des manoeuvres requises de la part du traducteur pour incorporer ces segments dans sa traduction, ce que nous appelerons la *contrainte ergonomique*.

Cette contrainte n'est pas prise en charge de façon explicite dans la procédure **max-prob** puisqu'en théorie, elle fait partie intégrante de la distribution $\Pr(R_P|P)$ sur laquelle elle se base. Toutefois, et comme nous le verrons clairement lors de notre évaluation (section 4.4), les modèles statistiques que nous avons à notre disposition rendent plutôt mal compte de la contrainte ergonomique. Pour pallier ce problème, nous pouvons fixer a priori un seuil sur la longueur minimale des segments t_p considérés lors de la recherche. En pratique, ce seuil pourrait être ajusté manuellement par l'utilisateur, en fonction de ses préférences personnelles.

Il existe toutefois une alternative, consistant à utiliser une procédure de sélection qui, au lieu de maximiser directement $\Pr(R_P|P)$, cherche une couverture maximale de P tout en tenant compte de la contrainte ergonomique énoncée ci-dessus. L'algorithme 3 ci-dessous décrit une telle méthode.

Dans cette procédure, la fonction de score optimisée est la taille $(j - i + 1)$ des séquences couvertes dans P, à laquelle on soustrait une pénalité K. Le *score* obtenu par la couverture R_P^* s'exprime alors:

$$\sum_{c_{ijk} \in R_P^*} (|p_{ij}| - K) \tag{4.4}$$

Dans cette équation, K peut être interprété comme le coût (en termes de mots) associé à l'utilisation d'une proposition du système. En maximisant l'équation 4.4, on se trouve donc à maximiser la couverture de P, tout en minimisant le nombre de segments requis (de ce fait, on maximise leur longueur). Si de plus on exige que ce score ne soit pas négatif, K peut être utilisé pour déterminer la taille minimale des

Algorithme 3 (**max-cover**): calcule la couverture optimale de P

Soit la phrase $P = w_1...w_n$ et, pour chaque sous-séquence $p_{ij} = w_i...w_j$ de P, l'ensemble (possiblement vide) des repérages correspondants $C_{ij} = \{c_{ij1}, ..., c_{ijm}\}$:

Soit une constante K

$Score[n+1] \leftarrow 0$

pour $i \leftarrow n, ..., 1$ **faire**

  $Score[i] \leftarrow 0$

  $Cover[i] \leftarrow 0$

  pour $j \leftarrow i, ..., n$ **faire**

    $x \leftarrow (j - i + 1) - K + Score[j + 1]$

    si $x > Score[i]$ **alors**

      $Score[i] \leftarrow x$

      $k^* = \text{argmax}_k \Pr(t_{ijk}|p_{ij}),\ c_{ijk} = \langle s_{ijk}, t_{ijk} \rangle \in C_{ij},$

      $Cover[i] \leftarrow c_{ijk^*}$

    fin si

  fin pour

fin pour

segments utilisés dans la couverture. Par exemple, en utilisant $K = 3$, on interdit dans les faits l'utilisation de segments de longueur inférieure à 3.

Finalement, en ne retenant dans chaque C_{ij} que le candidat pour lequel $\Pr(t_{ijk}|p_{ij})$ est maximal, on trouve donc en fait le R^*_P le plus probable étant donnée une couverture maximale soumise à la contrainte ergonomique K.

Ici encore, à la fin du calcul, le score de la couverture optimale se trouve dans $Score[1]$. Concrètement, celui-ci peut être interprété comme le nombre de mots couverts dans P, moins K fois le nombre de candidats impliqués dans la couverture. La sélection effectuée peut ici encore être récupérée par la procédure **get-cover** (algorithme 2).

4.4 Évaluation

Afin d'évaluer et de comparer la performance de différents mécanismes de sélection, nous avons mis en place une procédure d'évaluation, que nous décrivons dans cette section.

Rappelons à ce stade que notre objectif final est la mise sur pied d'un système d'aide à la traduction, permettant à un traducteur humain de récupérer des segments de traduction existants, extraits d'une mémoire de traduction, afin de produire la traduction d'un nouveau document. Le but de cette opération n'est pas seulement d'accélérer la traduction humaine, mais plus généralement d'en faciliter l'exercice, en suggérant des formulations qui ne seraient peut-être pas venues d'emblées à l'idée du traducteur.

Idéalement, l'évaluation devrait donc se faire dans cette perspective. Malheureusement, ce n'est pas envisageable pour l'instant, d'abord parce que nous avons délibérément escamoté toute la question de l'interface-utilisateur d'une telle application, ensuite parce que ce genre d'évaluation nécessiterait la mise sur pied d'un cadre expérimental coûteux et complexe.

Ce que nous souhaitons mesurer à ce stade, c'est simplement la proportion du matériel proposé susceptible d'être utile à un traducteur, à l'intérieur d'un ensemble de scénarios d'utilisation relativement généraux. La procédure d'évaluation que nous avons mise en place consiste donc en une simulation, impliquant d'une part une mémoire de traduction existante, et d'autre part un nouveau texte à traduire, pour lequel nous disposons déjà d'une *traduction oracle*. La procédure d'évaluation cherche à mesurer la proportion des candidats séléctionnés dans la mémoire de traduction qui sont effectivement utiles pour produire cette traduction.

Toutes les expériences qui suivent sont basées sur une mémoire de traduction constituée de 1919 paires de documents du Hansard canadien, c'est-à-dire la totalité des documents publiés entre avril 1986 et janvier 2002, pour un corpus totalisant près de 6 millions de phrases de chaque langue, soit près de 200 millions de mots en tout (il s'agit essentiellement du corpus *Hansard* du système *TransSearch*, décrit à la section 2.3.1, tel qu'il existait à la fin janvier 2002; ce corpus a également été utilisé dans les expériences décrites aux chapitres 2 et 3). Chaque document a subi un traitement préalable visant d'une part à le dépouiller de son balisage HTML, et d'autre part à localiser les frontières entre les paragraphes et les phrases. Les paires de documents correspondantes dans les deux langues (anglais et français) ont ensuite été alignées au niveau des phrases au moyen de la procédure SFIAL décrite à la section 2.5.3, générant un peu plus de 5,7 millions de couples de phrases. Le tout a été versé dans une base de données documentaire (MG [115] – voir la section 2.5.2), permettant ainsi l'extraction rapide de tous les couples contenant une certaine séquence de mots.

D'autre part, nous avons isolé un document bilingue, également tiré du Hansard, mais distinct des documents de la mémoire de traduction (il s'agit d'un document paru en mars 2002; ce même document a servi aux expériences des sections 2.6 et 3.7). De ce document, nous avons extrait 1000 paires de phrases alignées. C'est ce que nous appelons ici le *document-test*.

L'alignement des phrases de ce document a également été effectué par SFIAL,

puis vérifié à la main. La version anglaise doit jouer le rôle de *nouveau texte à traduire*, alors que la version française constitue la *traduction-oracle*: essentiellement, tous les scénarios d'expérimentation qui suivent supposent qu'un traducteur humain doit traduire la partie anglaise du document-test en français. Nous supposons alors que la version française est la traduction que le traducteur a en tête après lecture du texte-source, et donc qu'il souhaite produire.

Suivant la méthode décrite à la section 2.5, chaque phrase-source anglaise du document-test a d'abord été segmentée en tronçons syntaxiques; nous avons alors extrait de la mémoire de traduction tous les couples qui contenaient une séquence de tronçons du document-test; finalement, nous avons effectué un repérage de la traduction de ces séquences de tronçons, au moyen de la méthode de repérage de traduction *compositionnel contraint* (RT-CC) décrite à la section 3.5.5.

À partir de ce stade, nous étions en mesure d'effectuer une sélection parmi ces candidats, au moyen des procédures décrites précédemment. Nous avons alors mesuré, pour chaque sélection ainsi produite:

couverture-cible : la proportion (en mots) de la traduction-oracle qui pourrait être produite en utilisant directement (tels quels) les candidats retenus par la sélection; nous avons utilisé pour ce faire une variante de l'algorithme 3, qui calcule une couverture optimale de la traduction-oracle au moyen des repérages-cible sélectionnés. (Bien entendu, cette variante ignore la distribution $\Pr(t_p|p)$, pour ne considérer que la longueur et le nombre des candidats retenus t_p.)

précision : la proportion (en nombre de mots) des repérages-cible sélectionnés et effectivement utilisés pour produire la traduction-oracle.

longueur moyenne : la longueur moyenne (en mots) des repérages-cible utilisés.

À titre de points de référence, et en plus des performances enregistrées par les

procédures de sélection de la section précédente, nous avons évalués les deux méthodes de sélection suivantes:

sélection aléatoire : Il s'agit en fait de l'algorithme 1, dans lequel nous attribuons une probabilité égale à tous les candidats; cette expérience visait à établir le *niveau de performance minimal* (en anglais : *baseline*).

sélection oracle : Ici, on n'effectue en fait aucune sélection: tous les candidats de C_P sont considérés; cette expérience visait donc à établir le *niveau de performance maximal* auquel on pouvait s'attendre (en terme de couverture). Évidemment, dans ces expériences, il ne fait aucun sens de mesurer la précision (telle que définie ci-dessus), qu'on peut théoriquement considérer égale à 1.

Les expériences ont été effectuées sous différentes conditions, correspondant à différents niveaux de tolérance de l'utilisateur quant à la longueur minimale des séquences proposées. Ainsi, nous avons examiné ce qui se passerait si l'utilisateur refusait systématiquement d'utiliser des séquences de moins de 3 mots, de moins de 4 mots, et de moins de 5 mots (ces seuils nous semblaient réalistes). Comme ces niveaux de tolérance constituaient en fait des préférences, les candidats trop courts étaient filtrés avant même le processus de sélection. On retrouve dans le tableau 4.1 ci-dessous le nombre total de candidats extraits de la mémoire de traduction, sous chacun de ces trois scénarios.

candidat minimal	nombre de candidats
$\|t_p\| \geq 3$	371 976
$\|t_p\| \geq 4$	150 513
$\|t_p\| \geq 5$	57 160

Table 4.1. Nombre de candidats extraits de la mémoire de traduction

Les procédures de sélection que nous souhaitions tester reposent sur un estimé

de $\Pr(t_p|p)$. Dans les expériences qui suivent, le calcul de cette estimation a été effectué suivant la formule de l'équation (4.2), en nous basant sur les paramètres $t(f|e)$ d'un modèle de traduction de type IBM-1 (implantation du package *Egypt* [5]), entraîné sur un sous-ensemble du corpus Hansard constituant la mémoire de traduction, représentant environs 20% de celle-ci.

Quant à la constante C_0, représentant le coût associé à l'utilisation du *repérage vide* dans la procédure **max-prob**, nous avons arbitrairement fixé sa valeur à 100.

La tableau 4.2 résume les performances enregistrées par les différentes procédures de sélection de candidats décrits dans la section 4.3.

		Algorithme de sélection			
		aléatoire	**oracle**	**max-cover**	**max-prob**
$\|t_p\| \geq 3$	couverture (%)	6,19	40,5	10,92	13,22
	précision (%)	6,19	–	13,23	13,49
	taille moyenne	4,05	3,96	4,99	4,07
$\|t_p\| \geq 4$	couverture (%)	5,38	26,90	8,83	9,89
	précision (%)	6,49	–	11,47	12,15
	taille moyenne	5,05	5,11	5,92	5,15
$\|t_p\| \geq 5$	couverture (%)	3,58	16,76	6,52	7,24
	précision (%)	5,97	–	11,75	12,55
	taille moyenne	6,47	6,37	7,08	6,51

Table 4.2. Performance des algorithmes de sélection

Comme on le voit, **max-prob** (algorithme 1) couvre significativement mieux la cible que **max-cover**, tout en assurant une meilleure précision. On observe par ailleurs dans le **max-cover** l'effet d'incorporer explicitement la contrainte ergonomique dans le critère d'optimisation: les repérages-cible retenus sont typiquement de 10 à 20% plus long qu'avec le **max-prob**. Par ailleurs, cette tendance tend

à s'amenuiser à mesure que les repérages-cible courts sont filtrés à la source.

Autre constat: à part pour la sélection-oracle, la précision des ensembles de candidats choisis est toujours légèrement supérieure à la couverture. Ceci est tout à fait normal, puisque les procédures de sélection utilisées produisent normalement des ensembles de candidats dont la taille est légèrement inférieure à la phrase-source; à mesure qu'on augmente la taille minimale des repérages-cible, cette différence de taille tend à augmenter, parce qu'il devient de plus en plus difficile pour la procédure de sélection de couvrir la totalité de la phrase-source. Par conséquent, les taux de couverture tendent à diminuer, mais la précision demeure à peu près la même.

Comme on le mentionnait plus haut, on souhaiterait ultimement que l'utilisateur puisse régler la quantité de matériel proposé selon ses préférences. Étant donnée la nature des procédures de recherche proposées, il existe plusieurs façons d'implanter un tel contrôle. La méthode la plus simple consiste certainement à permettre à l'utilisateur de l'exprimer comme un pourcentage de la taille de P. Si la quantité de matériel dans R_P^* excède ce pourcentage, on peut éliminer des candidats en commençant par les plus courts ou par ceux dont $\Pr(t_p|p)$ est la plus faible; si à l'opposé le nombre de candidats est insuffisant, on peut aller chercher dans chaque C_{ij} correspondant aux séquences p_{ij} sélectionnées dans R_P^* les deuxièmes meilleurs candidats (en termes de $\Pr(t_p|p)$), possiblement en commençant par les séquences p les plus longues, puis les troisièmes meilleurs candidats, et ainsi de suite.

Pour connaître l'effet de ce mécanisme, nous avons effectué une série d'expériences dans lesquelles nous produisons, pour chaque C_{ij} sélectionné, les N meilleurs candidats en terme de $\Pr(t_p|p)$. La figure 4.3 illustre les résultats de ces expériences avec la méthode **max-prob** (notez l'échelle exponentielle en abscisse; les valeurs en ordonnées sont exprimées en pourcentage). En pratique, **max-cover** se comporte essentiellement de la même façon.

Comme on le constate, la couverture du texte-cible qu'on peut espérer atteindre augmente à peu près de façon log-linéaire à mesure qu'on accroît la taille de la

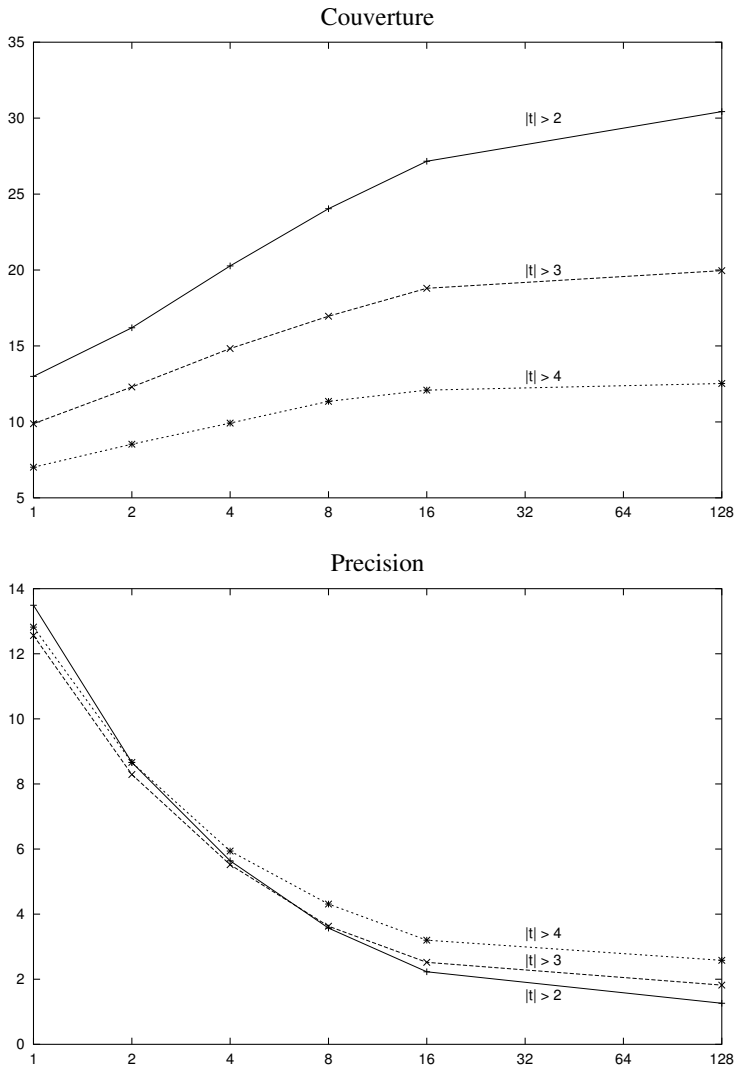

Figure 4.3. Couverture et précision observées pour des sélections $\max$-prob conservant les N meilleurs candidats pour chaque séquence p sélectionnée

sélection. Par contre, et comme on pouvait s'y attendre, cette augmentation se fait au prix d'une diminution très marquée de la précision (théoriquement, celle-ci devrait décroître en proportion inverse de N, mais en pratique cette décroissance s'atténue, tout simplement parce que la quantité de matériel disponible dans la mémoire de traduction n'est pas infinie). La couverture tend à plafonner environ 25% sous la barre de la couverture optimale. Ce plafonnement s'explique par le fait que la sélection est faite sur la base d'une couverture du texte-source, alors que la couverture-cible optimale a été calculée sans tenir compte d'une telle contrainte.

D'une façon générale, bien que les résultats obtenus avec ces procédures de sélection soient significativement supérieurs à la performance minimale attendue (algorithme *aléatoire*), ils demeurent bien en-dessous du maximum (*oracle*). Force est de constater que le problème de la sélection des candidats est difficile. Les facteurs qui font qu'un traducteur choisit une formulation plutôt qu'une autre sont multiples et parfois imprévisibles. À l'examen, on constate que les candidats de la *sélection-oracle* se voient parfois attribuer des probabilités très faibles par les modèles de traduction. Dans les faits, une faible probabilité n'atteste pas nécessairement d'une "mauvaise" traduction (cette notion n'existant pas pour les modèles statistiques), mais simplement d'une traduction relativement rare. Et par ailleurs, plus une séquence est rare, moins l'estimé de sa probabilité par des méthodes statistiques est fiable. Ce phénomène se manifeste de façon d'autant plus criante que les séquences sont longues. Ceci s'explique par le fait que nos modèles de traduction estiment les probabilités de façon compositionnelle, i.e. $\Pr(t_p|p)$ se calcule sur la base de paramètres $t(e|f)$ pour des mots individuels e et f. Or, il semble que cette hypothèse de compositionnalité de la traduction tend à s'affaiblir à mesure qu'on approche le niveau du mot.

4.5 Critères de sélection alternatifs

Les résultats des expériences de la section 4.4 mettent en relief les lacunes de nos modèles de traduction pour estimer $\Pr(R_P|P)$. Nous examinons ici différentes alternatives à leur utilisation pour la sélection des candidats.

4.5.1 Fréquences relatives des candidats-cible

Il est intéressant de noter que les ensembles de candidats C_P nous suggèrent une alternative prometteuse pour estimer la distribution $\Pr(R_P|P)$, dans la mesure où on les considère non pas comme des ensembles, mais plutôt comme des listes ou des *sacs de candidats*. Dès lors, certains segments-cible t_p peuvent apparaître plusieurs fois dans la liste C_p des candidats associés au segment p. Sous l'hypothèse que $\Pr(c_p|P) \approx \Pr(t_p|p)$, la fréquence de t_p dans C_p peut servir à estimer $\Pr(t_p|p)$: c'est le nombre de fois que t_p a été utilisé pour traduire une phrase de la mémoire de traduction contenant p. Si on note $F(t_p)$ la fréquence relative du segment t_p dans le C_p d'où il provient, on peut réécrire l'équation 4.3:

$$\Pr(R_P|P) \approx \prod_{t_p \in C_P} F(t_p) \tag{4.5}$$

En remplaçant la fonction de score de l'algorithme 1 (**max-prob**) par $-\log(F(t_p))$, on obtient un algorithme de sélection que nous appelons **max-freq**.

Alternativement, on peut modifier la procédure **max-cover** d'une façon analogue: pour chaque p sélectionné par l'algorithme **max-cover**, choisir le candidat t_p pour lequel $F(t_p)$ est maximal. On appelle cette méthode **max-cover-freq**.

Nous avons testé ces deux méthodes sur les données de la section 4.4. Le tableau 4.3 présente les résultats de ces expériences; nous y reproduisons également les résultats des méthodes **max-prob** pour fins de comparaison.

D'une façon surprenante, ces deux méthodes produisent des résultats supérieurs à ceux obtenus avec des modèles de traduction, à la fois en couverture, en précision,

		Algorithme de sélection		
		max-prob	**max-freq**	**max-cover-freq**
$\|t_p\| \geq 3$	couverture (%)	13,22	14,59	14,44
	précision (%)	13,49	16,53	16,73
	taille moyenne	4,07	4,66	4,77
$\|t_p\| \geq 4$	couverture (%)	9,89	11,28	11,09
	précision (%)	12,15	13,80	13,54
	taille moyenne	5,15	5,59	5,74
$\|t_p\| \geq 5$	couverture (%)	7,24	8,16	7,94
	précision (%)	12,55	13,10	12,78
	taille moyenne	6,51	7,05	7,13

Table 4.3. Méthodes max-prob, max-freq et max-cover-freq

et en terme de la longueur des segments sélectionnés. Comme on l'avait observé avec les modèles de traduction, une approche visant à maximiser directement la probabilité des segments (**max-freq**) se comporte généralement mieux qu'une approche qui maximise d'abord la couverture du texte-source.

4.5.2 Similitude des segments-source

En pratique, l'estimation de $\Pr(t_p|p)$ à même les fréquences des t_p dans les ensembles de candidats comporte toutefois une faille importante: plusieurs C_p contiennent très peu de paires, parfois une seule, mais plus souvent quelques segments-cible de très basse fréquence. Une estimation fiable de $\Pr(t_p|p)$ devient alors problématique, et en pratique il devient impossible de classer les candidats. Encore une fois, ce problème affecte plus particulièrement les séquences longues, qui sont plus rares.

Lorsque cette situation survient, une possibilité est de se rabattre sur le modèle de traduction pour estimer $\Pr(t_p|p)$. En pratique toutefois, on constate que d'autres

caractéristiques des candidats $\langle s_p, t_p \rangle$ pourraient être utiles pour choisir le "bon" candidat. Par exemple, le degré de similitude entre s_p et p est souvent un bon indicateur du potentiel d'un candidat. Partant du fait que les segments p sont toujours des sous-séquences des repérages-source s_p, ce degré de similitude peut se mesurer comme le ratio des longueurs $|p|/|s_p|$: ce ratio est égal à 1 lorsque les deux séquences sont identiques, et tend vers 0 à mesure que s_p grandit.

Le tableau 4.4 présente la performance d'une méthode de sélection obtenue en remplaçant la fonction de score de l'algorithme 1 par $-\log(|p|/|s_p|)$, méthode que nous appelons **max-similitude**. Pour fins de comparaison, nous présentons également les résultats obtenus par **max-prob** et **max-freq** sur les mêmes données.

		Algorithme de sélection		
		max-prob	**max-freq**	**max-similitude**
$\|t_p\| \geq 3$	couverture (%)	13,22	14,59	15,44
	précision (%)	13,49	16,53	16,51
	taille moyenne	4,07	4,66	3,81
$\|t_p\| \geq 4$	couverture (%)	9,89	11,28	10,49
	précision (%)	12,15	13,80	13,23
	taille moyenne	5,15	5,59	4,85
$\|t_p\| \geq 5$	couverture (%)	7,24	8,16	6,65
	précision (%)	12,55	13,10	11,78
	taille moyenne	6,51	7,05	6,11

Table 4.4. Méthodes max-prob, max-freq et max-similitude

Dans ces expériences, la méthode **max-similitude** se comporte donc légèrement mieux que toutes les méthodes précédentes lorsqu'on permet au système de proposer des séquences-cible de 3 mots et plus. Toutefois, cet avantage décroît rapidement lorsqu'on passe à 4 et à 5 mots. Si on observe de plus que les séquences sélectionnées

par **max-similitude** sont systématiquement plus courtes, on peut en conclure que, si cette caractéristique des candidats est bien adaptée pour départager les séquences courtes, elle l'est beaucoup moins pour les séquences longues.

4.6 Combinaisons de critères

Les expériences rapportées dans les sections précédentes démontrent qu'il existe plusieurs façons de caractériser les repérages candidats, plus ou moins adaptées pour guider le processus de sélection. Afin de prendre une décision éclairée, on aimerait pouvoir utiliser toute l'information disponible. Ceci suggère l'utilisation, en lieu et place d'un estimé de $\Pr(t_p|p)$, d'une fonction de score qui combinerait ces différentes sources d'information. Bien entendu, les possibilités de combinaison sont infinies, et c'est pourquoi nous proposons d'utiliser une méthode de combinaison générique pour laquelle il est possible de trouver la combinaison optimale par apprentissage automatique. Une telle approche est certainement envisageable, puisqu'on peut aisément produire un corpus d'exemples pour l'apprentissage, à partir de ressources similaires à celles que nous avons utilisées jusqu'à maintenant pour l'évaluation de la performance: un ensemble de phrases en langue-source, et pour chacune d'elle, des ensembles de candidats-cible et une traduction-oracle.

D'un point de vue apprentissage-machine, cette question sur la fonction de score peut évidemment être vue comme un problème d'*estimation de densité*: on aimerait trouver une fonction qui, étant donné un ensemble d'attributs caractérisant un candidat c_p, produit un estimé de $\Pr(c_p|P)$. Toutefois, et comme on l'a vu dans les expériences précédentes, cette distribution reste difficile à estimer, et il n'y a pas lieu de croire à ce stade que nous puissions y arriver mieux qu'avec un modèle de traduction statistique.

Alternativement, on peut voir le processus de sélection comme un problème de *classification*: séparer les "bons" candidats des "mauvais". Cette solution est égale-

ment problématique, pour deux raisons: d'abord, elle s'harmonise mal avec le type de procédure de recherche proposé à la section 4.3; ensuite, et d'une façon plus générale, on constate que les candidats parmi lesquels nous devons choisir sont rarement "mauvais". L'objectif de la sélection est plutôt de déterminer lesquels sont les plus susceptibles d'aider le traducteur.

En fin de compte, tout ce dont nous avons réellement besoin pour effectuer la sélection, c'est d'une fonction de score. L'interprétation exacte qu'on attribue aux valeurs produites n'a pas réellement d'importance, en autant que la fonction produise de meilleurs scores pour les candidats les plus prometteurs. On peut donc voir la question de déterminer la fonction de score la plus adéquate pour le processus de sélection comme un problème de *régression*: trouver la fonction qui, étant donné un ensemble d'attributs caractérisant un candidat c_p, produit une valeur d'autant plus grande que le candidat est susceptible d'être utile au traducteur.

De multiples approches sont possibles pour attaquer un tel problème, nous avons choisi de concentrer nos efforts sur une classe de méthodes relativement simple, les réseaux de neurones multicouches.

4.6.1 Réseaux de neurones multicouches

Les réseaux de neurones multicouches (RNM) sont des dispositifs bien connus dans le domaine de l'apprentissage-machine, et ont été employés dans de multiples applications de classification, de régression et d'estimation de densité [11]. Conceptuellement, un RNM est constitué d'un ensemble d'unités, dont le comportement se veut une approximation de celui des neurones dans le cerveau humain. Chaque unité prend en entrée un vecteur $x \in \mathcal{R}^n$ et produit en sortie une valeur unique $y(x) \in \mathcal{R}$, de la façon suivante:

$$y(x) = f(\sum_{i=1}^{n} w_i x_i + w_0) \tag{4.6}$$

où les $w_i \in \mathcal{R}$, $0 \leq i \leq n$, constituent les paramètres qui déterminent le comportement de l'unité. La fonction f est appelée la *fonction d'activation*: elle a pour objectif de ramener le résultat dans un intervalle spécifique, typiquement $[0, 1]$ ou $[-1, 1]$. Les fonctions d'activation les plus utilisées sont la fonction *signe* (qui retourne 1 lorsque son argument est positif, -1 sinon) et les fonctions *sigmoïde* et *tangeante hyperbolique* (*tanh*).

Dans un RNM, ces unités sont organisées en couches: chacune de ces couches reçoit en entrée le même vecteur x, et les sorties individuelles des unités constituent ensemble un vecteur de sortie $y \in \mathcal{R}^m$, où m est le nombre d'unités dans la couche. Plusieurs couches peuvent ainsi être superposées, chaque couche recevant en entrée la sortie de la couche précédente.

L'intérêt de ce genre de dispositif est qu'on peut ajuster la valeur de l'ensemble des paramètres θ (c'est-à-dire l'ensemble des paramètres w_i de toutes les unités du réseau) de façon à s'approcher du résultat souhaité. Cet ajustement (l'apprentissage) se fait habituellement en exposant le réseau à un ensemble d'exemples, constitué de paires $(x, \hat{y})$. On compare alors le résultat souhaité $\hat{y}$ avec la valeur $y_\theta(x)$ effectivement produite par le réseau. On mesure l'ampleur de la différence entre les deux (ce qu'on appelle l'*erreur*) , et on produit un nouvel ensemble de paramètres θ' qui tend à minimiser cette erreur. Typiquement, ce calcul se fait par descente de gradient (nous y revenons plus loin).

Cet apprentissage par *rétropropagation des erreurs* se fait donc de façon itérative: l'examen des exemples, la rétropropagation des erreurs et la production d'un nouvel ensemble de paramètres pour le réseau peuvent être répété plusieurs fois, jusqu'à ce qu'une certaine convergence soit observée. On distingue habituellement le mode d'apprentissage *en lot*, où tous les exemples sont examinés avant de procéder à la mise-à-jour des paramètres, du mode d'apprentissage *en-ligne*, ou *stochastique*, où les paramètres du réseau sont recalculés après chaque exemple présenté.

Nous nous intéresserons ici à des réseaux à deux couches, c'est-à-dire une couche

intermédiaire (dite *cachée*) et une couche de sortie [2]. Dans notre cas, cette dernière couche ne comportera qu'une seule unité, dont la sortie constituera le score utilisé par les procédures de sélection.

Chaque unité de nos réseaux comporte une fonction d'activation *tanh*. En pratique, nous avons observé que cette fonction n'était pas essentielle, mais que d'une part sa présence accélérait systématiquement la convergence du processus d'apprentissage, et que d'autre part elle se comportait mieux que l'alternative standard, la fonction *sigmoïde*.

Donc, pour un repérage-candidat c_p, on calcule une valeur de sortie du réseau $y(c_p)$ de la façon suivante:

$$y(c_p) = \tanh(\sum_{j=0}^{Z} w_j z_j(c_p)) \tag{4.7}$$

$$z_j(c_p) = \tanh(\sum_{i=0}^{X} w_{ij} x_i(c_p)) \tag{4.8}$$

Dans ces formules:

- les $x_i(c_p)$ constituent l'entrée du réseau, c'est-à-dire un vecteur descriptif du candidat c_p;
- X est la taille de ce vecteur;
- les $z_j(c_p)$ représentent la sortie de la couche cachée, qui alimente l'unité de sortie;
- Z est le nombre d'unités dans la couche cachée (et, par conséquent, la taille du vecteur z).

[2] Bien que la capacité d'un réseau augmente avec le nombre de couches, il est théoriquement possible de faire l'approximation de n'importe quelle fonction continue avec une seule couche cachée, et en pratique, peu de problèmes en nécessitent réellement plus qu'une couche.

On remarque par ailleurs que chaque couche comporte une unité *zéro*, qui joue le rôle de *biais*, c'est-à-dire que cette unité reçoit systématiquement une entrée égale à 1: $x_0(c) = 1$ et $z_0(c) = 1$. Ces unités se trouvent en fait à remplacer les paramètres w_0 de l'équation (4.6).

4.6.2 Valeurs en entrée du réseau

Ce réseau reçoit donc en entrée un vecteur $x(c_p)$ qui caractérise le candidat $c_p = \langle s_p, t_p \rangle$. Les caractéristiques que nous avons retenues sont les suivantes:

- Les caractéristiques proposées précédemment :
  - $x_1(c_p) = \mathrm{mt}(c_p)$: estimé de $\Pr(t_p|p)$, tel que produit par nos modèles de traduction.
  - $x_2(c_p) = F(c_p)$: fréquence relative de t_p au sein de l'ensemble de candidats C_p d'où il provient.
  - $x_3(c_p) = \mathrm{sim}(c_p) = |p|/|s_p|$: degré de similitude entre s_p et p.
- Des attributs relatifs à la taille des candidats. L'objectif visé avec ces attributs est double: d'une part, faire intervenir la longueur des repérages dans les critères de sélection, de façon à rendre compte de la *contrainte ergonomique* (section 4.2); d'autre part, favoriser des repérages dans lesquels les différents segments ont des longueurs comparables. Ce dernier critère s'appuie sur des observations en ce sens dans le cas des alignements de phrases [35].
  - $x_4(c_p) = |p|$: longueur de p (en mots).
  - $x_5(c_p) = |s_p|$: longueur de s_p (en mots).
  - $x_6(c_p) = |t_p|$: longueur de t_p (en mots).
  - $x_7(c_p) = (|s_p| - |t_p|)/(|s_p| + |t_p|)$: ratio des longueurs de s_p et t_p.

  - $x_8(c_p) = (|p| - |t_p|)/(|p| + |t_p|)$: ratio des longueurs de p et t_p.

- Des attributs rendant compte du *rang* d'un candidat à l'intérieur de C_p, relativement à un attribut donné; la fonction $\text{rang}(f(c_p))$ retourne la valeur 1 pour le candidat c de C_p qui affiche le "meilleur" $f(c_p)$, 2, pour le deuxième meilleur, etc. Il s'agit en somme d'une façon alternative de numériser certaines des caractéristiques des candidats. Nous avons constaté que, dans certaines circonstances, nos réseaux s'accommodaient mieux de ce genre de valeurs, c'est pourquoi elles ont été ajoutées.

  - $x_9(c_p) = \text{rang}(\text{mt}(c_p))$
  - $x_{10}(c_p) = \text{rang}(F(c_p))$
  - $x_{11}(c_p) = \text{rang}(\text{sim}(c_p))$
  - $x_{12}(c_p) = \text{rang}(|s_p|)$

- Un attribut booléen:

  - $x_{13}(c_p) = \text{cov}_\text{P}(c_p)$: p appartient-il à la couverture-source maximale telle que calculée par **max-cover**? $\text{cov}_\text{P}(c_p)$ retourne la valeur 1 si c'est le cas, 0 sinon.

- Deux attributs qui simulent le comportement des méthodes de sélection basées sur une couverture-source maximale préalable (**max-cover** et **max-freq-cover**). Le calcul de ces attributs est basé sur $\text{cov}_\text{P}(c_p)$:

  - $x_{14}(c_p) = \text{cov}_\text{P}(c_p)\text{mt}(c_p)$
  - $x_{15}(c_p) = \text{cov}_\text{P}(c_p)F(c_p)$

Pour satisfaire aux contraintes inhérentes aux RNM, tous ces attributs sont normalisés, afin de résider dans la plage $[-1, +1]$.

4.6.3 Valeurs de sortie du réseau

Bien que nous ayons formulé notre problème dans un cadre de régression, nous ne disposons pas a priori de valeurs-cible que nous aimerions voir le réseau produire, c'est-à-dire les valeurs y souhaitées pour chaque exemple d'apprentissage. On peut toutefois construire de telles valeurs artificiellement en utilisant la traduction-oracle. Par exemple, soit une phrase P en langue-source et sa traduction-oracle T_P, on pourrait statuer que la fonction de score doit attribuer une note $y_c(c_p)$ égale à 1 à chaque t_p qui couvre exactement une portion de T_P, et 0 sinon.

En pratique, cette stratégie revient à forcer le RNM à se comporter comme un classificateur, qui décide a priori si une séquence est "bonne" (= 1) ou "mauvaise" (= 0), tâche possiblement difficile étant donnée la caractérisation plutôt sommaire que le réseau reçoit en entrée (nous verrons plus loin comment cette méthode se comporte en pratique).

Alternativement, on peut construire des valeurs-cible qui donnent plus de lattitude au réseau dans l'expression de son opinion, quitte à ce que cette valeur soit moins efficace dans la tâche de sélection. Comme point de départ, nous proposons une mesure du *degré de couverture-cible* d'un candidat: soit une phrase P et sa traduction-oracle T_P, la *couverture-cible du candidat* c_p est la sous-séquence maximale t_p^* de t_p qui est également une sous-séquence de T_P. Le *degré de couverture-cible* $y_p(c_p)$ est simplement le rapport des longueurs $|t_p^*|/|t_p|$. Un rapport de 1 dénote un candidat directement utilisable dans la traduction-oracle, alors qu'un rapport de 0 dénote un candidat qui ne comporte aucun mot en commun avec T_P. Les valeurs intermédiaires identifient quant à elles des candidats qui seraient partiellement utilisables.

La figure 4.4 illustre le calcul des couverture-cible sur la sélection de la figure 4.2. Dans cet exemple, les crochets [...] dénotent les couvertures produites sur la source et sur la cible au moyen des repérages de la sélection.

On peut d'emblée évaluer la performance d'une fonction de score qui serait en

Phrase source:

$[_{p_1}$ The government is putting] $[_{p_2}$ a \$2.2 billion tax] on $[_{p_3}$ Canada 's most vulnerable industry] $[_{p_4}$, the airline industry] .

Repérages-cible:

$$t_{p_1} = \underbrace{\text{Le gouvernement}}_{t^*_{p1}} \text{ met} \qquad \rightarrow y_p(c_{p_1}) = 2/3 = 0,66$$

$$t_{p_2} = \underbrace{\text{une taxe de } 2,2 \text{ milliards}}_{t^*_{p2}} \qquad \rightarrow y_p(c_{p_2}) = 5/5 = 1$$

$$t_{p_3} = \underbrace{\text{industrie la plus vulnérable}}_{t^*_{p3}} \qquad \rightarrow y_p(c_{p_3}) = 4/4 = 1$$

$$t_{p_4} = \underbrace{\text{l}' \text{ industrie}}_{t^*_{p4}} \text{ aérienne} \qquad \rightarrow y_p(c_{p_4}) = 2/3 = 0,66$$

Traduction oracle et couverture:

$[_{t^*_{p_1}}$ Le gouvernement] impose $[_{t^*_{p_2}}$ une taxe de 2,2 milliards] de dollars à l' $[_{t^*_{p_3}}$ industrie la plus vulnérable] du Canada , $[_{t^*_{p_4}}$ l' industrie] du transport aérien .

Figure 4.4. Exemples de calcul des couvertures-cible des candidats

mesure d'estimer parfaitement l'une des valeurs-cible proposées ci-dessus. Partant du corpus de test utilisé précédemment (section 4.4), il suffit de calculer la valeur-cible pour chaque candidat, et de mesurer la performance d'une méthode de sélection qui utiliserait cette valeur en lieu et place de l'estimé de $\Pr(t_p|p)$ dans l'algorithme **max-prob**. Nous avons ainsi testé le comportement de fonctions de score estimant parfaitement $y_c(c_p)$ et $y_p(c_p)$, de même que différentes variantes de ces deux valeurs, faisant intervenir la longueur du repérage-cible $|t_p|$ correspondant. Les résultats de ces tests sont donnés dans le tableau 4.5.

		Valeur-cible				
		y_c	$y_c\|t_p\|$	y_p	$y_p\|t_p\|$	${y_p}^2\|t_p\|$
$\|t_p\| \geq 3$	couverture (%)	35,61	37,36	31,39	27,18	29,35
	précision (%)	91,24	92,87	40,26	32,28	34,91
	taille moyenne	3,70	4,27	3,73	4,51	4,45
$\|t_p\| \geq 4$	couverture (%)	24,56	25,47	20,37	18,92	19,90
	précision (%)	95,33	96,60	29,60	25,40	26,76
	taille moyenne	4,87	5,40	4,91	5,59	5,55
$\|t_p\| \geq 5$	couverture (%)	15,76	16,16	12,78	12,38	12,78
	précision (%)	98,34	98,81	26,82	24,18	24,99
	taille moyenne	6,10	6,68	6,20	6,85	6,83

Table 4.5. Performance comparée de différentes valeurs-cible

Comme on le voit, les sélections basées sur y_c sont meilleures que celles basées sur y_p, ce qui est assez naturel. La valeur-cible y_c seule ne rend pas compte de la longueur des segments, de telle sorte que la sélection tend à maximiser le nombre de segments dans la couverture plutôt que la couverture elle-même. C'est ce qui explique en partie que les couvertures obtenues ne sont pas aussi bonnes que les couvertures-cible optimales (tableau 4.2). $y_c|t_p|$ compense pour cette tendance, en

faisant intervenir dans le calcul la longueur du repérage-cible t_p, sans pour autant atteindre le maximum: il arrive que la sélection (qui est effectuée sur la base d'une couverture du texte-source P) produise des candidats-cible qui se chevauchent, et qui par conséquent ne sont pas tous utilisables simultanément. Ceci explique également pourquoi la précision de ces sélections n'est pas parfaite.

Les valeurs-cible basées sur le degré de couverture y_p, quoique moins performantes, se comportent quand même relativement bien, du moins du point de vue de la couverture (les précisions sont forcément beaucoup moins bonnes, parce que y_p n'élimine pas d'emblée les candidats qui ne recouvrent pas exactement une portion de la traduction-oracle). Les variantes faisant intervenir la longueur du segment-cible $|t_p|$ sont généralement moins performantes, mais tendent à favoriser les candidats plus longs, une caractéristique intéressante compte tenu de notre *contrainte ergonomique*.

4.6.4 Entraînement direct

Étant donné un choix de valeur-cible en sortie, on peut entraîner un RNM à produire ces valeurs. On choisit pour ce faire une fonction d'erreur rendant compte de la différence entre la valeur-cible voulue $\hat{y}(c_p)$ et la valeur $y(c_p)$ effectivement calculée par le réseau. Un choix classique est une fonction d'erreur quadratique:

$$E = \frac{1}{2} \sum_{c_p \in C} (y(c_p) - \hat{y}(c_p))^2 \tag{4.9}$$

L'entraînement du réseau se fait par descente de gradient, c'est-à-dire qu'on modifie itérativement les paramètres θ du réseau de façon à minimiser E. En pratique, on calcule pour chaque paramètre $w \in \theta$ la dérivée partielle $\frac{\partial E}{\partial w}$ et on modifie en conséquence la valeur de w:

$$w \leftarrow w - \eta \frac{\partial E}{\partial w},$$

où $\eta > 0$ est le *pas de gradient*. (Théoriquement, la valeur exacte de η n'a pas d'importance en autant qu'elle soit relativement petite, mais en pratique elle joue

un rôle sur la convergence de l'apprentissage; on fixera sa valeur empiriquement en effectuant quelques expériences préliminaires.) Tel que mentionné plus haut, ces gradients peuvent être calculés *en lot* pour l'ensemble des candidats c_p des exemples d'apprentissage; on modifie alors les paramètres en une fois après avoir examiné la totalité des exemples (*batch learning*). Alternativement, on peut modifier les paramètres après chaque exemple (*descente de gradient stochastique*).

Le calcul des gradients individuels se fait par rétropropagation de l'erreur. Pour les paramètres de la couche cachée w_j, partant de la fonction d'erreur (équation 4.9) et de la couche de sortie du réseau (équation 4.7):

$$\begin{aligned}
\frac{\partial E}{\partial w_j} &= \sum_k (y(c_p) - \hat{y}(c_p)) \frac{\partial y(c_p)}{\partial w_j} \\
\frac{\partial y(c_p)}{\partial w_j} &= (1 - y(c_p)^2) z_j(c_p)
\end{aligned}$$

Pour les paramètres w_{ij} de la couche cachée (équation 4.8):

$$\begin{aligned}
\frac{\partial E}{\partial w_{ij}} &= \sum_k (y(c_p) - \hat{y}(c_p)) \frac{\partial y(c_p)}{\partial w_{ij}} \\
\frac{\partial y(c_p)}{\partial w_{ij}} &= \frac{\partial y(c_p)}{\partial z_j(c_p)} \frac{\partial z_j(c_p)}{\partial w_{ij}} \\
\frac{\partial z_j(c_p)}{\partial w_{ij}} &= (1 - z_j(c_p)^2) x_k(c_p) \\
\frac{\partial y(c_p)}{\partial z_j c_p} &= (1 - y(c_p)^2) w_j
\end{aligned}$$

Nous avons entraîné ainsi et testé plusieurs réseaux de neurones de ce type. Toutes ces expériences ont été faites avec les mêmes données que pour l'évaluation de la section 4.4, à la différence qu'ici, une partie des données était utilisée pour entraîner le réseau, alors que le reste servait à l'évaluation. De façon à obtenir des résultats comparables à ceux des expériences précédentes, nous avons procédé à une validation

croisée (*K-fold cross-validation*), c'est-à-dire que le processus entraînement/test a été répété plusieurs fois sur des tranches de données différentes, de façon à obtenir une évaluation couvrant l'ensemble des données. Cette façon de faire a en outre l'avantage de nous instruire sur la stabilité de l'approche. Toutes les expériences ci-dessous ont été effectuées avec $K = 20$, c'est-à-dire qu'on entraînait sur 950 exemples et qu'on testait sur les 50 exemples restants.

Nous avons expérimenté avec plusieurs tailles différentes de couche cachée, allant de 2 unités seulement à 1000 unités. D'une façon générale, les meilleurs résultats ont été obtenus avec des réseaux comportant 50 unités dans la couche cachée. On a aussi constaté qu'une descente de gradient stochastique permettait une convergence plus rapide et un comportement plus stable du réseau (saturation des fonctions d'activation *tanh*). En général, les réseaux convergeaient après 3 ou 4 itérations sur la base d'exemples, après quoi on commençait à percevoir des signes de sur-entraînement (c'est-à-dire une diminution de l'erreur E à l'entraînement, mais une détérioration des performances sur les tranches de test).

Le tableau 4.6 rapporte la performance de réseaux entraînés sur les différentes valeurs-cible proposées ci-dessus.

Les différentes valeurs-cible proposées présentent toutes des performances très comparables. Globalement, ${y_p}^2|t_p|$ semble celle qui se comporte le mieux, à la fois en termes de couverture, de précision et de la taille des segments proposés. Mais les différences sont tellement minimes qu'en fin de compte, n'importe laquelle de ces valeurs peut faire l'affaire.

Si on compare ces résultats avec ceux présentés aux sections 4.4 et 4.5, on constate que les réseaux sont avantageux par rapport à toutes les alternatives envisagées. Lorsque $|t_p| \geq 3$, les sélections-réseau offrent une couverture comparable à la méthode **max-similitude**, tout en proposant des candidats qui sont typiquement 20% plus longs (les précisions sont également légèrement supérieures). Si on exige d'emblée des séquences plus longues, ($|t_p| \geq 4,5$), on observe une amélioration en couverture et en

		Valeur-cible				
		y_c	$y_c\|t_p\|$	y_p	$y_p\|t_p\|$	${y_p}^2\|t_p\|$
$\|t_p\| \geq 3$	couverture (%)	15,30	14,75	15,03	15,23	15,21
	précision (%)	18,14	17,12	18,29	17,12	17,60
	taille moyenne	4,48	4,58	4,37	4,73	4,59
$\|t_p\| \geq 4$	couverture (%)	11,91	11,62	11,96	12,20	12,25
	précision (%)	14,74	14,32	15,01	14,80	15,22
	taille moyenne	5,58	5,61	5,50	5,69	5,56
$\|t_p\| \geq 5$	couverture (%)	8,72	8,40	8,68	8,75	8,68
	précision (%)	14,42	13,75	14,54	14,29	14,29
	taille moyenne	6,89	6,95	6,80	6,97	6,92

Table 4.6. Performance comparée de réseaux entraînés sur différentes valeurs-cible

précision par rapport à **max-freq** dans l'ordre de 5%.

L'examen des expériences de validation individuelles révèle que les réseaux ci-dessus se comportent de façon relativement stable et que les gains qu'on observe globalement se matérialisent généralement sur les tranches de test individuelles de 50 exemples. Par exemple, lors des 20 entraînements du réseau visant à produire la valeur ${y_p}^2|t_p|$ pour $|t_p| \geq 4$ (dernière colonne dans le tableau 4.6, rangées du milieu), les sélections produites par le réseau ont surclassées la meilleure alternative (**max-freq**) 16 fois sur 20. On observe des comportements comparables lors des autres entraînements.

Malgré tout, les résultats obtenus sont encore très loin du maximum qu'on pouvait espérer, tel qu'il apparaît au tableau 4.5. Aussi, il faut bien admettre que les améliorations par rapport à des méthodes simples telles que **max-freq** sont un peu décevantes. L'explication la plus plausible réside dans la difficulté de la tâche,

dont nous avons déjà fait état à la section 4.3. Du point de vue de l'entraînement d'un réseau de neurones, cette difficulté se manifeste dans la variance des données d'entraînement: un même candidat proposé pour deux phrases distinctes P_1 et P_2, peut très bien se révéler utile dans la traduction de l'une mais pas dans celle de l'autre, ce qui entraîne des valeurs-cible parfois radicalement différentes pour des caractéristiques similaires. Dans ces conditions, il devient très difficile pour le réseau de départager effectivement les "bons" candidats des "mauvais".

4.6.5 Entraînement sur le classement

Nous avons exploré la possibilité d'élaborer une méthode d'entraînement du réseau qui nous affranchirait dans une certaine mesure des problèmes dûs à la variance excessive des valeurs-cible. Par exemple, on peut envisager une méthode d'entraînement dans laquelle le réseau établit lui-même ses propres valeurs-cible, en visant une ressemblance avec la sélection-oracle. Ce genre d'approche a notamment été utilisé avec succès dans une application de reconnaissance de l'écriture cursive [26].

Soit une phrase P, sa traduction T_P, un ensemble de candidats C_P. On peut calculer la couverture-source optimale au moyen de l'algorithme **max-prob**, en utilisant les sorties du réseau comme fonction de score; appelons cette couverture $R_P = \{c_{p_1}, ..., c_{p_m}\} \in C_P$. Le score calculé par **max-prob** pour R_P est:

$$\text{Score}(R_P) = -\sum_{c_p \in R_P} \log(y(c_p))$$

Par ailleurs, on peut également déterminer la meilleure couverture-cible qui puisse être obtenue pour T_P à partir des candidats de C_P, que nous appelons la *sélection-oracle*: $R^*_P = \{c^*_{p_1}, ..., c^*_{p_n}\} \in C_P$. Puisque **max-prob** a préféré R_P à R^*_P, on en conclut que le score de R_P est inférieur (dans un cadre de minimisation) à celui qu'aurait obtenue la sélection-oracle R^*_P:

$$\text{Score}(R^*_P) = -\sum_{c_p \in R^*_P} \log(y(c_p)) > \text{Score}(R_P)$$

Pour en arriver à un réseau qui produit des valeurs menant à la bonne sélection (c'est-à-dire R_P^*), on veut renverser cette tendance: on aimerait modifier les paramètres du réseau de façon à faire simultanément augmenter $\text{Score}(R_P)$ et diminuer $\text{Score}(R_P^*)$. Une façon d'aller dans cette direction est de choisir une fonction d'erreur

$$E = \text{Score}(R_P^*) - \text{Score}(R_P)$$

Intuitivement, cette façon de faire est séduisante: puisque l'entraînement se base sur le résultat final visé (la sélection des candidats), il concentre ses énergies sur les erreurs de la procédure de recherche. Les contributions des candidats identiques de R_P et R_P^* s'annulent dans le calcul du gradient, de telle sorte que seuls les candidats différents (*bruits* et *silences*) y contribuent effectivement.

L'entraînement du réseau se fait ici encore par descente de gradient. On commence par calculer pour chaque phrase P la sélection effectuée par le réseau R_P et la sélection-oracle R_P^*. Les gradients se calculent alors ainsi:

$$\begin{aligned}
\frac{\partial E}{\partial \theta} &= \frac{\partial \text{Score}(R_P^*)}{\partial \theta} - \frac{\partial \text{Score}(R_P)}{\partial \theta} \\
&= -\sum_{c_p \in R_P^*} \frac{\partial \log(y(c_p))}{\partial \theta} + \sum_{c_p \in R_P} \frac{\partial \log(y(c_p))}{\partial \theta} \\
&= -\sum_{c_p \in R_P^*} \frac{1}{y(c_p)} \frac{\partial y(c_p)}{\partial \theta} + \sum_{c_p \in R_P} \frac{1}{y(c_p)} \frac{\partial y(c_p)}{\partial \theta}
\end{aligned}$$

À partir de ce point, les $\frac{\partial y(c_p)}{\partial \theta}$ se calculent exactement comme précédemment.

Il y a différents problèmes avec cette approche. D'abord, la procédure a naturellement tendance à sous-utiliser les données d'entraînement, puisque pour chaque C_P, il n'y a finalement qu'une poignée de candidats qui servent ultimement au calcul des gradients.

Plus grave, il y a un problème de convergence dans cette procédure: lorsque le réseau s'approche de la solution optimale, il est possible qu'il produise des sélections

qui, sans être égales aux sélections-oracle, engendrent néanmoins des couvertures-cible acceptables. Les candidats ainsi sélectionnés ont souvent des caractéristiques qui se distinguent difficilement de celles des candidats des sélections-oracle. Néanmoins, la procédure considère systématiquement qu'un candidat qui n'appartient pas à R^*_P est "mauvais", quelles que soient ses caractéristiques et son utilité réelle, et pénalise le réseau en conséquence. Ceci entraîne le réseau dans un cycle de "montagnes russes" et empêche la convergence vers la solution souhaitée.

Ces problèmes nous amènent à considérer des alternatives qui, sans fixer a priori des valeurs-cible pour le réseau, font une meilleure utilisation des données d'entraînement, tout en ne pénalisant pas systématiquement les solutions sous-optimales.

La fonction de score utilisée par les procédures de sélection proposées ici détermine un ordre parmi les candidats d'un même ensemble de candidats C_P. Par exemple, si on dispose d'un estimé fiable de $\Pr(t_p|p)$, on peut établir a priori quel est le candidat le plus prometteur de chaque ensemble C_{ij}, et ne considérer que cet unique candidat lors de la procédure de recherche, avec exactement les mêmes résultats à la sélection finale.

À défaut d'un tel estimé, on peut espérer qu'une fonction de score qui imposerait aux candidats de chaque ensemble C_{ij} un classement identique à celui produit par $\Pr(t_p|p)$ donnerait lieu à des sélections comparables (on ne peut garantir une sélection optimale, parce que le score joue un rôle non seulement dans le classement des candidats entre eux, mais également dans le calcul de la sélection finale.)

Dans cette perspective, nous élaborons ci-dessous une procédure d'entraînement d'un réseau de neurone qui vise à minimiser les différences entre le classement des données produit par le réseau et un classement-cible. Cette procédure s'inspire d'une approche pour l'entraînement des *Support Vector Machines* (SVM) sur une base similaire [50]

Soit un ensemble $C_P = c_1, ..., c_n$; un *classement* des éléments de C_P est une relation $r \in C_P \times C_P$. On note $c_i <_r c_j$ l'appartenance d'un certaine paire d'éléments

(c_i, c_j) à ce classement; en d'autres termes, $c_i <_r c_j$ signifie que c_i apparaît "avant" c_j dans le classement r. Cette relation est transitive ($c_i <_r c_j$ et $c_j <_r c_k$ impliquent que $c_i <_r c_k$) et non-commutative ($c_i <_r c_j$ si et seulement si $c_j \not<_r c_i$; on parle donc d'un classement *strict*). Un tel classement de tous les éléments de C_P contiendra $\binom{n}{2}$ paires.

Il existe différentes façons de comparer deux classements r_a et r_b, mais la méthode la plus répandue est la mesure τ de Kendall [55]: il suffit en fait de compter le nombre d'éléments identiques P entre r_A et r_B, et le nombre d'éléments différents Q. τ s'exprime alors:

$$\tau(r_A, r_B) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{n}{2}}$$

En somme, τ varie entre -1 et 1; $\tau = 1$ désigne des classements identiques, alors que $\tau = -1$ désigne des classements inverses.

Si les classements r_A et r_B sont issus de deux fonctions f_A et f_B, c'est-à-dire que $f_A(c_i) < f_A(c_j) \Leftrightarrow c_i <_{r_A} c_j$, on peut calculer ainsi le nombre d'éléments différents Q entre r_A et r_B:

$$Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} |\text{signe}(f_A(c_i) - f_A(c_j)) - \text{signe}(f_B(c_i) - f_B(c_j))| \tag{4.10}$$

où la fonction signe(x) prend la valeur -1 si $x < 0$ et $+1$ si $x \geq 0$.

On peut mettre sur pied une procédure d'entraînement pour un réseau de neurones qui minimise directement τ, en établissant une fonction d'erreur s'inspirant directement de l'équation 4.10 ci-dessus:

$$E = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E_{ij} \tag{4.11}$$

où E_{ij} est l'erreur de classement commise par le réseau entre les candidats c_i et c_j:

$$E_{ij} = |\text{signe}(y(c_i) - y(c_j)) - \text{signe}(\hat{y}(c_i) - \hat{y}(c_j))|\,,$$

Ici, $y(c)$ est la sortie du réseau sur le candidat c et $\hat{y}(c)$ la valeur produite par une fonction de score produisant un classement correct des candidats (par exemple, une des variantes du degré de couverture-cible y_p proposées à la section 4.6.4).

L'entraînement d'un réseau qui chercherait à minimiser cette fonction d'erreur par descente de gradient comporte certains problèmes: d'une part, les fonctions *valeur-absolue* et *signe* ne sont pas dérivables en tous points, puisqu'elles comportent toutes deux une discontinuité au point 0; d'autre part, la fonction *signe* a une dérivée nulle en tous ses autres points.

Le premier problème (valeur absolue) n'est pas majeur, puisque la descente de gradient peut très bien s'accommoder d'un dérivée constante de part et d'autre du point zéro, et nulle en zéro. Toutefois, on préfèrera la remplacer par un carré, qui offre l'avantage d'une pente s'adoucissant à l'approche du minimum. Quant à la fonction *signe*, on peut la remplacer par une tangeante hyperbolique: la dérivée de cette fonction est bien définie en tout point, et son comportement général s'apparente à celui de *signe*. En somme:

$$E_{ij} = (\tanh(\Delta_{ij}) - \tanh(\hat{\Delta}_{ij}))^2, \tag{4.12}$$

où Δ_{ij} et $\hat{\Delta}_{ij}$ désignent les différences de score $y(c_i) - y(c_j)$ et $\hat{y}(c_i) - \hat{y}(c_j)$ respectivement.

Comme on le voit à la figure 4.5, si les valeurs de y et $\hat{y}$ se situent dans l'intervalle $[-1, 1]$, la surface d'erreur de E_{ij} est minimale le long de l'axe $x = y$, c'est-à-dire lorsque les deux fonctions y et $\hat{y}$ attribuent le même écart de score pour les candidats c_i et c_j, et la pente à l'entour du minimum s'accentue à mesure qu'on s'en éloigne.

Lors de l'entraînement, on calcule d'abord les valeurs $y(c)$ et $\hat{y}(c)$ pour tous les candidats c d'un ensemble de candidats. On peut alors procéder au calcul des E_{ij} et des gradients, qui se calculent alors ainsi:

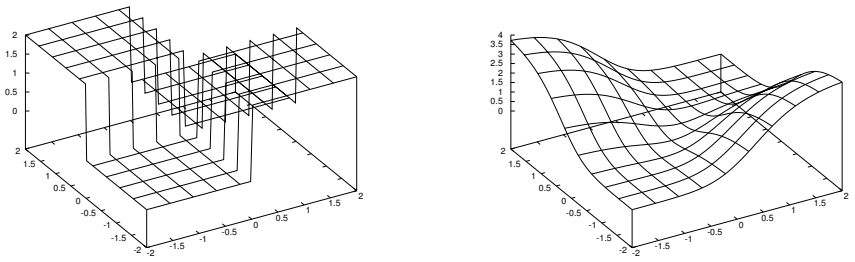

Figure 4.5. Fonctions d'erreur sur le classement: à gauche, une fonction mesurant directement l'erreur de classement; à droite, une approximation dérivable de cette erreur

$$
\begin{aligned}
\frac{\partial E}{\partial \theta} &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{\partial E_{ij}}{\partial \theta} \\
\frac{\partial E_{ij}}{\partial \theta} &= 2(\tanh(\Delta_{ij}) - \tanh(\hat{\Delta}_{ij}))(1 - \tanh(\Delta_{ij})^2)(\frac{\partial \Delta_{ij}}{\partial \theta}) \\
\frac{\partial \Delta_{ij}}{\partial \theta} &= \frac{\partial y(c_i)}{\partial \theta} - \frac{\partial y(c_j)}{\partial \theta}
\end{aligned}
$$

On repasse alors sur chaque c_i avec le réseau, et l'on rétropropage sur la sortie du réseau l'erreur Δ_i relative à ce candidat, c'est-à-dire tous les termes de $\frac{\partial E}{\partial \theta}$ qui font intervenir c_i:

$$\Delta_i = 2 \sum_{j=1...n, j \neq i} (\tanh(\Delta_{ij}) - \tanh(\hat{\Delta}_{ij}))(1 - \tanh(\Delta_{ij})^2)$$

À l'opposé de la méthode d'entraînement basée sur les différences de scores, cette procédure-ci est plus coûteuse que l'entraînement direct sur les valeurs-cible (section 4.6.4), parce que le calcul de l'erreur et des gradients fait intervenir un nombre de termes proportionnel au carré du nombre de candidats dans chaque ensemble de candidats C_{ij} associé aux séquences p_{ij} d'une phrase P. Par contre, elle capitalise mieux sur les données d'entraînement, qui contribuent toutes aux calculs des gradients.

Comme pour l'entraînement direct sur les valeurs-cible, les paramètres peuvent être modifiés *en lot* après chaque phrase P, ou immédiatement après le calcul de chaque Δ_i. Comme c'était le cas précédemment, on constate que cette dernière méthode assure une meilleure stabilité en pratique et une convergence plus rapide. Par contre, l'usage plus intensif des données semble se traduire par une tendance plus prononcée au surentraînement. Lors d'expériences préliminaires, nous avons établi qu'il était préférable de d'arrêter l'entraînement après deux itérations seulement.

Le *classement-oracle* nécessaire a l'entraînement peut être obtenu au moyen de n'importe laquelle des valeurs-cible proposées à la section 4.6.4, ou autre.

Le tableau 4.7 résume les résultats obtenus par validation croisée ($K = 20$) sur les données de la section 4.4. Pour fins de comparaison, nous reproduisons également les résultats obtenus par entraînement direct sur la même valeur-cible, de même que les sélections effectuées avec les fonctions de score $mt(c)$ et $F(c)$. Toutes ces expériences ont été effectuées avec la procédure **max-prob**.

Contrairement à ce qu'on avait observé avec l'entraînement direct, certaines valeurs-cible se comportent beaucoup mieux que d'autres. Ceci est tout à fait naturel: en particulier, y_c impose un ordre dichotomique sur l'ensemble des candidats (les "bons" et les "mauvais"), ce qui entraîne un comportement très instable de la fonction d'erreur. Ce phénomène semble d'autant plus accentué que le nombre de candidats à traiter est grand ($|t_p| \geq 3$). Incidemment, un phénomène similaire se produit avec y_p. Ceci s'explique par le fait que, en pratique, y_p tend à partitionner les candidats d'un même C_{ij} en un petit nombre de classes d'équivalences. Le facteur $|t_p|$ dans le calcul des valeurs-cible tend à répartir plus uniformément les candidats, ce qui facilite la convergence.

Globalement, les meilleurs résultats sont obtenus avec ${y_p}^2|t_p|$. Pour cette valeur-cible, les résultats obtenus sont comparables avec ceux du Tableau 4.6 (entraînement direct) ou très légèrement inférieurs. Il ne semblerait donc pas y avoir de gains immédiats à l'utilisation de cette technique d'entraînement. Il faut dire que la

		Valeur-cible				
		y_c	$y_c\|t_p\|$	y_p	$y_p\|t_p\|$	${y_p}^2\|t_p\|$
$\|t_p\| \geq 3$	couverture (%)	6,79	14,70	6,74	14,77	14,82
	précision (%)	7,01	17,30	6,24	16,80	17,23
	taille moyenne	5,13	4,52	5,70	4,67	4,57
$\|t_p\| \geq 4$	couverture (%)	9,49	11,60	6,36	11,81	12,01
	précision (%)	11,48	14,34	6,82	14,32	14,80
	taille moyenne	5,70	5,58	6,40	5,69	5,60
$\|t_p\| \geq 5$	couverture (%)	7,54	8,69	7,81	8,51	8,58
	précision (%)	12,44	14,33	12,62	13,75	14,00
	taille moyenne	6,88	6,90	7,04	7,05	6,98

Table 4.7. Performance comparée de réseaux entraînés sur le classement produit par différentes valeurs-cible

tangeante hyperbolique, telle qu'utilisée dans la fonction d'erreur ci-dessus, a possiblement pour effet de faire converger le réseau vers les mêmes valeurs que l'entraînement direct. En effet, on sait que pour des valeurs proches de zéro, $\tanh(x) \approx x$. Or les différences Δ_{ij} sont précisément le plus souvent dans cette région, surtout à mesure que le réseau s'améliore, de telle sorte que:

$$\begin{aligned} E_{ij} &= (\tanh(\Delta_{ij}) - \tanh(\hat{\Delta}_{ij}))^2 \\ &\approx ((y(c_i) - \hat{y}(c_i)) + (\hat{y}(c_j) - y(c_j)))^2 \end{aligned}$$

En fin de compte, il est fort probable que cette variante d'une fonction d'erreur quadratique guide le réseau vers des paramètres qui se rapprochent de ceux des entraînements directs.

Par ailleurs, on a déterminé qu'un entraînement de type stochastique (modifica-

tion des paramètres après chaque exemple) permettait une convergence plus rapide et plus stable qu'un entraînement *en lot*, et qu'il était préférable d'arrêter l'entraînement après deux itérations seulement. En fait, dans bien des cas, une seule itération aurait été suffisante. Ceci ouvre la voie à un entraînement *en continu* du réseau, à même l'application d'aide à la traduction. En effet, on peut imaginer un mécanisme qui modifie les paramètres du réseau à la volée, en tenant compte directement des interventions de l'utilisateur (par exemple, le choix que fait l'utilisateur parmi la sélection de candidats proposée et la nature de la traduction ultimement produite). Ce genre d'adaptation dynamique du système pourrait éventuellement s'avérer très utile pour ce type d'application.

4.7 Discussion

Globalement, les résultats présentés dans les sections précédentes sont relativement encourageants. Si on se rapporte aux résultats des expériences préliminaires présentés à la section 2.6, des couvertures du texte-cible allant de 8% à 15% (tableau 4.6), si minimes qu'elles puissent paraître à première vue, constituent quand même une amélioration très notoire lorsqu'on les compare à la couverture qu'on pouvait espérer avec des phrases complètes (environ 0,5%). Ceci représente somme toute 15 à 30 fois plus de matériel récupéré dans la MT.

On peut toutefois se demander si les résultats obtenus permettent effectivement d'envisager une application réelle, surtout en regard des taux de précision observés des sélections, qui excède rarement les 15%. En pratique, ceci signifie que pour s'épargner la saisie de 3 mots, l'utilisateur doit en lire 20.

Ces résultats doivent néanmoins être interprétés à la lumière de différentes caractéristiques de notre évaluation. Premièrement, on suppose implicitement qu'il n'existe pour chaque phrase-source du document-test qu'une seule traduction possible (la *traduction oracle*), et que toute autre traduction est incorrecte. Afin d'en

arriver à une meilleure estimation du potentiel des propositions, il faudrait considérer un ensemble de traductions possibles pour chaque phrase en langue-source: on pourrait alors soit calculer des moyennes des différentes statistiques (ce qui répondrait à la question: "Comment différents traducteurs utiliseraient-ils ces propositions?") ou les statistiques associées à la traduction-oracle pour laquelle on obtient la couverture maximale (ce qui répondrait à la question: "Comment un traducteur pourrait-il utiliser les propositions de manière optimale?"). Ce genre de méthode d'évaluation est de plus en plus utilisé en traduction automatique [78, 81], mais il requiert bien entendu un corpus de traductions multiples, qui ne nous était pas disponible au moment de l'évaluation.

Ensuite, notre évaluation est basée sur un scénario *tout-ou-rien*, c'est-à-dire qu'on fait la supposition que l'utilisateur doit soit accepter une proposition telle qu'elle est, soit la rejeter intégralement. Or, il est clair que dans un contexte d'utilisation réelle, un utilisateur pourrait choisir de réutiliser une proposition dont la couverture-cible (y_p) est inférieure à 100%, quitte à la modifier pour l'adapter à son nouveau contexte.

Une évaluation plus réaliste devrait donc tenir compte de cette possibilité. Par exemple, dans une série d'expériences similaires à celles rapportées ici, menée conjointement avec Philippe Langlais [104], nous avons examiné trois scénarios d'utilisation différents :

tout-ou-rien Un scénario d'utilisation essentiellement identique à celui rapporté ici, c'est-à-dire qu'on considère que l'utilisateur prend une proposition telle qu'elle est, ou ne la prend pas du tout.

copier-coller On suppose que l'utilisateur est disposé à effectuer des opérations de *copier-coller* afin de réutiliser les séquences proposées, c'est-à-dire de prélever, dans chaque proposition, la sous-séquence contiguë lui permettant de couvrir la plus grande partie possible de la traduction-oracle.

préfixe Il s'agit en quelque sorte d'un scénario mitoyen entre les deux précédents : on suppose que l'utilisateur est disposé à réutiliser des préfixes des propositions faites par le système. Concrètement, ceci revient à insérer intégralement une proposition dans la traduction, puis à en effacer un ou plusieurs mots, en partant de la droite, avant de poursuivre.

Les résultats obtenus lors de ces expériences ont permis d'établir qu'en autorisant l'utilisation de propositions dont seul un préfixe couvrait la traduction-oracle, on augmentait de près de 45% la couverture de cette dernière, alors qu'en autorisant les *copier-coller*, cette même couverture doublait (des augmentations similaires ont été observées concernant la précision des sélections).

Cette évaluation demeurait toutefois très superficielle, puisqu'elle ne considérait finalement que deux types de manoeuvres, et ne tenait aucun compte de l'ampleur des modifications apportées aux propositions, ni de leur coût en termes de manipulations. Idéalement, une évaluation plus réaliste devrait se baser sur une modélisation beaucoup plus fine de l'utilisateur. Par exemple, on pourrait chercher à estimer la probabilité qu'un utilisateur utilise une proposition donnée, en fonction de l'ampleur et de la nature des manoeuvres requises pour l'adapter à la traduction-oracle. Les différentes métriques (couverture, précision, etc.) pourraient alors être soit pondérées avec ces probabilités, soit calculées pour un "utilisateur moyen" (c'est-à-dire qu'on comptabilise toutes les séquences proposées pour lesquelles la probabilité ci-dessus est supérieure à 1/2).

Finalement, l'évaluation présentée ici ne tient aucun compte de l'apport cognitif du système, c'est-à-dire de la possibilité que les propositions oriente le traducteur vers certaines formulations (section 2.4). Cet apport est potentiellement non-négligeable, surtout pour des traducteurs qui travaillent sous pression, et pour lesquels la perspective d'économiser des frappes l'emporte sur l'ambition de produire la traduction "idéale".

En somme, les taux de couverture et de précision présentés dans les sections précédentes doivent en fait être vus comme une borne inférieure de la réutilisabilité des candidats proposés. En pratique, un examen visuel des sorties du système révèle que bon nombre des propositions sont en fait tout à fait pertinentes et potentiellement réutilisables. La figure 4.6 en donne quelques exemples particulièrement éloquents. Il s'agit des ensembles de candidats de plus de 3 mots ($t_p \geq 4$) sélectionnés pour quelques phrases du document-test, avec un RNM entraîné directement sur des valeurs-cible ${y_p}^2|t_p|$ (voir la section 4.6.4).

En fin de compte, seule une évaluation "sur le terrain" pourra fournir une réponse objective à la question de l'utilisabilité. À cet égard, on voudrait certainement s'inspirer du travail de Langlais et al. [59], qui visait à mesurer les gains de productivité que des traducteurs pouvaient espérer tirer du système *TransType* [33], un système de traduction automatique interactive. Les nombreuses similitudes entre l'interface-utilisateur de ce système et celles des SMT existants suggèrent qu'une évaluation sur des bases similaires serait tout à fait envisageable. Une telle évaluation nécessiterait toutefois la mise sur pied d'une interface-utilisateur pour le système proposé ici, qui reste encore à faire.

4.8 Conclusions

Le problème de la sélection des candidats consiste à choisir, parmi tous les segments de texte-cible extraits d'une mémoire de traduction pour une certaine phrase-source, ceux qui sont le plus susceptibles d'être utiles au traducteur. Cette sélection est essentielle si on veut éviter de submerger le traducteur de propositions inutiles ou redondantes.

Dans ce chapitre, nous avons développé une méthode générale de sélection des candidats à l'intérieur d'un cadre probabiliste: on considère $\Pr(c_p|P)$, la probabilité que le candidat c_p soit utile au traducteur pour produire une traduction de la phrase

	phrase-source		répérages-cible sélectionnés
1.	[As it is now] ,	→	[dans sa forme actuelle]
	[our equalization system]	→	[notre système de péréquation]
	is broken .		
2.	[It is unfortunate that we are seeing]	→	[il est regrettable que]
	[an increased level]	→	[un niveau accru de]
	of control		
	[of committees by]	→	[des délibérations des comités]
	[the Liberal whip]	→	[le whip du parti libéral]
	[as an extension of]	→	[comme un prolongement de]
	[the Prime Minister 's Office and]	→	[cabinet du premier ministre et]
	the dysfunctionality		
	[which is inherent in]	→	[qui est inhérente au]
	that .		
3.	[Mr. Scott Brison] :	→	[M. Scott Brison (]
	[Mr. Speaker , this will be]	→	[Monsieur le Président , c' est]
	[like shooting fish in a barrel].	→	[comme tirer sur des poissons dans]
4.	[First , I thank the hon. member]	→	[Premièrement , je remercie le député]
	[for his softball question].	→	[de sa question facile]
5.	[Mr. Speaker , even]	→	[Monsieur le Président , même]
	alcoholics		
	[know that the first step]	→	[que le premier pas]
	to recovery		
	[is to admit]	→	[c' est d' admettre]
	[that they have a problem].	→	[qu' ils ont un problème]
6.	That [said , we]	→	[je l' ai dit ,]
	[cannot revert to]	→	[nous ne pouvons pas revenir aux]
	Tory policies of destroying the resource ,		
	[as happened on]	→	[nous est arrivé avec]
	[the cod stocks]	→	[Les stocks de morue]
	[of the east coast],	→	[de la côte est]
	simply		
	[for political reasons].	→	[pour des raisons politiques]

Figure 4.6. Exemples de sélections

P. La méthode **max-prob** recherche parmi un ensemble de candidats C_P le sous-ensemble R_P^* qui recouvre le mieux la phrase-source P, tout en maximisant la probabilité $\Pr(R_P^*|P)$, sous l'hypothèse que les candidats sont indépendants les uns des autres. Cette probabilité est elle-même estimée au moyen d'un modèle statistique de traduction (modèles IBM).

Nous avons également mis sur pied un protocole d'évaluation visant à établir la productivité des sélections proposées. Ce protocole nous a permis de déterminer que des modèles de traduction employés ne fournissaient pas nécessairement le meilleur estimé de $\Pr(c_p|P)$ pour cette tâche, et que d'autres caractérisations des candidats - notamment la fréquence relative d'un candidat dans la mémoire de traduction, ou même une simple mesure de similitude entre le segment p de P pour lequel le candidat c_p a été proposé et le repérage-source s_p correspondant dans la mémoire de traduction - menaient parfois à des sélections plus judicieuses.

Partant de là, nous avons proposé une méthode pour évaluer le potentiel d'un candidat, combinant différents attributs descriptifs de ceux-ci en un score unique, au moyen d'un réseau de neurones multi-couche. Nous avons en outre proposé deux méthodes distinctes d'entraînement du réseau: la première (entraînement direct) visait à minimiser directement les différences entre les valeurs produite par le réseau et des valeurs-cible reflétant l'utilité effective des candidats dans une traduction existante; la seconde (entraînement sur le classement) cherchait plutôt à minimiser les différences dans le classement des candidats entre eux.

Par une série d'expériences, nous avons montré comment ces méthodes permettaient d'obtenir des sélections plus judicieuses des candidats. Malgré tout, nous avons dû constater que le problème de la sélection est difficile et que les meilleurs résultats obtenus étaient encore loin des solutions optimales.

En fin de compte, le problème de la sélection s'apparente de très près au problème de la génération du texte-cible en traduction automatique. On peut donc penser qu'une solution plus satisfaisante doit forcément passer par une meilleure modélisation

du processus de traduction lui-même. En particulier, une meilleure modélisation des séquences longues dans les modèles statistiques tels que ceux utilisés ici éviterait la sous-estimation systématique de celles-ci par le modèle.

À plus court terme toutefois, une meilleure caractérisation des repérages candidats permettrait sans doute une meilleure sélection. En particulier, la caractérisation proposée ne tient compte que de la ressemblance littérale entre la sous-séquence p de la phrase-source pour laquelle un candidat c_p est proposé et le repérage-source s_p de la phrase S de la mémoire de traduction d'où c_p est extrait. Une caractérisation plus fine pourrait tenir compte des contextes environnants de p et s_p, par exemple en mesurant la similitude générale entre P et S. On pourrait également faire intervenir des considérations d'ordre syntaxique: dans quelle mesure les analyses syntaxiques de p et s_p (étiquettage morpho-syntaxique, tronçonnage, etc.) sont-elles similaires? Ces considérations pourraient éventuellement s'étendre aux contextes des segments: les mots précédant ou suivant p et s_p ont-ils des caractéristiques linguistiques comparables?

D'autres méthodes de régression que les réseaux de neurones pourraient être utilisées pour estimer le potentiel des candidats. On pense notamment aux *Support Vector Machines* et autres méthodes *à noyau*. Certaines expériences préliminaires sont en cours dans cette direction.

Les résultats de nos expériences ont également mis en lumière certaines lacunes de notre méthode d'évaluation. Une mesure de la qualité des sélections sur la base d'une unique *traduction-oracle* tend forcément à sous-estimer le potentiel des sélections. À l'instar de ce qui se fait en traduction automatique, une évaluation faisant intervenir des références multiples permettrait une estimation plus juste de la performance attendue. En parallèle, un corpus de traductions multiples permettrait probablement d'améliorer aussi la performance des réseaux de neurones, en atténuant la variance des données d'apprentissage. Ceci met en relief l'importance de développer ce genre de ressource.

Chapitre 5

CONCLUSIONS

Dans cette thèse, nous avons exploré la possibilité de mettre sur pied un système de mémoire de traduction sous-phrastique, c'est-à-dire capable de proposer des segments de texte en langue-cible de taille variable, susceptibles d'être utiles à un traducteur dans la production d'une traduction.

Nous avons d'abord examiné la question de l'unité de traduction la plus appropriée pour ce genre de système. En nous appuyant notamment sur les résultats d'une étude sur l'utilisation du système TransSearch, nous avons proposé que ce genre de système pourrait chercher à extraire la traduction de segments contigus du texte-source, correspondant à des séquences de tronçons syntaxiques. Nous avons exposé comment ce genre d'extraction pouvait être effectué de façon efficace. Nous avons également démontré par quelques expériences l'ampleur des gains que l'unité de traduction proposée permettait d'envisager en terme de réutilisation des traductions.

Nous nous sommes ensuite penché sur la question du repérage automatique des traductions, c'est-à-dire la recherche dans les paires de phrases extraites de la mémoire de traduction du segment en langue-cible constituant la traduction du segment d'intérêt dans la portion en langue-source. Nous avons montré comment, dans le contexte spécifique de cette application, on pouvait intégrer à certaines méthodes statistiques d'alignement de mots des contraintes de contiguïté et de compositionnalité. Par différentes expériences, nous avons montré comment ces contraintes permettaient d'améliorer substantiellement la qualité des repérages.

Nous avons ensuite présenté un problème nouveau, celui de la sélection des propositions, visant à déterminer, parmi tous les segments en langue-cible extraits d'une mémoire de traduction pour un texte donné en langue-source, le sous-ensemble de

ces segments qui est le plus susceptible d'être utile à un traducteur. Nous avons exposé comment ce problème pouvait être présenté dans un cadre probabiliste, et comment différentes caractérisations des segments-cible pouvaient servir à approximer une sélection optimale. Nous avons finalement montré comment ces caractérisations pouvaient être combinées au moyen de réseaux de neurones, afin d'améliorer les sélections.

Les approches proposées et les résultats obtenus lors de l'étude de ces différentes questions, bien que développés dans le contexte d'une application spécifique, ont possiblement des débouchés plus généraux. En premier lieu, l'analyse de l'utilisation du concordancier bilingue *TransSearch*, par l'intermédiaire du registre des requêtes soumises par ses utilisateurs, est clairement porteuse d'un intérêt qui va bien au-delà des systèmes de mémoires de traduction. On pense notamment aux travaux de recherche visant à étudier le processus de traduction d'un point de vue psycholinguistique. Depuis quelques années déjà, les chercheurs dans ce domaine se sont tournés vers des méthodes expérimentales empiriques pour analyser ce processus, notamment les "protocoles de verbalisation" (*think-aloud protocols*, ou *TAP*). Or ces méthodes ont fait l'objet de nombreuses critiques, et leur validité même a été sérieusement remise en question, principalement au regard de la quasi-impossibilité de maintenir des conditions d'expérimentation "neutres", c'est-à-dire telles que le regard de l'observateur n'affecte pas le processus qu'il tente d'observer. L'analyse des registres de requêtes de *TransSearch*, sans se prêter à la même flexibilité qu'un protocole expérimental fait sur mesure, constitue néanmoins un point de vue alternatif sur le travail des traducteurs, sans s'exposer aux critiques formulées à l'égard de l'objectivité des protocoles de verbalisation. Elle présente en outre un avantage notoire sur ces derniers, de par l'imposante quantité de données disponible, littéralement des millions de requêtes.

Les méthodes de repérage de traduction, bien que peu étudiées dans la littérature (il semble que les chercheurs aient préféré s'attaquer au problème plus général de l'alignement des mots), ont des applications qui débordent le cadre des systèmes de mémoire de traduction classiques. L'une des premières qui vient à l'esprit est

le système *TransSearch*. Présentement, ce système n'a pas la capacité d'identifier la région du texte-cible qui correspond à l'expression en langue-source recherchée par l'utilisateur. L'ajout d'un mécanisme de repérage du type de ceux présentés ici représenterait une importante amélioration à cette application.

Par ailleurs, l'apport des contraintes de compositionnalité dans la performance de ces mêmes méthodes démontre l'importance de cette notion. Jusqu'à maintenant, et à quelques exceptions près, elle avait plus ou moins été évacuée dans les récents travaux de modélisation statistique de la traduction. Par ailleurs, des chercheurs ont déjà souligné l'importance de déterminer le "seuil de la compositionnalité de la traduction", c'est-à-dire la limite en-deça de laquelle ce principe ne s'applique plus, en tentant de répertorier, par exemple, les "suites non-compositionnelles" dans une langue ou un corpus de texte spécifique. Ces travaux, comme ceux présentés ici, soulignent l'importance de cette notion et le besoin de l'intégrer de façon plus fine dans les modèles de traduction.

Le problème de la sélection finale des segments-cible candidats à être présentés au traducteur est, bien entendu, spécifique au système de mémoire de traduction présenté ici. On peut toutefois voir qu'il existe des rapprochements possibles entre celui-ci et le problème plus général de la génération dans les systèmes de traduction automatique. Ce parallèle est particulièrement évident dans certaines approches à la TA basées sur les exemples, notamment des approches telles que celle proposée par Brown [18], qui font intervenir très peu de prétraitement de nature linguistique sur la base d'exemples.

Par ailleurs, dans le cadre de travaux plus récents, nous avons effectué certaines expériences visant à évaluer le potentiel d'une fusion entre une mémoire de traduction et un système de TA statistique [60]. Dans un tel contexte, avant d'effectuer la traduction d'une nouvelle phrase, on extrait des segments-cible pertinents d'une mémoire de traduction, qui sont ensuite fournis au décodeur (la procédure de génération) du système de TA; ce dernier essaie alors de construire une traduction de probabilité

maximale “à l’entour” de ces segments. Les segments issus de la mémoire de traduction servent donc en quelque sorte à construire des “canevas” de départ, que le décodeur cherche à compléter. Bien que ces expériences ne soient pas entièrement concluantes, il semble néanmoins exister des situations où un tel mécanisme permet d’améliorer la qualité des traductions. Par contre, le nombre de canevas à examiner croît très rapidement quand la taille des ensembles de candidats extraits de la MT augmente. Des mécanismes de sélection tels que décrits ici permettent alors de réduire efficacement l’espace de recherche.

D’une façon générale, les résultats rapportés au chapitre 4 quant à la couverture obtenue pour la traduction d’un nouveau texte sont très encourageants et démontrent l’ampleur du potentiel de récupération inexploité dans les systèmes de mémoire de traduction existants. Ces résultats sont d’autant plus convaincants lorsqu’on considère, pour le type de documents utilisés dans nos expériences, la faible proportion des phrases complètes qu’on peut retrouver intégralement dans la mémoire de traduction (section 2.6). Globalement, le fait de rechercher des séquences de tronçons syntaxiques plutôt que des phrases complètes permet de multiplier le taux de récupération par un facteur variant entre 15 et 30, dépendant des préférences de l’utilisateur quant à la longueur minimale des segments de texte proposés.

Toutefois, les faibles taux de précision observés (la proportion du matériel proposé qui s’avère finalement utilisé) laissent à penser que les mécanismes proposés pourraient encore faire l’objet d’améliorations. Nous avons souligné par exemple certaines défaillances de la tronçonneuse utilisée pour l’extraction des séquences au chapitre 2. Les procédures de repérages présentées au chapitre 3 reposent par ailleurs sur des modèles de traduction qui ne représent pas nécessairement le dernier cri en la matière.

Mais d’une façon plus générale, il serait instructif d’effectuer un examen plus poussé des sorties du système, afin possiblement de dégager une meilleure caractérisation de ce qui distingue les segments effectivement réutilisés. Par exemple, on a déjà ob-

servé que certaines séquences de tronçons apparaissent plus souvent que d'autres dans les requêtes soumises au système TransSearch (notamment les suites *NP*, *PP-NP*, *VP*, *VP-NP*, etc.). Il est possible qu'un phénomène analogue soit observable au niveau de l'utilisation des segments proposés, et que certaines séquences de tronçons s'avèrent systématiquement plus productives que d'autres. Si tel était le cas, on pourrait en tenir compte soit dans le processus d'extraction initial, soit dans le processus de sélection des candidats.

D'autre part, il y a plusieurs facteurs dont le mécanisme de sélection des candidats ne tient présentement pas compte, et qui pourraient jouer un rôle important. Par exemple, on n'a pas pris en compte le degré d'adéquation entre les tronçonnages du point de mire (le segment du texte à traduire pour laquelle on recherche une traduction) et du repérage-source (le segment-source correspondant dans la mémoire de traduction), ou encore entre les tronçonnages des repérages-source et cible. L'intégration de ces facteurs au processus de sélection permettrait possiblement d'écarter des segments de texte qui, bien qu'en apparence indentiques au point de mire, représentent en fait des emplois différents d'une même séquence de mots.

On a également fait la supposition que les segments proposés à l'utilisateur pouvaient être choisis indépendamment les uns des autres. Un raffinement possible à la procédure de sélection consisterait à y faire intervenir une mesure du degré de compatibilité entre les différents segments, par exemple en évaluant au moyen d'un modèle de langue statistique la probabilité que deux segments-cible correspondant à des portions adjacentes du texte-source constituent ensemble une séquence de texte cohérente. On peut possiblement envisager également l'emploi de modèles de co-occurrences pour évaluer d'un point de vue plus global la cohérence d'un ensemble de candidats.

Finalement, et tel que souligné en introduction, la question de l'interface-utilisateur n'a pas été abordée dans le cadre de ce travail. On peut certainement imaginer, moyennant quelques modifications, que les interfaces existantes pourraient permet-

tre des propositions telles que celles produites par les composantes décrites ici. Par exemple, dans un éditeur de texte dédié, l'ensemble des candidats retenus pour une phrase donnée du texte-source pourrait être présenté dans une zone réservée au bas de l'écran ou dans une fenêtre transitoire. On pourrait utiliser un codage de couleurs dans le texte-source et dans la liste des propositions pour indiquer quelle proposition correspond à quel segment du texte-source. En somme l'utilisation d'une telle interface ne serait sans doute pas beaucoup plus complexe que celle des systèmes existants.

On peut toutefois imaginer des modes d'interaction radicalement différents, s'inspirant, par exemple, du système de traduction automatique interactive *TransType* de Foster et al. [33]. Dans ce système, c'est l'utilisateur qui initie le processus de traduction, en écrivant au moyen d'un traitement de texte dédié la traduction d'une phrase-source donnée. Le système observe l'utilisateur à mesure que celui-ci tape sa traduction, tente constamment d'anticiper ses intentions, et propose des "suites" aux portions de texte déjà tapées. L'utilisateur est libre d'accepter ces propositions telles quelles, de les modifier ou simplement de les ignorer. Dans le système TransType, les propositions sont calculées dynamiquement au moyen d'un modèle de traduction statistique. Mais on peut très bien imaginer comment elles pourraient provenir directement d'une mémoire de traduction. Alternativement, on pourrait envisager des façons de combiner les deux sources d'information, pour en arriver à un système hybride de traduction assistée par ordinateur.

En somme, l'approche proposée ici s'inscrit naturellement dans un cheminement tel que celui proposé par Martin Kay [53]: partant d'un environnement bien adapté aux besoins et aux attentes des traducteurs, intégrer graduellement, voire imperceptiblement, les éléments d'automatisation que l'on sait bien faire.

Little steps for little feet...

RÉFÉRENCES

[1] Emile Aarts et Jan Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, Chichester, UK, 1989.

[2] Anne Abeillé, Lionel Clément, et Alexandra Kinyon. Building a treebank for French. Dans Proceedings of LREC-2 [91], pages 87–94.

[3] Steven Abney. Parsing by Chunks. Dans R.C. Berwick, editeur, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.

[4] Steven Abney. Prosodic Structure, Performance Structure and Phrase Structure. Dans *Proceedings, Speech and Natural Language Workshop*, pages 425–428, San Mateo, USA, 1992. Morgan Kaufmann Publishers.

[5] Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah H. Smith, et David Yarowsky. Statistical Machine Translation - Final Report, JHU Workshop 1999. Rapport technique, Johns Hopkins University, 1999.

[6] P. J. Arthern. Aids unlimited: the scope for machine aids in a large organization. *Machine Aids for Translators*, 33(7–8):309–319, 1981.

[7] Ricardo Baeza-Yates et Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New-York, USA, 1999.

[8] L. E. Baum. An Inequality and Associated Maximization Technique in Statistical Estimations of Probabilistic Functions of Markov Processes. *Inequalities*, 3:1–8, 1972.

[9] Silvia Bernardini. Think-aloud Protocols in Translation Research: Achievements, Limits, Future Prospects. *Target*, 13(2):241–263, 2001.

[10] Ann Bies, Mark Ferguson, Karen Katz, et Robert MacIntyre. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Rapport technique, University of Pennsylvania, 1995.

[11] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[12] Sotiris Boutis et Stelios Piperidis. Aligning Clauses in Parallel Texts. Dans *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 17–26, Granada, Spain, 1998.

[13] J. Brousseau, C. Drouin, G. Foster, P. Isabelle, R. Kuhn, Y. Normandin, et P. Plamondon. French Speech Recognition in an Automatic Dictation System for Translators: the TransTalk Project. Dans *Proceedings of Eurospeech 95*, pages 193–196, Madrid, Spain, 1995.

[14] Peter Brown, Jennifer C. Lai, et Robert Mercer. Aligning Sentences in Parallel Corpora. Dans Proceedings of ACL-29 [84], pages 169–173.

[15] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, et Paul S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, June 1990.

[16] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John D. Lafferty, et Robert L. Mercer. Analysis, Statistical Transfer, and Synthesis in Machine Translation. Dans Proceedings of TMI-4 [92], pages 83–100.

[17] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, et Robert L. Mercer. The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[18] Ralf D. Brown. Example-Based Machine Translation in the Pangloss System. Dans Proceedings of COLING-96 [88], pages 169–174.

[19] Ralf D. Brown. Adding Linguistic Knowledge to a Lexical Example-Based Translation System. Dans Proceedings of TMI-8 [93], pages 22–32.

[20] Sabine Buchholz. README for Perl script `chunklink.pl`. URL: `http://pi0657.kub.nl/ sabine/chunklink/README.html`, February 2000.

[21] Claire Cardie, Walter Daelemans, Claire Nédellec, et Erik Tjong Kim Sang, editeurs. *Proceedings of the Fourth Conference on Computational Natural Language Learning*, Lisbon, Portugal, September 2000.

[22] Michael Carl. Combining Invertible Example-Based Machine Translation with Translation Memory Technology. Dans Proceedings of AMTA-4 [86].

[23] Jiang Chen et Jian-Yun Nie. Parallel Web text mining for cross-language IR. Dans *Actes de la Conférence sur la recherche d'information assistée par ordinateur*, pages 62–77, Paris, France, 2000.

[24] Kenneth W. Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. Dans *Proceedings of the 2nd Conference on Applied Natural Language Processing (ANLP)*, Austin, USA, 1988.

[25] Kenneth W. Church. `Char_align`: A Program for Aligning Parallel Texts at the Character Level. Dans Proceedings of ACL-31 [85].

[26] Yann Le Cun, Léon Bottou, Yoshua Bengio, et Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278.

[27] Ido Dagan et Ken W. Church. TERMIGHT: Identifying and Translating Technical Terminology. Dans *Proceedings of the 4th Conference on Applied Natural Language Processing (ANLP)*, pages 34–40, Stuttgart, Germany, 1994.

[28] Ido Dagan, Kenneth W. Church, et William A. Gale. Robust Bilingual Word Alignment for Machine Aided Translation. Dans *Proceedings of the 1st ACL Workshop on Very Large Corpora (WVLC)*, pages 1–8, Columbus, USA, 1993.

[29] Béatrice Daille, Éric Gaussier, et Jean-Marc Langé. Towards Automatic Extraction of Monolingual and Bilingual Terminology. Dans *Proceedings of the 15th International Conference on Computational Linguistics (COLING) 1994*, pages 515–521, Kyoto, Japan, Août 1994.

[30] Fathi Debili et Elyès Sammouda. Appariement des phrases de textes bilingue francais-anglais et francais-arabe. Dans Proceedings of COLING-92 [87], pages 517–524.

[31] Gerald Dennett. Translation Memory: Concept, products, impact and prospects. Thèse de maîtrise, South Bank University - School of Electrical, Electronic and Information Engineering, 1995.

[32] Ted Dunning. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):62–74, 1993.

[33] George Foster, Pierre Isabelle, et Pierre Plamondon. Target-Text Mediated Interactive Machine Translation. *Machine Translation*, 21(1-2), 1997.

[34] George F. Foster. Statistical Lexical Disambiguation. Msc thesis, McGill University, School of Computer Science, 1991.

[35] William A. Gale et Kenneth W. Church. A Program for Aligning Sentences in Bilingual Corpora. Dans Proceedings of ACL-29 [84], pages 177–184.

[36] William A. Gale et Kenneth W. Church. Identifying Word Correspondences in Parallel Texts. Dans *Proceedings of the DARPA Workshop on Speech and Natural Language*, pages 152–157, Pacific Grove, USA, 1991.

[37] Eric Gaussier. *Modèles statistiques et patrons morphosyntaxiques pour l' extraction de lexiques bilingues*. PhD thesis, Université de Paris 7, janvier 1995.

[38] Eric Gaussier. Flow Network Models for Word Alignment and Terminology Extraction for Bilingual Corpora. Dans Proceedings of COLING-98 [89], pages 444–450.

[39] Eric Gaussier et Jean-Marc Langé. Modèles statistiques pour l'extraction de lexiques bilingues. *Traitement automatique des langues*, 36(1–2):133–155, 1995.

[40] James Paul Gee et François Grosjean. Performance Structures: A Psycholinguistic and Linguistic Appraisal. *Cognitive Psychology*, 15:411–458, 1983.

[41] Ralph Grishman. Iterative Alignment of Syntactic Structures for a Bilingual Corpus. Dans Proceedings of WVLC-2 [94], pages 57–68.

[42] H. Altay Güvenir et Ilyas Cicekli. Learning Translation Templates from Examples. *Information Systems*, 23(6):353–363, 1998.

[43] Brian Harris. Are You Bi-textual? *Language Technology*, 7:41, 1988.

[44] John Hutchins. The Origins of the Translator's Workstation. *Machine Translation*, 13(4):287–307, 1998.

[45] Nancy Ide, G Priest-Dorman, et Jean Véronis. Corpus Encoding Standard. URL: `http://www.cs.vassar.edu/CES/`, 1995.

[46] Industries de services Industrie Canada. Aperçus des industries de service: Industrie de la traduction. URL: `http://strategis.ic.gc.ca/SSI/bpf/transl-fre.pdf`, October 2001.

[47] Pierre Isabelle, Marc Dymetman, George Foster, Jean-Marc Jutras, Elliott Macklovitch, François Perrault, Xiabo Ren, et Michel Simard. L'analyse de traduction et l'automatisation de la traduction. Dans Louisette Emirkanian et Lorne H. Bouchard, editeurs, *Traitement automatique du français écrit*, Les cahiers scientifiques, pages 211–240. Acfas, Montréal, Canada, 1996.

[48] Pierre Isabelle, Marc Dymetman, George Foster, Jean-Marc Jutras, Elliott Macklovitch, François Perrault, Xiaobo Ren, et Michel Simard. Translation Analysis and Translation Automation. Dans *Proceedings of the 5th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 201–217, Kyoto, Japan, 1993.

[49] Pierre Isabelle et Michel Simard. Propositions pour la représentation et l'évaluation des alignements de textes parallèles. URL: `http://www-rali.iro.umontreal.ca/arc-a2/PropEval`, 1996.

[50] Thorsten Joachims. Optimizing search engines using clickthrough data. Dans *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 128–137, Edmonton, Canada, 2002.

[51] Hiroyuki Kaji, Yuudo Kida, et Yasutsugu Morimoto. Learning Translation Templates from Bilingual Text. Dans Proceedings of COLING-92 [87], pages 672–678.

[52] Ronald M. Kaplan. The formal architecture of Lexical-Functional Grammar. *Journal of Information Science and Engineering*, 5(4):305–322, 1989.

[53] Martin Kay. The Proper Place of Men and Machines in Language Translation. *Machine Translation*, 12(1-2):3–23, 1997. Initialement paru en 1980 sous la forme d'un document de travail de Xerox PARC.

[54] Martin Kay et Martin Röscheisen. Text-translation Alignment. *Computational Linguistics*, 19(1):121–142, 1993.

[55] Maurice G. Kendall et Jean D. Gibbons. *Rank Correlation Methods*. Oxford University Press, Oxford, UK, 1990.

[56] Friedrich Krollmann. Linguistic Data Banks and the Technical Translator. *Meta*, 16:117–124, 1971.

[57] Jean-Marc Langé, Éric Gaussier, et Béatrice Daille. Bricks and Skeletons: Some Ideas for the Near Future of MAHT. *Machine Translation*, 12(1–2):39–51, 1997.

[58] Philippe Langlais. Opening Statistical Translation Engines to Terminological Resources. Dans *Proceedings of 7th International Workshop on Applications of Natural Language to Information Systems (NLDB)*, pages 191–202, Stockholm, Sweden, 2002.

[59] Philippe Langlais, Marie Loranger, et Guy Lapalme. Translators at work with TransType: Resource and evaluation. Dans *Proceedings of the Third Interna-*

tional Conference on Language Resources & Evaluation (LREC), pages 2128–2134, Las Palmas de Gran Canaria, Spain, 2002.

[60] Philippe Langlais et Michel Simard. Merging Example-Based and Statistical Machine Translation: an experiment. Dans *From Research to Real Users – Proceedings of the Fifth Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 104–114, Tiburon, USA, 2002.

[61] Elliott Macklovitch. TransCheck — or the Automatic Validation of Human Translations. Dans *Machine Translation Summit V*, Luxembourg, 1995. (original sans pagination).

[62] Elliott Macklovitch. Peut-on vérifier automatiquement la cohérence terminologique? *META*, 41(3):299–316, 1996.

[63] Elliott Macklovitch et Graham Russell. What's been Forgotten in Translation Memory. Dans Proceedings of AMTA-4 [86], pages 137–146.

[64] Elliott Macklovitch, Michel Simard, et Philippe Langlais. TransSearch: A Free Translation Memory on the World Wide Web. Dans Proceedings of LREC-2 [91], pages 1201–1208.

[65] Daniel Marcu. Towards a Unified Approach to Memory- and Statistical-Based Machine Translation. Dans *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 378–385, Toulouse, France, Juillet 2001.

[66] Mitchell P. Marcus, Beatrice Santorini, et Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–331, June 1993.

[67] Yuji Matsumoto, Hiroyuki Ishimoto, et Takehito Utsuro. Structural Matching of Parallel Texts. Dans Proceedings of ACL-31 [85], pages 23–30.

[68] Kevin McTait. Memory-Based Translation Using Translation Patterns. Dans *Proceedings of the 4th Annual CLUK Colloquium*, pages 43–52, Sheffield, UK, 2001.

[69] Kevin McTait, Maeve Olohan, et Arturo Trujillo. A Building Blocks Approach to Translation Memory. Dans *Proceedings of the 21st ASLIB International Conference on Translating and the Computer*, London, UK, 1999. (original sans pagination).

[70] Kevin McTait et Arturo Trujillo. A Language-Neutral Sparse-Data Algorithm for Extracting Translation Patterns. Dans Proceedings of TMI-8 [93], pages 98–108.

[71] I. Dan Melamed. Automatic Construction of Clean Broad-coverage Translation Lexicons. Dans *Expanding MT horizons – Proceedings of the Second Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 125–134, Montréal, Canada, 1996.

[72] I. Dan Melamed. Automatic Discovery of Non-compositional Compounds in Parallel Data. Dans *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 97–108, Providence, USA, 1997.

[73] I. Dan Melamed. Manual Annotation of Translational Equivalence: The Blinker Project. Rapport Technique 98-06, IRCS, Philadelphia, USA, 1998.

[74] I. Dan Melamed. Word-to-Word Models of Translational Equivalence. Rapport

Technique 98-08, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, USA, 1998.

[75] Alan K. Melby. A Bilingual Concordance System and its Use in Linguistic Studies. Dans W. Gutwinski et G. Jolly, editeurs, *Proceedings of the Eighth LACUS Forum*, pages 541–549, Toronto, Canada, 1981. Linguistic Association of Canada and the United States, Hornbeam Press.

[76] Alan K. Melby. Sharing of Translation Memory Databases Derived From Parallel Text. Dans Véronis [111], pages 347–368.

[77] Frédéric Meunier. Découpage de Phrases et Alignement de Sous-Phrases dans un Corpus Bilingue. DEA informatique fondamentale, Université Paris 7, UFR Informatique, septembre 1993.

[78] Sonja Niessen, Franz Josef Och, Gregor Leusch, et Hermann Ney. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. Dans Proceedings of LREC-2 [91], pages 39–45.

[79] Franz Josef Och et Hans Weber. Improving Statistical Natural Language Translation with Categories and Rules. Dans Proceedings of COLING-98 [89], pages 985–989.

[80] Miles Osborne. Shallow Parsing as Part-of-Speech Tagging. Dans Cardie et al. [21], pages 145–147.

[81] Kishore Papinemi, Salim Roukos, Todd Ward, et Wei-Jing Zhu. Bleu: a method for automatic evalution of machine translation. Dans *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, USA, Juillet 2002.

[82] Emmanuel Planas. Extending Translation Memories. Dans *EAMT Machine Translation Workshop*, page (original sans pagination), Ljubljana, Slovenia, May 2000.

[83] Emmanuel Planas et O. Furuse. Formalizing Translation Memories. Dans *Machine Translation Summit VII*, pages 331–339, Singapore, 1999.

[84] *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berkeley, USA, Juin 1991.

[85] *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL)*, Columbus, USA, Juin 1993.

[86] *Envisioning Machine Translation in the Information Future – Proceedings of the Fourth Conference of the Association for Machine Translation in the Americas (AMTA)*, Cuernavaca, Mexico, 2000.

[87] *Proceedings of the 14th International Conference on Computational Linguistics (COLING) 1992*, Nantes, France, Août 1992.

[88] *Proceedings of the 16th International Conference on Computational Linguistics (COLING) 1996*, Copenhagen, Denmark, Août 1996.

[89] *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL) and 17th International Conference on Computational Linguistics (COLING) 1998*, Montréal, Canada, Août 1998.

[90] *Proceedings of the International Conference on Spoken Language Processing (ICSLP) 1996*, Philadelphia, USA, 1996.

[91] *Proceedings of the Second International Conference on Language Resources & Evaluation (LREC)*, Athens, Greece, 2000.

[92] *Proceedings of the 4th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Montréal, Canada, 1992.

[93] *Proceedings of the 8th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Chester, UK, 1999.

[94] *Proceedings of the 2nd ACL Workshop on Very Large Corpora (WVLC)*, Las Cruces, USA, 1994.

[95] L. R. Rabiner et B. H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, pages 4–16, Jan 1986.

[96] Adwait Ratnaparkhi. A Maximum Entropy Part-Of-Speech Tagger. Dans *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–142, Philadelphia, USA, 1996.

[97] Deborah S. Ray et Eric J. Ray. Good, Fast, Cheap: Translation Memory Systems Offer the Potential for All Three. *Technical Communication*, 46(2):280–285, May 1999. URL: `http://www.raycomm.com/techwhirl/translationmemory.html`.

[98] Philip Resnik et Noah Smith. The Web as a Parallel Corpus. Rapport Technique UMIACS-TR-2002-61, University of Maryland, 2002.

[99] Klaus Ries, Finn Dag Buø, et Alex Waibel. Class Phrase Models For Language Modeling. Dans Proceedings of ICSLP-96 [90], pages 398–401.

[100] Erik F. Tjong Kim Sang et Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. Dans Cardie et al. [21], pages 127–132.

[101] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656, October 1948.

[102] Michel Simard, George Foster, et Pierre Isabelle. Using Cognates to Align Sentences in Bilingual Corpora. Dans Proceedings of TMI-4 [92], pages 67–82.

[103] Michel Simard, George Foster, et François Perrault. TransSearch : un concordancier bilingue. Rapport technique, Centre d'innovation en technologies de l'information (CITI), Laval, Canada, 1993.

[104] Michel Simard et Philippe Langlais. Sub-sentential Exploitation of Translation Memories. Dans *Machine Translation Summit VIII*, pages 335–339, Santiago de Compostela, Spain, September 2001.

[105] Michel Simard et Pierre Plamondon. Bilingual Sentence Alignment: Balancing Robustness and Accuracy. *Machine Translation*, 13(1):59–80, 1998.

[106] Harold Somers. Review Article: Example-based Machine Translation. *Machine Translation*, 14:113–157, 1999.

[107] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, UK, 2nd edition édition, 1979.

[108] Stephan Vogel, Hermann Ney, et Christoph Tillmann. HMM-Based Word Alignment in Statistical Machine Translation. Dans Proceedings of COLING-96 [88], pages 836–841.

[109] Jean Véronis. Tagging guidelines for word alignment. URL: `http://www.up.univ-mrs.fr/ veronis/arcade/2nd/word/guide/index.html`, April 1998.

[110] Jean Véronis. From the Rosetta Stone to the Information Society - A Survey of Parallel Text Processing. Dans Véronis 99 [111], pages 1–24.

[111] Jean Véronis, editeur. *Parallel Text Processing*. Text, Speech and Language Technology. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[112] Jean Véronis et Philippe Langlais. Evaluation of Parallel Text Alignment Systems – The ARCADE Project. Dans Véronis [111], pages 369–388.

[113] Robert A. Wagner et Michael J. Fischer. The String-to-string Correction Problem. *Journal of the ACM*, 21(1):168–173, 1974.

[114] Ye-Yi Wang, John Lafferty, et Alex Waibel. Word Clustering With Parallel Spoken Language Corpora. Dans Proceedings of ICSLP-96 [90], pages 2364–2367.

[115] Ian H. Witten, Alistair Moffat, et Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishing, San Francisco, USA, 2ième édition, 1999.

[116] Dekai Wu. Trainable coarse bilingual grammars for parallel text bracketing. Dans Proceedings of WVLC-2 [94], pages 69–82.

[117] Dekai Wu. An algorithm for simultaneously bracketing parallel texts by aligning words. Dans *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 244–251, Cambridge, USA, Juin 1995.

[118] Dekai Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–404, September 1997.