université PARIS-SACLAY

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir®

Université Paris-Saclay

École Doctorale des Sciences et Technologies de l'Information et de la Communication
Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur

Discipline : Informatique

Thèse de doctorat

Benjamin Marie

Exploitation d'informations riches pour guider la traduction automatique statistique

Thèse soutenue le 25 mars 2016





— Lingua et Machina —
L'écrit multilingue dans l'entreprise

Composition du jury :

Président :	François Yvon	Professeur (Université Paris-Saclay, Paris-Sud)
Rapporteurs :	Marine Carpuat	Assistant Professor (University of Maryland)
	Philippe Langlais	Professeur (Université de Montréal)
Examinateur :	Laurent Besacier	Professeur (Université Joseph Fourier)
Invité :	François Brown de Colstoun	Président de *Lingua et Machina*
Directrice de thèse :	Anne Vilnat	Professeure (Université Paris-Saclay, Paris-Sud)
Co-encadrant :	Aurélien Max	Maître de Conférences (Université Paris-Saclay, Paris-Sud)

NNT : 2016SACLS066

Remerciements

Je tiens tout d'abord à remercier mes encadrants Anne Vilnat et Aurélien Max qui m'ont permis de réaliser les travaux présentés dans ce manuscrit. Aurélien a été mon guide scientifique depuis mon arrivée au LIMSI. Durant presque 4 ans, il a su orienter mes recherches en fonction de mes compétences et m'a donné l'envie d'entamer une carrière professionnelle dans ce domaine. Anne de son côté m'a orienté administrativement tout au long des 3 années de thèse, puis m'a beaucoup aidé à parfaire ma soutenance de thèse et ce manuscrit.
Je remercie également François Brown de Colstoun qui en m'embauchant dans son entreprise, Lingua et Machina, a permis le financement puis la réalisation de cette thèse. Lingua et Machina ayant été hébergé par l'Inria j'ai aussi eu la chance de partager les locaux d'Alpage à Rocquencourt, grâce notamment à Éric de la Clergerie que je remercie.
Je remercie aussi François Yvon, tout d'abord pour m'avoir accueilli au sein du groupe traduction puis pour avoir présidé le jury de ma soutenance.
Mes remerciements s'adressent également à Marine Carpuat, Philippe Langlais et Laurent Besacier qui ont acceptés d'évaluer mon travail et se sont rendus disponibles pour assister à ma soutenance.

Dès mon arrivée au LIMSI j'ai eu la chance de côtoyer de nombreux doctorants déjà bien installés et qui m'ont donné l'envie de faire une thèse : Hai Son, Thiago, Nadi, Penny et Li. Je remercie aussi particulièrement toutes les personnes travaillant en traduction pour leur expérience dont j'ai beaucoup bénéficié : Marianna, Hélène, Thomas, Guillaumme et Alex. Je me souviendrai que c'est par l'intermédiaire d'Alex que je suis arrivé au LIMSI, me mettant en contact avec Aurélien alors que j'étais encore en Master à l'INaLCO.
Mes années passées au LIMSI auront été très agréables en grande partie grâce à mes camarades de bureau. Nicolas et Yong durant mon stage, puis Marco, Hai Son, Li, Khanh, Elena et Rachel durant ma thèse. J'en suis absolument convaincu, pour faire une bonne thèse il faut un bon bureau. Je n'oublierai pas de remercier les derniers doctorants arrivés qui ont participé à la conservation d'une bonne ambiance de travail : Julia, Laurianne, Matthieu, Kévin, Pierre.

Enfin, je remercie mes sœurs Anne-gaëlle et Claire-line, mon frère Réginald et mes parents pour leur soutien tout au long de mes études et sans qui ce travail n'aurait pu être accompli.

Table des matières

1 Introduction **1**

2 Vue d'ensemble sur la traduction automatique statistique **5**

2.1 Le problème de la traduction automatique statistique 8

2.2 Alignements mot à mot 9

2.3 La traduction automatique statistique fondée sur les segments 11
2.3.1 Les bisegments comme unités de traduction 11
2.3.2 Une fonction de score qui met en jeu des modèles 13
2.3.2.1 Une fonction de score pour évaluer les traductions 13
2.3.2.2 Des tables de traduction pour collecter les bisegments . . 13
2.3.2.3 Modèle de langue 14
2.3.2.4 Des modèles de réordonnancement pour réorganiser les traductions 15
2.3.3 Parcours de l'espace de recherche 16

2.4 Les métriques d'évaluation automatique de la traduction 17
2.4.1 Rôles des métriques d'évaluation automatique 17
2.4.2 BLEU et ses variantes 18
2.4.3 TER 19
2.4.4 METEOR 19
2.4.5 Les limites de l'évaluation automatique 20

2.5 L'optimisation d'un système de traduction 20
2.5.1 La recherche des meilleurs poids à attribuer aux modèles 20
2.5.2 L'algorithme d'optimisation `KB-MIRA` 22

Résumé 24

3 Améliorer la sortie d'un système de traduction automatique **27**

3.1 Données et systèmes de traduction de référence 29
3.1.1 Systèmes de traduction d'articles d'actualité (`news`) 29
3.1.2 Systèmes de traduction de textes médicaux (`médical`) 30
3.1.3 Systèmes de traduction de présentations orales (`TED Talks`) . . . 30

3.2 Le décodage et ses limites 31

3.2.1 Un parcours de l'espace de recherche qui doit être efficace 31
3.2.2 Des modèles trop complexes pour être intégrés au décodage 33
3.2.2.1 Modèles nécessitant une hypothèse complète pour être calculés 34
3.2.2.2 Modèles trop coûteux pour être intégrés au cours du décodage 35
3.2.2.3 Modèles indisponibles au cours d'un premier décodage . 35

3.3 Le reclassement des n-meilleures hypothèses de traduction 36
3.3.1 Le reclassement et ses limites 36
3.3.2 Systèmes de traduction de référence avec reclassement d'hypothèses 40
3.3.2.1 Modèles complexes utilisés 40
3.3.2.2 Résultats des expériences de reclassement 41

3.4 Regénérer une traduction après un premier décodage 43

3.5 Au-delà du décodage : la recherche locale 45

3.6 La post-édition pour améliorer une traduction 48
3.6.1 La post-édition par un humain 48
3.6.2 La post-édition automatique 49

Résumé 51

4 Une recherche locale enrichie avec des modèles complexes 53

4.1 Évaluation du potentiel d'amélioration des traductions produites 55
4.1.1 La recherche locale oracle 55
4.1.1.1 Une métrique d'évaluation automatique comme fonction de score 55
4.1.1.2 Opérations de réécriture 55
4.1.2 Systèmes de référence `europarl-intersect` et `BTEC` 56
4.1.3 Expériences et analyse 60
4.1.3.1 Mise en évidence du potentiel d'amélioration des traductions 60
4.1.3.2 Effets de la réduction des données d'entraînement et du filtrage des tables de traduction 63
4.1.3.3 Impact de la langue cible 64
4.1.3.4 Une opération `paraphrase` pour améliorer l'atteignabilité des traductions de référence 65
4.1.3.5 Les dérives d'une recherche locale oracle guidée par sBLEU 67

4.2 Réécriture automatique de traductions à l'aide de modèles complexes . . 70
4.2.1 Une réécriture guidée par des modèles inutilisables lors du décodage 70
4.2.1.1 Table de réécriture 70
4.2.1.2 Optimisation de la nouvelle fonction de score 71
4.2.1.3 Le système de réécriture : `rewriter` 71
4.2.2 Expériences 72
4.2.2.1 Systèmes de référence 72

4.2.2.2 Résultats . . . 72
4.2.3 Analyse . . . 74
4.2.3.1 Impact de la table de réécriture et des exemples d'entraînement . . . 74
4.2.3.2 Performance de `rewriter` en fonction de la qualité de la traduction à réécrire . . . 78
4.2.3.3 Expériences semi-oracle : ne pas réécrire ce qui est déjà correct . . . 78
4.2.3.4 Analyse manuelle des erreurs effectuées . . . 80
4.2.3.5 Une réécriture très locale et parfois trop limitée . . . 81

Résumé . . . 83

5 Un décodage mieux informé guidé par des modèles complexes 87

5.1 Un décodage à passes multiples . . . 89
5.1.1 Guidage du décodeur par le résultat d'un reclassement d'hypothèses 89
5.1.2 Exploitation des informations nouvelles d'un reclassement par partitionnement des tables de traduction . . . 90
5.1.2.1 Partitionnement des tables par identification des bisegments à privilégier ou à pénaliser . . . 90
5.1.2.2 Traduction spécialisée pour chaque token . . . 91
5.1.3 Fonctionnement global du décodage à passes multiples . . . 92
5.1.3.1 Accumulation des modèles et des hypothèses produites . 92
5.1.3.2 Optimisation du système . . . 93

5.2 Expériences . . . 93
5.2.1 Systèmes de référence . . . 93
5.2.2 Résultats . . . 95

5.3 Analyse . . . 99
5.3.1 Des listes d'hypothèses plus diversifiées . . . 99
5.3.1.1 Analyse au niveau des hypothèses . . . 99
5.3.1.2 Analyse au niveau lexical . . . 100
5.3.1.3 Une meilleure diversité pour un reclassement d'hypothèses amélioré . . . 103

5.4 Combinaison du décodage à passes multiples et de la recherche locale . . 104
5.4.1 Ajout d'une nouvelle étape de réécriture après chaque décodage . 105
5.4.2 Résultats et analyse . . . 107

5.5 Analyse des différences entre les traductions produites par les systèmes . 108

Résumé . . . 113

6 Regénérer une traduction à l'aide de l'humain : la pré-post-édition 117

6.1 La pré-post-édition . . . 119

6.1.1 Description de l'approche . . . 119
6.1.2 Partitionnement des modèles en fonction des annotations . . . 120
6.1.2.1 Modèles de langue . . . 120
6.1.2.2 Tables de traduction . . . 122
6.1.3 Fonctionnement global . . . 123
6.1.3.1 Utilisation des nouveaux modèles . . . 123
6.1.3.2 Optimisation du système . . . 124
6.2 Expériences . . . 124
6.2.1 Le paradigme de la post-édition simulée . . . 124
6.2.2 TER_{PPE} : une métrique pour l'évaluation de la pré-post-édition . . . 125
6.2.3 Systèmes de référence . . . 126
6.2.4 Résultats . . . 126
6.3 Analyse . . . 128
6.3.1 Évaluation de l'importance des modèles ajoutés . . . 128
6.3.2 Ajout d'une passe de reclassement d'hypothèses après chaque itération . . . 129
Résumé . . . 132
7 Conclusion **135**
Appendices **141**
Annexe A Extraits de bicorpus **143**
A.1 Extrait du bicorpus d'évaluation utilisé pour la tâche `news` . . . 143
A.2 Extrait du bicorpus d'évaluation utilisé pour la tâche `médical` . . . 145
A.3 Extrait du bicorpus d'évaluation utilisé pour la tâche `TED Talks` . . . 146
Annexe B Exemples de tokens hors-vocabulaire **148**
B.1 Tokens hors-vocabulaire du corpus d'évaluation de la tâche `news` . . . 148
B.2 Tokens hors-vocabulaire du corpus de test de la tâche `médical` . . . 149
B.3 Tokens hors-vocabulaire du corpus de test de la tâche `TED Talks` . . . 149
Annexe C Exemples de traductions produites **151**
Annexe D Publications de l'auteur **154**

1 Introduction

Le besoin d'accéder à des informations indisponibles dans sa langue, ainsi que de communiquer avec des personnes parlant d'autres langues, n'a cessé de s'accroître depuis l'essor d'Internet, que ce soit dans le milieu professionnel ou dans la sphère privée. L'accélération de la mondialisation des échanges économiques et des migrations de populations bouleverse les modes de communication en rendant indispensable l'accès à des services de traduction rapides et gratuits. Si l'on considère, par exemple, la situation de l'Union Européenne, où l'on dénombre 24 langues officielles, qui représentent 552 directions de traduction possibles, un nombre considérable de traducteurs et interprètes professionnels sont nécessaires pour ne finalement traduire qu'une petite partie de ce qui devrait l'être. Les coûts de traduction, les difficultés de disponibilité de traducteurs humains pour certaines directions de traduction, ainsi que le temps nécessaire pour produire une traduction manuelle rendent indispensable le recours à des systèmes automatiques de traduction.

De prime abord, la traduction automatique présente de nombreux avantages relativement à la traduction humaine : son coût est bien moins élevé, elle produit des traductions de manière instantanée, et peut attaquer la traduction de n'importe quel type de texte. Elle offre également une solution appropriée lorsqu'il s'agit de traduire de nouveaux types de textes à portée éphémère, tels que les *tweets*, messages de forum, courriels, SMS, dialogues de messagerie instantanée, etc.

Cependant, s'il est indéniable que de nos jours la traduction automatique facilite la communication entre langues, ses productions n'ont pas, dans la majorité, des cas le niveau de qualité de celles obtenues par recours à des traducteurs humains. Toutefois, ces deux modes de traduction peuvent utilement se compléter, la manière la plus répandue consistant à produire de façon automatique un *brouillon* de traduction qui sera ensuite repris et corrigé par un traducteur humain. Si cette approche permet des gains de produc-

tivité important, elle impose à l'humain la réalisation d'une tâche relativement artificielle, et parfois laborieuse, révèlant que les systèmes de traduction automatique actuels sont toujours loin de produire des traductions parfaites, et cela bien qu'ils soient l'aboutissement de plus d'un demi-siècle de recherche.

En effet, on fait remonter les débuts de la recherche en traduction automatique aux années 1950, sous l'influence du mathématicien américain Warren Weaver. La tâche de traduction automatique fut initialement formulée comme un simple problème de *décodage*, le texte en langue étrangère étant considéré comme un message codé qu'il faut transformer en anglais. S'en sont suivies des décennies de recherches, ponctuées de désillusions (Hutchins et Somers, 1992), durant lesquelles différentes approches pour mettre au point des systèmes de traduction automatique ont vu le jour. L'accroissement très important du nombre de documents bilingues disponibles a mené à une situation où dominent les approches fondées sur l'utilisation de ces grandes masses de connaissances. Aujourd'hui, les systèmes de traduction dits *statistiques* sont très rapides, peu coûteux à mettre en place, et permettent sous certaines conditions d'obtenir des traductions de bonne qualité.

Les travaux que nous décrirons dans ce manuscrit s'ancrent dans cette approche, dont les principales caractéristiques seront décrites dans le chapitre 2. Cette description mettra en relief les limites actuelles de la traduction statistique, qui incluent notamment le fait qu'un système, qui va considérer un très grand nombre d'*hypothèses* de traductions possibles, ne pourra mettre en œuvre que des informations de certains types pour déterminer celles qui sont les meilleures. Comme nous le décrirons, certains types d'informations plus riches, qui rendent mieux compte de la qualité d'une traduction, ne peuvent être exploitées qu'après une première génération d'un ensemble d'hypothèses les plus prometteuses. Prenons l'exemple d'une phrase en anglais et de sa traduction automatique en français, générée par un système à l'état de l'art accessible librement sur Internet [1], donnée par la Figure 1.1. Dans cet exemple, des erreurs fréquemment commises par de tels systèmes sont observées, ici particulièrement au niveau d'accords en genre des participes passés (indiquées en gras). En effet, les informations relativement simples que le système peut utiliser ne permettent qu'une modélisation très locale de la grammaticalité de la traduction générée. Ainsi le participe passé « pelés » est probablement trop éloigné du nom « pommes de terre » pour qu'il puisse être accordé correctement étant données les informations utilisées par le système. De plus un système statistique génère typiquement ses traductions de gauche à droite, par conséquent « Avant d'être cuits » est généré avant « pommes de terre », et le système n'a pas d'information sur comment réaliser l'accord sur la traduction de « cooked », qui n'existe pas encore à ce stade de la traduction.

Afin, notamment, de corriger ce type d'erreurs, différentes approches qui visent à une amélioration *a posteriori* des traductions produites par un système à l'état de l'art sont possibles, et seront décrites dans le chapitre 3. Nous accompagnerons ces descriptions d'expériences pour évaluer la performance de chacune d'elles. Ces expériences seront conduites sur trois types de textes et pour deux directions de traduction afin de mieux

1. https://translate.google.fr/

phrase à traduire	*Before being cooked, the potatoes must be washed and peeled.*
traduction humaine	Avant d'être cuites, les pommes de terre doivent être lavées et pelées.
traduction automatique	Avant d'être **cuits**, les pommes de terre doivent être **lavés** et **pelés**.

FIGURE 1.1 – Exemple d'erreurs typiquement observées dans une traduction produite automatiquement, de l'anglais vers le français, par un système à l'état de l'art.

rendre compte des performances ainsi évaluées. Nous reprendrons les mêmes configurations expérimentales dans l'ensemble des chapitres suivants afin d'aider dans l'interprétation de la portée des résultats que nous présenterons.

Nos contributions s'articuleront autour de trois chapitres. Nous introduirons tout d'abord, dans le chapitre 4, un système de réécriture guidé par des informations riches qui applique des réécritures successives à la meilleure traduction proposée par un système statistique à l'état de l'art. Plus précisément, nos points de comparaison correspondront aux meilleures traductions selon une étape de *reclassement* des traductions exploitant ces mêmes informations riches. De la sorte, nous visons principalement à faire un meilleur usage de telles informations[2] en introduisant une procédure efficace pour construire et évaluer de nouvelles traductions.

Bien que la première approche que nous décrirons obtiendra de bonnes performances en évaluation, nous avons été en mesure de mettre en évidence certaines de ses limites. Essentiellement, celle-ci ne bénéficie pas des avantages apportés par une exploration très large des traductions possibles. L'originalité de l'approche que nous décrirons dans le chapitre 5 consistera principalement à refaire, par *passes multiples*, une telle exploration mais en exploitant des informations dérivées de l'évaluation des traductions produites antérieurement à l'aide d'informations riches. Nous serons à nouveau en mesure de montrer de bonnes performances empiriques pour cette approche, et nous montrerons également comment nos deux propositions des chapitres 4 et 5 peuvent être combinées.

La dernière contribution de notre travail, que nous décrirons dans le chapitre 6, portera sur l'hypothèse de l'existence d'un type d'information idéalisé parfait qui permettrait de déterminer quelles parties d'une traduction issue d'un système est correcte. Il s'agit donc, en quelque sorte, d'apporter une indication de la meilleure performance atteignable avec les approches introduites précédemment si les informations riches disponibles décrivaient parfaitement ce qui constitue une bonne traduction. Nous serons toutefois en mesure de décrire cette approche sous forme d'une traduction *interactive*, laquelle serait réduite à sa forme la plus simple : un système produit la meilleure hypothèse de traduction qu'il peut générer pour une phrase, puis un humain apporte la connaissance des parties qui sont correctes, et un nouveau système exploite cette information pour tirer parti de la nouvelle connaissance apportée. Cette contribution originale, que nous nommerons *pré-post-édition*, permettra d'obtenir des gains dans les mesures d'évaluation automatique

2. Il est notable que nous ne chercherons pas, dans ce travail, à introduire de nouveaux types d'informations riches qui n'auraient pas été étudiés jusque-là. Nous réutiliserons des types d'informations habituellement utilisés dans les systèmes de reclassement, et nous rapporterons les performances individuelles de chacun.

de très forte ampleur sur l'ensemble des configurations de traduction étudiées dans ce travail.

Nous apporterons finalement une conclusion générale dans le chapitre 7. Nous y décrirons également les principaux problèmes dont la résolution permettrait, selon notre expérience acquise en menant notre travail, de pousser plus loin l'exploitation d'informations riches en traduction statistique, ainsi que de nouveaux modes de collaboration entre l'homme et la machine.

2

Vue d'ensemble sur la traduction automatique statistique

Sommaire

2.1 Le problème de la traduction automatique statistique **8**

2.2 Alignements mot à mot **9**

2.3 La traduction automatique statistique fondée sur les segments **11**
- 2.3.1 Les bisegments comme unités de traduction 11
- 2.3.2 Une fonction de score qui met en jeu des modèles 13
- 2.3.3 Parcours de l'espace de recherche 16

2.4 Les métriques d'évaluation automatique de la traduction **17**
- 2.4.1 Rôles des métriques d'évaluation automatique 17
- 2.4.2 BLEU et ses variantes 18
- 2.4.3 TER 19
- 2.4.4 METEOR 19
- 2.4.5 Les limites de l'évaluation automatique 20

2.5 L'optimisation d'un système de traduction **20**
- 2.5.1 La recherche des meilleurs poids à attribuer aux modèles . . . 20
- 2.5.2 L'algorithme d'optimisation `KB-MIRA` 22

Résumé **24**

La Traduction Automatique (TA ou *MT* pour *Machine Translation*) vise à produire automatiquement la traduction d'un texte d'une langue vers une autre langue. Il s'agit en fait de l'une des toutes premières applications du Traitement Automatique des Langues (TAL). Depuis les premières tentatives de construction d'un système de TA, souvent illustrées avec le système MÉTÉO (Chevalier *et al.*, 1978), développé pour la traduction automatique de bulletins météorologiques entre l'anglais et le français, et par les efforts de l'entreprise IBM fournis pour la traduction automatique du russe vers l'anglais [1]de nombreuses et différentes approches ont été introduites pour construire des systèmes de traduction de plus en plus performants pour arriver jusqu'à ceux que nous connaissons aujourd'hui.

Au cours des années 1970, la traduction automatique fondée sur des règles (*Rule-Based* MT) était l'approche dominante. Elle repose sur une analyse du texte à traduire aux niveaux morphologique, syntaxique et parfois sémantique, puis sur la transformation, ou le transfert, des structures obtenues afin de finalement générer le texte en langue cible. Sa particularité essentielle réside dans son recours à l'intervention d'experts linguistes humains pour la construction des ressources nécessaires, telles que des dictionnaires et des grammaires. Par conséquent, la mise en place des systèmes peut s'avérer longue et coûteuse, surtout si l'on souhaite couvrir plusieurs paires de langues et plusieurs types de textes.

C'est notamment le besoin de faciliter le développement de systèmes de TA et d'en réduire le coût qui a motivé de nouvelles approches reposant sur l'exploitation de données (*data-driven* MT). Ceci n'a toutefois été permis que par la disponibilité grandissante de textes traduits, qui sont assemblés sous forme de corpus dits *parallèles* et permettent de mettre en correspondance des unités de texte allant de documents à des mots. Dans l'approche de traduction automatique fondée sur des exemples (*Example-Based* MT) (Carl et Way, 2003), apparue au début des années 1980, le texte à traduire est fragmenté et les fragments obtenus sont utilisés pour interroger une *mémoire de traduction* contenant des fragments de textes et leur traduction produite par des humains. Les fragments de traduction extraits de la mémoire sont ensuite recombinés pour obtenir la traduction complète du texte à traduire. Progressivement, l'augmentation continue de la quantité de textes traduits disponibles, conjuguée à l'augmentation de la puissance de calcul et de la mémoire de travail des ordinateurs, a permis le développement d'une approche de traduction automatique dite *statistique* (*Statistical* MT)(Koehn, 2010; Allauzen et Yvon, 2011), dont la naissance est généralement datée au début des années 1990 (Brown *et al.*, 1990).

Dans l'approche statistique, plusieurs *modèles* sont appris sur des corpus parallèles et des corpus monolingues de la langue vers laquelle on traduit. Au cours de la traduction, ces modèles sont utilisés pour calculer des scores qui permettent d'identifier la traduction de meilleur score qui servira de réponse du système (sa meilleure *hypothèse*).

1. Voir les travaux de Hutchins (1997) pour plus de détails sur l'histoire des premiers systèmes de TA.

Le développement de ce type d'approche a été grandement facilité grâce à l'apparition d'implémentations de systèmes libres d'utilisation ainsi que de *métriques* automatiques permettant d'évaluer rapidement la qualité des hypothèses de traduction produites. L'accroissement de la taille des données disponibles et de la puissance de calcul font que les systèmes de TA statistiques sont aujourd'hui devenus très performants et constituent l'état de l'art de nos jours.

Le rôle de ce chapitre est d'apporter une vue d'ensemble de la traduction automatique statistique, en n'abordant que les principes nécessaires à la compréhension du fonctionnement des systèmes ou en lien direct avec les travaux présentés dans ce manuscrit. Nous décrirons tout d'abord le problème que la traduction automatique statistique cherche à résoudre (section 2.1). Nous présenterons ensuite brièvement le problème du calcul des alignements entre mots de deux langues (section 2.2), puis le fonctionnement d'un système de traduction fondé sur des groupes de mots (*segments*, section 2.3). Nous nous pencherons également sur la question de l'évaluation de la traduction automatique par la présentation des principales métriques utilisées (2.4), ainsi que sur celle de l'*optimisation* des systèmes qui repose sur l'utilisation de ces métriques (2.5).

2.1 Le problème de la traduction automatique statistique

La traduction automatique statistique (TAS) consiste à résoudre le problème probabiliste suivant (Brown *et al.*, 1990) :

$$e_{best} = \arg\max_{e} \; p(e|f) \tag{2.1}$$

e_{best} étant la meilleure traduction telle que pour la phrase à traduire f dans la langue *source* on ait la phrase e dans la langue *cible* vers laquelle on traduit. [2] Ce problème peut être décomposé en appliquant la règle de Bayes :

$$e_{best} = \arg\max_{e} \; p(f|e)p(e) \tag{2.2}$$

Cette décomposition fait apparaître $p(f|e)$, soit la probabilité d'associer la phrase f avec une traduction générée automatiquement e. Cette probabilité peut être calculée via différents modèles (voir section 2.3.2), dont un *modèle de traduction*, qui correspond en pratique lui-même un ensemble de modèles, appris typiquement sur des *corpus bilingues parallèles*. Dans de tels corpus, les textes sont alignés phrase à phrase de telle sorte que les phrases du texte original sont associées avec leur traduction. [3] Les paires de phrases ainsi formées sont généralement appelées *biphrases*. Un exemple de corpus parallèle est présenté par la Table 2.1 pour la paire de langues anglais-français.

L'équation 2.2 fait également apparaître $p(e)$, la probabilité que la phrase e produite existe dans la langue cible vers laquelle on traduit. Si la phrase e est grammaticalement incorrecte ou plus généralement peu probable étant donné un grand corpus représentatif de la langue en question, alors la valeur de $p(e)$ sera faible. Cette probabilité est donnée par un *modèle de langue* (voir la section 2.3.2.3).

Le problème de la génération d'une traduction ne peut cependant pas se résumer à chercher la traduction d'une phrase présente dans un corpus bilingue. Cela reviendrait en effet à considérer que toutes les phrases qu'il est possible de former ont déjà été traduites par des humains et que nous avons accès à cette traduction. Pour tenter de résoudre ce problème, les premiers systèmes de TA statistiques découpaient la phrase en mots : on les décrit donc comme systèmes *fondés sur les mots* (*word-based*).

2. Les notations e et f sont utilisées pour des raisons historiques : la notation f est utilisée pour signifier *foreign* (i.e. la langue depuis laquelle on traduit), et la notation e *English* (i.e. la langue vers laquelle on traduit).

3. Il faut noter que, dans la majorité des cas, la langue d'origine ayant servi à obtenir le texte source par traduction n'est pas connue (ou, du moins, n'est pas prise en compte par les systèmes de traduction), alors qu'il a été montré que la direction d'une traduction a une grande influence sur la performance des systèmes de TA statistique (Kurokawa *et al.*, 2009; Lembersky *et al.*, 2012). Il est même possible qu'aucune de ces deux langues ne soit la vraie langue d'origine.

anglais	**français**
Mr Frazetta Snr, aged 81, is famed for his depiction of characters such as Conan the Barbarian and Tarzan.	M. Frazetta Senior, âgé de 81 ans, est célèbre pour ses dessins de personnages tels que Conan le Barbare et Tarzan.
The artist was in Florida at the time of the incident, the Associated Press news agency reported.	Au moment de l'incident, l'artiste était en Floride, a rapporté l'agence de presse Associated Press (AP).
AP quoted an unnamed police official as saying the younger Mr Frazetta may have been motivated by a family feud.	AP a cité un responsable de la police selon lequel le jeune M. Frazetta peut avoir été motivé par une querelle familiale.

TABLE 2.1 – Exemple d'alignements phrase à phrase dans un corpus parallèle anglais-français.

2.2 Alignements mot à mot

Les systèmes de traduction fondés sur les mots reposent sur la décomposition de la phrase à traduire en mots, ou plus généralement en *tokens*, par un traitement de *tokénisation* (ou *segmentation en tokens*) [4]. Cette segmentation permet de déterminer des alignements mot à mot tel qu'illustré par la Figure 2.1. Les modèles IBM (Brown *et al.*, 1993) permettent de déterminer quel mot de la langue cible est aligné avec quel mot de la langue source. Il existe cinq modèles IBM [5] avec un degré de complexité croissant. Le modèle 1 considère tous les alignements possibles comme étant équi-probables, sans tenir compte de l'ordre des mots, tandis que le modèle 2 ajoute un modèle de réordonnancement basé sur la position absolue des mots dans la phrase cible et la phrase source ; les détails de ces modèles IBM sont donnés par Brown *et al.* (1993). En considérant par exemple le plus simple de ces modèles, le modèle 1, la probabilité $p(f|e)$ se calcule avec la formule suivante :

$$p(f, a|e) = \frac{\epsilon}{(l_e + 1)^{l_f}} \prod_{j=1}^{l_e} t(f_j|e_{a(j)}) \tag{2.3}$$

où l_f est la longueur de la phrase f à traduire, l_e la longueur de la traduction e avec un alignement de chaque token e_i avec chaque token f_j selon la fonction d'alignement $a : i \to j$.

Le modèle de traduction estime ici la probabilité d'associer chaque mot de la phrase

4. Ce traitement est relativement peu ambigu pour de nombreuses langues qui séparent explicitement les mots, telle que l'anglais ou le français. Il n'en va pas de même pour des langues très agglutinantes (par ex. finnois, tamoul) ou qui n'ont pas de tels séparateurs (par ex. chinois, japonais).

5. L'entreprise IBM est à l'origine de ces cinq modèles, mais un sixième modèle IBM, introduit dans les travaux de Och et Ney (2003), est parfois mentionné dans la littérature. Un modèle d'alignement HMM (*Hidden Markov Model*) (Vogel *et al.*, 1996) est aussi souvent utilisé en association avec les premiers modèles IBM.

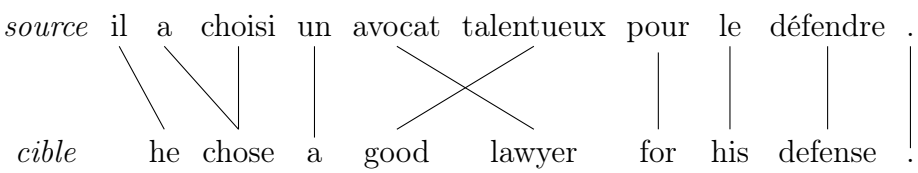


FIGURE 2.1 – Exemple d'alignements au niveau des mots entre le français et l'anglais.

en langue source avec chacune de ses traductions en langue cible. Cependant, cette modélisation se confronte à divers problèmes, notamment à celui de la polysémie lexicale et de l'homographie. Dans une traduction du français vers l'anglais, si on a par exemple le mot « avocat », le système de traduction devra décider s'il le traduit par « lawyer » ou par « avocado ». En exploitant certains corpus parallèles bilingues, le modèle de traduction pourrait par exemple donner des probabilités telles que : $p(\text{lawyer}|\text{avocat}) = 0.8$, et $p(\text{avocado}|\text{avocat}) = 0.2$. Concrètement, la traduction « avocado » ne pourrait être effectivement choisie que si, en particulier, le modèle de langue $p(e)$ avait une nette préférence pour cette traduction dans le contexte où elle est générée (voir la Table 2.2).

$\mathbf{f}$	*il a choisi un avocat talentueux pour le défendre .*
$\mathbf{e}_1$	*he chose a good lawyer for his defense .*
$\mathbf{p(f\|e_1)}$	... $p(\text{un}\|\text{a})p(\text{talentueux}\|\text{good})p(\text{avocat}\|\mathbf{lawyer})$ ...
$\mathbf{e}_2$	*he chose a good avocado for his defense .*
$\mathbf{p(f\|e_2)}$	... $p(\text{un}\|\text{a})p(\text{talentueux}\|\text{good})p(\text{avocat}\|\mathbf{avocado})$ ...

TABLE 2.2 – Problème du choix d'une traduction pour un mot polysémique, « avocat » en français, ayant deux traductions distinctes, « lawyer » et « avocado » en anglais.

D'autres problèmes peuvent rendre difficile l'estimation des probabilités de traduction mot à mot. Un token rare, peu observé dans les corpus parallèles utilisés, aura des probabilités de traduction peu fiables. Cette rareté peut être naturelle, notamment dans les langues morphologiquement riches où un mot peut avoir différentes formes dépendant de la fonction grammaticale qu'il occupe dans la phrase. Mais cette rareté peut être aussi la conséquence d'une mauvaise tokenization. En outre, les corpus parallèles peuvent être bruités, c'est-à-dire contenir des paires de phrases qui ne sont pas des traductions l'une de l'autre. [6]

Finalement, un autre problème important concerne l'idiomaticité de certaines traductions : il est en effet souvent impossible de traduire un mot par un mot exactement. Par exemple, le mot composé « pomme de terre » se traduit en anglais par « potatoe », et

6. On notera toutefois qu'en pratique les systèmes de TAS sont très robutes à ce type de bruit. Goutte *et al.* (2012) ont ainsi observé qu'il est possible de construire des systèmes à partir de corpus parallèles contenant jusqu'à 30% de paires de phrases mal alignées sans observer de dégradations significatives de la qualité des traductions produites.

plus généralement il est fréquemment nécessaire de traduire un groupe de tokens par un groupe de tokens. La traduction statistique dite *fondée sur les segments* (*phrase-based* SMT[7]) tente d'apporter des solutions à ce problème.

2.3 La traduction automatique statistique fondée sur les segments

2.3.1 Les bisegments comme unités de traduction

Contrairement à la traduction fondée sur les mots, la traduction fondée sur les segments (Och *et al.*, 1999; Och et Ney, 2003; Koehn *et al.*, 2003) utilise comme unités de traduction des groupes de tokens dont la taille maximale est bornée[8]. Un segment peut donc capturer des éléments de contexte local, tant que ceux-ci appartiennent au segment, permettant ainsi de diminuer les risques d'ambiguïtés lexicales. Par exemple, le segment « avocat d' affaires », si sa traduction a été correctement extraite ("business lawyer" en anglais), permet de résoudre l'ambiguïté décrite plus haut sur la traduction de « avocat ».

À partir des alignements mot à mot calculés sur un corpus parallèle (voir la section 2.2), il est possible d'extraire des appariements entre *segments source* (segments de la phrase à traduire) et un ou plusieurs *segments cible* correspondant. La paire de segments associant un segment source à un segment cible est généralement appelée *bisegment* (*biphrase*). Pour chaque bisegment, un ensemble de scores peuvent être calculés, par exemple :

- la probabilité de traduction directe : $p(e|f)$
- la probabilité de traduction inverse : $p(f|e)$
- la pondération lexicale directe : $lex(e|f)$
- la pondération lexicale inverse : $lex(f|e)$

La probabilité de traduction évalue la probabilité du bisegment (e, f) selon un corpus parallèle. Dans sa forme la plus simple elle est donnée par la fréquence relative dans le corpus parallèle de cette association rapportée à toutes les occurrences de f :

$$p(e|f) = \frac{F(e,f)}{\Sigma_{e_k} F(e_k, f)} \tag{2.4}$$

où $F(e, f)$ compte le nombre de fois où le bisegment (e, f) a été observé et $F(e_k, f)$ compte le nombre de fois où est observé f pour les k segments source avec lesquels il a été associé.

7. Dans l'expression *phrase-based*, le sens de *phrase* correspond véritablement à la traduction « segment », et non « syntagme », car les unités considérées n'ont pas à respecter de définition pour un constituant linguistique.

8. Une valeur typique de taille maximale des segments est de 6 tokens pour une paire de langues telle que anglais-français.

Cette estimation s'avère toutefois trop simple pour déterminer à elle seule la probabilité d'un bisegment (e, f), notamment pour les segments rares ou lorsqu'on utilise des corpus parallèles de petite taille. Dans ces situations, on peut en effet avoir à l'extrême, lorsqu'un bisegment n'est observé qu'une seule fois dans les données utilisées, $p(f|e) = p(e|f) = 1$, ce qui correspondra fréquemment à une mauvaise modélisation. Pour améliorer la qualité de ces estimations, le travail de Koehn *et al.* (2003) ajoute un score dit de *pondération lexicale* (*lexical weighting*) pour évaluer la qualité des alignements des mots qui composent le bisegment :

$$lex(e|f, A) = \prod_{i=i_d}^{i_f} \frac{1}{F(j|A(i,j)=1)} \sum_{j|A(i,j)=1} \frac{F(e_i, f_i)}{F(f_i)} \tag{2.5}$$

où $F(e_i, f_j)$ et $F(f_j)$ correspondent respectivement à la fréquence d'alignement du mot e_i avec le mot f_i et à la fréquence du mot f_j. i_d et i_f sont les indices dans la phrase du début (d) et de la fin (f) du segment. A correspond à la matrice d'alignements pour la paire de phrases (e, f).

La Table 2.3 présente un extrait d'appariements possibles entre segments et les différents scores qui leur sont associés. Nous pouvons observer que ceux-ci sont parfois très bruités ; c'est par exemple le cas de l'association de « le salaire moyen d' un » avec « an average », qui obtient une probabilité $p(e|f)$ relativement élevée de 0,2, relativisée toutefois par un faible score de pondération lexicale $lex(e|f)$ de $0,00021122$.

bisegments					
Segments source	**Segments cible**	$p(e\|f)$	$lex(e\|f)$	$p(f\|e)$	$lex(f\|e)$
le salaire moyen d' un	the average salary of a	0,2	0,0012	0,25	3,1405x10^{-4}
	the average wage of a ,	0,2	9,3451x10^{-4}	1,0	2,4754x10^{-4}
	an average	0,2	2,1122x10^{-4}	1,1457x10^{-4}	3,9517x10^{-9}
	average salary for an	0,2	9,6648x10^{-5}	1,0	2,3409x10^{-5}
le deuxième	The second a	0,0087	0,0016	0,0021	0,0464
	the second	0,4649	0,3328	0,0011	0,0863
	the second one	0,0087	6,7489x10^{-4}	0,0037	0,0863
	the third	0,0087	8,8716x10^{-4}	4,9935x10^{-5}	3,0798x10^{-4}
rétrogradé	demoted in rank	0,0204	2,4835x10^{-4}	1,0	0,098
	reduced	0,0204	0,0196	2,6524x10^{-5}	2,2014x10^{-5}
	stipulating that	0,0204	2,7048x10^{-4}	0,0030	0,0014
	demoted	0,204	0,2549	0,1785	0,1969
	reduction	0,0204	0,0196	2,0738x10^{-5}	1,9228x10^{-5}
temps ,	weather ,	0,0098	0,0072	0,0011	0,0936
	time ,	0,3333	0,4276	8,5928x10^{-4}	0,2564

TABLE 2.3 – Exemples de bisegments pour des segments source en français et des segments cible en anglais.

2.3.2 Une fonction de score qui met en jeu des modèles

2.3.2.1 Une fonction de score pour évaluer les traductions

Comme décrit dans la section 2.1, un système de TAS cherche la meilleure hypothèse e_{best} telle que décrite par l'équation 2.1. Les systèmes décomposent typiquement ce problème en combinant plusieurs modèles, ce qui donne pour un système fondé sur les segments :

$$e_{best} = \arg\max_{e} \; p_{lm}(e)p_{tm}(f,e)p_w p_{dis} \tag{2.6}$$

Cette équation met en jeu le score d'un modèle de langue $p_{lm}(e)$ (décrit dans la section 2.3.2.3), des scores de modèles de traduction $p_{tm}(f,e)$ (cf. section 2.3.2.2), un score dit de *distorsion* p_{dis} (décrit dans la section 2.3.2.4), ainsi qu'un score dit de *pénalité lexicale* p_w qui modélise la longueur de l'hypothèse de traduction. Toutefois, chaque modèle peut prendre une importance différente lors de la recherche de la meilleure traduction (le *décodage*, décrit section 2.3.3). Ceci est obtenu en attribuant un poids λ propre à chaque score de modèle :

$$e_{best} = \arg\max_{e} \; p_{lm}(e)^{\lambda_{lm}} p_{tm}(f,e)^{\lambda_{tm}} p_w^{\lambda_w} p_{dis}^{\lambda_d} \tag{2.7}$$

Chaque terme de cette équation pouvant avoir une valeur numérique extrêmement faible, la multiplication de leur valeur peut produire des résultats dont la précision serait difficile à représenter dans un ordinateur. Pour éviter ce problème, l'équation peut être réécrite sous la forme d'une combinaison linéaire de logarithmes des scores de modèles (Och et Ney, 2002) (combinaison traditionnellement décrite comme « log-linéaire » (*log-linear*)) :

$$e_{best} = \arg\max_{e} \; \lambda_{lm} \log p_{lm}(e) + \lambda_{tm} \log p_{tm}(f,e) + \lambda_d \log p_{dis} + \lambda_w \log p_w \tag{2.8}$$

Pour attribuer un score à une hypothèse de traduction complète, on peut donc utiliser la fonction suivante :

$$score(f,e) = \lambda_{lm} \log p_{lm}(e) + \lambda_{tm} \log p_{tm}(f,e) + \lambda_d \log p_{dis} + \lambda_w \log p_w \tag{2.9}$$

Le processus d'optimisation des poids λ sera décrit dans la section 2.5.

2.3.2.2 Des tables de traduction pour collecter les bisegments

Un système de traduction fondé sur les segments repose sur des informations sur les bisegments qui sont regroupées dans une *table de traduction*. Celle-ci contient tous les bisegments extraits lors d'une phase préliminaire, lesquels sont associés à un ensemble de scores, dont la probabilité de traduction directe, qui caractérisent l'association entre le segment source et le segment cible du bisegment. Lors de la génération d'une traduction, les scores de tous les bisegments utilisés sont combinés en tenant compte de leur poids

	table de traduction A	table de traduction B
	bisegment X dans A	bisegment X absent de B
scores associés à X	a b c	1,0 1,0 1,0 1,0
au cours du décodage	$\alpha \log(a) + \beta \log(b) + \gamma \log(c)$	0,0 + 0,0 + 0,0 + 0,0

TABLE 2.4 – Illustration de la situation dans laquelle un bisegment X, utilisé au cours du décodage, et provenant d'une table de traduction A, est absent de la table de traduction B. a, b et c sont les scores associés au bisegment X et α, β et γ leur poids respectif.

respectif. De façon plus générale, le score de traduction d'une hypothèse e se calcule donc de la façon suivante à partir des scores issus de la table de traduction utilisée :

$$\begin{aligned} p_{tm}(f,e) &= \sum_{i=1} \lambda_{p(e|f)} \log p(e_i|f_i) \\ &+ \sum_{i=1} \lambda_{lex(e|f)} \log lex(e_i|f_i) \\ &+ \sum_{i=1} \lambda_{p(f|e)} \log p(f_i|e_i) \\ &+ \sum_{i=1} \lambda_{lex(f|e)} \log lex(f_i|e_i) \end{aligned} \tag{2.10}$$

où (f_i, e_i) dénote le $i^{\text{ème}}$ bisegment dans l'ordre de la phrase source segmentée.

Dans certaines situations on peut être amené à exploiter plusieurs tables de traduction simultanément au cours du décodage. Cela peut par exemple être utilisé pour *adapter* un système à un domaine particulier (Koehn et Schroeder, 2007; Axelrod *et al.*, 2011; Bisazza *et al.*, 2011) ; une table de traduction apprise sur des données génériques et une seconde table apprise sur des données du domaine visé peuvent par exemple être utilisées. La première sera typiquement plus complète, mais la seconde contiendra vraisemblablement des bisegments et des scores plus adaptés aux textes à traduire. Puisqu'un bisegment peut n'apparaître que dans l'une des deux tables, la combinaison log-linéaire des scores des deux tables est permise par l'ajout virtuel des bisegments absents d'une des deux tables avec des scores fixés à 1, tel qu'illustré dans la Table 2.4. [9] Chaque score de chaque table de traduction reçoit son propre paramètre λ qui obtiendra donc une valeur propre après optimisation. On s'attendra typiquement à ce que, par exemple, le modèle de traduction issu d'une table de traduction apprise depuis un corpus adapté prenne plus d'importance qu'un modèle de traduction appris depuis un corpus peu adapté.

2.3.2.3 Modèle de langue

Le modèle de langue, typiquement de type n-gramme, tente d'évaluer la « vraisemblance » de la phrase produite dans la langue cible. Il calcule la probabilité pour chaque

9. L'utilisation de plusieurs de tables de traduction au cours du décodage est notamment décrite par Birch *et al.* (2007).

mot de la phrase dans le contexte des $n-1$ mots qui le précèdent. Ceci est illustré dans l'exemple suivant pour un modèle trigramme ($n = 3$), après le passage en logarithmes [10] :

$$\begin{aligned}\log P(I, saw, the, red, house) &= \log P(I| < s >, < s >) \\ &+ \log P(saw| < s >, I) \\ &+ \log P(the|I, saw) \\ &+ \log P(red|saw, the) \\ &+ \log P(house|the, red) \\ &+ \log P(< /s > |red, house) \end{aligned} \tag{2.11}$$

Si, lors du calcul, une séquence de mots contient un n-gramme inconnu du modèle de langue, celui-ci recevrait par défaut une probabilité nulle, ce qui donnerait également une probabilité nulle à la phrase entière qui le contient. Pour éviter cette situation un lissage (*smoothing*) est effectué pour attribuer une probabilité non nulle aux n-grammes absents des données utilisées pour l'apprentissage du modèle de langue. Les deux principales méthodes de lissage utilisées sont Witten-Bell et Kneser-Ney (Chen et Goodman, 1998).

Il faut également noter que plus une phrase compte de mots, et plus il est possible que le score calculé par le modèle de langue devienne faible [11]. Pour contrebalancer et effet, la fonction de score 2.9 utilise également une *pénalité lexicale* p_w, qui compte simplement le nombre de mots présents dans la phrase. [12]

2.3.2.4 Des modèles de réordonnancement pour réorganiser les traductions

Comme illustré par la Figure 2.1, les tokens dans la langue cible peuvent être dans un ordre différent de celui des tokens qu'ils traduisent dans la langue source. Il existe donc des différences dans l'ordre des mots entre langues qui doivent être prises en compte. Prenons l'exemple du japonais où le verbe se situe le plus souvent en fin de phrase. Une traduction mot à mot vers l'anglais ne tenant pas compte de cette spécificité commettrait l'erreur de mettre le verbe en fin de phrase dans la traduction en anglais. Des différences plus locales existent également, par exemple concernant la position des adjectifs relativement au nom modifié en anglais et en français. La solution proposée dans l'approche statistique standard est d'ajouter à l'équation 2.2 un ou plusieurs modèles dits de *réordonnancement* (voir la section 2.3.2.4), qui estiment diverses probabilités de déplacer les mots relativement à une traduction *monotone* de la phrase source.

Ainsi, dans la fonction de score 2.9 intervient également un score de distorsion p_{dis} visant à autoriser des réordonnancements dans la traduction produite relativement à

10. `<s>` représente le début de la phrase, et `</s>` la fin de la phrase.
11. La combinaison de l'équation 2.9 utilisant les logarithmes des scores, les valeurs données obtenues sont négatives.
12. Dans la mesure où les autres scores de la fonction 2.9 sont tous négatifs, puisqu'il s'agit de logarithmes de probabilités, afin de conserver des scores de modèles relativement homogènes généralement p_w est en réalité le nombre de tokens de l'hypothèse soustrait à 0 pour obtenir également une valeur négative.

l'ordre de la phrase source. Dans sa forme la plus simple, il est calculé de la façon suivante :

$$p_d = d(a_i - b_{i-1} - 1) \tag{2.12}$$

où a_i représente la position du début du segment à traduire dans le $i^{\text{ème}}$ segment de la phrase traduite, et b_{i-1} la position de la fin du segment à traduire dans le $(i-1)^{\text{ème}}$ segment de la phrase traduite. Le score obtenu est généralement soustrait à 0 afin d'obtenir une valeur de p_d négative pour les mêmes raisons que p_w (voir section 2.3.2.3). Une limite de distorsion est souvent imposée au décodeur pour éviter les réordonnancements de longue portée trop souvent arbitraires car mal modélisés par les modèles standard utilisés.[13] Cette limite de distorsion permet généralement d'améliorer les performances du système en termes de qualité et de rapidité[14]

D'autres manières plus complexes de modéliser les réordonnancements, lexicalisées (Tillmann, 2004; Koehn *et al.*, 2005) et/ou hiérarchiques (Galley et Manning, 2008), sont couramment utilisées en complément du score de distorsion, et peuvent se montrer très utiles pour des paires de langues où l'ordre des mots diffèrent beaucoup. Ici, nous ne discuterons que du fonctionnement des modèles de réordonnancement lexicalisés qui sont bien plus utilisés que les modèles hiérarchiques pour la paire de langues anglais-français.

Les modèles de réordonnancement lexicalisés considèrent une séquence d'orientations $o = (o_1 ... o_n)$ de différents types, et reposent sur les fréquences auxquelles chaque bisegment est trouvé avec chaque type d'orientation considéré :

$$p_o(o|e,f) = \frac{\text{count}(o,e,f)}{\sum_o \text{count}(o,e,f)} \tag{2.13}$$

où o est une séquence d'orientations d'un type particulier, e et f les segments source et cible du bisegment considéré. Généralement, 3 types d'orientation sont utilisés et donc associés à une probabilité pour chaque bisegment :

- monotone : le bisegment suit directement le bisegment qui le précéde
- inversion (*swap*) : le bisegment est placé avant celui qui le précéde
- discontinue (*discontinuous*) : le bisegment n'est pas adjacent au bisegment qui le précéde

2.3.3 Parcours de l'espace de recherche

Trouver l'hypothèse de traduction ayant le meilleur score tel que défini par l'équation 2.9 est un problème NP-difficile (Knight, 1999), qui nécessite de chercher le meilleur compromis entre tous les modèles présents dans la fonction de score utilisée. Les tailles

13. Généralement cette limite de distorsion est empiriquement fixée à 6 pour la paire de langues anglais-français.

14. En effet, limiter les réordonnancements possibles réduit significativement la taille de l'espace de recherche du système.

de la table de traduction, du modèle de langue et de l'espace des réordonnancements possibles étant particulièrement grandes, seul un algorithme de parcours heuristique de l'espace de recherche pourra produire une traduction en un temps raisonnable durant la phase dite de *décodage* (*decoding*). La solution employée par les *décodeurs* consiste à parcourir l'espace des préfixes en langue source en concaténant et réordonnant les segments au fur et à mesure de la construction des hypothèses de traduction. Une fois le décodage terminé, un *treillis de mots* (*word lattice*) peut être compilé à partir des chemins du graphe de recherche effectivement parcouru pour générer la traduction.

Dans le but de réduire le temps de calcul, plusieurs techniques visant à réduire la taille de l'espace de recherche ont été proposées reposant sur une recherche de type A* (Och *et al.*, 2001) ou sur des algorithmes gloutons (Germann *et al.*, 2001a; Germann, 2003a). La technique de recherche par faisceau (*beam search*) (Koehn *et al.*, 2003) figure encore aujourd'hui parmi les plus utilisées [15]. Celle-ci permet de ne conserver en mémoire que les k meilleures solutions trouvées jusque-là, focalisant ainsi la recherche sur un nombre limité d'hypothèses les plus prometteuses. Lorsque la phrase source a été entièrement traduite, on peut extraire la meilleure hypothèse de l'espace de recherche effectivement parcouru selon la fonction de score utilisée, voire les n meilleures (on parle alors informellement de *liste de n meilleures hypothèses (*n-best lists*)*. Une méthode plus récente qui combine recherche par faisceau et *cube pruning* (Huang et Chiang, 2007) permet de générer une traduction plus rapidement tout en produisant une traduction de qualité équivalente. Les problèmes posés par ces types de parcours heuristiques, notamment pour l'utilisation de certains modèles, seront discutés dans la section 3.2.1.

2.4 Les métriques d'évaluation automatique de la traduction

2.4.1 Rôles des métriques d'évaluation automatique

La question de l'évaluation des systèmes de traduction s'avère être un problème relativement complexe (Koehn, 2010), qui génère régulièrement de nouvelles propositions (par exemple, (Federmann, 2010; Graham *et al.*, 2015)). De nombreuses métriques d'évaluation automatiques de la performance des systèmes de traduction ont été développées, tel que celles proposées lors des récentes campagnes d'évaluation de l'atelier internation en traduction automatique statistique WMT (Machacek et Bojar, 2014; Stanojević *et al.*, 2015) . Celles-ci dispensent notamment du recours à des évaluations humaines, qui sont coûteuses, difficiles à mettre en place (Graham *et al.*, 2015) et non reproductibles. Les métriques BLEU [16] (Papineni *et al.*, 2002), TER [17] (Snover *et al.*, 2006) et

15. Cette technique est notamment employée par les décodeurs *open source* très utilisés que sont `cdec` (Dyer *et al.*, 2010), `moses` (Koehn *et al.*, 2007), `Joshua` (Post *et al.*, 2013) ou `phrasal` (Green *et al.*, 2014a).
16. *BiLingual Evaluation Understudy*
17. *Translation Error Rate* (aussi parfois appelée *Translation Edit Rate*

METEOR[18] (Denkowski et Lavie, 2014) sont parmi les plus utilisées. Outre l'évaluation automatique d'une traduction, ces métriques sont également utilisées pour optimiser les systèmes (voir la section 2.5).

2.4.2 BLEU et ses variantes

La métrique BLEU[19] (Papineni *et al.*, 2002) mesure la ressemblance entre une hypothèse de traduction et une ou plusieurs traductions humaines dites *de référence*. Son calcul se compose de précisions n-grammes et d'une *pénalité de concision* (*brevity penalty* selon l'équation 2.14 :

$$BLEU = BP \cdot \left(\prod_{n=1}^{N} p_n \right)^{\frac{1}{N}} \tag{2.14}$$

La pénalité de concision est donnée par l'équation 2.15. Elle permet à BLEU la prise en compte dans son calcul de la taille de l'hypothèse évaluée par rapport à la taille de la traduction de référence. c et r étant respectivement la longueur (en nombre de tokens) de la traduction produite et de la traduction de référence, et p_n la précision n-gramme[20].

$$BP = \begin{cases} 1 & \text{si } c > r \\ e^{1-r/c} & \text{si } c \leq r \end{cases} \tag{2.15}$$

BLEU est une métrique qui a été conçue pour l'évaluation de la traduction de corpus[21]. p_n, c et r correspondent donc à des sommes prenant en compte l'ensemble des phrases du corpus évalué. Cela en fait une métrique très mal adaptée pour l'évaluation de la traduction de textes très courts, constitués par exemple d'une seule phrase. En effet, si l'on utilise BLEU en mesurant la précision jusqu'aux 4-grammes et si l'hypothèse de traduction ne contient aucun des 4-grammes de la traduction de référence, son score sera nul. Pour corriger cela, des variantes de BLEU ont été développées (parmi lesquelles celles proposées par Lin et Och (2004); Nakov *et al.* (2012); Chen et Cherry (2014)), et sous différentes appellations : *s*BLEU, *sentence-level* BLEU, *smoothed*-BLEU ou encore BLEU*+1*. L'idée commune à la plupart de ces variantes repose sur l'« hallucination » d'un n-gramme supplémentaire pour tout n-gramme de la traduction de référence, assurant ainsi qu'aucune précision n-gramme p_n ne soit nulle.

18. *Metric for Evaluation of Translation with Explicit ORdering*
19. La dénomination de cette métrique est parfois précisée de façon à rendre plus explicite son calcul : BLEU-IBM (BLEU tel qu'il est décrit par Papineni *et al.* (2002)), BLEU-4 (4 étant ici la taille maximale des n-grammes pris en compte dans le calcul) ou encore %BLEU (pour rendre plus explicite le fait que le score BLEU, normalement compris entre 0 et 1, est en fait multiplié par 100). Dans ce manuscrit, tout comme dans la plupart des travaux en TAS, les scores BLEU présentés sont donnés après leur multiplication par 100 pour plus de lisibilité.
20. En général, on utilise 4 pour la valeur de N.
21. Généralement, au moins plusieurs centaines de phrases. Par exemple, les corpus utilisés pour les campagnes d'évaluation WMT pour l'évaluation des systèmes de traduction ont une taille d'environ 3 000 phrases.

Pour une meilleure prise en compte de la variété des traductions possibles, il est possible de calculer BLEU avec plusieurs traductions de référence. On parle alors de multi-BLEU. Dans cette situation, tous les n-grammes de toutes les traductions sont accumulés, et la longueur retenue pour calculer la pénalité de concision sera la plus proche de la longueur de l'hypothèse évaluée. Cependant, produire une traduction de référence est une tâche coûteuse, si bien que les situations où l'on dispose de plusieurs traductions de référence sont rares.

2.4.3 TER

La métrique TER (Snover *et al.*, 2006) mesure un coût de transformation d'une hypothèse de traduction en une traduction de référence à l'aide d'opérations d'édition élémentaires sur les tokens : substitution, insertion, suppression et déplacement. Le score se calcule selon l'équation 2.16. [22]

$$TER = \frac{\text{nombre d'opérations effectuées}}{\text{nombre de mots de la traduction de référence}} \tag{2.16}$$

À la différence de BLEU, TER peut donc être utilisée pour évaluer la traduction d'un corpus ou d'unités de texte bien plus petites.

Une variante de TER, nommée TERp (Snover *et al.*, 2009), utilise des ressources linguistiques (base de synonymes, de paraphrases et raciniseur) pour considérer notamment les cas où le token est bien traduit même si sa traduction n'est pas exactement celle présentée dans la traduction de référence mais, par exemple, un synonyme.

2.4.4 METEOR

Le calcul de la métrique METEOR (Denkowski et Lavie, 2014) repose sur l'alignement des tokens entre une hypothèse de traduction et une traduction de référence. Cet alignement est effectué en utilisant des ressources linguistiques tels que des outils de lemmatisation, un ensemble de paraphrases ou des bases de données lexicales (*wordnet* (Fellbaum, 1998) par exemple). Elle permet notamment de détecter les cas de synonymie ce que ne font pas TER et BLEU. Ces caractéristiques en font une métrique qui met l'accent sur le *rappel* plutôt que la *précision* en récompensant plus fréquemment les traductions proposées [23]. Cette métrique obtient de meilleures corrélations avec le jugement humain que des métriques telles que BLEU ou TER (Machacek et Bojar, 2014). Elle reste cependant moins utilisée, notamment du fait de la possible indisponibilité des ressources linguistiques nécessaires à son calcul pour certaines langues, mais également parce qu'il s'agit d'une métrique relativement coûteuse à calculer.

22. Il faut donc noter que, contrairement à BLEU par exemple, plus un score TER est bas et meilleure la traduction est supposée être.

23. Parfois à tort étant donné notamment l'absence de prise en compte de l'ambiguïté lexicale (Apidianaki et Marie, 2015) et la présence de bruit dans les ensembles de paraphrases utilisés.

où se trouve l' arrêt de bus pour l' hôtel de ville ?
where is the bus stop for city hall ?
where is the bus stop for buses going to city hall ?
where do I have to get off for the City Hall ?
where can I get off for the City Hall ?
which stop is for the City Hall ?
where can I catch the bus that goes to City Hall ?
where does the bus , that goes to City Hall , pick up passengers ?
where is the bus stop for buses bound for City Hall ?
where can I catch a bus for city hall ?
do you know where the bus stop for city hall is ?
where is the stop for the bus that goes to City Hall ?
where would I catch the bus to go to City Hall ?
where can I catch a bus that goes to City Hall ?
where is the bus stop for City Hall located ?
which way to the bus stop for City Hall ?
could you tell me where to take the bus for City Hall ?

TABLE 2.5 – Extrait du corpus BTEC (Takezawa *et al.*, 2002) d'une phrase en français à traduire et de 15 traductions possibles en anglais.

2.4.5 Les limites de l'évaluation automatique

Les métriques automatiques partagent de nombreux défauts. Le plus notable est la contrainte de similarité, très restrictive, à des traductions de référence déjà produites. Il peut en effet, comme l'illustre la Table 2.5, exister un grand nombre de traductions acceptables pour une même phrase à traduire. En outre, l'interprétation de ces scores s'avère souvent difficile : un score BLEU élevé, par exemple, n'implique pas qu'une hypothèse sera *globalement* ainsi que *localement* correcte au niveau du sens (*adequacy*) ni de la langue (*fluency*). On peut également remarquer qu'une traduction de référence, considérée comme *étalon or* par ces métriques, peut contenir des imperfections imputables notamment à des erreurs commises par le traducteur humain les ayant produites.

2.5 L'optimisation d'un système de traduction

2.5.1 La recherche des meilleurs poids à attribuer aux modèles

Comme nous l'avons vu lors de la présentation de l'équation 2.10, chaque modèle de la fonction de score est associé à un poids λ. Ces poids nécessitent d'être optimisés afin que le système de traduction puisse produire la meilleure traduction possible en fonction des modèles utilisés. Cette optimisation (*tuning*) vise donc à trouver le meilleur compromis entre les différents modèles.

Pour effectuer cette optimisation, un corpus parallèle bilingue dit de *développement* (*development/tuning set*) est utilisé. Ces corpus sont typiquement de taille limitée (au plus quelques milliers de paires de phrases) et ne doivent pas contenir de paires de phrases qui seront utilisées pour l'apprentissage des modèles ou pour l'évaluation du système. Il est néanmoins important qu'il soit représentatif du type de texte que le système sera amené à traduire, car l'optimisation des poids sera fortement influencée par des caractéristiques propres au type de texte considéré.

Il existe principalement deux types d'algorithme d'optimisation en TAS, dont les performances sont similaires (Green *et al.*, 2013b) : par lots (*batch*) ou en ligne (*online*). Dans ce travail seuls des algorithmes d'optimisation par lots ont été utilisés, nous ne décrirons donc que ce type d'algorithme ici. [24]. En effet, ceux-ci sont plus faciles à mettre en œuvre que les algorithmes d'optimisation en ligne tout en donnant des performances équivalentes. Le décodeur génère une première fois les n meilleures hypothèses de traduction du corpus de développement avec des poids λ initiaux fixés manuellement. L'algorithme cherche ensuite à réévaluer ces n meilleures hypothèses en trouvant les poids maximisant le score donné par une métrique d'évaluation automatique (BLEU le plus souvent). Le décodeur produit ensuite une nouvelle liste de n meilleures hypothèses avec les nouveaux poids trouvés, laquelle est concaténée (sans doublons) à la liste d'hypothèses générées jusque-là. L'algorithme d'optimisation est relancé sur cette nouvelle liste enrichie d'hypothèses, qui est supposée contenir davantage de *diversité*, ce qui doit permettre une meilleure optimisation des poids. Cet enchaînement décodage/optimisation est répété jusqu'à convergence, soit lorsque le décodeur ne produit plus de nouvelles hypothèses par rapport aux itérations précédentes, ou lorsqu'un nombre d'itérations fixé à l'avance est atteint.

Pour l'évaluation du système, les poids λ utilisés sont ceux obtenus lors de la dernière itération de l'optimisation, ou bien ceux de l'itération ayant permis d'obtenir le meilleur score de la métrique utilisée. Il est à noter qu'on obtient de meilleurs résultats si la métrique utilisée pour l'évaluation est la même que celle utilisée lors de l'optimisation du système (Cer *et al.*, 2010). Il existe aujourd'hui de nombreux algorithmes d'optimisation aux performances comparables sous certaines conditions. `MERT` (Och, 2003) est encore très utilisé malgré son caractère inadapté pour l'optimisation d'une combinaison de plus d'une vingtaine de modèles (Chiang *et al.*, 2008) et sa forte propension à donner des résultats très variables d'une optimisation à l'autre. Des algorithmes plus stables tels que `PRO` (Hopkins et May, 2011) et `KB-MIRA` (Cherry et Foster, 2012) (voir section 2.5.2 pour les détails de l'algorithme) permettent d'optimiser les poids de très nombreux modèles dont des modèles parcimonieux (*sparse*).

24. Pour plus de détails, l'algorithme d'optimisation en ligne MIRA est décrit par Watanabe *et al.* (2007).

2.5.2 L'algorithme d'optimisation KB-MIRA

Les expériences décrites dans ce manuscrit utilisent principalement l'algorithme KB-MIRA pour optimiser les poids des différentes fonctions de score introduites. En effet, cet algorithme d'optimisation, qui est à l'état de l'art (Cherry et Foster, 2012), permet d'optimiser des fonctions de score mettant en jeu de nombreux modèles[25]. KB-MIRA repose sur l'algorithme MIRA (*Margin Infused Relaxed Algorithm*) (Crammer *et al.*, 2006) introduit en TAS par Watanabe *et al.* (2007). MIRA utilise, dans sa forme la plus simple, la fonction de perte structurée (*structured hinge loss*) suivante :

$$l_i(\overrightarrow{w}) = \max_{e \in \varepsilon_i} \left[\Delta_i(e) + \overrightarrow{w} \cdot \left(\overrightarrow{h_i}(e) - \overrightarrow{h_i}(e_i^*) \right) \right] \tag{2.17}$$

où $\overrightarrow{w}$ est le vecteur de poids attribué aux modèles, ε l'ensemble des hypothèses de l'espace de recherche, e_i^* une traduction trouvée en ayant connaissance de la traduction de référence et avec un coût défini tel que $\Delta_i(e) = BLEU_i(e_i^*) - BLEU_i(e_i)$ avec $\Delta_i(e_i^*) = 0$. $l_i(\overrightarrow{w})$ vaut 0 si $\overrightarrow{w}$ sépare chaque $e \in \varepsilon_i$ de e_i^* d'une marge proportionnelle à la différence de leur score BLEU. MIRA est un algorithme d'apprentissage en ligne qui répète les étapes suivantes :

1. sélectionner un exemple i
2. décoder avec $\overrightarrow{w}$
3. mettre à jour $\overrightarrow{w}$ pour réduire $l_i(\overrightarrow{w})$

Ces étapes sont répétées un nombre de fois fixé à l'avance sur le corpus de développement. Chaque mise à jour effectue le plus petit changement de $\overrightarrow{w}$, limité par un paramètre de régularisation C, pour séparer la meilleure traduction produite (selon la référence) des exemples négatifs . Si e_i' est le $e \in \varepsilon_i$ qui maximise $l_i(\overrightarrow{w})$, alors la mise à jour de $\overrightarrow{w}$ est effectuée avec :

$$\begin{aligned} \eta_t &= \min \left[C, \frac{l_i(\overrightarrow{w})}{\| \overrightarrow{h_i}(e_i^*) - \overrightarrow{h_i}(e_i') \|^2} \right] \\ \overrightarrow{w}_{t+1} &= \overrightarrow{w}_t + \eta_t \left(\overrightarrow{h_i}(e_i^*) - \overrightarrow{h_i}(e_i') \right) \end{aligned} \tag{2.18}$$

Le coût Δ_i est défini avec un BLEU *pseudo-corpus* utilisant les comptes des n-grammes des meilleures traductions obtenues au cours des dernières mises à jour.

Le principal avantage de l'apprentissage en ligne est qu'il permet d'apprendre en ayant directement le décodeur dans le processus d'apprentissage, les poids étant mis à jour après chaque phrase décodée, éliminant ainsi le besoin d'utiliser une approximation de l'espace de recherche telle que représentée par une liste de n meilleures hypothèses. Cependant, les algorithmes d'apprentissage en ligne sont connus pour être difficiles à paralléliser. Or au cours de l'optimisation l'étape le plus coûteuse est le décodage, la non-parallélisation de celui-ci rend donc l'optimisation avec MIRA très coûteuse. Les travaux de Chiang *et al.*

25. Nous avons utilisé l'implémentation de KB-MIRA fournie avec le décodeur moses que nous utilisons dans nos expériences.

(2008); McDonald *et al.* (2010); Hasler *et al.* (2011), par exemple, proposent une adaptation de MIRA permettant une telle parallélisation mais au prix d'une complexification significative de l'implémentation de MIRA. C'est pourquoi Cherry et Foster (2012) ont proposé la variante KB-MIRA reposant sur un apprentissage par lots et s'intégrant dans la procédure d'optimisation au même endroit que MERT ou PRO. Cette variante s'avère donc beaucoup donc facile à intégrer dans les systèmes existants que MIRA. L'optimisation qu'elle effectue comporte les étapes suivantes :

1. décoder de façon parallélisée : $[\tilde{\varepsilon}']_1^n = k\text{-meilleures}([f, \varepsilon]_1^n, \overrightarrow{w})$
2. accumuler : $[\tilde{\varepsilon}]_1^n = [\tilde{\varepsilon}]_1^n \cup [\tilde{\varepsilon}']_1^n$
3. apprendre : $\overrightarrow{w} = \text{BatchMIRA}([f, R, \tilde{\varepsilon}]_1^n, \overrightarrow{w})$
4. répéter les étapes 1 à 3 m fois

où BatchMIRA() correspond à la version adaptée de MIRA et $\tilde{\varepsilon}$ à l'approximation de l'espace de recherche utilisée, soit les k meilleures hypothèses [26] produites par le décodeur. Le pseudo-code de BatchMIRA() est présenté par l'algorithme 1.

Algorithme 1 Algorithme de KB-MIRA

Entrée : $[f, R, \tilde{\varepsilon}]_1^n$, $\overrightarrow{w}$, C, nombre d'époques J, pondération du pseudo-corpus γ
  Init pseudo-corpus BG avec de petits comptes positifs
  Init $t = 1$, $\overrightarrow{w}_t = \overrightarrow{w}$
  pour tout j de 1 à J **faire**
    pour tout i de 1 à n dans un ordre aléatoire **faire**
      $e_t^* = \arg\max_{e \in \varepsilon_i} [\overrightarrow{w}_t \cdot \overrightarrow{h_i}(e) + \text{BLEU}_i(e)]$
      $e_t' = \arg\max_{e \in \varepsilon_i} [\overrightarrow{w}_t \cdot \overrightarrow{h_i}(e) - \text{BLEU}_i(e)]$
      # mise à jour des poids
      $\Delta_t = \text{BLEU}_i(e_t^*) - \text{BLEU}_i(e_t')$
      $\eta_t = \min\left[C, \frac{\Delta_i + \overrightarrow{w} \cdot \left(\overrightarrow{h_i}(e) - \overrightarrow{h_i}(e_i^*)\right)}{\|\overrightarrow{h_i}(e_i^*) - \overrightarrow{h_i}(e_i')\|^2}\right]$
      $\overrightarrow{w}_{t+1} = \overrightarrow{w}_t + \eta_t \left(\overrightarrow{h_i}(e_i^*) - \overrightarrow{h_i}(e_i')\right)$
      # mise à jour des statistiques
      $BG = \gamma BG +$ statistiques de BLEU pour e_t^* et R_i
      $t = t + 1$
    fin pour
    $\overrightarrow{w}_j^{avg} = \frac{1}{nj} \sum_{t'=1}^{nj} \overrightarrow{w}_{t'}$
  fin pour
  retourner $\overrightarrow{w}_j^{avg}$ qui maximise le score BLEU du corpus de développement

Le meilleur ensemble de poids maximisant le score BLEU du corpus de développement, selon une traduction de référence R, obtenu après m décodages (m fixé à l'avance) est finalement utilisé lors de l'évaluation du système sur un corpus de test.

26. Ainsi, pour MIRA, $\tilde{\varepsilon} = \varepsilon$.

Résumé

Dans ce chapitre nous avons brièvement décrit les principales notions impliquées dans le développement d'un système de TAS fondé sur les segments : apprentissage des alignements mot à mot, modèles utilisés, parcours de l'espace de recherche, optimisation de la fonction de score et évaluation des traductions produites.

Le calcul d'alignements mot à mot repose sur des modèles IBM appris sur de grands corpus parallèles. Les alignements obtenus permettent d'extraire des bisegments, qui permettent de traduire des séquences de tokens, et qui servent ensuite unités de traduction de base pour construire des hypothèses de traduction au cours du décodage. Leur utilisation au cours du décodage est guidée par une fonction de score utilisant différents modèles capables d'évaluer efficacement les nombreuses hypothèses de traductions partielles qui sont construites : modèle de langue, modèles de traduction et modèles de réordonnancement. Chacun de ces modèles est associé à un poids dont l'optimisation par un algorithme tel que `KB-MIRA` est effectuée sur un corpus de développement, distinct du corpus qui sera utilisé pour évaluer le système. La mesure de la performance d'un système, lors de l'optimisation ou de l'évaluation, nécessite des métriques d'évaluation automatiques qui comparent les hypothèses de traductions produites à une ou plusieurs traductions de référence produites par un traducteur humain.

Les systèmes de TAS, qui sont l'aboutissement de plus de 40 ans de recherche en TA, sont cependant encore loin de produire des traductions utilisables en l'état. Chacun des mécanismes présentés dans ce chapitre fait l'objet de tentatives répétées d'amélioration. Par ailleurs, il existe de nombreuses manières d'améliorer des traductions, certaines impliquant un traducteur humain, telle que la post-édition, d'autres reposant sur l'utilisation de modèles plus complexes que ceux utilisés au cours du décodage. Nous nous intéresserons dans le chapitre 3 tout particulièrement aux raisons qui font que certains types de modèles ne peuvent pas être utilisés au cours du décodage ainsi qu'aux différentes façons de les exploiter pour améliorer les traductions produites par un décodeur standard à l'état de l'art.

Ce premier chapitre nous permet finalement d'illustrer de manière simple les étapes de production d'une traduction par un système de TAS au moyen de la Figure 2.2. Nous compléterons cette illustration au fur et à mesure des chapitres de ce manuscrit et des propositions d'améliorations qu'ils introduisent.

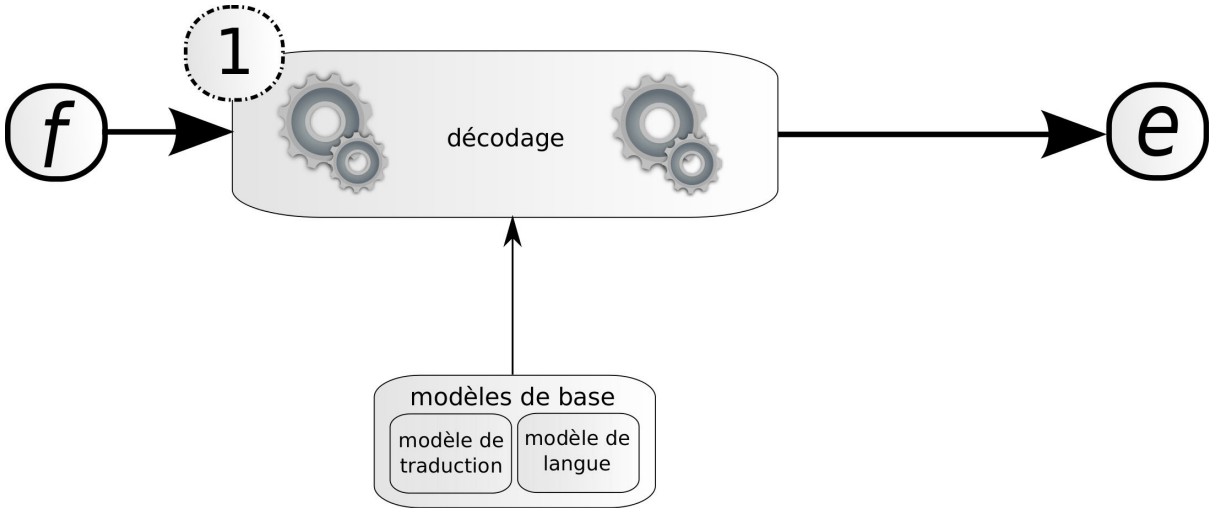


FIGURE 2.2 – Fonctionnement d'un système de TAS standard effectuant un simple décodage. Le décodage (1) d'une phrase f produit sa traduction e.

3

Améliorer la sortie d'un système de traduction automatique

Sommaire

3.1 Données et systèmes de traduction de référence **29**
3.1.1 Systèmes de traduction d'articles d'actualité (`news`) 29
3.1.2 Systèmes de traduction de textes médicaux (`médical`) 30
3.1.3 Systèmes de traduction de présentations orales (`TED Talks`) . 30

3.2 Le décodage et ses limites **31**
3.2.1 Un parcours de l'espace de recherche qui doit être efficace . . 31
3.2.2 Des modèles trop complexes pour être intégrés au décodage . 33

3.3 Le reclassement des n-meilleures hypothèses de traduction **36**
3.3.1 Le reclassement et ses limites 36
3.3.2 Systèmes de traduction de référence avec reclassement d'hypothèses 40

3.4 Regénérer une traduction après un premier décodage **43**

3.5 Au-delà du décodage : la recherche locale **45**

3.6 La post-édition pour améliorer une traduction **48**
3.6.1 La post-édition par un humain 48
3.6.2 La post-édition automatique 49

Résumé **51**

Le chapitre 2 a introduit les principes de fonctionnement de la traduction automatique statistique. L'approche statistique correspond aujourd'hui à l'état de l'art en TA, mais ses productions demeurent en général largement perfectibles. Jusqu'à présent, nous avons décrit la TAS comme un processus limité à un unique décodage : le texte à traduire est donné en entrée au décodeur pour produire sa meilleure traduction qui sera ensuite exploitée. Différentes stratégies peuvent venir modifier ce processus en intervenant *a priori* sur le texte à traduire, par exemple pour le rendre plus facile à traiter pour le décodeur, ou en intervenant *a posteriori* sur la traduction produite par le décodeur.

La modification du texte à traduire peut notamment permettre de simplifier son décodage. La *pré-édition* (*pre-editing*), par exemple, fait intervenir un humain pour corriger ou simplifier le texte à traduire. En effet, dans certaines situations le texte à traduire peut être imparfait et contenir des erreurs. Les mots mal orthographiés doivent notamment être corrigés afin que le décodeur ne se heurte pas à ces mots inconnus qu'il serait incapable de traduire. Cette situation est typique lorsqu'il s'agit de traduire des contenus informels générés par des utilisateurs (*user-generated content*) tels que des messages de forums (Gerlach *et al.*, 2013b,a) ou des commentaires d'articles d'actualité. Plus généralement, il est utile de chercher à diminuer le nombre de mots qui sont dits *hors-vocabulaire*, c'est-à-dire absents des données d'apprentissage du système ou pour lesquels aucune traduction n'a pu être apprise. D'autres approches *a priori* dites de *préordonnancement* (Xia et McCord, 2004; Collins *et al.*, 2005) (*preordering*) modifient l'ordre des mots dans le texte à traduire pour le rendre le plus proche possible de l'ordre de leur traduction dans la langue cible, facilitant ainsi le travail de réordonnancement du décodeur dont les modèles dédiés à cette tâche sont parfois insuffisants.

Nous nous sommes au contraire concentré dans notre travail sur les approches intervenant *a posteriori* du décodage pour améliorer les traductions produites. Ce chapitre commence par un état de l'art des différentes approches permettant, en particulier, d'améliorer une traduction à l'aide de modèles plus complexes que ceux utilisés par la fonction de score du décodeur et difficiles ou impossibles à utiliser au cours du décodage. Notre revue de l'état de l'art sera accompagnée d'expériences effectuées sur trois tâches de traduction différentes présentées dans la section 3.1 et qui seront utilisées tout au long du manuscrit. Nous mettrons ensuite en évidence dans la section 3.2 que la contrainte d'efficacité du parcours de l'espace de recherche est à l'origine des difficultés d'utilisation de certains types de modèles, que nous qualifierons en conséquence de *modèles complexes*, exploitant des informations au cours du décodage. Nous verrons dans la section 3.3 que de tels modèles complexes sont utilisés en TA *a posteriori* du décodage pour reclasser des listes d'hypothèses produites par le décodeur. Nous étudierons également une approche de *regénération* d'hypothèses, présentée dans la section 3.4, qui réutilise le décodeur pour regénérer une nouvelle traduction grâce à de nouveaux modèles appris après un premier décodage, ainsi qu'une approche de réécriture, présentée dans la section 3.5, qui met en jeu un algorithme de recherche locale. Nous donnerons ensuite dans la section 3.6 un aperçu des méthodes de post-édition impliquant un intervenant humain.

3.1 Données et systèmes de traduction de référence

Nous avons utilisé plusieurs jeux de données différents pour la paire de langues anglais-français pour construire les systèmes de traduction de référence sur lesquels sont basées les expériences décrites dans ce manuscrit. Les corpus parallèles utilisés sont différents pour chaque système : ils appartiennent à des domaines différents et contiennent des quantités de données différentes. Cette variété nous permettra de mieux apprécier les résultats obtenus par les différentes approches étudiées dans ce manuscrit à l'aide des configurations différentes ainsi obtenues.

Tous les corpus ont été segmentés en tokens avec le segmenteur développé au LIMSI et décrit dans (Déchelotte *et al.*, 2008). Tous les systèmes utilisent des tables de traduction extraites à partir d'alignements mot à mot appris à l'aide de l'outil `MGIZA++` (Gao et Vogel, 2008) en utilisant l'heuristique de symétrisation des alignements `grow-diag-final-and` [1]. L'extraction des tables de traduction à partir des alignements symétrisés est effectuée à l'aide des outils mis à disposition par le projet `moses` (Koehn *et al.*, 2007). L'optimisation des systèmes est réalisée par l'algorithme `KB-MIRA`, en utilisant l'implémentation fournie avec `moses`, avec la métrique BLEU sur des listes de 200 meilleures hypothèses. Le nombre maximal d'itérations pour `KB-MIRA` a été fixé empiriquement à 25 itérations, et les poids des modèles utilisés pour l'évaluation sont ceux de l'itération ayant obtenu le meilleur score BLEU au cours de l'optimisation. Le décodage en évaluation est effectué de la même manière avec `moses` et ses valeurs de paramètres par défaut. La performance des systèmes est mesurée avec les métriques d'évaluation BLEU [2] et TER. Pour faciliter l'implémentation des différents algorithmes testés dans ce manuscrit, les modèles de réordonnancement lexicalisés n'ont pas été activés au cours des expériences. L'outil `SRILM` (Stolcke, 2002) a été utilisé pour entraîner des modèles de langue 4-grammes avec un lissage de type Kneser-Ney (Chen et Goodman, 1998).

Nous présentons dans les trois sections suivantes les spécificités des différents systèmes étudiés. Des textes d'exemples pour chaque configuration sont donnés dans l'annexe A.1.

3.1.1 Systèmes de traduction d'articles d'actualité (`news`)

Pour la construction de la table de traduction, les systèmes `news` utilisent une grande quantité de données d'entraînement, qui correspondent à celles utilisées par le LIMSI lors la campagne d'évaluation en traduction automatique WMT 2013 [3] (Allauzen *et al.*, 2013). Le modèle de langue utilisé est également celui utilisé pour le système soumis par

1. L'heuristique `grow-diag-final-and` calcule d'abord l'intersection des alignements $e \to f$ et $f \to e$ puis ajoute des liens d'alignement provenant de l'union des alignements $e \to f$ et $f \to e$, notamment afin d'ajouter des liens sur les tokens restés non alignés.
2. Les scores BLEU donnés dans ce manuscrit sont calculés, sauf mention contraire, avec le script `multi-bleu.perl` fourni avec `moses`.
3. Toutes les données utilisées pour construire les systèmes `news` sont disponibles à l'adresse : `http://statmt.org/wmt13/`

le LIMSI, lequel a été entraîné sur une quantité de données monolingues particulièrement importante afin d'assurer une bonne modélisation de la langue cible. Pour l'optimisation des systèmes, les corpus de test officiels de la campagne d'évaluation précédente WMT 2009 ont été utilisés (`dev`), et l'évaluation est effectuée sur les corpus de test officiels de la campagne WMT 2011 (`test`). Ces corpus sont constitués d'articles d'actualité traitant de différents sujets. Les systèmes `news` ne sont donc pas spécialisés dans la traduction de textes d'un thème particulier. Il est en outre important de noter que les corpus de développement et d'évaluation utilisés, tout comme les données utilisées pour entraîner les modèles, contiennent des phrases qui sont déjà le résultat de traductions faite par des traducteurs humains dont la langue d'origine n'est pas connue.

La Table 3.2 décrit l'ensemble des données utilisées, qui sont de taille raisonnablement importante, et la Table 3.1 donne les performances de traduction obtenues par `moses`.

3.1.2 Systèmes de traduction de textes médicaux (`médical`)

Les systèmes `médical` utilisent les données utilisées pour la soumission du LIMSI à la campagne d'évaluation WMT 2014 pour la tâche de traduction médicale [4] (Pécheux *et al.*, 2014). Les corpus de développement (`dev`) utilisés pour l'optimisation des systèmes et les corpus d'évaluation (`test`) sont les corpus officiels de la campagne d'évaluation 2014. L'apprentissage des alignements mot à mot et l'entraînement du modèle de langue médical n'utilisent que des données fournies spécifiquement pour la tâche de traduction médicale de WMT 2014. Afin d'améliorer la performance des systèmes, les modèles de langue appris sur de grandes quantités de données utilisés par les systèmes `news` sont également utilisés par les systèmes `médical` (qui utilisent donc deux modèles de langue). Au contraire des systèmes `news`, les systèmes `médical` sont donc spécialisés pour la traduction de textes d'un domaine particulier, tout en utilisant une grande quantité de données d'entraînement.

La Table 3.2 décrit les données utilisées et la Table 3.1 donne les performances de traduction obtenues par `moses`.

3.1.3 Systèmes de traduction de présentations orales (`TED Talks`)

Les systèmes `TED Talks` utilisent uniquement les données spécifiques à cette tâche [5] fournies pour la campagne d'évaluation IWSLT 2010 [6] pour l'apprentissage des tables de

4. Les données utilisées pour construire les systèmes `médical` sont disponibles à l'adresse : http://www.statmt.org/wmt14/medical-task/

5. Il s'agit d'une collection de textes disponible publiquement qui correspondent à des transcriptions de présentations données dans le cadre des conférences organisées par la fondation TED, enregistrées en vidéo, et portant sur des thèmes très variés.

6. http://iwslt2010.fbk.eu/. Une version à jour des données est disponible à l'adresse suivante : https://wit3.fbk.eu/

traduction et les mêmes modèles de langue que ceux utilisés par les systèmes `news`[7]. Ce système a pour particularité d'utiliser des quantités de données d'entraînement beaucoup plus petites que les systèmes `news` et `médical`, et est en outre spécialisé pour la traduction de transcriptions de présentations orales préparées. Les corpus de développement (*dev*) et d'évaluation (`test`) utilisés sont ceux fournis pour la campagne d'évaluation IWSLT 2010.

La Table 3.2 décrit les données utilisées et la Table 3.1 donne les performances de traduction obtenues par `moses`. Si l'on se limite à la seule interprétation des scores de métriques obtenus, la tâche `médical` serait la plus facile, la tâche `news` la plus difficile, et la tâche `TED Talks` serait légèrement plus facile que la tâche `news`.

Système	fr→en		en→fr	
	BLEU	TER	BLEU	TER
`news`	28,6	52,4	31,1	51,8
`médical`	37,3	41,8	38,5	44,5
`TED Talks`	32,2	47,8	31,8	49,8

TABLE 3.1 – Scores BLEU et TER des systèmes de traduction `news`, `médical` et `TED Talks`, fr→en et en→fr.

3.2 Le décodage et ses limites

3.2.1 Un parcours de l'espace de recherche qui doit être efficace

Comme nous l'avons vu dans la section 2.3.3, le parcours de l'espace de recherche généré par le décodeur nécessite le recours à des approches heuristiques, telles que la recherche par faisceau, pour être effectué en un temps raisonnable.

Parmi les défauts des approches fondées sur des heuristiques, on note qu'une recherche par faisceau élimine les hypothèses les moins prometteuses, alors que celles-ci pourraient en fait permettre d'atteindre des hypothèses globalement meilleures. Cet élagage (*pruning*) des hypothèses les moins prometteuses se fait en tenant compte du coût courant de l'hypothèse partielle produite, c'est-à-dire de son score tel que donné par l'équation 2.9[8], ainsi que de son *coût futur* (*future cost*), qui est une estimation du coût global de la traduction une fois celle-ci complétée, calculée à l'aide d'une fonction de score appauvrie peu coûteuse[9]. En pratique, les approximations implémentées sont cependant devenues très

7. Les modèles de langue utilisés n'incluent donc pas la partie cible des données d'entraînement des systèmes `TED Talks`.
8. Plus le score de l'hypothèse est bas et plus son coût est élevé.
9. Une fonction de coût future typique consiste à calculer le coût correspondant à la traduction des

Tâche	Corpus	Phrases	Tokens (fr-en)	% OOV (fr-en)
`news`	entraînement	12 M	383 M - 318 M	
	dev	2 525	73 k - 65 k	1,8% - 1,8%
	test	3 003	85 k - 74 k	1,8% - 2,1%
	modèle de langue		2,5 G - 6 G	
`médical`	entraînement	4,9 M	91 M - 78 M	
	dev	500	12 k - 10 k	0,9% - 1,2%
	test	1 000	26 k - 21 k	1,0% - 1,2%
	modèle de langue médical		146 M - 78 M	
	modèle de langue générique		2,5 G - 6 G	
`TED Talks`	entraînement	107 758	2,2 M - 2 M	
	dev	934	20 k - 20 k	2,4% - 2,0%
	test	1 664	34 k - 31 k	1,5% - 1,5%
	modèle de langue		2,5 G - 6 G	

TABLE 3.2 – Description des données utilisées pour construire et évaluer les systèmes de traduction `news`, `médical` et `TED Talks`, fr→en et en→fr. La colonne « % OOV » donne pour les corpus de développement et d'évaluation le pourcentage de tokens source hors vocabulaire, i.e. non couverts par la table de traduction, et qui seront donc directement recopiés dans la traduction. Des exemples de tokens hors-vocabulaires sont donnés dans l'annexe B.

efficaces : il a été montré, par exemple, que le nombre d'erreurs de recherche produites par une recherche par faisceau relativement à un décodage exact était très faible (Aziz *et al.*, 2014).

En outre, le parcours de l'espace des préfixes d'une phrase à traduire produit des hypothèses de traduction *partielles*, tel qu'illustré par la Figure 3.1. Les hypothèses étant partielles, il est nécessaire que les scores des modèles utilisés par le décodeur soient *décomposables*, comme c'est le cas pour les modèles présentés dans la section 2.3.2. En particulier, il est donc impossible de calculer des scores portant sur une hypothèse complète en cours de décodage. Cela peut néanmoins être fait lors d'une nouvelle étape dite de *reclassement* (*reranking*) effectuée sur la liste des n meilleures hypothèses produites par le décodeur (voir la section 3.3.1). Cela a toutefois la limitation importante de restreindre ce calcul aux seules n meilleures hypothèses obtenues par recherche heuristique avec les modèles utilisés par le décodeur.

tokens restants en sélectionnant directement pour chacun d'eux leur traduction la plus probable en conservant l'ordre des mots.

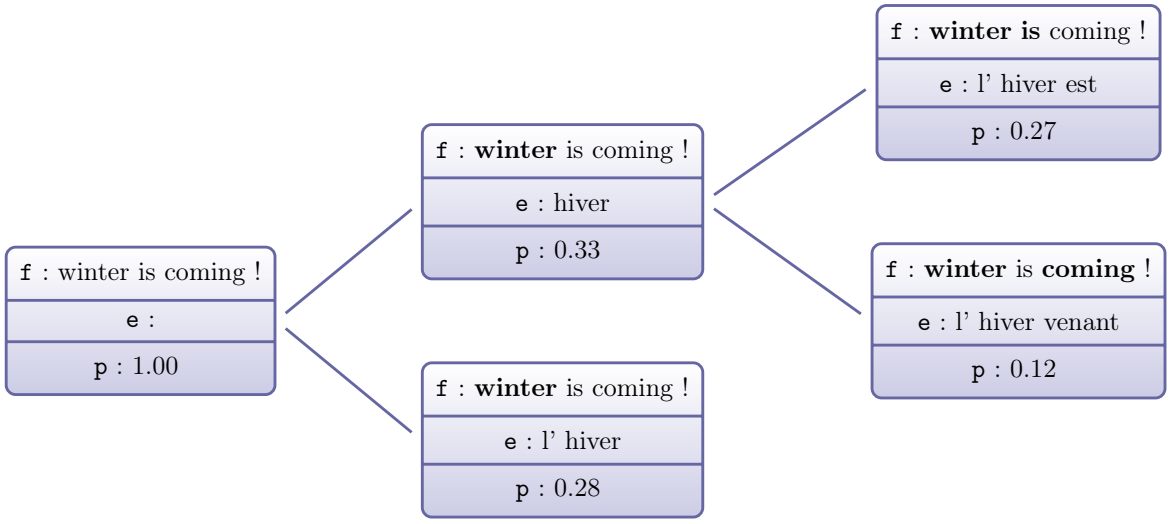


FIGURE 3.1 – Exemple de génération d'hypothèses de traduction partielles au cours du décodage pour la traduction partielle en français f de la phrase e en anglais « Winter is coming ! ». p correspond au score de l'hypothèse partielle produite traduisant les mots en gras de la phrase source.

3.2.2 Des modèles trop complexes pour être intégrés au décodage

Le mode de fonctionnement du décodage fait que de nombreux modèles seraient trop *complexes* pour être utilisés en cours de décodage. Différemment, et possiblement mieux informés, des modèles que l'on peut qualifier de complexes sont appris à partir d'*informations riches* : syntaxiques, sémantiques, discursives, ou encore obtenues *a posteriori* d'un premier décodage. De telles informations seraient probablement difficiles ou impossibles à utiliser lors du décodage, mais pourraient s'avérer utiles pour apprendre à mieux capturer certains aspects d'une traduction que ne le font les modèles effectivement utilisés lors du décodage. Par conséquent, leur utilisation, bien qu'elle n'assure pas que des gains en qualité de traduction seront obtenus, est souhaitable.

Nous décrivons dans cette section les trois principaux freins à l'intégration de tels modèles complexes lors du décodage : le besoin d'une hypothèse complète pour le calcul (section 3.2.2.1), le coût prohibitif d'un calcul à très large échelle (section 3.2.2.2), et l'impossibilité du calcul en cours de décodage (section 3.2.2.3). Cette description s'accompagne de la présentation de certains modèles que l'on considérera comme complexes dans le cadre de notre étude. Comme cela a été dit plus tôt, nous ne ferons aucune proposition originale de modèles, et nous contenterons de mettre en œuvre des modèles bien étudiés et performants et qui sont très utilisés en TA. Nous les considérerons donc uniquement comme des paramètres expérimentaux variables.

Il faut également noter que d'autres aspects du fonctionnement d'un système de TAS, non abordés ici, tel que le fait que ce type de système traduit les phrases de façon indépendante, peuvent aussi limiter l'utilisation de certains modèles, nécessitant par exemple

la prise en compte du niveau discursif (Carpuat, 2009; Hardmeier *et al.*, 2012).

3.2.2.1 Modèles nécessitant une hypothèse complète pour être calculés

Comme nous l'avons expliqué dans la section 3.2.1 et illustré à l'aide de la Figure 3.1, le décodage génère des hypothèses partielles que les modèles présentés dans la section 2.3.2 permettent d'évaluer par décomposition de leur score. Cependant, d'autres types de modèles n'ont de sens que si on les applique à une unité linguistique complète bien formée, typiquement une phrase. [10]

Prenons l'exemple d'un modèle informé par la syntaxe qui serait capable de prendre en compte des dépendances entre mots possiblement éloignés dans une hypothèse de traduction, ce que ne peut typiquement pas faire un modèle de langue de type n-gramme pour de petites valeurs de n. Une hypothèse partielle peu prometteuse selon les modèles de la fonction de score classique serait donc supprimée, même si celle-ci avait été la seule hypothèse pouvant mener à une hypothèse complète comportant toutes les dépendances que seul un modèle syntaxique complexe aurait pu reconnaître. Divers travaux ont certes eu recours à différentes techniques pour permettre l'inclusion de modèles syntaxiques au cours du décodage, mais au prix d'une forte augmentation des temps de calcul (Schwartz *et al.*, 2011). En pratique, de tels modèles sont uniquement utilisés pour reclasser des listes d'hypothèses (Carter et Monz, 2011).

Le problème est similaire pour d'autres types d'analyses. Les analyseurs sémantiques, par exemple, nécessitent également une hypothèse de traduction complète pour être performants, ce qui fait qu'en pratique ils ne sont utilisés qu'après un décodage sur des hypothèses complètes déjà formées (Wu et Fung, 2009). Il existe de nombreux autres modèles nécessitant une hypothèse complète pour être calculés, tels qu'une variante des modèles IBM 1 au niveau de la phrase qui prend en compte tous les tokens de la traduction produite (Hildebrand et Vogel, 2008) (en ne prenant pas en compte, toutefois, leur alignement avec le ou les tokens source qu'ils traduisent). Un tel modèle se calcule de la façon suivante :

$$P_{IBM1}(e|f_1^J) = \frac{1}{J+1}\sum_{j=0}^{J} p(e|f_j) \tag{3.1}$$

où f_1^J est la phrase source, J le nombre de tokens de la phrase source, f_0 le token source *nul* et $p(e|f_j)$ est la probabilité d'avoir le token cible e comme traduction du token source f_j. Une variante proposée par Hildebrand et Vogel (2008) remplace la somme des probabilités par la recherche d'une valeur maximale pour ne tenir compte que des tokens ayant la plus forte probabilité (voir l'équation 3.2). Des tels modèles ne peuvent être utilisés de façon optimale au cours du décodage car ils doivent avoir accès à tous les

10. À nouveau, ce raisonnement pourrait également s'appliquer à une unité plus longue telle qu'un document complet.

tokens de la traduction qui n'est pas encore complètement produite.

$$P_{IBM1max}(e|f_1^J) = \max_{j=0,\ldots,J} p(e|f_j) \tag{3.2}$$

3.2.2.2 Modèles trop coûteux pour être intégrés au cours du décodage

Pour d'autres modèles, le premier frein à leur intégration au cours du décodage est lié à leur coût en temps de calcul. En effet, et malgré l'utilisation d'heuristiques de recherche, le nombre d'hypothèses partielles produites et à évaluer au cours du décodage reste très élevé, ce qui en interdit le calcul, par exemple pour des modèles utilisant des réseaux neurones dont l'inférence peut être coûteuse [11]. C'est par exemple le cas des modèles neuronaux SOUL disposant d'une couche de sortie structurée (*Structured OUtput Layer*), modèles de langue (Le *et al.*, 2011) ou de traduction (Le *et al.*, 2012), qui peuvent être utilisés uniquement pour reclasser la liste des n meilleures hypothèses produites par le décodeur. Certains modèles ayant recours à des analyseurs syntaxiques, outre le fait qu'ils peuvent nécessiter une hypothèse complète, sont aussi très coûteux à calculer, et donc *a fortiori* pour un grand nombre d'hypothèses.

3.2.2.3 Modèles indisponibles au cours d'un premier décodage

Certains modèles ne peuvent être calculés qu'à l'issue d'un premier décodage. C'est en particulier le cas pour des modèles cherchant à évaluer le déroulement d'un décodage. Prenons l'exemple de la probabilité *a posteriori* qu'un mot soit présent dans la traduction produite (Blatz *et al.*, 2004). Un mot très présent par rapport aux autres mots dans la liste des n meilleures traductions révélera une certaine confiance du décodeur que ce mot doit figurer dans la traduction. [12]

Plusieurs méthodes de calcul ont été proposées par Ueffing et Ney (2007) pour déterminer la probabilité d'un mot *a posteriori*. Prenons l'exemple de la méthode fondée sur les comptes des mots (*count-based*) présents dans les n meilleures hypothèses produites :

$$p_e(|f_1^J) = \frac{p_e(n, f_1^J)}{\sum_{n'=0}^{n_{max}} p_e(n'|f_1^J)} \tag{3.3}$$

où

$$p_e(n, f_1^J) = \sum_{I, e_1^I} \delta(n_e, n) p(f_1^J, e_1^I) \tag{3.4}$$

11. Il faut cependant noter que des travaux récents, tels que ceux de Vaswani *et al.* (2013); Devlin *et al.* (2014), permettent désormais, grâce à différentes approximations, d'utiliser efficacement des modèles basés sur des réseaux de neurones au cours du décodage. Les modèles neuronaux que nous utiliserons dans ce manuscrit sont trop coûteux pour être utilisés au cours du décodage mais permettent d'obtenir des résultats similaires, sinon meilleurs, que des modèles neuronaux utilisés au cours du décodage (Bojar *et al.*, 2015).

12. A condition que le mot en question ne soit pas hors-vocabulaire (*Out-Of-Vocabulary*, *OOV*), c'est-à-dire que ce mot était présent dans les données parallèles utilisées pour entraîner le système de traduction. Dans le cas contraire celui-ci ne sera pas traduit et effectivement reporté dans toutes les hypothèses de traduction produites.

où n_e est le nombre de fois où le token e apparaît dans la traduction de la phrase f_1^J, J étant le nombre de phrases source à traduire. e_1^I est l'hypothèse de traduction considérée parmi les I meilleures hypothèses produites par le décodeur. $p(f_1^J, e_1^I)$ est le score donné par le décodeur à la traduction de f_1^J par l'hypothèse e_1^I. $\delta(n_e, n)$ est le delta de Kronecker [13] et n_{max} le nombre maximal de fois où un même token est observé dans une même hypothèse. $p_e(n|f_1^J)$ est donc la probabilité qu'un mot e apparaisse n fois dans la traduction de la phrase f_1^J. Par conséquent chaque fois que le token e apparaît n fois dans une hypothèse, la probabilité de $p_e(n|f_1^J)$ est incrémentée du score attribué par le décodeur à l'hypothèse.

Ce type de modélisation *a posteriori* peut également être étendu aux alignements utilisés par le décodeur, aux réordonnancements des mots dans la traduction, ou encore aux n-grammes produits (Zens et Ney, 2006; Gispert *et al.*, 2012). Une fois la probabilité *a posteriori* d'un mot ou d'un n-gramme connue il est ainsi possible de déterminer un nouveau score global pour la traduction produite étant donné, par exemple, les mots ou n-grammes qui la composent. Ce type de score a déjà pu montrer son efficacité pour le reclassement de listes de meilleures hypothèses (Ueffing et Ney, 2007). Il existe par ailleurs d'autres modèles plus complexes, estimant par exemple à l'aide d'un classifieur la *confiance* du syst-me dans le fait qu'un token soit correct ou incorrect, et qui repose en partie sur des calculs de probabilités *a posteriori* et sur d'autres modèles parfois coûteux à calculer. Il a été montré que ce type de modèles permet également d'améliorer la traduction *a posteriori* d'un premier décodage (Luong *et al.*, 2014b,a).

3.3 Le reclassement des n-meilleures hypothèses de traduction

3.3.1 Le reclassement et ses limites

Habituellement, les modèles complexes qui sont difficiles à intégrer au cours du décodage (tels que ceux présentés dans la section 3.2.2) sont utilisés en TAS par un système de reclassement d'hypothèses (Och *et al.*, 2004). Ce type de système intervient donc après un décodage, et calcule des scores pour les n meilleures hypothèses produites par le décodeur. [14] Les n meilleures hypothèses produites par le décodeur, initialement classées selon leur score calculé par le décodeur avec la fonction de l'équation 2.9, sont donc reclassées par le système de reclassement (*reranker*) selon le score calculé par une nouvelle fonction qui met typiquement en jeu un ou plusieurs nouveaux modèles. Grâce à l'utilisation de nouvelles informations riches, non exploitées lors du décodage, la nouvelle fonction de score a le potentiel de proposer un classement rendant mieux compte de la qualité réelle des hypothèses de traduction. Tout particulièrement, on espère que la nouvelle hypothèse

13. Cette fonction donne : $\delta(n_e, n) = \begin{cases} 0 & \text{si } n_e \neq n \\ 1 & \text{si } n_e = n \end{cases}$

14. On observe généralement une valeur de 1 000 pour n, mais il est aussi courant d'utiliser des valeurs plus petites telles que 100 ou 300 ; à l'inverse, n peut être beaucoup plus grand lorsque les modèles ajoutés sont peu coûteux à calculer (Ueffing et Ney, 2007).

de rang 1 correspondra à une meilleure traduction que la meilleure hypothèse du décodeur. La nouvelle fonction de score réutilise typiquement les modèles utilisés au cours du décodage auxquels seront combinés, par exemple linéairement, les nouveaux modèles. Pour apprendre les nouveaux poids associés à chaque modèle dans cette fonction de score, il est possible d'utiliser les mêmes algorithmes d'optimisation que ceux utilisés pour l'optimisation des systèmes tels que `MERT`, `PRO` ou `KB-MIRA` (voir la section 2.5).

La Figure 3.2 illustre le potentiel d'une telle approche pour les systèmes `news`, `médical` et `TED Talks`, fr→en et en→fr, en présentant les scores *oracle*[15] calculés avec *s*BLEU[16] sur la liste des n meilleures hypothèses produites par un système de TAS, `moses`, pour différentes valeurs de n. On constate que de meilleures hypothèses que celle préférée par le décodeur sont présentes dans cette liste, et cela même pour de petites valeurs de n. Par exemple, l'oracle avec n=2 pour le système `news` en→fr est déjà supérieur de 1,3 point BLEU au score de `moses`. Quel que soit le système étudié, l'oracle semble tendre vers un maximum après n=300, atteignant pour n=1 000 des gains importants par rapport à `moses` de 7,4 et 12,6 points BLEU respectivement pour les systèmes `news` et `médical` en→fr.

On comprend donc qu'une meilleure fonction de score pourrait ainsi préférer certaines de ces meilleures hypothèses. Toutefois, ces résultats mettent en évidence une limite importante de la performance des systèmes de reclassement, pour lesquels les meilleurs scores BLEU atteignables restent encore très éloignés des scores oracle calculés sur l'espace de recherche complet du décodeur (Wisniewski *et al.*, 2010; Wisniewski et Yvon, 2013). Augmenter la valeur de n ne semble pas être une solution efficace pour obtenir rapidement de meilleures hypothèses, le score BLEU de l'oracle ayant une progression en fonction de n de plus en plus faible.

Ainsi, s'ils permettent l'utilisation de modèles complexes difficiles ou impossibles à intégrer lors d'un décodage, les systèmes de reclassement ne proposent qu'une solution sous-optimale pour l'exploitation de tels modèles. Tout d'abord, les n meilleures hypothèses, même pour de grandes valeurs de n, ne représentent qu'une partie infime de l'espace de recherche parcourue par le décodeur, et *a fortiori* de l'espace de recherche complet. Les modèles complexes utilisés n'auront donc qu'un impact fortement limité et ne pourront pas toujours s'exprimer au mieux. De plus, la diversité présente parmi les meilleures hypothèses est très limitée : même pour des valeurs telles que n=1000, il n'est pas rare que ces listes ne contiennent en réalité que quelques dizaines d'hypothèses sensiblement différentes. On notera notamment que, si elles ne sont pas lexicales, les différences observées entre hypothèses peuvent se situer au niveau des alignements entre l'hypothèse

15. Un oracle a pour but de trouver la meilleure solution dans un espace de recherche en connaissant la solution idéale que l'on souhaite atteindre. Ici, parmi les n meilleures hypothèses on cherche donc celle qui sera la plus proche de la traduction de référence selon une métrique d'évaluation automatique particulière.

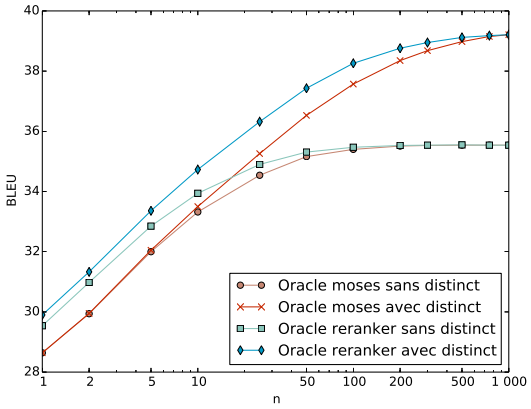
16. *s*BLEU est donc utilisé pour sélectionner la meilleure traduction de chaque phrase. Finalement, un score BLEU standard est calculé sur l'ensemble des traductions sélectionnées pour obtenir le score de l'oracle sur le corpus d'évaluation.

et la phrase traduite, et deux hypothèses constituées d'une même séquence de tokens mais avec des alignements lexicaux différents auront potentiellement un score différent. Les nombreux modèles complexes s'appuyant en partie sur les tokens de la traduction pour être calculés n'auront donc que peu de possibilités pour trouver une meilleure hypothèse dans une liste lexicalement très peu diversifiée.

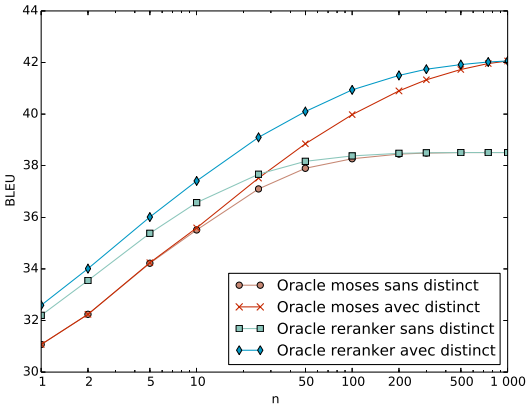
Un tel manque de diversité s'avère également problématique pour l'entraînement du système de reclassement. Le travail décrit par Chatterjee et Cancedda (2010) a consisté à sélectionner aléatoirement dans l'espace de recherche parcouru par le décodeur des hypothèses de traduction qui ont ensuite été ajoutées aux n meilleures hypothèses précédemment produites. Cette diversification peut par la suite permettre une meilleure optimisation des poids de la fonction de score et ce en dépit de la piètre qualité moyenne des hypothèses ajoutées. Gimpel *et al.* (2013); Cer *et al.* (2013) ont eux proposé une approche pour générer des hypothèses différentes d'hypothèses déjà produites auparavant par le décodeur tout en essayant d'obtenir des hypothèses de la meilleure qualité possible. La diversité générée, utilisée pour l'apprentissage d'un système de reclassement, permet là encore d'améliorer légèrement ses performances. Chen *et al.* (2007) ont proposé de compléter la liste des n meilleures hypothèses à l'aide d'une méthode d'expansion de n-grammes (*n-gram expansion*). Celle-ci consiste à générer de nouvelles hypothèses en recombinant les n-grammes des meilleures hypothèses produites par le décodeur. Le reclassement de ces nouvelles hypothèses mélangées aux n meilleures hypothèses produites par le décodeur permet là aussi d'obtenir des améliorations modestes. Cependant cette méthode a pour principal défaut de produire des hypothèses qui ne peuvent pas être évaluées par la fonction de score du décodeur du fait de l'absence d'alignements, normalement produits par le décodeur, entre les tokens source et les tokens cible des n-grammes *fabriqués* dans les nouvelles hypothèses générées. Les hypothèses ainsi fabriquées ne sont donc plus segmentés en bisegments, rendant inutilisables les scores de la table de traduction.

Enfin, les n meilleures hypothèses sont générées par le décodeur en utilisant des modèles dont la nature peut être totalement différente des modèles complexes utilisés en reclassement. En travaillant sur cette liste, le système de reclassement hérite d'un type d'hypothèses, généré avec une fonction de score plus simple, qui peut ne pas être adapté pour faire s'exprimer au mieux les modèles complexes ajoutés.

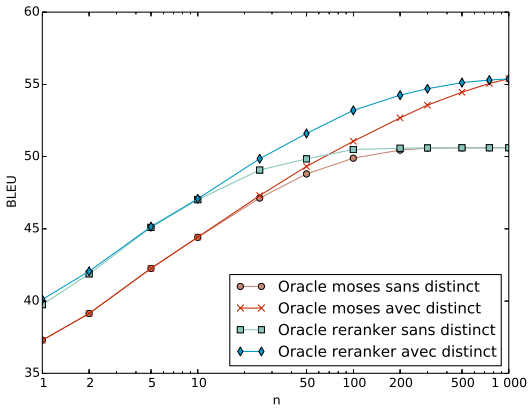
En dépit des limitations que nous venons de décrire et donc, en particulier, de l'exploitation sous-optimale de nouveaux modèles, les systèmes de reclassement d'hypothèses restent très largement utilisés et présentent de bonnes garanties d'amélioration des traductions initialement produites par un décodeur (Och *et al.*, 2004; Zens et Ney, 2006; Ueffing et Ney, 2007; Carter et Monz, 2011; Le *et al.*, 2012; Garmash et Monz, 2015).



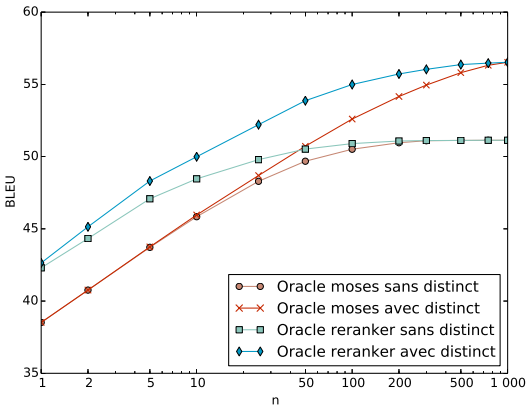
(a) `news` fr→en



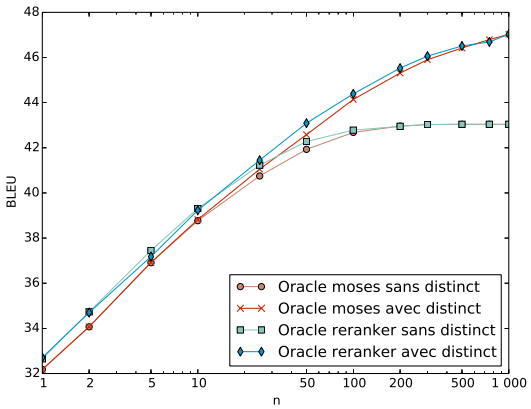
(b) `news` en→fr



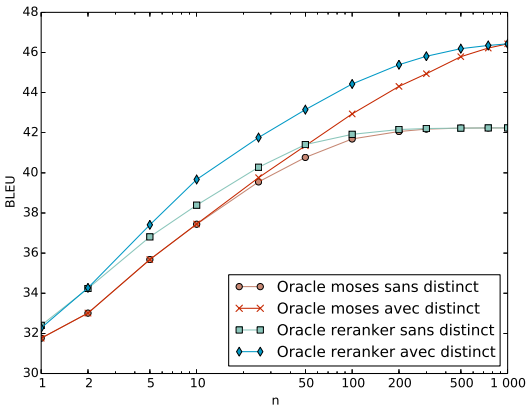
(c) `médical` fr→en



(d) `médical` en→fr



(e) `TED Talks` fr→en



(f) `TED Talks` en→fr

FIGURE 3.2 – Résultats oracle (scores BLEU) calculés sur les n meilleures hypothèses produites par les systèmes `news`, `médical` et `TED Talks` en fonction de n. Le système `reranker` est présenté dans la section 3.3.2 et la production des n-meilleures hypothèses `distinct` est décrite dans la section 3.3.2.2.

3.3.2 Systèmes de traduction de référence avec reclassement d'hypothèses

Cette section présente et analyse les résultats obtenus lors du reclassement des 1 000 meilleures hypothèses produites par les systèmes `news`, `médical` et `TED Talks`. Les nouveaux modèles utilisés sont présentés dans la section 3.3.2.1 et les résultats dans la section 3.3.2.2.

3.3.2.1 Modèles complexes utilisés

Les modèles complexes [17] suivants ont été utilisés :

— **IBM1** : les scores IBM1 calculés pour l'hypothèse complète avec les équations 3.1 et 3.2 (présentées dans la section 3.2.2.1) pour les deux sens de traduction.

— **MosesNorm** : tous les scores des modèles utilisés lors du décodage ainsi que le score global de l'hypothèse normalisés par la longueur de l'hypothèse en nombre de tokens.

— **SOUL** : les modèles `SOUL` sont des modèles neuronaux disposant d'une couche de sortie structurée (*Structured OUtput Layer*) qui ont montré leur capacité à améliorer une traduction par le reclassement d'hypothèses, notablement à l'occasion des campagnes d'évaluation WMT Allauzen *et al.* (2013); Pécheux *et al.* (2014); Marie *et al.* (2015). Le calcul des modèles `SOUL` s'avère trop coûteux pour rendre possible leur intégration sans allonger significativement la durée du décodage. Les modèles de langue Le *et al.* (2011) et de traduction Le *et al.* (2012) utilisés sont des modèles 10-grammes entraînés sur les données fournies lors de la campagne d'évaluation WMT'12. [18]

— **Syntax** : profondeur, nombre de nœuds et nombre de règles unaires de l'arbre syntaxique de l'hypothèse analysée, normalisés par la longueur de l'hypothèse en nombre de tokens (Carter et Monz, 2011).

— **POSLM** : modèle de langue 6-gramme entraîné sur les catégories morpho-syntaxiques des tokens de la partie cible du corpus d'apprentissage du système de traduction. Ce modèle utilise un lissage Witten-Bell [19]. Ce type de modèle est difficile à intégrer au décodage du fait de l'indisponibilité d'un contexte suffisant ou fiable lors de la construction de la traduction par le décodeur pour pouvoir étiqueter avec précision les tokens de l'hypothèse avec leur catégorie morpho-syntaxique.

— **TagRatio** : nombre de tokens de l'hypothèse étiquetés en tant que *nom*, *verbe* ou *adjectif* divisé par le nombre de tokens ayant la même catégorie morpho-syntaxique dans la phrase source.

17. Un modèle peut fournir plusieurs scores. Par exemple le modèle `SOUL` ajoute 5 scores : 1 pour le modèle monolingue et 4 pour le modèle bilingue.

18. Les modèles `SOUL` utilisés ici n'ont pas été spécifiquement adaptés à la tâche. Pour cela, une approche d'adaptation telle que celle proposée par Do *et al.* (2015) pourrait être utilisée afin d'obtenir de meilleures performances.

19. Un lissage Witten-Bell est utilisé ici car bien qu'il soit moins performant que le lissage Kneser-Ney, ce dernier requiert le calcul de statistiques souvent indisponibles pour de longs n-grammes tels que des 6-grammes.

— **WPP** : les tokens des hypothèses sont associés à leur probabilité postérieure. Cette probabilité est calculée avec la méthode *count-based* (voir l'équation 3.3).

— **PPP** : les bisegments sont associés à leur probabilité postérieure (*Phrase Posterior Probability*). Cette probabilité est calculée par l'équation 3.5, où E est l'ensemble des n meilleures hypothèses générées par le décodeur correspondant à la phrase source f, et E_α le sous-ensemble de E tel que les alignements mot-à-mot de la paire de phrases (e', f), $\forall e' \in E_\alpha$ permettent l'extraction du bisegment α. $H(e, f)$ est le score de la paire (e, f) assigné par le décodeur. Une fois la probabilité postérieure de chaque bisegment obtenue, les valeurs des logarithmes des probabilités $P(\alpha|F)$ de chaque bisegment sont additionnées pour calculer un score au niveau de l'hypothèse complète.

$$P(\alpha|F) = \frac{\sum_{e' \in E_\alpha} exp(H(e', f))}{\sum_{e'' \in E} exp(H(e'', f))} \tag{3.5}$$

L'étiquetage morpho-syntaxique pour les modèles **POSLM** et **TagRatio** est effectué avec le `Log-linear Part-Of-Speech Tagger` de l'Université de Stanford [20] (Toutanova et Manning, 2000). L'analyse syntaxique utilisée pour le modèle **Syntax** est effectuée avec l'analyseur syntaxique `Shift-Reduce` de l'université de Stanford [21] (Zhu *et al.*, 2013).

3.3.2.2 Résultats des expériences de reclassement

La Table 3.3 présente le résultat des reclassements des 1 000 meilleures hypothèses produites par `moses` pour les systèmes présentés dans la section 3.1. Ces listes d'hypothèses sont produites en utilisant l'option `distinct` de `moses` qui permet d'éliminer les hypothèses identiques dont seuls les alignements sont différents. [22] À titre de comparaison, des résultats sans cette option activée sont aussi donnés. L'apprentissage du système de reclassement est effectué avec `KB-MIRA` [23] sur les 1 000 meilleures hypothèses produites par `moses` lors de la traduction du corpus de développement utilisé pour l'optimisation du système de traduction.

Le système de reclassement combinant tous les modèles, `reranker`, améliore significativement le score BLEU pour les systèmes `news` et plus encore pour les systèmes `médical`

20. http://nlp.stanford.edu/software/tagger.shtml
21. http://nlp.stanford.edu/software/srparser.shtml
22. Pour générer des hypothèses différentes, `moses` utilise un facteur r et génère d'abord la liste des $n*r$ meilleures hypothèses. Ensuite sont conservées, dans leur ordre de première apparition, les hypothèses de traduction différentes. `moses` utilise par la défaut la valeur $r = 20$, valeur qui a été conservée pour les expériences décrites dans ce manuscrit. Lorsque les 1 000 meilleures hypothèses distinctes sont produites, `moses` construit donc les 20 000 meilleures hypothèses qui seront donc dédoublonnées. En pratique, il est donc courant d'avoir une liste contenant moins de 1 000 hypothèses pour chaque phrase avec l'option `distinct`, la liste des 20 000 meilleures hypothèses contenant en général un grand nombre de doublons. Cependant, puisque `moses` produit et évalue ces 20 000 hypothèses, la production de 1 000 meilleures hypothèses `distinct` est significativement plus coûteuse que la production des 1 000 meilleures non-`distinct`.
23. L'algorithme d'optimisation `PRO` a également été testé, mais les résultats obtenus avec cet algorithme ne sont pas donnés dans ce manuscrit, ceux-ci étant généralement inférieurs (de l'ordre de 0,1 à 0,2 point BLEU) à ceux obtenus avec `KB-MIRA`.

avec des gains de 2,9 et 4,1 points BLEU respectivement pour les directions fr→en et en→fr. `reranker` améliore également le score BLEU des systèmes `TED Talks` mais dans une moindre mesure.

L'importance des gains de `reranker` pour les systèmes `médical` par rapport aux autres systèmes peut s'expliquer en analysant les résultats des oracle calculés sur les n meilleures hypothèses produites par `moses`. Ces résultats, contenus dans la Figure 3.2, révèlent en effet un potentiel d'amélioration en reclassement d'hypothèses bien plus net sur cette tâche. De plus, après reclassement des hypothèses, les scores oracle recalculés sont meilleurs pour toutes les valeurs de n considérées et sur toutes les tâches, montrant la capacité de la nouvelle fonction de score à produire un meilleur classement d'hypothèses que la fonction de score du décodeur. Avec une liste d'hypothèses moins diversifiée générée sans activer l'option `distinct`, les résultats obtenus par `reranker` sont légèrement inférieurs sur les tâches `news` et `médical`. L'augmentation de la diversité semble donc bénéfique, comme cela avait déjà montré dans les travaux de Chatterjee et Cancedda (2010); Gimpel *et al.* (2013) [24]. Ce résultat peut être expliqué en comparant les scores oracle calculés sur les listes des n meilleures hypothèses produites présentés par la Figure 3.2. Les scores oracle calculés sur les 1 000 meilleures hypothèses obtenues avec l'option `distinct` obtiennent un score BLEU bien plus élevé que ceux calculés sur les 1 000 meilleures hypothèses produites sans l'option `distinct`, avec par exemple un écart de 5,4 points BLEU sur la tâche `médical` en→fr.

Pour tous les systèmes, les modèles **SOUL** sont ceux qui contribuent le plus à la bonne performance de `reranker`. La performance de `reranker` est même quasiment égale à celle de la configuration utilisant uniquement **SOUL** parmi les modèles complexes utilisés pour les systèmes `news` et `TED Talks`, le modèle **POSLM** contribuant également à l'amélioration des performances sur les systèmes `médical`, et les autres modèles n'impactant eux que très légèrement la performance de `reranker`. Il est cependant intéressant de constater que nous n'observons plus les gains qui avaient été auparavant observés par Ueffing et Ney (2007); Zens et Ney (2006) en reclassement d'hypothèses avec le modèle **WPP** ou encore ceux qu'avaient observés Och *et al.* (2004) avec les modèles **IBM1**. Malgré l'absence d'améliorations significatives avec ces modèles, ceux-ci ont été conservés pour les expériences décrites tout au long de ce manuscrit afin d'illustrer la capacité des différents algorithmes et systèmes décrits à prendre en compte et bénéficier de l'utilisation d'un grand nombre de modèles complexes de différentes natures.

24. Il faut toutefois noter que, dans leurs travaux, Chatterjee et Cancedda (2010); Gimpel *et al.* (2013) ont effectué leurs expériences avec `moses` sans activer l'option `distinct`. Les approches qu'ils ont développées sont donc comparées à des systèmes utilisant des listes d'hypothèses non-dédoublonnées et donc moins riches en diversité.

Modèles	news		médical		TED Talks	
	fr→en	en→fr	fr→en	en→fr	fr→en	en→fr
`moses` (*réf.*)	28,6	31,1	37,3	38,5	32,2	31,8
+ **IBM1**	28,8	30,9	37,4	38,8	32,1	31,8
+ **MosesNorm**	28,7	31,1	37,4	38,6	32,2	31,8
+ **POSLM**	28,9	31,1	37,9	38,9	32,3	31,8
+ **PPP**	28,6	31,2	37,3	38,5	32,2	31,8
+ **SOUL**	29,8	32,5	39,1	41,7	32,7	32,3
+ **Syntax**	28,9	31,2	37,5	38,5	32,2	31,8
+ **TagRatio**	28,8	31,2	37,6	38,6	32,2	31,8
+ **WPP**	28,6	31,2	37,4	38,8	32,3	31,9
`reranker` sans `distinct`	$29,5_{(+0,9)}$	$32,2_{(+1,1)}$	$40,1_{(+2,8)}$	$42,3_{(+3,8)}$	$32,7_{(+0,5)}$	$\mathbf{32,4}_{(+0,6)}$
`reranker`	$\mathbf{29,9}_{(+1,3)}$	$\mathbf{32,5}_{(+1,4)}$	$\mathbf{40,2}_{(+2,9)}$	$\mathbf{42,6}_{(+4,1)}$	$\mathbf{32,7}_{(+0,5)}$	$32,3_{(+0,5)}$

TABLE 3.3 – Résultats (scores BLEU) du reclassement des 1 000 meilleures hypothèses, produites avec l'option `distinct`, en utilisant des modèles complexes difficiles à intégrer au décodage. Chaque ligne ajoute de façon incrémentale un modèle à la fonction de score originale de `moses`. `reranker` donne la performance du reclassement obtenu en utilisant tous les modèles. « sans `distinct` » donne le résultant de `reranker` entraîné et évalué sur une liste des 1 000 meilleures hypothèses générées sans l'option `distinct` activée. Les écarts inscrits entre parenthèses sont donnés par rapport au système de référence `moses`, dénoté (*réf.*), n'utilisant aucun modèle complexe.

3.4 Regénérer une traduction après un premier décodage

Pour contourner les limitations d'un système de reclassement, d'autres approches ont été proposées afin de mieux exploiter certains de types modèles en TAS. Ces approches réutilisent le décodeur pour accéder à la totalité de son espace de recherche et le reparcourir à l'aide d'une nouvelle fonction de score contenant de nouveaux modèles. Ces approches de *regénération d'hypothèses* (*hypothesis regeneration*) (Chen *et al.*, 2008b,a, 2010) ne s'appliquent toutefois que pour de nouveaux modèles fondés sur le calcul de probabilités *a posteriori* peu coûteux à calculer et calculables pour des hypothèses partielles.

Prenons l'exemple d'un modèle de langue entraîné sur la concaténation des n meilleures hypothèses produites par un décodeur (Chen *et al.*, 2008a). Un tel modèle peut ensuite être facilement combiné linéairement aux autres modèles de la fonction de score du décodeur et être utilisé au cours d'un nouveau décodage. L'exploitation de ce type d'informations ayant par ailleurs mené à des améliorations en reclassement d'hypothèses (Ueffing et Ney, 2007; Zens et Ney, 2006), une exploitation au cours d'un nouveau décodage peut permettre d'obtenir des améliorations plus importantes.

Ce processus de regénération peut se dérouler sur plusieurs itérations, dont le nombre est fixé à l'avance. À l'issue de chaque nouveau décodage, les nouveaux modèles basés sur

des probabilités calculées *a posteriori* sont mis à jour. Le décodeur est donc potentiellement guidé par ces nouveaux modèles vers la sous-partie de l'espace de recherche qui est la plus probable *a posteriori*. Des améliorations des traductions sont possibles par cette approche, mais celles-ci restent néanmoins très limitées par le type de modèles qu'il est possible d'exploiter. Dans leurs travaux, Chen *et al.* (2008b) vont jusqu'à reconstruire de nouvelles tables de traduction et de réordonnancement lexicalisées en réapprenant des alignements à l'aide de `GIZA++` sur les listes des n meilleures hypothèses où chaque hypothèse de traduction est alignée avec sa phrase source correspondante. La nouvelle table de traduction ainsi créée est utilisée au cours d'un nouveau décodage. Une nouvelle liste de m meilleures hypothèses est créée et concaténée à la liste des n meilleures hypothèses initialement produites par le décodeur (`moses`). Un reclassement d'hypothèses est ensuite effectuée avec de nouveaux modèles sur cette liste enrichie. Nous avons reproduit ces expériences pour nos trois tâches `news`, `médical` et `TED Talks`, en respectant le protocole suivant :

1. produire les n meilleures hypothèses de traduction avec `moses`
2. lancer `MGIZA++` pour réapprendre des alignements entre la phrase source et les n meilleures hypothèses
3. construire une nouvelle table de traduction sur ces nouveaux alignements
4. remplacer la table de traduction du décodeur par cette nouvelle table
5. réoptimiser les poids des modèles de la fonction de score du décodeur avec `KB-MIRA` sur le même corpus de développement que celui utilisé pour optimiser la fonction de score initiale de `moses`
6. redécoder le texte source avec la nouvelle fonction de score pour produire les m meilleures hypothèses
7. fusionner les listes des $n + m$ meilleures hypothèses en éliminant les doublons
8. apprendre un système de reclassement d'hypothèses avec `KB-MIRA` et les modèles présentés par la section 3.3.2.1 sur les $n + m$ meilleures hypothèses produites sur le corpus de développement
9. reclasser les $n + m$ meilleures hypothèses produites sur le corpus de test

Pour ces expériences, les valeurs de n et m utilisées sont identiques à celles utilisées par Chen *et al.* (2008b) : $n = 800$ et $m = 400$. Il faut également noter que les listes d'hypothèses produites sont dédoublonnées dans ces expériences : l'option `distinct` a été activée conformément aux expériences de Chen *et al.* (2008b).

Les résultats obtenus après reclassement du mélange des $n + m$ meilleures hypothèses sont donnés par la Table 3.4. Nous pouvons constater une légère amélioration des scores BLEU, du même ordre que celle rapportée par Chen *et al.* (2008b), relativement à un reclassement réalisé sur les n meilleures hypothèses. Des améliorations peuvent donc être obtenues avec une telle approche, en revanche celles-ci s'avère être assez modestes, ce qui peut peut-être être notamment attribué au fait qu'un seul redécodage est effectué pour produire m nouvelles hypothèses.

Il existe de nombreuses autres approches permettant de regénérer une hypothèse en utilisant des informations indisponibles lors du décodage. Par exemple, la combinaison de

Système	news		médical		TED Talks	
	fr→en	en→fr	fr→en	en→fr	fr→en	en→fr
`moses`	28,6	31,1	37,3	38,5	32,2	31,8
`reranker` (*réf.*)	29,9	32,5	**40,2**	42,6	32,7	32,3
`reranker` 800 + 400	**30,0**$_{(+0,1)}$	**32,7**$_{(+0,2)}$	**40,2**$_{(+0,0)}$	**42,8**$_{(+0,2)}$	**32,8**$_{(+0,1)}$	**32,6**$_{(+0,3)}$

TABLE 3.4 – Résultats (scores BLEU) du reclassement des 800 meilleures hypothèses produites par `moses` mélangées aux 400 meilleures hypothèses produites après un redécodage. Le système de reclassement utilise les mêmes modèles que ceux utilisés par `reranker` (voir section 3.3.2.1). Les écarts inscrits entre parenthèses sont donnés relativement à la configuration de `reranker`, dénotée (*réf.*).

systèmes (Bangalore, 2001; Rosti *et al.*, 2008) permet de combiner plusieurs hypothèses produites par différents systèmes de traduction ; l'utilisation d'hypothèses obtenues par la traduction par pivot *via* d'autres langues Crego *et al.* (2010) permet d'informer un décodeur par les hypothèses générées par d'autres systèmes sur des tâches de traduction de complexité et de performance différentes. Devlin et Matsoukas (2012) ont eux proposé une approche consistant à effectuer de nombreuses petites modifications de la fonction de score du décodeur, afin de biaiser le parcours de l'espace de recherche vers des hypothèses de différentes natures[25]. Chaque modification de la fonction de score est donc associée à un nouveau décodage qui produira une nouvelle liste de n meilleures hypothèses. Les différentes hypothèses pour l'ensemble des listes produites sont finalement combinées pour extraire une hypothèse de meilleure qualité. Une autre approche de nature très différente a été décrite par Mirkin *et al.* (2013) : elle consiste à exploiter des paraphrases de la phrase source pour générer et proposer des hypothèses de traduction alternative à un humain. Toutefois, ces différentes approches ne permettent pas aisément l'ajout explicite de nouveaux modèles. Ceci est permis en revanche par l'approche de *recherche locale* (*local/greedy search*) qui parcourt le voisinage d'une meilleure hypothèse guidé par une fonction de score qui peut exploiter de nouveaux modèles (Langlais *et al.*, 2007). Nous décrivons plus précisément cette approche dans la section suivante.

3.5 Au-delà du décodage : la recherche locale

Nous avons vu dans les sections précédentes différentes approches permettant l'amélioration d'une hypothèse produite par le décodeur. La recherche locale, en particulier, consiste à améliorer itérativement une solution par génération d'hypothèses de traduction dans le voisinage immédiat d'une hypothèse *amorce*. La recherche locale appliquée à la TAS fut initialement utilisée pour le décodage dans le système `ReWrite` (Germann *et al.*, 2001b; Germann, 2003b), qui reposait sur un modèle de traduction fondé sur les mots[26]. La recherche locale durant le décodage n'a cependant pas été beaucoup plus étudiée car

25. Ces hypothèses peuvent par exemple avoir une taille légèrement plus petite ou plus grande que les hypothèses qu'aurait produites le décodeur naturellement.

26. Les modèles fondés sur les segments ne s'étant réellement répandus que quelques années plus tard.

les algorithmes à base de programmation dynamique ont rapidement montré une plus grande efficacité (Foster *et al.*, 2003).

Une autre manière d'utiliser la recherche locale consiste à la mettre en œuvre sur une traduction complète déjà produite. Cette recherche est guidée par une fonction de score qui, si elle est identique à celle du décodeur initialement utilisée, peut permettre de corriger des erreurs de recherche commises par ce dernier pour des raisons décrites dans la section 3.2.1. Un algorithme de recherche locale (voir la description dans l'Algorithme 2) prend en entrée une hypothèse amorce, ici la meilleure hypothèse produite par un système, puis parcourt son voisinage en lui appliquant différentes opérations de *réécriture*. Parmi l'ensemble des opérations possibles, l'algorithme *glouton* applique celle qui maximise à chaque itération la fonction de score utilisée (voir l'équation 2.9), et s'arrête dès qu'aucune amélioration n'est obtenue.

Le travail de Langlais *et al.* (2007) propose 6 opérations qui ont pour objectif de visiter des hypothèses proches de l'amorce qui sont susceptibles de contenir des améliorations ou corrections de celle-ci. Les opérations `move` (rapprochement de deux segments cible traduisant deux segments adjacents dans la phrase source) et `swap` (inversion de deux segments) déplacent des segments dans une hypothèse. Elles opèrent donc une réorganisation des segments d'une hypothèse dans le but d'améliorer son score. Les autres opérations (`split`, `merge`, `replace` et `bireplace`) permettent quant à elles de remplacer certains segments de l'hypothèse par d'autres bisegments présents dans la table de traduction. La complexité de ces opérations ainsi qu'une brève description sont données dans la Table 3.6. Toutes ces opérations respectent la même limite de distorsion que celle utilisée par le décodeur initial.

Dans leur travail, Langlais *et al.* (2007) avaient observé une légère amélioration des traductions obtenues, mettant ainsi en évidence des erreurs de recherche effectuées par le décodeur. Ce travail utilisait `Pharaoh` (Koehn, 2004), décodeur à l'état de l'art de l'époque, pour produire les amorces servant de point de départ à la recherche locale. La progression de la puissance de calcul des ordinateurs permettant aujourd'hui d'évaluer un plus grand nombre d'hypothèses au cours du décodage[27], et les différentes améliorations apportées aux algorithmes de parcours de l'espace de recherche font que les erreurs de recherche commises par un décodeur plus avancé tel que `moses` sont quasiment inexistantes (Arun *et al.*, 2009; Aziz *et al.*, 2014).

27. En augmentant en particulier la taille du faisceau utilisé au cours de la recherche.

Algorithme 2 Algorithme de recherche locale

ENTRÉE : *source* une phrase à traduire.

```
courant ← AMORCE(source)
boucler
  sCourant ← SCORE(source)
  s ← sCourant
  pour tout h ∈ VOISINAGE(courant) faire
    c ← SCORE(h)
    si c > s alors
      s ← c
      meilleur ← h
    fin si
    si s = sCourant alors
      retourner courant
    sinon
      courant ← meilleur
    fin si
  fin pour
fin boucle
```

Système	news		médical		TED Talks	
	fr→en	en→fr	fr→en	en→fr	fr→en	en→fr
moses	28,6	31,1	37,3	38,5	32,2	31,8
recherche locale	28,6	**31,1**$_{(+0,0)}$	**37,4**$_{(+0,1)}$	38,5	32,2	31,8

TABLE 3.5 – Résultats (BLEU) d'une recherche locale appliquée à une traduction générée par moses.

La Table 3.5 donne les résultats d'une recherche locale effectuée à partir des traductions générées par moses pour les systèmes news, médical et TED Talks. Afin de limiter la taille de l'espace de recherche, à l'instar du travail de Langlais *et al.* (2007), seules les 10 meilleures traductions de chaque segment source ont été considérées durant la recherche. On observe qu'aucune amélioration des traductions initiales ne sont obtenues. De plus, très peu d'opérations sont effectuées (moins d'une dizaine pour chaque système, voire aucune pour les systèmes TED Talks).

Opération	Complexité	Description
`move`	O(N)	rapproche deux segments cible distants si les deux segments sources correspondant sont adjacents
`swap`	O(N)	inverse deux segments cible adjacents
`replace`	O(N x T)	change la traduction d'un segment source par une autre traduction présent dans la table de traduction
`merge`	O(T x N)	fusionne deux segments sources et remplace les deux segments cibles corresopndants par une traduction du nouveau segment source créé
`split`	O(N x S x T^2)	scinde un segment source en deux parties et remplace la segment cible correspondant par des traductions de deux nouveaux segments créés
`bireplace`	O(T^2 x N)	change la traduction de deux segments adjacents simultanément

TABLE 3.6 – Opérations utilisées par la recherche locale de Langlais *et al.* (2007). **N** : nombre de segments dans la phrase source ; **T** : nombre maximum de traductions par segment ; **S** : nombre moyen de tokens par segment.

Aujourd'hui la recherche locale effectuée avec une fonction de score identique à celle du décodeur n'a donc plus grand intérêt, ce dernier commettant trop peu d'erreurs de recherche. Toutefois, il faut noter qu'une telle approche peut être utilisée pour exploiter des modèles complexes n'ayant pas pu être utilisés au cours du décodage initial (voir section 4.2), en combinant par exemple de façon linéaire ces modèles avec ceux de la fonction de score d'origine. Par exemple, Langlais *et al.* (2007) ont ajouté à la fonction de score du décodeur un modèle de langue *inversé* qui prédit les tokens de droite à gauche. Un tel modèle ne peut pas être utilisé simplement au cours du décodage étant donné que le suffixe de l'hypothèse partielle en cours de construction est inconnu.

3.6 La post-édition pour améliorer une traduction

3.6.1 La post-édition par un humain

Bien que la qualité des traductions produites par les systèmes de traduction automatique ait fortement progressé au cours de la dernière décennie (Graham *et al.*, 2014), celles-ci sont encore aujourd'hui difficilement diffusables en l'état. L'intervention d'un humain, idéalement un traducteur professionnel formé pour la tâche visée, reste nécessaire pour corriger et améliorer la traduction produite. Cette *post-édition* (*post-editing*) permet donc d'obtenir une traduction potentiellement de haute qualité tout en étant glo-

balement moins coûteuse que ne l'aurait été une traduction uniquement manuelle (Green *et al.*, 2013a). La post-édition peut être facilitée par l'interface utilisée, par exemple par l'ajout d'informations telles que des estimations de la confiance dans le fait que chaque mot d'une hypothèse de traduction soit correct (Bach *et al.*, 2011).

Il est important de noter qu'une traduction post-éditée peut aussi être utilisée pour réentraîner le système de traduction de façon incrémentale pour prévenir la réapparition d'erreurs déjà corrigées auparavant par le post-éditeur (Denkowski *et al.*, 2014), et plus généralement enrichir la couverture et améliorer la spécialisation d'un système Gong *et al.* (2014).

Afin d'améliorer l'efficacité du traducteur humain, des approches de *traduction interactive* (*Interactive Machine Translation* (IMT)) peuvent également être mises en œuvre (par exemple, (Langlais *et al.*, 2000; Foster *et al.*, 2002; Koehn et Haddow, 2009)). Il s'agit, par exemple, de proposer des continuations à la traduction en cours d'écriture par le traducteur que celui-ci peut ignorer ou accepter. Green *et al.* (2014b) ont montré, dans un travail récent, qu'une approche de traduction interactive efficace pouvait toutefois se révéler plus coûteuse en temps qu'un recours à une post-édition classique, bien que les traductions produites soient de meilleure qualité.

Le recours à un intervenant humain qualifié rend quoi qu'il en soit la post-édition coûteuse et difficile à mettre en place. De plus, la post-édition ne peut pas être envisagée lorsque les hypothèses de traduction initiales sont de trop mauvaise qualité (Specia et Farzindar, 2010). La post-édition automatique, décrite dans la section suivante, correspond à une tentative visant à réduire le coût de la post-édition en essayant d'améliorer en amont, en tenant compte des limites d'un système particulier, la qualité de la traduction.

3.6.2 La post-édition automatique

La post-édition automatique (*Automatic Post-Editing* (APE)) (Knight et Chander, 1994) a pour objectif de corriger automatiquement les erreurs systématiques d'un système de traduction automatique. On peut en effet espérer des améliorations car des erreurs commises par un système de traduction dont le fonctionnement peut nous être inconnu peuvent être corrigées en exploitant les corrections effectuées sur des traces de post-édition humaine. Parton *et al.* (2012) ont par exemple montré que la post-édition automatique fondée sur des règles permet l'amélioration de la *précision* (*accuracy*) des traductions.

Les approches statistiques pour la post-édition automatique ont bénéficié de plus d'attention ces dernières années (Chatterjee *et al.*, 2015) comparées aux approches à base de règles (Rosa *et al.*, 2012) considérées comme plus coûteuses à mettre en place et difficiles à développer pour plusieurs langues ou domaines. Celles-ci consistent généralement à apprendre un système de TAS en utilisant comme données parallèles les traductions pro-

duites en tant que langue source et la post-édition réalisée par des traducteurs humains en tant que langue cible. Un tel système cherche donc à « traduire » une traduction produite automatiquement en une traduction post-éditée. Cette approche possède l'inconvénient de nécessiter une grande quantité de traductions post-éditées pour apprendre des alignements mot à mot et une table de traduction de bonne qualité. En pratique, la post-édition automatique reste une tâche difficile qui peut même aller jusqu'à dégrader la qualité des hypothèses de traduction initiales, en particulier si les données utilisées pour son apprentissage sont mal sélectionnées ou mal filtrées (Bojar *et al.*, 2015).

Résumé

Dans ce chapitre, nous avons mis en évidence les limites du parcours de l'espace de recherche effectué par le décodeur d'un système de TAS à l'état de l'art. En effet, tel qu'il est effectué, ce parcours rend difficile l'exploitation de certains types de modèles complexes, en particulier : ceux qui ne peuvent se calculer que sur une hypothèse de traduction complète ; ceux qui sont trop coûteux à calculer lors d'un décodage ; ou encore ceux qui ne peuvent être estimés qu'à l'issue d'un premier décodage. Un état de l'art des différentes approches visant l'amélioration automatique d'une traduction *a posteriori* par l'exploitation de ce type de modèles a été présenté. Nous avons notamment vu que ces modèles complexes sont principalement utilisés en TAS par l'intermédiaire de systèmes reclassant les n meilleures hypothèses de traduction produites par un décodeur. Cependant, si ce type de système s'avère efficace pour améliorer une traduction, l'exploitation de modèles complexes par un reclassement d'hypothèses semble encore loin d'être optimale. En effet, les hypothèses de traduction contenues dans les listes reclassées sont trop peu différentes et ne représentent qu'une partie relativement restreinte de l'ensemble des traductions possibles. Différentes approches ont été proposées afin d'introduire de la diversité en vue d'augmenter les performances des systèmes de reclassement, mais les améliorations apportées par ces approches, réutilisant pour la plupart le décodeur, restent relativement faibles et ne sont pas particulièrement orientées par l'objectif de production d'une diversité optimale pour une fonction de score enrichie en modèles complexes. D'autres approches génèrent elles des hypothèses de traduction par redécodage en utilisant des modèles appris après d'un premier décodage, et nous avons pu constater que les améliorations apportées étaient là aussi relativement modestes, alors que la variété des types de modèles qu'elles permettent d'exploiter demeure très limitée. Finalement, nous avons également décrit une approche de réécriture fondée sur un algorithme de recherche locale. Une telle approche a la capacité de corriger les erreurs de recherche du décodeur en réutilisant sa fonction de score et a donc le potentiel d'améliorer la traduction produite. Toutefois, les erreurs commises par le décodeur sont aujourd'hui trop peu nombreuses pour observer un effet de cette réécriture sur la qualité des traductions obtenues. La recherche locale reste cependant intéressante puisqu'elle donne la possibilité de reparcourir une partie de l'espace de recherche du décodeur avec de nouveaux modèles venant enrichir la fonction de score du décodeur.

Dans les expériences des chapitres qui suivront, nous prendrons comme systèmes de référence un décodage suivi d'une passe de reclassement d'hypothèses effectuée avec la même liste de modèles complexes. L'approche de reclassement d'hypothèses a été préférée aux autres approches pour construire les systèmes de référence du fait des fortes améliorations de la qualité des traductions observées avec les modèles complexes que nous avons utilisés. De plus, comme nous l'avons vu, cette approche est très utilisée en TAS, à l'inverse par exemple des approches effectuant de la regénération d'hypothèses, qui sont en outre plus coûteuses à mettre en œuvre. Le fonctionnement des systèmes de traduction de référence pour le reste de notre travail est résumé par la Figure 3.3.

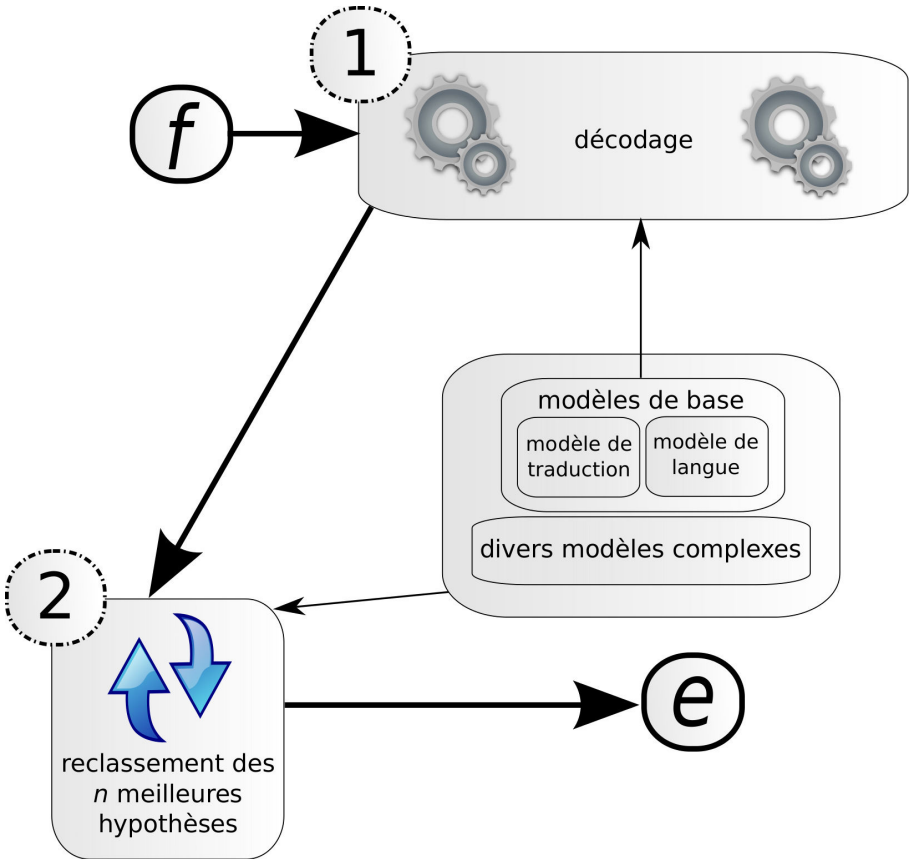


FIGURE 3.3 – Résumé du fonctionnement de système de TAS impliquant une passe de reclassement d'hypothèses. Une phrase source f est décodée en utilisant une fonction de score impliquant uniquement les modèles de base du décodeur (1). Le décodeur produit une liste de n meilleures hypothèses qui est reclassée à l'aide d'une nouvelle fonction de score exploitant un nouvel ensemble de modèles, qui sont les mêmes que ceux utilisés par le décodeur combinés à de nouveaux modèles complexes n'ayant pu être utilisés au cours du décodage (2). Après reclassement, la nouvelle meilleure hypothèse est utilisée comme e de la phrase f.

4

Une recherche locale enrichie avec des modèles complexes

Sommaire

4.1 Évaluation du potentiel d'amélioration des traductions produites . . . **55**
4.1.1 La recherche locale oracle . . . 55
4.1.2 Systèmes de référence `europarl-intersect` et `BTEC` . . . 56
4.1.3 Expériences et analyse . . . 60

4.2 Réécriture automatique de traductions à l'aide de modèles complexes **70**
4.2.1 Une réécriture guidée par des modèles inutilisables lors du décodage . . . 70
4.2.2 Expériences . . . 72
4.2.3 Analyse . . . 74

Résumé . . . **83**

Un état de l'art des différentes solutions permettant une amélioration *a posteriori* d'une traduction automatique produite par un système de TAS a été présenté dans le chapitre 3. Certaines de ces approches ont recours à des modèles plus complexes que ceux utilisés par la fonction de score du décodeur, qui sont difficiles à intégrer au décodage pour les raisons que nous avons évoquées dans la section 3.2.2. Aujourd'hui, de tels modèles sont principalement utilisés en TAS dans une seconde étape suivant le décodage dans une étape de reclassement des meilleures hypothèses produites par le décodeur. Cependant, nous avons également montré que cette solution est loin d'être optimale, notamment parce qu'elle se limite à la réévaluation d'une liste d'hypothèses de traduction qui ne représente qu'une très petite partie de l'espace des hypothèses. Aucune réexploration de l'espace de recherche n'est effectuée, limitant ainsi par nature les améliorations atteignables à l'aide d'une fonction de score mieux informée.

Nous avons également vu dans la section 3.5 une approche de recherche locale opérant des réécritures qui modifient les hypothèses de traduction d'un système par l'application d'un ensemble d'opérations. À chaque itération, l'opération de réécriture effectivement appliquée sur la traduction est celle qui maximise le score donné par la fonction de score du décodeur. La recherche continue aussi longtemps que des réécritures permettent d'améliorer le score de la traduction. Nos expériences n'avaient toutefois pas permis d'observer d'améliorations avec une telle approche sur les traductions produites par nos systèmes de référence `news`, `médical` et `TED Talks`.

Le nombre d'erreurs de recherche étant négligeable, il apparaît donc que la fonction de score du décodeur n'est pas suffisamment bien informée pour identifier une meilleure traduction dans le voisinage de celle produite par le décodeur. En revanche, une recherche locale guidée par une fonction de score enrichie par des modèles complexes pourrait elle permettre d'identifier de meilleures traductions, tout en permettant d'élargir la diversité d'hypothèses de traductions considérées, les voisinages générés par la recherche locale n'étant pas nécessairement inclus dans la liste des n meilleures hypothèses produites par le décodeur. Néanmoins, le nombre d'hypothèses de réécriture contenues dans un tel voisinage étant particulièrement grand, une recherche locale guidée par des modèles complexes, parfois très coûteux à calculer, ne pourrait être envisageable qu'après une forte réduction de la taille des voisinages générés.

Dans ce chapitre, nous allons tout d'abord évaluer le potentiel d'amélioration d'une approche de réécriture par l'intermédiaire d'expériences *oracle* de recherche locale dans la section 4.1.1. De nouveaux systèmes de traduction de référence, construits spécialement pour ces expériences, seront introduits par la section 4.1.2, et les résultats et l'analyse de ces expériences oracle seront décrits dans la section 4.1.3. Dans la section 4.2.1, nous sortirons du cadre oracle pour passer dans un cadre automatique en introduisant un système de réécriture exploitant une fonction de score mettant en jeu des modèles complexes non utilisés au cours du premier décodage. Les expériences et l'analyse des performances de ce système de réécriture seront présentées par les sections 4.2.2 et 4.2.3.

4.1 Évaluation du potentiel d'amélioration des traductions produites

Dans cette section, nous décrivons une étude visant à mettre en évidence le potentiel d'amélioration d'une traduction produite par le décodeur qui peut être atteinte à l'aide d'un algorithme de recherche locale lorsque celle-ci est guidée par une connaissance parfaite des traductions de référence à produire.

4.1.1 La recherche locale oracle

4.1.1.1 Une métrique d'évaluation automatique comme fonction de score

En disposant d'une meilleure fonction de score que la fonction de score du décodeur, il est attendu qu'une recherche locale permette la réécriture de la traduction produite par le décodeur en une traduction de meilleure qualité. On souhaite ici mesurer le potentiel d'une telle approche de réécriture : nous allons donc utiliser comme fonction de score une métrique d'évaluation de la traduction ayant accès à la traduction de référence à produire. On peut donc qualifier cette recherche locale d'*oracle*. Celle-ci cherchera à produire des hypothèses proches de la traduction de référence par le jeu d'applications successives d'opérations de réécriture qui maximisent le score *s*BLEU de la traduction. Une forte augmentation du score *s*BLEU de la traduction après réécriture sera donc indicatrice d'un fort potentiel d'amélioration.

À l'aide d'un tel oracle, nous cherchons donc à confirmer le fait qu'un algorithme de recherche locale peut effectivement améliorer une traduction. Il s'agit d'un oracle car le système a accès à la traduction de référence. En revanche, nous n'avons pas la garantie d'atteindre la meilleure traduction présente dans l'espace de recherche du décodeur. En effet, l'algorithme glouton du système peut se heurter à un maximum local de la fonction de score et arrêter par conséquent la séquence de réécritures sans avoir trouvé la traduction atteignable de meilleur score. Il est donc attendu que la traduction produite par une recherche locale oracle obtiendra un score BLEU moins élevé que celui d'une traduction produite par un oracle exact (Wisniewski et Yvon, 2013) mais difficile à atteindre, voire inatteignable, *via* un algorithme de recherche locale.

4.1.1.2 Opérations de réécriture

Pour effectuer notre recherche locale oracle (`GOS`, *Greedy Oracle Search*), un ensemble d'opérations différent de celui du travail de Langlais *et al.* (2007) a été utilisé. Ces opérations sont présentées dans la Table 4.1 et ne sont pas restreintes par la limite de distorsion qui a été utilisée durant le décodage (voir la section 2.3.2.4). La complexité des opérations est exprimée en fonction du nombre maximum d'hypothèses qu'il est possible de produire étant donnée l'hypothèse amorce à réécrire. Certaines des opérations introduites ici ont une complexité bien supérieure à celle mise en œuvre dans Langlais *et al.* (2007), le

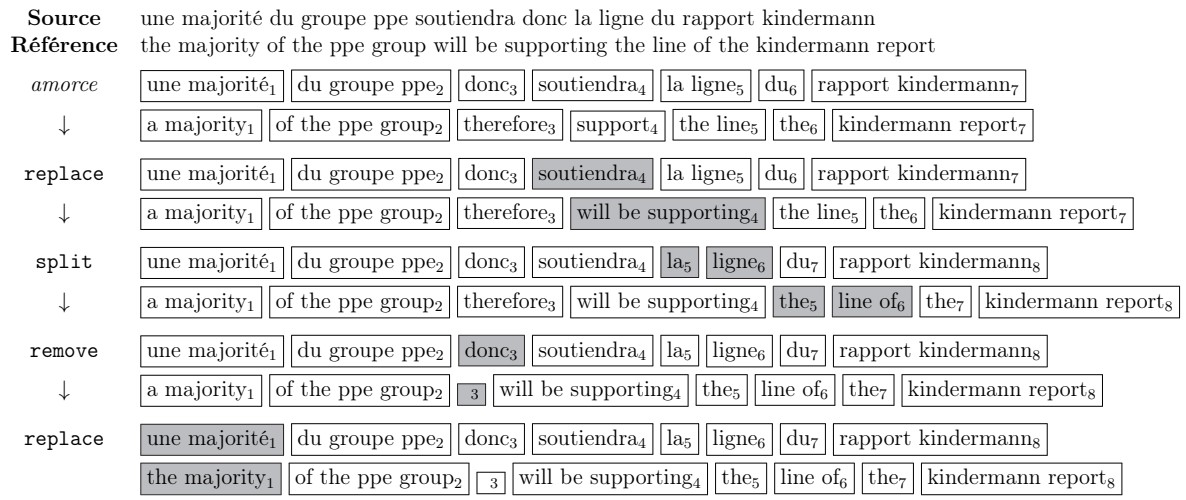


FIGURE 4.1 – Séquence de 4 opérations (`replace`, `split`, `remove` et `replace`) effectuées lors d'une recherche locale oracle guidée par *s*BLEU atteignant la traduction de référence visée. Les indices précisés dans les boîtes contenant les segments indiquent la position du segment source correspondant de la phrase source.

nombre d'hypothèses plus important qu'elles génèrent augmentant ainsi mécaniquement la probabilité de trouver une meilleure traduction dans le voisinage de l'hypothèse. Les voisinages générés par les opérations décrites par Langlais *et al.* (2007) sont inclus dans ceux générés par les opérations utilisées ici. La recherche locale oracle choisira toujours la meilleure traduction présente dans ce voisinage au sens de la métrique utilisée.

La Figure 4.1 illustre une trace de recherche locale oracle illustrant l'effet de certaines opérations de réécriture. Dans cet exemple, `GOS` permet d'atteindre, en 4 opérations, la traduction de référence à partir d'une traduction automatique produite par le décodeur `moses`. La Figure 4.2 présente elle des exemples complémentaires d'opérations de réécriture impliquant les opérations `merge`, `move` et `paraphrase`.

4.1.2 Systèmes de référence `europarl-intersect` et `BTEC`

Dans le but d'obtenir des résultats pour un grand nombre de paires de langues et dans des conditions comparables en termes de quantités de données et de domaines, nous avons utilisé la sous-partie du corpus Europarl de débats parlementaires [1] qui est disponible en 11 langues. [2] Nous avons ensuite extrait des corpus d'entraînement, de développement et de test sur les mêmes données pour développer des systèmes de traduction ; les données et les systèmes `europarl-intersect` correspondant sont décrits dans la Table 4.2. Nous avons développé 10 systèmes avec le français (fr) en langue source, et en langues cibles le danois (da), l'allemand (de), le grec (el), l'anglais (en), l'espagnol (es), le finnois (fi),

1. http://statmt.org/europarl
2. Cette extraction de sous-corpus a été réalisée en utilisant l'anglais comme langue pivot.

Opération	Complexité	Description
`move`	O(N^2)	déplace le segment cible d'un bisegment à toutes les positions possibles dans l'hypothèse de traduction
`replace`	O(N x T)	identique au `replace` de Langlais *et al.* (2007) : change la traduction d'un segment source par une autre traduction présente dans la table de traduction
`merge`	O(N x T)	identique au `merge` de Langlais *et al.* (2007) : fusionne deux segments source adjacents et remplace les deux segments cibles correspondants par une traduction du nouveau segment source créé
`split`	O(N x S x T^2)	identique au `split` de Langlais *et al.* (2007) : scinde un segment source en deux parties et remplace chacun des segments cibles correspondant par une autre traduction
`remove`	O(N)	supprime le segment cible d'un bisegment dans l'hypothèse de traduction ; le segment cible est remplacé par une chaîne de caractères vide, mais reste disponible pour subir une nouvelle opération au cours des itérations suivantes
`paraphrase`	O(N x R x T)	remplace le segment source d'un bisegment par un autre segment source et remplace le segment cible par l'une des traductions du nouveau segment source

TABLE 4.1 – Opérations utilisées par notre recherche locale oracle. **N** : nombre de segments dans la phrase source ; **T** : nombre maximum de traductions par segment dans la table de traduction ; **S** : nombre moyen de tokens par segment ; **R** : nombre maximum de bisegments monolingues par segment source dans la table de paraphrase.

précédent	... le projet qui ferait gagner le plus de temps sur un ferroviaire$_{15}$ trajet$_{16}$ très long$_{17}$
	... the project which would win the more time on a rail$_{15}$ route$_{16}$ very long$_{17}$
`move`	... le projet qui ferait gagner le plus de temps sur un ferroviaire trajet très long
	... the project which would win the more time on a very long rail route
précédent	il est évident que parler d' intermodalité présuppose un profond changement de la culture d' entreprise .
	it is clear that speak intermodality presupposes a profound change in the business culture .
`paraphrase`	il est évident que débat d' intermodalité présuppose un profond changement de la culture d' entreprise .
	it is clear that discussion on intermodality presupposes a profound change in the business culture .
précédent	qu' il me soit$_3$ permis$_4$ dès lors de le placer dans une perspective plus historique .
	it would therefore be$_3$ allowed$_4$ to put it into a more a more historical perspective .
`merge`	qu' il me soit permis dès lors de le placer dans une perspective plus historique .
	it would therefore be permitted to put it into a more a more historical perspective .

FIGURE 4.2 – Exemples complémentaires d'opérations de réécriture non illustrées dans la séquence d'opérations de la Figure 4.1.

l'italien (it), le néerlandais (nl), le portugais (pt) et le suédois (sv). Toutefois, l'étude de la performance de la recherche locale oracle se focalisera principalement sur la direction de traduction fr→en. Pour analyser la performance de `GOS` en fonction de la quantité de données utilisées pour l'entraînement du système produisant la traduction à réécrire, des systèmes de traduction fr→en ont été entraînés en divisant par 2, 4, 8 et 16 la taille des données d'entraînement utilisées.

De plus, comme la configuration `europarl-intersect` n'utilise qu'une seule traduction de référence pour calculer le score *s*BLEU de la métrique automatique, nous avons également fait l'étude d'un système français→anglais entraîné sur les données d'expressions touristiques du corpus `BTEC` (Takezawa *et al.*, 2002) utilisant 16 traductions de référence pour l'optimisation et 7 traductions de référence pour l'évaluation. Les 7 traductions de référence disponibles ont également été utilisées en cours de recherche locale pour calculer les scores *s*BLEU [3] de chaque hypothèse de réécriture générée.

Les systèmes `europarl-intersect` et `BTEC` utilisent le décodeur `moses` avec une optimisation réalisée par `MERT` sur des listes de 200 meilleures hypothèses. Les modèles de langue de chaque système sont des modèles trigrammes, appris avec un lissage Kneser-Ney, entraînés uniquement sur la partie cible des données parallèles utilisées pour l'apprentissage des tables de traduction. Les scores BLEU et TER de ces systèmes de référence sont donnés dans la Table 4.2.

3. On parle alors de multi-*s*BLEU, un *s*BLEU qui utilise de multiples traductions de référence.

	apprentissage	**dev**	**test**		
	# M tokens	# k tokens	# k tokens	BLEU	TER
			`europarl-intersect`		
fr	10,2	32,8	32,8	-	-
en	8,8	28,3	28,6	29,1	54,0
/2	4,4	|	|	28,6	54,4
/4	2,2	|	|	27,6	55,4
/8	1,1	|	|	26,1	56,8
/16	0,5	|	|	25,2	58,4
da	8,4	27,0	27,2	23,2	61,3
de	8,4	27,1	27,1	17,0	68,0
el	8,8	28,5	28,5	23,5	62,2
es	9,2	29,5	29,7	35,9	49,7
fi	6,4	20,6	20,5	11,2	79,7
it	10,2	28,9	29,0	31,6	55,3
nl	8,9	28,2	28,7	21,2	64,6
pt	9,1	29,4	29,3	33,4	52,8
sv	7,9	25,7	25,8	21,0	62,7
			`BTEC`		
fr	0,2	0,5	0,5	-	-
en	0,2	0,5*	0,5**	59,6	24,6

TABLE 4.2 – Description des données utilisées par les systèmes `europarl-intersect` et `BTEC`. Les scores BLEU et TER sont données pour chaque système et paire de langues utilisés, dans le sens de traduction fr→`xx` (`xx` étant l'une des autres langues présentées ici que le français). Le système `BTEC` utilise 16 références (*) pour l'optimisation et 7 (**) pour l'évaluation.

4.1.3 Expériences et analyse

Les expériences présentées dans cette section sur la recherche locale oracle ont été effectuées en utilisant *s*BLEU comme fonction de score et les opérations de réécriture décrites par la Table 4.1. L'hypothèse amorce réécrite par la recherche locale correspond à la meilleure hypothèse retournée par le décodeur `moses`. La configuration par défaut s'effectue avec un faisceau de recherche de taille k fixée à 1 ; d'autres expériences avec des tailles de faisceau plus grandes seront également décrites. Puisque le voisinage des k meilleures hypothèses est généré à chaque itération, la recherche est initialisée avec la liste constituée des k meilleures hypothèses produites par le décodeur `moses`.

Compte tenu de la taille importante des tables de traduction, et même si la fonction de score *s*BLEU est peu coûteuse à calculer, construire et évaluer, chaque hypothèse du voisinage sans filtrage demande des temps de calcul conséquents. Puisqu'il s'agit pour nous essentiellement de mettre en évidence le potentiel d'amélioration des traductions, on peut se contenter d'une limite basse : la taille des voisinages a ainsi été limitée en ne considérant que les segments cible de la table de traduction contenant au moins un token présent dans la traduction de référence de la phrase en cours de traitement, si ce token ne figure pas parmi les 50 tokens les plus fréquents de la partie cible du corpus d'apprentissage.

4.1.3.1 Mise en évidence du potentiel d'amélioration des traductions

La Table 4.3 présente les résultats de la recherche locale oracle en activant soit toutes les opérations, soit chaque opération séparément pour les systèmes `europarl-intersect` et `BTEC` sur la direction de traduction français→anglais.

On observe tout d'abord que les opérations de réécriture ayant le plus d'impact sont celles accédant directement à la table de traduction, `replace` et `split`, avec d'importants gains oracle de respectivement 13,1 et 16,7 points BLEU pour le système `europarl-intersect`. L'opération `split` a un impact nettement plus fort que l'opération plus simple `replace`[4]. Cette opération permet d'atteindre une nouvelle traduction par composition de deux bisegments non atteignables autrement si le bisegment correspondant à cette composition n'existe pas dans la table de traduction. Au contraire, l'opération `merge` s'avère quant à elle très peu utilisée. L'opération `remove` seule n'a qu'un impact très faible, cette opération activée seule ne pouvant apporter des améliorations qu'en éliminant des tokens absents de la traduction de référence en rapprochant la taille de l'hypothèse de celle de la traduction de référence, ou encore en permettant la formation d'un n-gramme présent dans la référence mais coupé en deux précédemment par un token qui ne s'y trouve pas. L'opération `move` a également un impact très limité, ce que l'on peut probablement attribué en grande partie à la proximité dans l'ordre des mots entre le français et l'anglais (voir la section 4.1.3.3), donnant ainsi peu de possiblités à

4. En effet, chaque opération `split` opère potentiellement deux opérations `replace` (une pour chaque bisegment créé par la séparation du segment initial).

	`europarl-intersect` **fr→en (1 réf.)**							`BTEC` **fr→en (7 réfs.)**						
		BLEU				**TER**	**moy. #**		**BLEU**				**TER**	**moy. #**
	score	1g	2g	3g	4g	score	**itérations**	score	1g	2g	3g	4g	score	**itérations**
`moses`	29,0	63,2	35,5	22,6	14,6	54,0	-	59,6	85,1	67,1	53,3	41,5	24,6	-
taille du faisceau = 1														
`merge`	31,8	65,3	38,3	25,2	16,9	51,7	0,75	60,4	85,4	67,8	54,3	42,4	24,3	0,07
`move`	32,0	63,2	39,1	25,8	17,3	53,3	1,00	61,7	85,1	69,5	55,8	43,9	24,6	0,16
`remove`	29,7	67,1	39,2	25,6	16,9	50,0	1,03	59,6	85,1	67,1	53,3	41,5	24,6	0,00
`replace`	42,1	73,9	48,8	34,8	25,1	42,5	4,40	66,	88,7	73,4	60,9	49,3	23,7	0,91
`rewrite`	29,8	64,5	36,2	23,0	14,0	53,5	0,38	59,7	85,1	67,1	53,5	41,6	24,7	0,04
`split`	45,7	74,3	52,7	39,1	28,6	41,3	4,46	69,3	88,1	75,4	64,6	53,9	27,1	1,24
tous	66,5	88,2	73,8	62,6	53,0	23,1	11,04	77,3	91,2	81,7	73,8	65,0	23,5	1,92
taille du faisceau = 2														
tous	66,6	88,1	73,9	62,8	53,2	23,0	11,19	77,9	91,4	82,4	74,4	65,7	23,2	2,28
taille du faisceau = 5														
tous	**67,8**	**88,5**	**74,9**	**64,3**	**55,0**	**22,3**	11,26	**79,1**	**91,9**	**83,3**	**75,7**	**67,5**	**23,0**	2,12

TABLE 4.3 – Effets (scores BLEU et TER) des opérations de réécriture en recherche locale oracle avec différentes tailles de faisceau sur les systèmes `europarl-intersect` et `BTEC`. Les colonnes « 1g », « 2g », « 3g » et « 4g » donnent respectivement les précisions 1-, 2-, 3- et 4-grammes de BLEU.

	news				médical				TED Talks			
	fr→en		en→fr		fr→en		en→fr		fr→en		en→fr	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
moses	28,6	52,4	31,1	51,8	37,3	41,8	38,5	44,5	32,2	47,8	31,8	49,8
recherche locale oracle	**64,0**	**24.0**	**61,6**	**26,9**	**76,1**	**20,2**	**77,8**	**22,1**	**63,2**	**23,6**	**62,8**	**24,7**

TABLE 4.4 – Résultats (scores BLEU et TER) de recherche locale oracle, effectuée à partir d'une hypothèse amorce produite par `moses`, pour les systèmes `news`, `médical` et `TED Talks`.

la recherche locale de faire de grands déplacements de segments qui seront récompensés en score *s*BLEU. Nous considérerons l'opération `paraphrase` et ses effets plus en détails dans la section 4.1.3.4.

L'activation de toutes les opérations montre un important potentiel d'amélioration, respectivement de 37,5 et 17,7 points BLEU pour les systèmes `europarl-intersect` et `BTEC`. Pour les deux systèmes utilisés ici, la plupart des améliorations sont obtenues en utilisant des segments de plus petite taille *via* `split` et en choisissant des traductions de segment source plus appropriées *via* `replace`. La Table 4.4 présente des améliorations du même ordre pour les systèmes `news`, `médical`, et `TED Talks`. Cependant il nous faut tempérer ces résultats : ces très nettes améliorations du score BLEU ne nous assurent pas de l'obtention de traductions quasi parfaites (voir section 4.1.3.5). Notamment, un token correspondant à une traduction correcte mais absent de la traduction de référence peut être réécrit sans qu'il ne s'agisse systématiquement d'une amélioration de la traduction. Le système `BTEC`, qui dispose de 7 traductions de référence, est moins sensible à ce type de phénomène, ce qui se traduit par des gains en scores BLEU nettement moins importants.

La Figure 4.3 illustre la distribution des opérations de réécriture effectuées pour chaque quart des séquences d'opérations complètes [5] dans la configuration où tous les types d'opérations sont activés pour le système `europarl-intersect`. Le premier quart des opérations effectuées permet d'obtenir à lui seul presque la moitié des gains complets. Au cours de ce premier quart, les opérations `split` et `replace` sont principalement utilisées. L'opération `move` prend une plus grande importance dans les quarts suivants, tandis que `remove` agit plutôt en fin de séquences sur les segments pour lesquels les autres opérations n'ont pu améliorer le score des traductions.

Finalement, la Table 4.3 présente les résultats obtenus en utilisant différentes tailles de faisceau pour la recherche locale. Comme attendu, plus celle-ci est grande et plus l'amélioration obtenue est importante. Toutefois, afin de réduire significativement les temps de calcul, toutes les expériences suivantes seront réalisées avec une taille de faisceau fixée à 1.

5. Les résultats donnés correspondent à une moyenne pour tout le corpus de test. Ainsi, en moyenne nos séquences contiennent 11,04 opérations, chaque quart contient donc 2,76 opérations.

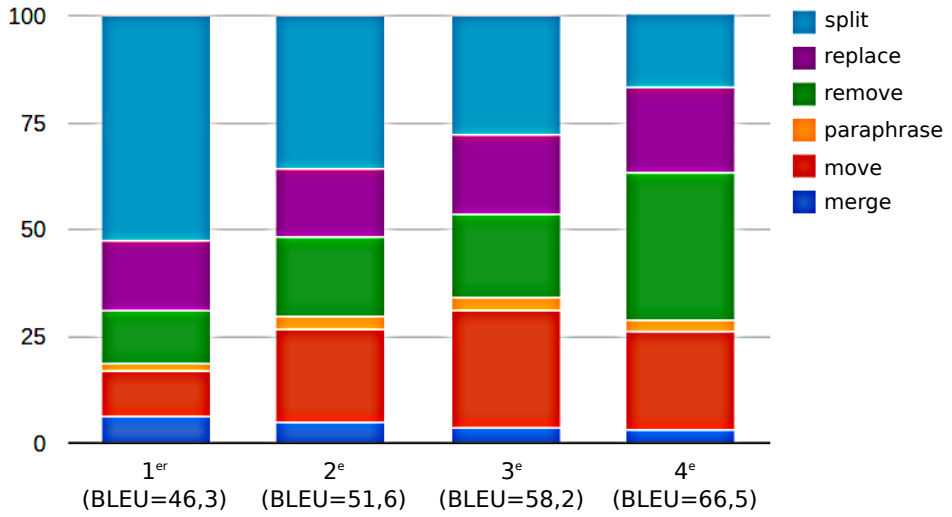


FIGURE 4.3 – Distribution des types d'opérations pour chaque quart des séquences d'opérations complètes effectuées durant la recherche locale oracle sur la tâche `europarl-intersect` fr→en.

4.1.3.2 Effets de la réduction des données d'entraînement et du filtrage des tables de traduction

La Table 4.5 présente l'impact de la taille des données utilisées pour apprendre la table de traduction sur la performance de l'oracle. Ces résultats montrent qu'à chaque réduction de moitié de la quantité des données d'entraînement, la performance du système `moses` diminue d'environ 1 point BLEU, alors qu'elle diminue de près de 5 points BLEU pour la recherche locale oracle. S'il a déjà été montré que le doublement de la quantité de données d'entraînement n'a qu'un impact limité sur la performance d'un système de traduction (Kolachina *et al.*, 2012), les résultats de recherche locale oracle présentés ici montrent que la traduction de référence devient rapidement plus difficile à atteindre.

Cela est aussi le cas lorsqu'une table de traduction est filtrée par une méthode telle que celle proposée par Johnson *et al.* (2007), qui ne conserve qu'une petite partie de la table de traduction contenant des bisegments répondant à certains critères : test de significativité ainsi que les bisegments observés plus d'une fois dans le corpus d'apprentissage et dont les segments source et cible apparaissent plus d'une fois[6]. Un tel filtrage permet ici pour le système `europarl-intersect` de réduire drastiquement la taille de la table de traduction, en ne conservant que 27% des bisegments, tout en obtenant un score BLEU similaire (résultat donné par la ligne `sigtest` de la Table 4.5). L'intuition derrière un tel filtrage veut que soient supprimés les bisegments dont les probabilités sont mal estimées car ils sont trop rarement observés ou peuvent correspondre à du bruit. Toutefois, il apparaît

6. Ces bisegments sont informellement qualifiés de 1-1-1 : 1 occurrence du segment cible, 1 occurrence du segment source, et qui apparaissent tous les deux dans la même biphrase du corpus d'apprentissage. Le filtrage a été effectué ici avec l'outil `filter-pt` de `moses` en utilisant les paramètres par défaut `-l a+e -n 30`.

	BLEU						TER		# bisegments
	référence	oracle	1g	2g	3g	4g	référence	oracle	**dans la table de traduction**
`complet`	**29,1**	**65,9**	87,9	73,3	61,9	52,4	**54,0**	**23,5**	735 273
/2	28,6	60,8	85,5	68,8	56,5	46,4	54,4	27,5	419 716
/4	27,6	55,6	82,8	64,3	51,0	40,7	55,4	31,1	239 647
/8	26,1	51,1	79,8	60,0	46,3	36,0	56,8	35,1	137 719
/16	25,2	46,0	76,9	55,2	41,1	30,7	58,4	39,0	79 837
`sigtest`	**29,1**	54,7	81,4	63,0	50,3	40,4	54,1	32,3	203 672

TABLE 4.5 – Effets de variations sur la taille des données d'entraînement ainsi que du filtrage de la table de traduction pour le système `europarl-intersect` fr→en. Pour ces expériences, l'ensemble des opérations de réécriture sont activées excepté `paraphrase`. `complet` correspond à la configuration utilisant toutes les données disponibles pour apprendre la table de traduction, et `sigtest` correspond à la configuration utilisant une table de traduction filtrée par test de significativité. Les précisions n-grammes de BLEU données sont celles de l'oracle.

qu'avec une table ainsi filtrée le score BLEU de la recherche locale oracle est nettement moins élevé que celui obtenu avec une table non filtrée, ce qui correspond à une perte de 11,2 points BLEU et une augmentation de 8,8 points TER. Bien que l'oracle améliore toujours très nettement la traduction, cette perte indique qu'une partie des entrées filtrées auraient été utiles pour atteindre la traduction de référence avec une meilleure fonction de score que celle de `moses`. Nous devons tout de même considérer ces résultats avec précaution : certains des bisegments filtrés pouvaient effectivement correspondre à de mauvaises associations qui pouvaient participer de manière aberrante à l'augmentation du score BLEU.

4.1.3.3 Impact de la langue cible

La performance d'un système de traduction peut varier de façon importante en fonction de la paire de langues traitée, ce qui a par exemple été mis en évidence par le travail de Koehn *et al.* (2009). Nous pouvons, compte tenu des paires de langues utilisées dans nos expériences, étudier l'impact sur la performance de la recherche locale oracle en fonction de la proximité entre langues source et cible. Si le français et l'espagnol peuvent par exemple être considérées comme deux langues relativement proches, le français et le finnois n'ont aucun lien de parenté. Il nous est ainsi possible d'observer l'impact de paire de langues avec des structures grammaticales très différentes sur la performance de la recherche locale oracle.

Les résultats de la recherche locale oracle pour différentes paires de langues pour `europarl-intersect` sont donnés dans la Table 4.6. Les améliorations obtenues par l'oracle par rapport au système de référence `moses` sont très variées, allant de +106% (pour l'espagnol) à +311% (pour le finnois). Ce dernier résultat met en évidence le fait que, en dépit d'un score BLEU relativement bas atteint par la recherche locale oracle, l'importante amélioration de la traduction générée par `moses` traduit le fait que de nom-

breuses entrées de la table de traduction sont utiles mais souffrent d'une mauvaise modélisation par `moses` en l'état.[7] Un autre résultat intéressant concerne la paire de langues français-espagnol, pour laquelle l'amélioration relative en BLEU obtenue par la recherche locale oracle est la plus faible par rapport aux autres paires de langues (+106%) alors qu'au contraire elle est la plus forte lorsque la traduction est évaluée avec TER (-63%).

Un autre axe d'analyse intéressant concerne la répartition des opérations de réécriture effectuées en fonction de la paire de langues. Cette répartition est donnée par la Figure 4.4. L'opération `replace` apparaît comme uniformément utile pour toutes les paires de langues, mettant en évidence la relative faiblesse des modèles de traduction pour guider le décodeur vers la bonne traduction. Les opérations `split` sont plus souvent employées pour les paires de langues ayant déjà un bon niveau de qualité, par exemple en traduction vers l'anglais et le portugais. Cela peut être attribué à une utilisation excessive par le système de traduction de bisegments comportant des segments cible trop longs, et peut-être trop peu souvent observés dans les données d'apprentissage pour être considérés comme fiables et bien modélisés. La recherche locale oracle a donc tendance à scinder de tels grands segments afin de progresser vers la traduction de référence. On observe une tendance légèrement différente pour les langues romanes et le grec pour lesquelles l'opération `merge` est légèrement plus utilisée que pour les autres langues. Pour ces langues, le décodeur `moses` a donc utilisé plus souvent de plus petits bisegments pour traduire par composition des fragments de la phrase source qui ont finalement été retraduits par l'oracle. Une possibilité parmi d'autres pour corriger ce problème serait d'utiliser un meilleur modèle de langue afin de mieux modéliser la qualité d'une séquence de segments cible. De façon attendue, l'opération `move` est plus utilisée pour des langues cible ayant un ordre des mots très différent du français telles que l'allemand, et dans une moindre mesure le néerlandais et les langues scandinaves. L'utilisation cette l'opération met en évidence les difficultés du décodeur à trouver les bons réordonnancements à effectuer, surtout lorsqu'ils doivent se faire sur de longues distances. Finalement, l'opération `remove` est très utilisée pour le finnois, ce qui illustre principalement la mauvaise qualité de la tokenisation effectuée imputable à la richesse morphologique de cette langue, qui rend très difficile la capture de bisegments pertinents.

4.1.3.4 Une opération `paraphrase` pour améliorer l'atteignabilité des traductions de référence

Jusqu'à présent, nous n'avons pas considérer dans nos analyses l'opération `paraphrase` : celle-ci a pour particularité de réécrire localement la phrase *source*, ce que ne font pas les autres opérations qui ne modifient que le texte cible et/ou uniquement la segmentation du texte source. Ces réécritures des phrases à traduire permet de rendre atteignables de nouvelles traductions locales, qui pourraient parfois correspondre à celles attendues dans la traduction de référence. La Table 4.6 montre que l'utilisation de cette opération en isolation, pour la paire de langues français-anglais, permet d'obtenir une amélioration modeste de 0,8 point BLEU, mais laquelle est néanmoins obtenue

7. Cependant, il est à noter que la mauvaise qualité de la traduction produite peut aussi être en grande partie due à la faiblesse du modèle de langue utilisé, celui-ci ayant été entraîné avec très peu de données relativement à des tailles de corpus monolingues plus communément utilisées.

		BLEU							TER			# itérations
		score		+paraphrase	1g	2g	3g	4g	score		+paraphrase	moy. par phrase
da	moses	23,2			57,2	28,9	17,3	10,7	61,3			-
	oracle	58,4	↑+151%	+0,9	84,0	66,5	53,8	43,6	29,5	↑-52%	-0,8	10,7
de	moses	17,0			52,6	22,1	11,6	6,3	68,0			-
	oracle	55,1	↑+224%	+1,4	83,7	64,2	50,1	39,0	32,0	↑-53%	-1,2	13,3
el	moses	23,5			53,8	28,9	17,8	11,1	62,2			-
	oracle	62,8	↑+167%	+1,0	85,2	70,3	58,8	49,3	26,5	↑-57%	-0,6	11,5
en	moses	29,0			63,2	35,5	22,6	14,7	54,0			-
	oracle	66,5	↑+129%	+0,6	88,2	73,8	62,6	53,0	23,1	↑-57%	-0,4	11,0
es	moses	35,9			65,3	41,5	29,4	21,2	49,7			-
	oracle	74,0	↑+106%	+0,5	90,6	79,8	70,3	62,5	18,2	↑**-63%**	-0,5	10,7
fi	moses	11,2			40,1	15,5	7,2	3,5	79,7			-
	oracle	46,1	↑**+311%**	+1,2	77,0	55,7	41,4	30,7	38,1	↑-52%	-1,2	11,3
it	moses	31,6			60,1	37,2	25,5	17,9	55,2			-
	oracle	71,2	↑+125%	+1,1	89,3	77,6	66,3	57,7	20,4	↑**-63%**	**-1,7**	11,3
nl	moses	21,2			56,4	26,8	15,4	9,5	64,6			-
	oracle	56,3	↑+165%	**+1,6**	83,7	64,8	51,5	41,0	32,4	↑-50%	-0,7	12,9
pt	moses	33,4			62,3	38,7	26,9	19,1	52,8			-
	oracle	69,8	↑+109%	+0,7	88,6	76,3	66,3	57,7	21,5	↑-59%	-0,5	10,2
sv	moses	21,0			55,0	26,3	15,2	9,2	62,7			-
	oracle	59,9	↑+185%	+1,0	85,0	67,7	55,3	45,2	27,8	↑-55%	-1,1	11,2

TABLE 4.6 – Effets de la langue cible sur le corpus `europarl-intersect` en scores BLEU et TER. +`paraphrase` indique la contribution de cette opération relativement aux configurations où elle n'est pas activée. Les colonnes donnant des pourcentages indiquent l'amélioration du score obtenu avec l'oracle relativement à celui obtenu avec `moses`.

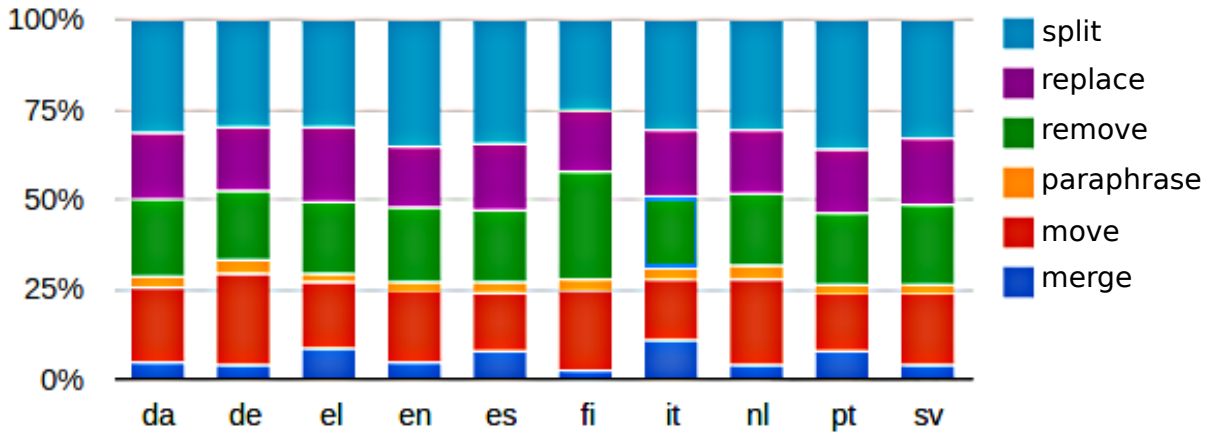


FIGURE 4.4 – Distribution des opérations par langue cible sur la tâche `europarl-intersect` fr→xx.

avec seulement 0,38 application en moyenne par phrase. Les améliorations atteignables *via* cette opération sont assez différentes en fonction des paires de langues tel qu'illustré par la Table 4.6. En traduction depuis le français, celles-ci varient entre 0,5 point BLEU (pour l'espagnol) et 1,6 point BLEU (pour le néerlandais). On observe que les langues qui bénéficient le plus de cette opération sont celles pour lesquelles la performance du système de référence `moses` est relativement basse.

Les applications de l'opération `paraphrase`, comme montré par les travaux de Schroeder *et al.* (2009) et de Onishi *et al.* (2010), incluent une typologie complexe de réécritures qui ne sont pas limitées à des phénomènes de réécriture sans changement de sens. La Table 4.5 montre, par exemple, que de nombreuses applications de cette opération impliquent le remplacement d'un nom par un verbe ou l'inverse. L'étude des patrons de réécriture montre que, pour le français, doté d'un système d'inflection des verbes riche, les verbes sont majoritairement réécrits par d'autres verbes, dont la traduction est probablement plus facile. Les réécritures de noms par des noms, de noms par des verbes et d'adverbes par des noms apparaissent comme beaucoup moins fréquentes.

4.1.3.5 Les dérives d'une recherche locale oracle guidée par *s*BLEU

Dans les expériences oracle des sections précédentes, la recherche locale cherche à maximiser le score *s*BLEU de la traduction à réécrire étant donnée une traduction de référence particulière. Nous allons succinctement illustrer dans cette section des situations d'améliorations de *s*BLEU, parfois très fortes, qui à l'extrême correspondent à des dégradations de la qualité de traduction initiale.

Tout d'abord, des situations où la phrase à traduire et la traduction de référence possèdent un sens différent mèneront possiblement plus facilement la recherche locale oracle à dégrader la traduction, comme illustré dans les exemples de la Table 4.7. Dans

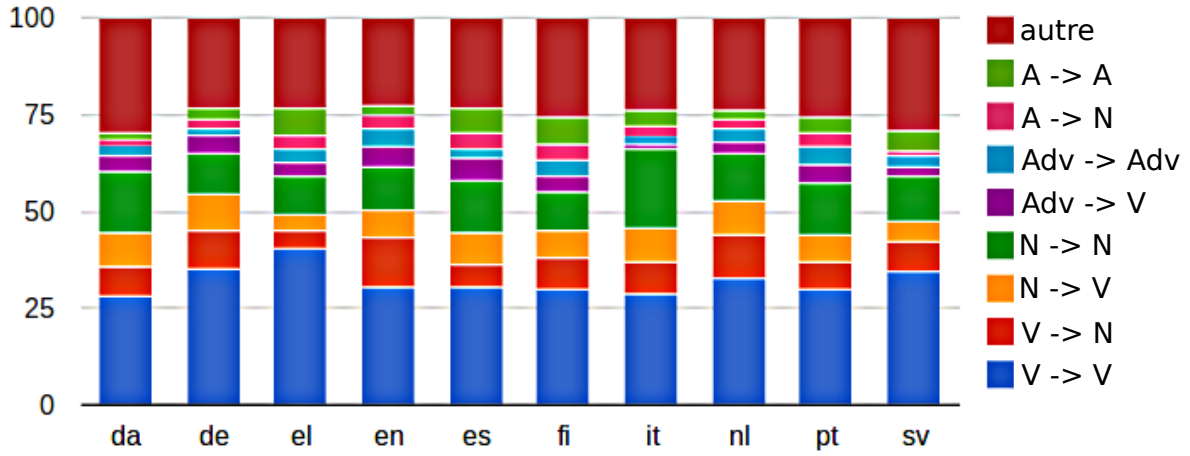


FIGURE 4.5 – Distribution des principaux changements de catégories morpho-syntaxiques des tokens source réécrits par l'opération `paraphrase`.

l'exemple 1, la traduction produite par `moses` s'avère être correcte [8], mais assez éloignée de la traduction de référence. La recherche locale oracle peine à se rapprocher de la référence et produit finalement une traduction avec un meilleur score *s*BLEU mais au prix d'une dégradation de la grammaticalité de la traduction.

Pour ces exemples, même un oracle calculé sur l'espace de recherche du décodeur (Wisniewski et Yvon, 2013) resterait assez loin de la traduction de référence. En effet, la table de traduction ne contient probablement pas les bisegments nécessaires tant la présence de certains tokens dans la phrase cible (par exemple « couronnes » dans l'exemple 1) paraît improbable étant donnée la phrase source.

La Table 4.8 présente deux autres exemples de dégradation de la traduction produite par `moses` en dépit d'une nette augmentation de leur score *s*BLEU. L'exemple 3 illustre une recherche locale ayant formé de nouveaux n-grammes présents dans la traduction de référence, et notamment des 4-grammes en déplaçant le token « . », mais en dégradant fortement la grammaticalité de la traduction. Celle-ci commence désormais par le bigramme aberrant « un une ». L'exemple 4 montre une traduction produite par `moses` sur laquelle la recherche locale oracle a supprimé 2 tokens, la rendant ainsi agrammaticale. Cette suppression est le résultat de l'application d'une opération `remove` effectuée pour faire correspondre la longueur de l'hypothèse à celle de la traduction de référence, la recherche oracle n'ayant par ailleurs pas permis de produire le token « verrons ». Dans les cas de ces deux nouveaux exemples, l'algorithme de recherche peut ici être mis en cause du fait de son incapacité à reproduire la traduction de référence étant donnée la segmentation initiale de la traduction amorce à réécrire et l'ensemble des opérations de réécriture dont il dispose. On peut imaginer ici qu'une traduction plus proche de la traduction de référence, obtenant donc un meilleure score *s*BLEU, existe dans l'espace de recherche du décodeur.

Si la recherche locale oracle exhibe bien un fort potentiel d'amélioration des traduc-

8. La traduction est correcte au moins hors contexte, le pronom « they » peut par exemple être mal traduit.

		score *s*BLEU
	Exemple 1	
source	annually they must then also take extra costs in account .	
amorce `moses`	chaque année , ils doivent aussi prendre en compte les coûts supplémentaires .	10,5
oracle	chaque année , on peut donc également en les coûts supplémentaires .	22,2$_{(+11,7)}$
référence	on peut donc compter également une centaine de couronnes en plus .	
	Exemple 2	
source	the candidate probably received financial compensation .	
amorce `moses`	le candidat a reçu une compensation financière .	10,8
oracle	le candidat est sans doute fait une compensation financière .	16,2$_{(+5,4)}$
référence	et il encaissait sans doute par le fait de se retirer .	

TABLE 4.7 – Résultats d'une recherche locale oracle à partir d'une traduction produite par le système `news` dont les réécritures rapprochent d'une traduction de référence de mauvaise qualité.

		score *s*BLEU
	Exemple 3	
source	a viable company must be able to take even such unpopular measures sometimes .	
amorce `moses`	une entreprise viable doit être capable de prendre des mesures impopulaires telles même parfois .	25,2
oracle	un une société viable de , il faut prendre des mesures impopulaires . telles même parfois	47,6$_{(+22,4)}$
référence	une société viable doit savoir que de temps en temps il faut prendre des mesures impopulaires .	
	Exemple 4	
source	" we'll see . "	
amorce `moses`	" nous allons voir " .	36,6
oracle	" nous " .	45,2$_{(+8,6)}$
référence	" nous verrons " .	

TABLE 4.8 – Résultats d'une recherche locale oracle à partir d'une traduction produite par le système `news` dont les réécritures dégradent la qualité de la traduction.

tions selon *s*BLEU, il est donc possible de trouver des exemples de traductions ayant été dégradées localement par celle-ci. Ceci illustre, pour certaines situations, le caractère inadapté des métriques d'évaluation automatique utilisées, notamment lorsqu'il s'agit d'évaluer la qualité de traduction d'une phrase isolée.

4.2 Réécriture automatique de traductions à l'aide de modèles complexes

4.2.1 Une réécriture guidée par des modèles inutilisables lors du décodage

La section 4.1 nous a permis de mettre en évidence le fait qu'il existe un fort potentiel d'amélioration des traductions produites par un système de TAS à l'état de l'art (`moses`) en utilisant un algorithme de recherche locale qui serait guidé parfaitement. Nous avons par ailleurs également montré empiriquement dans la section 3.5 qu'une recherche locale réexploitant la fonction de score du décodeur ne permet plus d'obtenir d'améliorations depuis de telles traductions. Une meilleure fonction de score, de performance intermédiaire entre celle utilisée par le décodeur et celle utilisée par l'oracle, devrait donc permettre d'obtenir certaines améliorations. Une façon simple d'améliorer la fonction de score est d'y ajouter des modèles complexes que le décodeur n'a pas pu utiliser (voir section 3.2.2). Chaque nouveau modèle dont les scores sont calculés au niveau de la phrase peut être ajouté à la combinaison de scores linéaire utilisée par le décodeur. La nouvelle fonction obtenue peut ensuite être utilisée pour guider la recherche locale à la place de la fonction de score du décodeur, telle qu'utilisée dans le travail de Langlais *et al.* (2007).

La section 4.2.1.1 présente la méthode utilisée pour extraire une *table de réécriture* à partir de la table de traduction du décodeur, laquelle sera ensuite utilisée par la recherche locale. La section 4.2.1.2 détaille la façon dont sont optimisés les poids des modèles de la nouvelle fonction de score utilisée pour guider la recherche locale. La section 4.2.1.3 présente elle le fonctionnement global de notre système de réécriture.

4.2.1.1 Table de réécriture

En vue d'assurer des temps de calcul raisonnables, l'utilisation de la totalité de la table de traduction du décodeur comme table de réécriture n'est pas envisageable. [9] Les voisinages générés seraient en effet trop grands pour pouvoir être efficacement évalués avec une fonction de score utilisant des modèles complexes. Dans nos expériences décrites dans la section 3.5, la recherche locale avait été effectuée avec une table de réécriture contenant les dix meilleures traductions de chaque segment source. Un tel filtrage a permis de réduire suffisamment la taille des voisinages générés pour permettre une évaluation des hypothèses

9. `moses` n'utilise d'ailleurs pas tous les bisegments de la table de traduction au cours du décodage : par défaut, seuls les 20 meilleurs segments cible sont retenus pour chaque segment source considéré.

qu'ils contiennent avec une fonction de score relativement simple telle que celle utilisée par `moses`. En revanche, l'utilisation d'une fonction de score incluant des modèles bien plus coûteux rend ce filtrage insuffisant pour des calculs en temps raisonnable. Une solution proposée et évaluée dans cette section consiste à n'utiliser que les bisegments exploités par le décodeur pour construire ses n meilleures hypothèses de traduction pour chaque phrase. Ces bisegments ont la particularité d'avoir été appréciés par le décodeur car celui-ci les a utilisés dans les meilleures traductions qu'il peut produire. La conservation des bisegments ayant servi à la construction de la traduction amorce du système de réécriture, associée à l'exploitation d'une fonction de score réputée meilleure, semble offrir de bonnes garanties pour ne pas dégrader la qualité de la traduction amorce. En outre, cela procure également l'avantage de ne retenir en pratique qu'un nombre relativement faible de bisegments en fonction du nombre de n meilleures hypothèses choisi, comme nous l'illustrerons dans la section 4.2.3.1.

4.2.1.2 Optimisation de la nouvelle fonction de score

Les poids des modèles de la nouvelle fonction de score utilisée pour ordonner par qualité décroissante les hypothèses du voisinage peuvent être optimisés de la même manière que ceux utilisés dans la fonction de score d'un système de reclassement (voir section 3.3.1). La façon dont sont générés les exemples d'entraînement doit toutefois être adaptée. L'optimisation du reclassement des hypothèses générées par la recherche locale, qui réutilise `KB-MIRA` dans nos expériences, est effectuée sur des voisinages générés par une première itération de recherche locale utilisant une table de réécriture créée avec la méthode décrite dans la section 4.2.1.1. L'optimisation est effectuée sur le même corpus de développement que celui utilisé pour optimiser le système `moses` initial. Il est donc intéressant de noter que notre approche ne requiert pas de nouvelles données parallèles pour son entraînement.

4.2.1.3 Le système de réécriture : `rewriter`

Le fonctionnement global de notre système de réécriture, `rewriter`, qui utilise une fonction de score enrichie par des modèles complexes, est présenté par la Figure 4.6. Dans la section 3.5, la recherche locale présentée, qui était guidée par la fonction de score du décodeur, réécrivait la meilleure hypothèse trouvée par le décodeur. `rewriter` utilise lui comme traduction amorce la meilleure hypothèse trouvée lors du reclassement des n meilleures hypothèses générées par le décodeur [10]. Le système de reclassement et la recherche locale utilisent donc la même fonction de score, dont seuls les paramètres (poids des modèles) diffèrent, ceux-ci ayant été optimisés à l'aide d'exemples d'entraînement différents. On s'assure donc ainsi que le système de réécriture réécrira la meilleure hypothèse qu'il est possible d'obtenir avec un système de traduction à l'état de l'art effectuant un reclassement d'hypothèses avec une fonction de score améliorée.

10. Un système de reclassement standard, tel que décrit dans la section 3.3.1, est utilisé pour cette étape.

Nous n'avons pas effectué de nouvelles expériences de recherche locale utilisant la fonction de score de `moses`, aucune amélioration n'ayant précédemment été constatée (cf. section 3.5). Dans les configurations que nous avons étudiées, `rewriter` n'utilise que les opérations de réécriture les plus prometteuses pour améliorer la traduction (selon nos résultats décrits dans la section 4.1.3.1) tout en conservant un espace de recherche de taille raisonnable : `split`, `merge` et `replace`. En phase d'évaluation, `rewriter` itère sur toutes les phrases jusqu'à convergence, c'est-à-dire aussi longtemps que l'application d'une opération de réécriture permet une augmentation du score de l'hypothèse.

4.2.2 Expériences

4.2.2.1 Systèmes de référence

Nos expériences utilisent les systèmes de référence `news` (fr↔en), `médical` (en↔fr), et `TED Talks` (fr↔en) présentés dans la section 3.1 [11]. Dans les expériences qui vont suivre, les systèmes de reclassement (`reranker`) et de réécriture (`rewriter`) utilisent les mêmes modèles complexes **IBM1**, **POSLM**, **PPP** et **SOUL**, qui sont décrits dans la section 3.3.2.1.

Les scores BLEU et TER obtenus par ces systèmes de référence sont présentés dans la Table 4.9. L'analyse des résultats obtenus sera principalement effectuée avec le système `médical` en→fr.

4.2.2.2 Résultats

Système	news				médical				TED Talks			
	fr→en		en→fr		fr→en		en→fr		fr→en		en→fr	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
`moses`	29,4	52,4	31,8	51,8	38,2	41,8	38,7	44,5	32,3	49,9	32,5	47,7
`reranker`	30,3	51,5	32,9	50,8	40,6	39,6	42,8	40,9	32,8	49,4	33,0	47,3
`rewriter`	**30,8**	**51,0**	**33,5**	**50,2**	**41,6**	**39,3**	**44,0**	**39,9**	**33,7**	**49,3**	**33,4**	**47,4**

TABLE 4.9 – Scores BLEU et TER des systèmes de référence (`moses` et `reranker`) et de `rewriter` pour les systèmes `news`, `médical` et `TED Talks` pour les deux directions de traductions fr→en et en→fr.

Les résultats de `rewriter` présentés par la Table 4.9 sont ceux obtenus en utilisant la meilleure configuration trouvée de façon empirique, sur un corpus de développement, parmi celles testées (voir section 4.2.3.1) : la table de traduction est extraite

11. Il est à noter que les scores BLEU et TER de la traduction générée par `moses` présentés dans la section 3.1 peuvent être différents des scores présentés dans ces nouvelles expériences et celles qui suivront. Cette différence est uniquement due à la variation obtenue entre différentes exécutions de l'algorithme d'optimisation `KB-MIRA` ainsi qu'à l'utilisation d'un autre script, uniquement dans cette section, le `mteval13a.pl` de `moses`, pour calculer les scores BLEU.

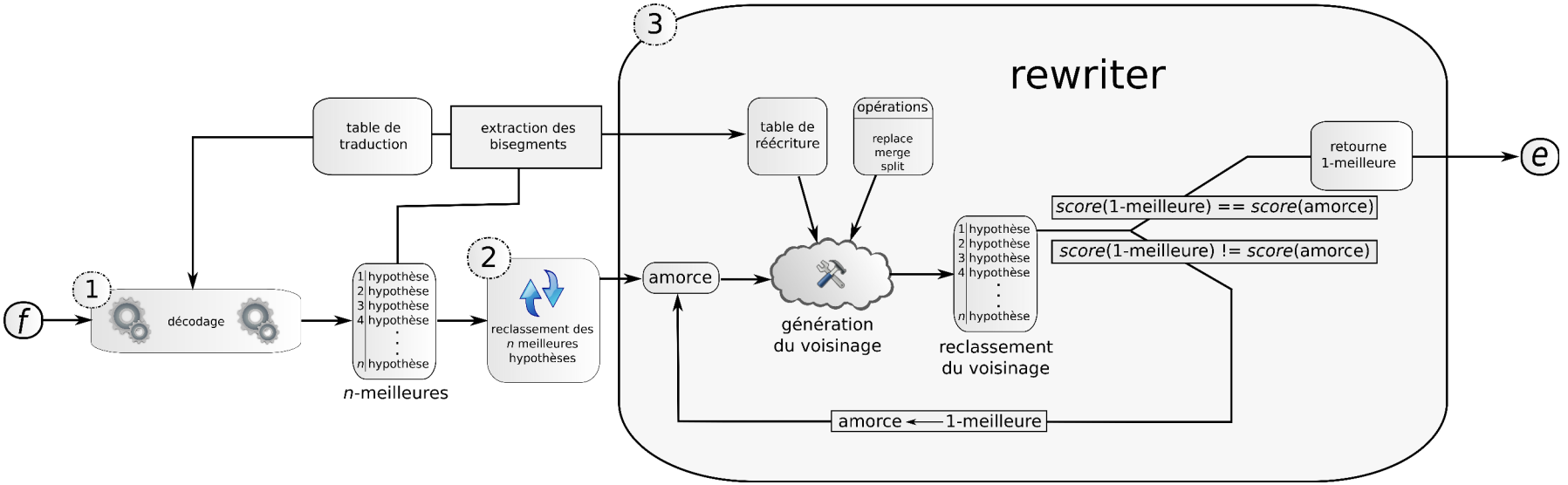


FIGURE 4.6 – Illustration du fonctionnement de `rewriter`. La liste initiale des n meilleures hypothèses de traduction de la phrase source f est générée par le décodeur `moses` (1). Cette liste est reclassée par `reranker` à l'aide d'une nouvelle fonction de score (2), puis une table de réécriture en est extraite. La nouvelle meilleure traduction obtenue après reclassement est alors réécrite par `rewriter` (3), qui applique une opération de réécriture par itération. `rewriter` s'arrête lorsque le score de la meilleure traduction obtenue après une itération n'augmente plus. La traduction de meilleure score e est retournée par le système.

des 10 000 meilleures hypothèses générées par `moses` et les exemples d'entraînement sont des voisinages des n meilleures hypothèses générées par une itération de recherche locale (voir la section 4.2.3.1 pour plus de détails sur les hypothèses utilisées en entraînement). Les listes des n meilleures hypothèses ont été produites par `moses` sans activer l'option `distinct`. Ce choix a été effectué pour privilégier la diversité des segmentations à la diversité lexicale pour l'extraction de la table de réécriture, `rewriter` opérant au niveau des bisegments.

Les résultats obtenus montrent une amélioration importante par `rewriter` de la traduction produite par `reranker`. Pour le système `médical` en→fr, l'amélioration est de 1,2 point BLEU et -1 point TER par rapport à `reranker`, contre 1 point BLEU et -0,3 point TER d'amélioration pour la direction fr→en. Pour les systèmes `TED Talks`, l'amélioration obtenue est plus faible : 0,9 et 0,4 point BLEU respectivement pour en→fr et fr→en. Sur la tâche `news`, les améliorations obtenues sont du même ordre avec des gains de 0,6 et 0,5 point BLEU pour en→fr et fr→en. L'amplitude plus faible des gains obtenus par `rewriter` sur `TED Talks` et `news` relativement à ceux obtenus sur `médical` peut s'expliquer par la moins bonne performance des modèles complexes utilisés, les améliorations obtenues en reclassement d'hypothèses étant en effet nettement moins importantes que celles obtenues sur `médical`.

4.2.3 Analyse

Notre analyse a été menée sur la tâche `médical` en→fr car il s'agit de la configuration pour laquelle les modèles complexes utilisés sont les plus prometteurs tel qu'indiqué par les résultats obtenus avec `reranker`. Le système `médical` en→fr utilisé ici est différent des systèmes `médical` construits jusqu'à présent : le modèle de langue générique appris sur des données monolingues hors-domaine n'a pas été utilisé, et seul le modèle de langue spécialisé a donc été utilisé dans les expériences de cette section. [12]

4.2.3.1 Impact de la table de réécriture et des exemples d'entraînement

Nous nous intéressons tout d'abord à la question de l'impact de variations sur la table de réécriture et les exemples d'entraînement utilisés sur les performances de `rewriter`. Les résultats des différentes configurations testées sont présentés dans la Table 4.10 pour la tâche `médical` en→fr.

12. Ce qui explique notamment l'obtention d'un score BLEU légèrement moins élevé que celui présenté par la Table 4.9 sur la tâche `médical` en→fr où le système de traduction était mieux informé grâce à l'utilisation d'un modèle de langue supplémentaire appris sur bien plus de données.

Système de réf.				dev	test		
				BLEU	BLEU	TER	BLEU GOS
`moses`				40,9	38,3	44,6	
`reranker` (*réf.*)				44,1	41,8	41,6	
exemples d'apprentissage	table de réécriture	# bisegments uniques	taille du faisceau				
1 000-meilleures `moses`	`conf10k`	38 455	1	44,1	$39,2_{(-2,6)}$	$43,8_{(+2,2)}$	58,7
10pefNeigh	`conf10k`	38 455	1	43,9	$40,9_{(-0,9)}$	$41,2_{(-0,4)}$	58,7
10-bestNeigh + 10pefNeigh	`conf10k`	38 455	1	43,8	$40,9_{(-0,9)}$	$41,2_{(-0,4)}$	58,7
30-bestNeigh + 10pefNeigh	`conf10k`	38 455	1	44,2	$43,2_{(+1,4)}$	$40,6_{(-1,0)}$	58,7
50-bestNeigh + 10pefNeigh	`rpt5pef`	85 530	1	44,5	$41,0_{(-0,8)}$	$42,0_{(+0,4)}$	50,6
=	`rpt10pef`	149 887	1	44,5	$41,1_{(-0,7)}$	$42,1_{(+0,5)}$	54,5
=	`conf10`	21 398	1	44,5	$42,4_{(+0,6)}$	$41,0_{(-0,6)}$	45,9
=	`conf100`	28 730	1	44,5	$42,9_{(+1,1)}$	$40,8_{(-0,8)}$	50,2
=	`conf1k`	33 929	1	44,5	$43,0_{(+1,2)}$	$40,6_{(-1,0)}$	53,3
=	`conf10k`	38 455	1	44,5	$\mathbf{43,4}_{(+1,6)}$	$\mathbf{40,4}_{(-1,2)}$	58,7
= (`opti`)	`conf10k`	38 455	10	44,5	$\mathbf{43,7}_{(+1,9)}$	$\mathbf{40,1}_{(-1,5)}$	59,6
90-bestNeigh + 10pefNeigh	`conf10k`	38 455	1	44,4	$43,4_{(+1,6)}$	$40,4_{(-1,2)}$	58,7

TABLE 4.10 – Résultats de `rewriter` obtenus sur `médical` en→fr pour différentes configurations d'entraînement, tables de réécriture et tailles de faisceau. `opti` correspond à la meilleure configuration obtenue avec un faisceau de taille 1. Les écarts inscrits entre parenthèses sont donnés par rapport à `reranker` (dénoté (*réf.*)).

Le meilleur résultat est obtenu en utilisant une table de réécriture extraite des 10 000 meilleures hypothèses (`conf10k`). Toutefois, il faut noter que dans la configuration où la table de réécriture est extraite à partir des 10 meilleures hypothèses (`conf10`), un gain de 0,6 point BLEU peut déjà être obtenu, alors que le nombre de bisegments contenus dans la table de réécriture est relativement faible.

Une méthode de sélection très simple, que nous avions utilisée lors de nos premières expériences de recherche locale (voir section 3.5), pour constituer la table de réécriture consiste à utiliser les k meilleures traductions pour chaque segment source dans la table de traduction. Pour k=5 et k=10, le nombre de bisegments conservés est nettement plus grand, mais mène à une dégradation de respectivement 0,7 et 0,8 point BLEU relativement à `reranker`. De manière intéressante, la recherche locale oracle (`GOS`) révèle également un plus fort potentiel d'amélioration en utilisant une table de réécriture de type `conf`k, avec une amélioration possible allant jusqu'à 16,9 points BLEU (`conf10k`), qu'en utilisant une table obtenue avec un tel filtrage, avec une amélioration possible allant jusqu'a 12,7 points BLEU (`rpt10pef`). Ces résultats de nos expériences automatiques et oracle soulignent l'intérêt de conserver dans la table de réécriture les bisegments pour lesquels `moses` a été le plus confiant. De plus, dans la meilleure configuration utilisant `conf10k`, la taille des voisinages générés reste relativement petite : 116 hypothèses sont utilisées en moyenne par phrase et par itération, contre 788 avec la table de réécriture `rpt10pef`. En moyenne donc, peu d'hypothèses sont produites ce qui permet un calcul relativement rapide des modèles complexes, comparé par exemple au calcul de ces mêmes modèles sur les 1000 meilleures hypothèses produites par `moses`. En outre, parmi les hypothèses produites par `rewriter`, seulement 30% d'entre elles sont des hypothèses figurant parmi les 1000 meilleures hypothèses produites par `moses`. La Figure 4.7 illustre l'augmentation de la performance de `rewriter` en fonction de l'augmentation de la valeur du paramètre n qui définit le nombre de meilleures hypothèses de `moses` utilisées pour construire la table de réécriture. [13]

Le choix des exemples d'entraînement a également une importance déterminante dans la performance de `rewriter`. En effet, en utilisant les mêmes exemples pour entraîner `rewriter` que ceux ayant été utilisés pour entraîner `reranker`, c'est-à-dire les n meilleures hypothèses produites par `moses`, `rewriter` dégrade significativement la qualité de la traduction obtenue par `reranker` de -2,6 points BLEU et de 2,2 points TER. En revanche, l'utilisation d'exemples d'entraînement générés par une itération de recherche locale permet un meilleur entraînement de `rewriter` qui améliore ensuite la qualité de la traduction en évaluation. La meilleure configuration d'entraînement utilise des exemples issus d'un mélange d'hypothèses générées par recherche locale avec deux tables de réécriture différentes pour augmenter la diversité : `conf10k` et `rpt10pef`, pour générer respectivement les voisinages des 50 meilleures hypothèses et de la meilleure hypothèse produites par `moses`. Ce résultat nous permet de confirmer l'importance d'utiliser des exemples d'entraînement qui sont similaires à ceux qui seront évalués lors de l'évaluation. Il est à noter

13. Nous n'avons pas testé de valeurs de n supérieures à 10 000 afin de conserver des temps de calculs raisonnables. Augmenter la valeur de n allonge la durée du décodage `moses` qui doit produire et évaluer les listes d'hypothèses utilisées, ainsi que la taille de la table de réécriture extraite.

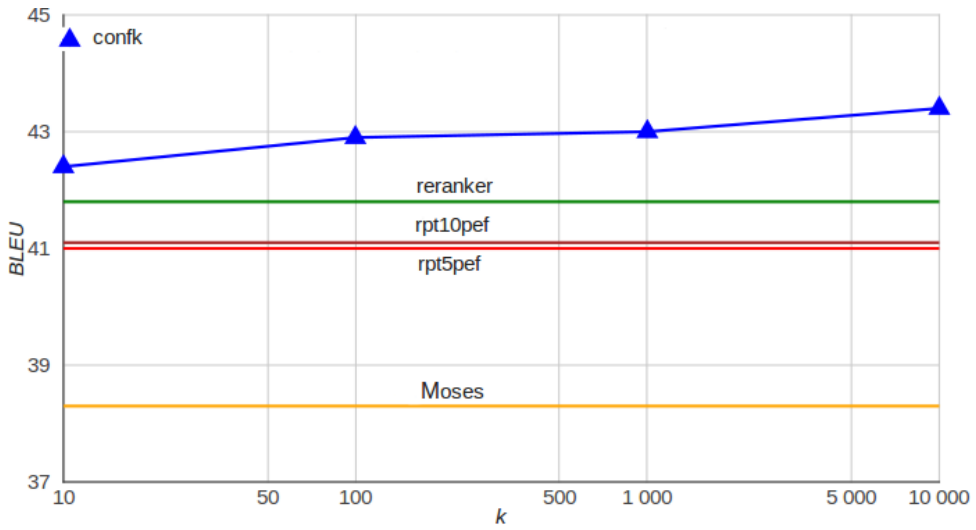


FIGURE 4.7 – Résultats de `rewriter` avec `rpt5pef`, `rpt10pef` et différentes valeurs de n pour l'extraction des tables de réécriture `confk` pour la tâche `médical` en→fr.

que l'ajout du voisinage des hypothèses placées au-delà des 50 meilleures hypothèses aux exemples d'entraînement ne permet plus d'améliorer la qualité des traduction obtenues par `rewriter`.

Utiliser un faisceau de taille supérieure permet d'améliorer les résultats obtenus : en particulier, l'augmentation de la taille de faisceau de 1 à 10 permet d'obtenir un gain de 0,3 point BLEU supplémentaire, soit une amélioration de 1,9 point BLEU relativement à `rewriter`.[14] L'utilisation des modèles complexes par `reranker` puis `rewriter` permet donc dans cette configuration une importante amélioration de 5,4 points BLEU par rapport à la traduction produite par `moses`.

Comme illustré par la Figure 4.8, `rewriter` obtient l'essentiel de son amélioration du score BLEU durant ses trois premières itérations, puis n'obtient plus d'améliorations après neuf itérations. La Figure 4.9a illustre l'évolution du score sBLEU par phrase et met en évidence le fait que toutes les traductions ne sont pas améliorées par `rewriter` : on constate en effet que la configuration `opti` permet l'amélioration pour 40,8% des traductions mais mène à une dégradation pour 29,2% des traductions fournies par `reranker`.

14. Il faut cependant noter que l'utilisation d'un faisceau de taille 10 est devient prohibitif, interdisant l'expérimentation avec des tailles de faisceau supérieures. En utilisant 12 threads de processeurs, 25 minutes ont été nécessaires pour le calcul d'une itération de `rewriter` avec un faisceau de taille 1, contre 3 heures avec un faisceau de taille 10. Cette augmentation est en grande partie imputable aux calculs effectués pour le calcul des scores de `SOUL`, l'élargissement du faisceau augmentant la diversité des 10-grammes à évaluer et rendant ainsi le système de mémoire cache utilisé par l'implémentation de `SOUL` moins efficace.

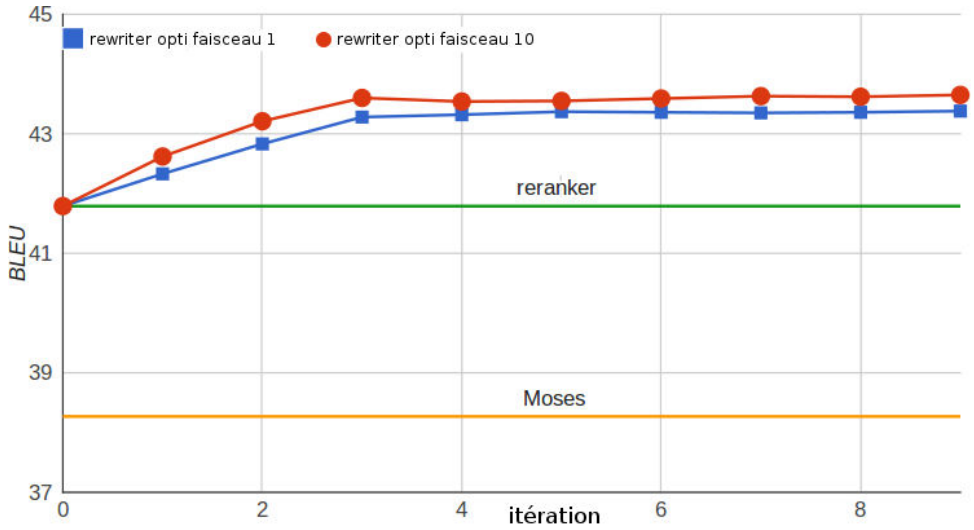


FIGURE 4.8 – Itérations effectuées par `rewriter` sur le corpus d'évaluation de la tâche `médical` en→fr avec la configuration `opti` et deux tailles de faisceau : 1 et 10.

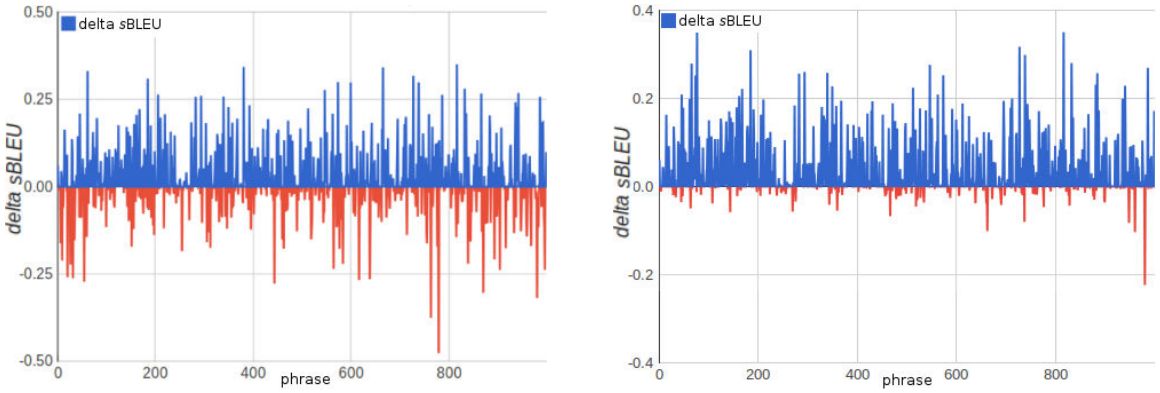
4.2.3.2 Performance de `rewriter` en fonction de la qualité de la traduction à réécrire

Comme le montre la Figure 4.10, la performance de `rewriter` est fortement dépendante de la qualité initiale des phrases à réécrire. Le partitionnement des phrases traduites par `moses` en quartiles selon leur score *s*BLEU révèle que `reranker` puis `rewriter` seront d'autant plus performants que la qualité de la traduction `moses` était faible. En particulier, `rewriter` améliore de 8,6 points BLEU la qualité de la traduction par rapport à `moses` sur le premier quartile, celui de plus basse qualité, tandis que cette amélioration n'est plus que de 1,3 point BLEU sur le dernier quartile. L'amélioration obtenue par `rewriter` pourrait donc être potentiellement plus grande si l'entraînement de `rewriter` était, par exemple, adapté en fonction de la qualité de la traduction à réécrire, laquelle pourrait être estimée automatiquement [15].

4.2.3.3 Expériences semi-oracle : ne pas réécrire ce qui est déjà correct

Nous avons montré dans la section 4.2.2.2 qu'en dépit d'une nette amélioration globale de la qualité de traduction obtenue par `rewriter`, une proportion non négligeable des traductions de phrases sont en réalité dégradées (voir les différences de score *s*BLEU entre `reranker` et `rewriter` dans la Figure 4.9a). Ce résultat signifie que certains des segments cible apparaissant dans la traduction de référence ont été modifiés. Afin de mettre en évidence le potentiel d'un système de réécriture sachant identifier les segments cible à ne pas modifier, une configuration *semi-oracle* a été mise en place dans laquelle les segments cible de l'amorce qui apparaissent exactement dans la traduction de référence sont figés et

15. Par exemple à l'aide d'un système d'estimation de qualité tel que celui utilisé pour construire le système de référence (*baseline*) utilisé lors des campagnes d'estimation de qualité organisées dans le cadre de WMT (Specia *et al.*, 2013)



(a) `rewriter opti` (b) `rewriter` semi-oracle

FIGURE 4.9 – Différences de scores *s*BLEU, pour chaque phrase, entre la meilleure hypothèse de `reranker` et sa réécriture par `rewriter` utilisant la configuration `opti` (4.9a) ou semi-oracle (4.9b).

deviennent ainsi non modifiables, et ainsi non localement dégradables, par `rewriter`. À chaque nouvelle itération de `rewriter`, tout nouveau segment cible réécrit par `rewriter` et présent exactement dans la traduction de référence se trouve également figé pour les itérations ultérieures de `rewriter`. Dans les faits, cette situation revient à simuler une configuration dans laquelle le système disposerait d'une mesure de confiance parfaite au niveau des bisegments indiquant à `rewriter` de ne plus toucher aux segments validés. [16] Hormis ce figement des bisegments corrects, le fonctionnement de `rewriter` demeure entièrement automatique et reprend la configuration `opti` sans nouvel apprentissage. [17]

système	taille du faisceau	test BLEU	test TER
`reranker`		41.8	41.6
`rewriter opti` (*réf.*)	1	43.4	40.4
`semi-oracle`	1	**44.9**$_{(+1.5)}$	**39.2**$_{(-1.2)}$
`semi-oracle`	10	**44.9**$_{(+1.5)}$	**39.0**$_{(-1.4)}$

TABLE 4.11 – Résultats de la configuration semi-oracle utilisant `opti` sur le système `médical` en→fr. La nature oracle de cette configuration correspond au figement des bisegments dont la partie cible appartient exactement à la traduction de référence disponible. Les écarts inscrits entre parenthèses sont donnés par rapport à la meilleure configuration de `rewriter`, dénotée (*réf.*), utilisant une taille de faisceau de 1.

Les résultats de cette configuration semi-oracle sont présentés dans la Table 4.11. À l'aide d'une telle information de confiance parfaite portant sur les bisegments, un gain

16. Ce qui ne donne pas pour autant la garantie qu'avoir un tel bisegment à l'itération courante de la réécriture est optimal en vue d'atteindre la meilleure solution.

17. Il est vraisemblable qu'une réoptimisation de la fonction de score avec des exemples de voisinages générés par cette configuration `semi-oracle` mènerait à de meilleurs résultats.

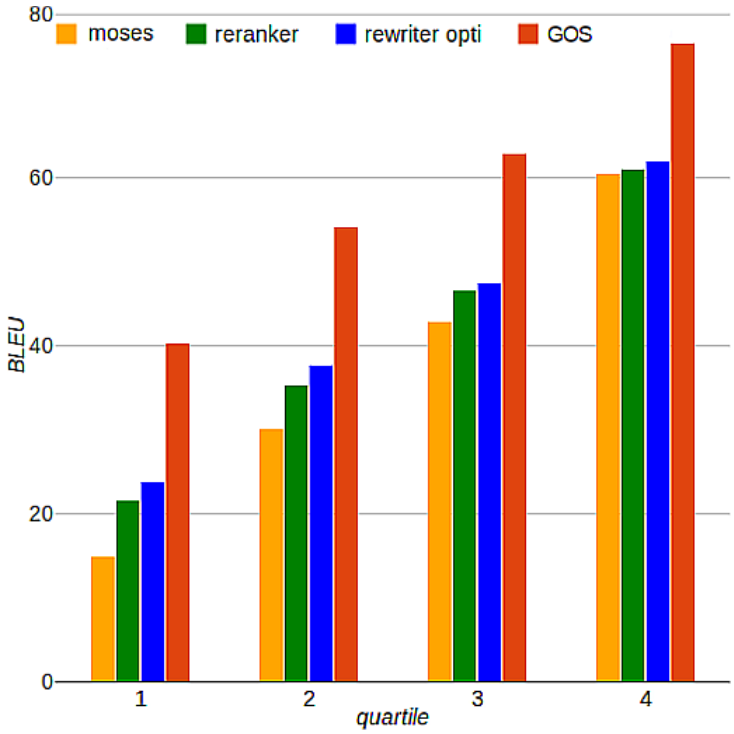


FIGURE 4.10 – Performance de la traduction de `moses`, `reranker`, `rewriter` et `GOS` pour chaque quartile de phrases, où les quartiles sont définis de taille égale selon le score *s*BLEU de l'hypothèse de `moses`.

additionnel de 1,5 point BLEU est obtenu pour `médical`, portant le gain par rapport à `reranker` à 3,1 points BLEU, par rapport à `moses` de 6,6 points BLEU. On note par ailleurs que l'amplitude de ce gain ne s'accroît pas pour une taille de faisceau supérieure à 10. Dès la première itération de `rewriter`, environ 65,6% des segments se retrouvent figés, et 70,5% après la dernière itération. Cette augmentation de la proportion des segments figés illustre la capacité de `rewriter` à exploiter de façon utile les informations fiables qui lui sont apportées. La Figure 4.9b met également en évidence que dans une telle configuration la proportion de traductions dégradées chute significativement.

4.2.3.4 Analyse manuelle des erreurs effectuées

Afin de mieux comprendre les différences qualitatives entre nos quatre principales configurations, `moses`, `reranker`, `rewriter` (`opti`) and `GOS`, un échantillon commun de leurs traduction a été extrait afin d'en analyser les erreurs. Cette analyse a été effectuée sur la tâche `médical` (en→fr) par un annotateur ayant le français comme langue maternelle. Les traductions des 70 premières phrases du corpus de test données par chacun des 4 systèmes ont été annotées en utilisant une typologie d'erreurs dérivée de celle de Vilar *et al.* (2006).

Les résultats de cette analyse manuelle sont reportés dans la Table 4.12. Les résultats les plus marquants concernent les erreurs de désambiguïsation (*désamb.*) et de morpho-

logie (*morph.*). Pour ces deux types d'erreurs, une forte réduction de leur nombre peut être observée entre `moses` et `reranker`, qui démontre l'impact globalement positif des modèles complexes utilisés par `reranker`. Une réduction du nombre d'erreurs d'ordre comparable est également observée entre `reranker` et `rewriter`, démontrant ici la capacité de `rewriter` à mieux exploiter les modèles complexes que ne le fait `reranker`. Enfin, `GOS` réduit à nouveau le nombre de telles erreurs par rapport à `rewriter`. Ce dernier résultat suggère donc que `rewriter` a le potentiel d'améliorer plus fortement encore la qualité des traductions si celui-ci disposait d'une meilleure fonction de score ou de mesures de confiance au niveau des segments performantes comme c'est le cas pour notre configuration semi-oracle (voir section 4.2.3.3).

Les autres types d'erreurs semblent moins informatifs ici. Il ne semble pas exister de différences significatives sur le nombre d'erreurs de style (*style*), même pour `GOS` qui ne parvient pas à en corriger. Concernant l'ordre des mots (*ordre*), `reranker` réduit le nombre d'erreurs, mais `rewriter` ne le réduit pas davantage. Ce résultat n'est pas très surprenant dans la mesure où `rewriter` ne dispose que de peu d'opérations susceptibles de modifier significativement l'ordre des mots, l'opération `move` n'étant en particulier pas activée. `reranker` et `rewriter` font tous les deux décroître légèrement le nombre de mots cible surnuméraires (*extra*) dans la traduction produite par `moses`, alors qu'au contraire `GOS` augmente ce nombre de façon artificielle afin de limiter l'importance de la valeur de pénalité de concision de la mesure BLEU obtenue lorsqu'une hypothèse de traduction est plus courte que la traduction de référence.

	extra	*manquant*	*incorrecte*				*inconnu*	
	mot	mot	désamb.	morph.	style	ordre	mot	**tous**
`moses`	11	**1**	57	91	13	31	10	214
`reranker`	5	3	47	73	**11**	**19**	10	168
`rewriter opti`	**4**	4	40	55	12	**19**	10	144
`GOS conf10k`	19	2	**26**	**44**	14	22	10	137

TABLE 4.12 – Résultats de l'analyse manuelle des erreurs sur les 70 premières phrases du corpus de test `médical` (en→fr). La configuration `GOS` correspond à une recherche locale oracle effectuée avec une table de réécriture `conf10k` sur la meilleure hypothèse donnée par `reranker`.

4.2.3.5 Une réécriture très locale et parfois trop limitée

Nos expériences décrites dans ce chapitre ont mis en évidence qu'il était possible d'améliorer une traduction automatique à l'aide d'un algorithme de recherche locale, si celle-ci dispose d'une fonction de score meilleure que celle du décodeur utilisée pour générer la traduction. Nos expériences oracle révèlent un fort potentiel d'amélioration avec un tel système de réécriture, laissant penser que des fonctions de score à nouveau améliorées,

par exemple par l'ajout de nouveaux modèles complexes performants, pourrait améliorer davantage la traduction réécrite.

Cependant, il convient de souligner les limites inhérentes à une telle approche de réécriture. Tout d'abord, `rewriter` utilise un algorithme glouton, choisissant itérativement la meilleure solution, celle qui maximise localement la fonction de score dans le voisinage immédiat de la traduction réécrite. Ce type de recherche possède le défaut de pouvoir terminer sur un maximum local de la fonction de score qui ne correspond pas à une solution optimale en général. Ce risque pourrait être partiellement limité par l'introduction d'opérations de réécriture *conjointes*. L'opération `bireplace`, telle que proposée par Langlais *et al.* (2007), peut être considérée comme une telle opération : deux opérations `replace` sont réalisées simultanément sur deux bisegments distincts. Le voisinage se trouve ainsi enrichi par des hypothèses de réécriture qu'il n'aurait peut être pas été possible de générer autrement lors d'une recherche gloutonne. Toutefois, les opérations conjointes ont l'inconvénient d'augmenter de façon très importante la taille de l'espace de recherche à parcourir.

Une autre limitation importante de notre recherche locale concerne sa dépendance à la segmentation de l'amorce utilisée. La réécriture étant réalisée au niveau des bisegments, la recherche de meilleures traductions se trouve fortement influencée par la segmentation initiale produite par `moses`. Une solution consisterait à utiliser une recherche par faisceau sur plusieurs amorces adoptant une segmentation différente, mais celle-ci s'avère très vite coûteuse pour des tailles de faisceau même modérées (voir section 4.2.3.1). Une réécriture au niveau des tokens, et non plus au niveau des segments, pourrait également être envisagée, mais cela empêcherait alors l'exploitation de modèles performants portant sur les segments dans la fonction de score (notamment, les probabilités $p(e|f)$ et $p(f|e)$ de la table de traduction).

Finalement, il faut également rappeler que la table de réécriture utilisée, qui permettra une recherche d'autant plus efficace qu'elle est petite, est le résultat d'un filtrage très important de la table de traduction qui limite très fortement le nombre et la nature des hypothèses pouvant être obtenues par réécriture. En conséquence, les voisinages générés ne correspondent qu'à un infime sous-ensemble de l'espace de recherche du décodeur initial.

Résumé

Ce chapitre nous a permis de mettre en évidence l'importance du potentiel d'amélioration des traductions en utilisant un algorithme de recherche locale, guidée par une métrique d'évaluation de la traduction, pour réécrire une traduction produite par un système de TAS. Les résultats de cette recherche locale oracle nous ont donc confirmé qu'une meilleure traduction est effectivement atteignable au moyen d'une telle approche si une meilleure fonction de score que celle du décodeur était disponible. En dehors de cette fonction de score, les améliorations atteignables dépendent de la qualité de la traduction amorce produite par le décodeur, de la langue cible, de la taille des données d'entraînement utilisées par le décodeur mais aussi d'un éventuel filtrage de la table de traduction utilisée pour réécrire la traduction amorce.

À la suite de ces observations, nous avons donc utilisé pour guider la recherche locale une fonction de score utilisant les mêmes modèles que le décodeur auxquels ont été ajoutés des modèles complexes. L'ensemble des modèles utilisés s'avère identique à celui que nous avons utilisé en reclassement d'hypothèses. La meilleure hypothèse donnée par le système de reclassement a alors pu être améliorée par `rewriter`. Pour cela, la fonction de score de `rewriter` nécessite d'avoir des poids optimisés sur des exemples de réécriture produits sur un corpus de développement. De plus, la table de réécriture, produite à partir de la table de traduction du décodeur, doit être filtrée de façon à réduire la taille des voisinages générés tout en conservant des bisegments utiles à l'amélioration de la traduction. À ce titre, nous avons pu montrer qu'une table de réécriture contenant les bisegments utilisés dans les n meilleures hypothèses produites par le décodeur représente un bon compromis permettant à `rewriter` d'améliorer la traduction tout en évaluant des voisinages de petites tailles.

L'analyse de la performance de `rewriter` en fonction de la qualité initiale de la traduction amorce réécrite a mis en évidence que plus la traduction initiale est de mauvaise qualité et plus `reranker` puis `rewriter` amélioreront chacun la traduction. Cette observation suggère l'entraînement de différents `rewriter` spécialisés pour la réécriture de traduction de différents niveaux de qualité. Le choix du `rewriter` à utiliser ensuite lors de l'évaluation pourrait se faire *via* un classifieur, utilisant par exemple des estimations de qualité calculées au niveau phrastique (Specia *et al.*, 2013).

En complément, des expériences semi-oracle qui figent les segments de l'hypothèse de traduction présent dans la traduction de référence ont également montré que disposer d'une mesure de confiance parfaite au niveau des segments pouvait permettre à `rewriter` d'améliorer davantage la traduction. Inclure dans `rewriter` la possibilité d'évaluer automatiquement si un bisegment doit être modifié, avec par exemple des estimations de confiance calculées au niveau des tokens ainsi qu'une indication du type d'opération de réécriture à effectuer (Bach *et al.*, 2011), semble une piste prometteuse pour améliorer davantage la traduction. De plus, cela permettrait de réduire la taille de l'espace de recherche pour compenser par exemple une augmentation de la taille du faisceau de re-

cherche qui donne l'opportunité à `rewriter` de trouver une traduction encore meilleure mais au prix d'une forte augmentation des temps de calcul.

Néanmoins, nous avons également pu constater les limites d'une telle approche de réécriture qui ne parcourt qu'un espace de recherche très petit par rapport à celui parcouru par le décodeur. Si cette petite taille nous permet de calculer les scores de modèles coûteux pour évaluer les hypothèses contenues dans les voisinages produits, elle est en revanche très restrictive et `rewriter` demeure confiné à n'explorer qu'un espace de recherche dans le voisinage des hypothèses de traduction que le décodeur a préféré. Une telle exploration aura de fait beaucoup de difficultés à permettre la remise en cause complète des choix de traduction effectués par le décodeur. `rewriter` ne peut en effet effectuer que des réécritures très locales du fait du caractère glouton de l'algorithme qu'il utilise associé à des opérations de réécriture n'opérant qu'au niveau des bisegments.

Afin de tenter de corriger ces limitations, le chapitre 5 propose une nouvelle approche reparcourant l'espace de recherche par l'intermédiaire d'un redécodage, guidé par des modèles complexes, pour regénérer une nouvelle traduction.

La Figure 4.11 résume le fonctionnement d'un système de TAS utilisant une passe de réécriture fondée sur un algorithme de recherche locale guidé par des modèles complexes.

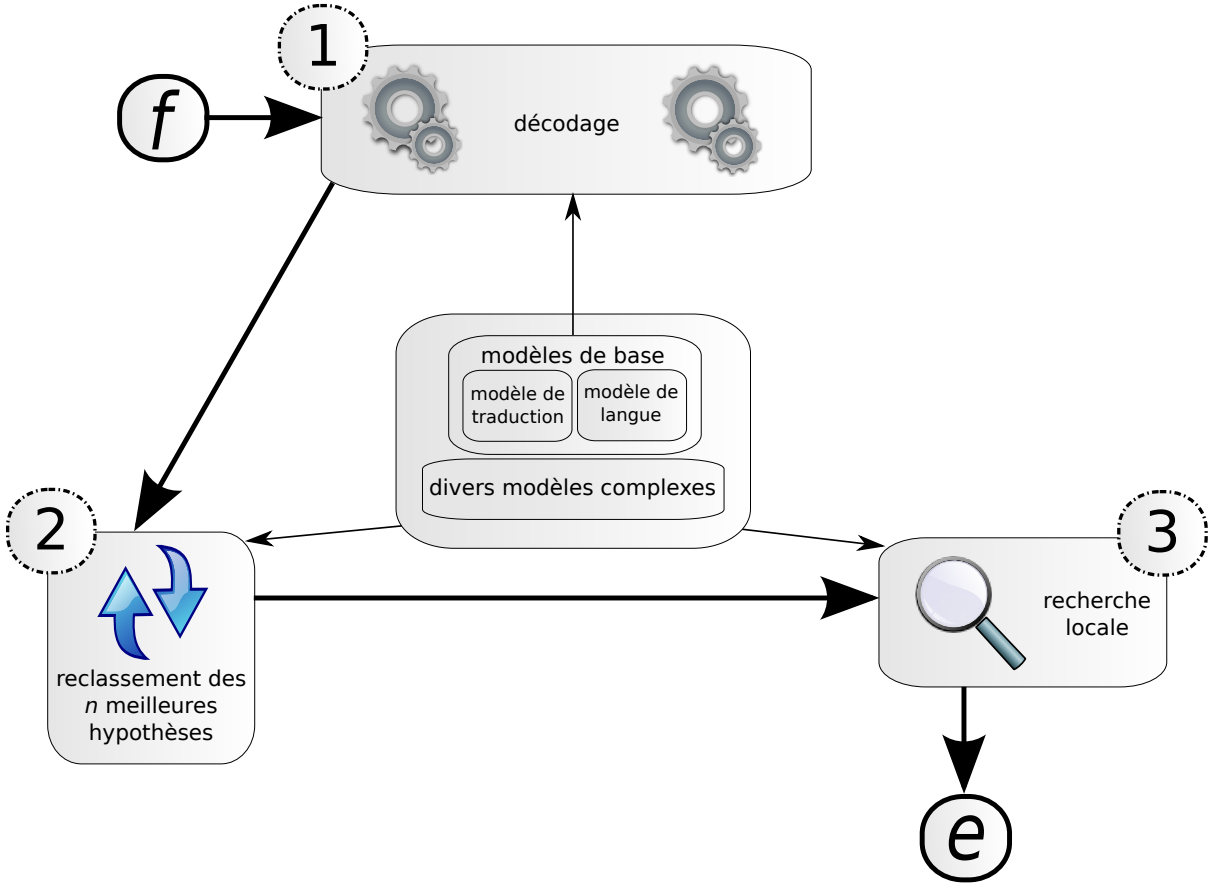


FIGURE 4.11 – Résumé du fonctionnement de système de TAS impliquant un système de réécriture. Une phrase source f est décodée en utilisant une fonction de score impliquant uniquement les modèles de base du décodeur (1). Le décodeur produit la liste des n meilleures hypothèses qui est reclassée à l'aide d'une nouvelle fonction de score associant modèles de base et de nouveaux modèles complexes non utilisables au cours du décodage (2). Un système de réécriture basé sur un algorithme de recherche locale va réécrire la meilleure hypothèse obtenue en étant guidé par une nouvelle fonction de score utilisant les mêmes modèles que ceux exploités lors de l'étape reclassement (3). Cette recherche locale produit la traduction e de f.

5

Un décodage mieux informé guidé par des modèles complexes

Sommaire

5.1 Un décodage à passes multiples **89**
5.1.1 Guidage du décodeur par le résultat d'un reclassement d'hypothèses 89
5.1.2 Exploitation des informations nouvelles d'un reclassement par partitionnement des tables de traduction 90
5.1.3 Fonctionnement global du décodage à passes multiples 92

5.2 Expériences **93**
5.2.1 Systèmes de référence 93
5.2.2 Résultats 95

5.3 Analyse **99**
5.3.1 Des listes d'hypothèses plus diversifiées 99

5.4 Combinaison du décodage à passes multiples et de la recherche locale **104**
5.4.1 Ajout d'une nouvelle étape de réécriture après chaque décodage 105
5.4.2 Résultats et analyse 107

5.5 Analyse des différences entre les traductions produites par les systèmes **108**

Résumé **113**

La section 4.2.3.5 a mis en évidence les limites d'un système de réécriture fondé sur un algorithme de recherche locale. Malgré les améliorations de la qualité des traductions obtenues par une telle approche, l'espace de recherche parcouru ne correspond qu'à une très petite partie de celui parcouru par le décodeur initial, ce qui fait qu'un grand nombre d'hypothèses de traduction ne sont pas considérées. Un nouveau parcours de l'espace de recherche à l'aide de la fonction de score mieux informée que nous avions utilisée pour guider le système de réécriture serait intéressant pour atteindre et évaluer ces hypothèses, mais pour les raisons évoquées dans la section 3.2 cela n'est pas envisageable. Un compromis serait alors de reparcourir l'espace de recherche du décodeur avec une fonction de score qui utiliserait des modèles exploitables au cours du décodage mais construits à partir d'informations riches qu'il n'était pas possible d'utiliser au cours d'un premier décodage. Le décodeur serait donc guidé pour produire un type de listes de n meilleures hypothèses plus adaptées au reclassement à l'aide de modèles complexes que les listes produites avec la fonction de score standard du décodeur. Chen *et al.* (2008b,a, 2010) ont par exemple proposé une méthode de regénération de listes d'hypothèses, que nous avions décrite dans la section 3.4, qui permet l'exploitation au cours d'un redécodage d'informations acquises à l'issue d'un premier décodage. Le redécodage effectué avec ces nouvelles informations permet *de facto* le guidage du décodeur vers la partie de l'espace de recherche préférée lors du décodage précédent pour en permettre une réexploration plus fine en générant des listes de n meilleures hypothèses plus représentative de cette partie de l'espace de recherche.

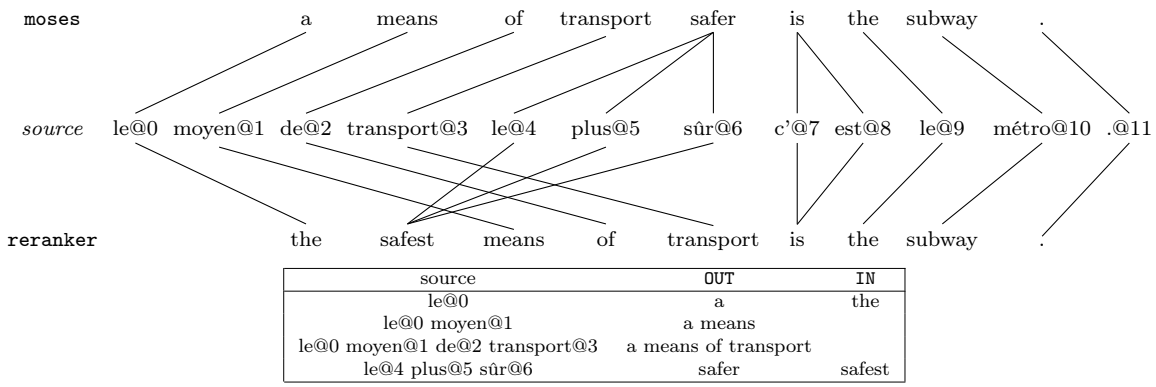
Ce chapitre présente un système adoptant une stratégie similaire ayant pour objectif de guider un décodeur à l'aide de modèles complexes afin d'obtenir de meilleures traductions. Dans une première section (5.1), nous décrirons ce nouveau : celui-ci repose sur un décodage à passes multiples et sur un partitionnement des tables de traduction utilisées par le décodeur. Dans la section 5.2, nous donnerons les résultats obtenus avec un tel système sur les tâches `news` et `médical`. Leur analyse sera effectuée dans la section 5.3 où nous verrons notamment les particularités des listes de n meilleures hypothèses produites par un tel système ainsi que leur impact lors de leur utilisation pour apprendre un système de reclassement d'hypothèses. Dans la section 5.4, nous présentons et analysons l'impact de l'introduction d'une étape de recherche locale dans ce décodage à passes multiples. Enfin, dans la section 5.5, nous présentons une analyse des différences entre les traductions produites par les systèmes que nous avons proposés.

5.1 Un décodage à passes multiples

5.1.1 Guidage du décodeur par le résultat d'un reclassement d'hypothèses

Nous avons montré dans la section 3.3.1 qu'il est possible d'obtenir une amélioration globale et significative de la qualité de traduction à l'aide de modèles complexes en reclassant les meilleures hypothèses produites par le décodeur. Dans la section 3.3.2.2, nous avions présenté des résultats oracle calculés sur les n meilleures hypothèses produites par `moses` avant et après reclassement par `reranker` (voir la Figure 3.2). Nous avions alors observé une amélioration sensible dans le classement des hypothèses après reclassement avec les modèles complexes utilisés. Les différences existantes entre la liste ordonnée d'hypothèses du décodeur et celle obtenue par reclassement correspondent probablement à des informations utiles en vue d'améliorer la qualité de traduction.

Considérons en particulier les différences qui existent entre la meilleure hypothèse trouvée par `moses` et celle trouvée par `reranker`. Ces différences sont le résultat de la remise en cause par `reranker` des préférences du décodeur. De manière simplifiée, si la meilleure hypothèse proposée par `moses` contient un token particulier A qui n'est pas présent dans la meilleure hypothèse de `reranker`, mais qu'inversement un token B n'est présent que dans la meilleure hypothèse de `reranker`, cela suggère que le décodeur aurait pu bénéficier du fait de privilégier des hypothèses contenant le token B et ne contenant pas le token A. Il est ainsi possible d'apprendre un modèle, propre à chaque phrase à traduire, sur la base des informations *a posteriori* fournies par le décodeur et le système de reclassement, et de l'utiliser lors d'un nouveau décodage qui exploiterait ainsi de façon indirecte la connaissance exprimée par les modèles complexes utilisés lors du reclassement. Cette idée est relativement proche des travaux de regénération d'hypothèses effectués par Chen *et al.* (2008b,a) (voir section 3.4) : le décodeur construit itérativement des espaces de recherche à l'aide de modèles extraits des n meilleures hypothèses produites à l'itération précédente. L'intention y est bien également d'exploiter des informations disponibles *a posteriori* afin de mieux guider ensuite une nouvelle tentative de construction d'une meilleure hypothèse par le décodeur. Si, dans les travaux de Chen *et al.* (2008b,a), un reclassement d'hypothèses à l'aide de modèles complexes est bien effectué, celui-ci n'est effectué qu'une seule fois à la fin des itérations de décodage sur l'ensemble des meilleures hypothèses obtenues. Le résultat du reclassement n'est donc pas exploité pour guider le décodeur. Il est intéressant de noter que le reclassement effectué produit une traduction de meilleure qualité qu'un reclassement effectué sur les seules meilleures hypothèses obtenues par le premier décodage.



source	OUT	IN
le@0	a	the
le@0 moyen@1	a means	
le@0 moyen@1 de@2 transport@3	a means of transport	
le@4 plus@5 sûr@6	safer	safest

FIGURE 5.1 – Exemple d'extraction des tables de traduction `IN` et `OUT` à partir des n-grammes qui diffèrent entre les meilleures hypothèses obtenues par `moses` et `reranker`.

Le système que nous présentons dans la suite de ce chapitre utilise au contraire le résultat du reclassement d'hypothèses produit par un décodage après chaque itération d'un décodage *à passes multiples*. Les modèles extraits du résultat du reclassement des hypothèses après chaque itération de décodage sont présentés dans la section 5.1.2 et le fonctionnement global du système ainsi que son entraînement sont présentés dans la section 5.1.3.

5.1.2 Exploitation des informations nouvelles d'un reclassement par partitionnement des tables de traduction

5.1.2.1 Partitionnement des tables par identification des bisegments à privilégier ou à pénaliser

Dans une configuration classique, l'optimisation d'un système de traduction s'effectue de façon globale en considérant tous les bisegments d'une unique table de traduction pour laquelle sera calculé un unique ensemble de poids optimisés. Les différences marquées qui existent entre la qualité de traduction d'un décodage classique et celle d'un décodage oracle laisse supposer que le calcul du score pour certains bisegments est inadapté. Séparer les bisegments d'une table de traduction en plusieurs tables peut, si ce partitionnement est fondé, permettre une optimisation améliorée pour les bisegments de chaque nouvelle table ainsi construite. Le principal défi est alors de déterminer quels sont les bisegments qui doivent être isolés des autres. Nous proposons de fonder cette décision sur le résultat d'un reclassement qui exploite une fonction de score qui est mieux informée que celle utilisée par le décodeur.

La fonction de score de `reranker` étant meilleure que celle de `moses`, on peut supposer que les bisegments présents uniquement dans la meilleure hypothèse de `moses` ne devraient pas être utilisés dans la meilleure traduction pouvant être générée. De la même manière, on peut supposer que les bisegments présents uniquement dans la meilleure hypothèse de `reranker` auraient dû être davantage privilégiés par `moses`. Si les deux en-

sembles de bisegments que nous venons de décrire ont été évalués différemment par `moses` et `reranker`, la possibilité de leur attribuer un jeu de poids différents pourrait permettre de mieux les évaluer lors d'un nouveau parcours de l'espace de recherche par `moses`.

Le succès de cette approche repose sur la cohérence des bisegments isolés dans chaque table, et le fait qu'ils doivent majoritairement et à juste titre être plutôt utilisés, ou non, pour construire une bonne hypothèse de traduction. Elle laisse toutefois une certaine flexibilité à la procédure d'optimisation, qui peut ne pas systématiquement favoriser, par exemple, les bisegments de la table issue de la meilleure hypothèse de `reranker`. En effet, malgré une amélioration globale de la traduction avec `reranker` par rapport à `moses`, nous avons constaté que la meilleure hypothèse de `reranker` possède occasionnellement un score *s*BLEU moins élevé que celle de `moses`, et donc privilégier de façon *dure* ces bisegments serait inapproprié. [1]

Pour effectuer ce partitionnement, nous comparons donc les meilleures hypothèses obtenues par `moses` et `reranker` et identifions les n-grammes qui diffèrent entre les deux. En utilisant les alignements au niveau des mots fournis par le décodeur, nous extrayons les bisegments qui sont *compatibles* avec ces n-grammes [2] et nous les répartissons dans deux nouvelles tables de traduction qui contiennent respectivement :

— `IN` : les bisegments **présents** dans la meilleure hypothèse de `reranker` mais **absents** de la meilleure hypothèse de `moses`.

— `OUT` : les bisegments **absents** de la meilleure hypothèse de `reranker` mais **présents** dans la meilleure hypothèse de `moses`.

Nous supprimons par ailleurs les bisegments ainsi extraits de la table de traduction principale. L'exploitation de ces nouvelles tables de traduction lors d'un nouveau décodage est détaillée dans la section 5.1.3.

5.1.2.2 Traduction spécialisée pour chaque token

Jusqu'à présent, nous n'avons évoqué que la situation *globale* des bisegments tels qu'ils apparaissent dans une table de traduction. Or un token source peut apparaître plusieurs fois dans un document ou même une phrase à traduire. Puisque le partitionnement décrit dans la section précédente 5.1.2.1 est réalisé phrase par phrase, il est nécessaire de rendre unique chaque token à traduire afin qu'il puisse bénéficier, ou non, des adaptations que nous proposons. En effet, `reranker` et `moses` peuvent traduire un même mot de la même façon, tout comme ils peuvent le traduire de façon différente. Prenons l'exemple du mot

1. Par exemple, dans les expériences de la section 3.3.2.2, `reranker` produit pour la tâche `médical` en→fr une traduction dont le score *s*BLEU est meilleur que le score obtenu par `moses` pour 59,3% des phrases, mais moins bon que celui de `moses` pour 24,3% des phrases. De plus, les phrases mieux traduites par `reranker` possèdent également des traductions qui sont localement moins bonnes que celles de `moses`.

2. En TAS fondée sur les segments, les bisegments utilisés ont typiquement une longueur maximale en nombre de tokens, le plus souvent 6 pour la paire de langues anglais-français, ce qui correspond donc ici à la valeur maximale de n.

anglais très courant « a » qui apparaîtrait deux fois dans une même phrase source en anglais, dont la première occurrence serait traduite par le mot français « un » par les deux systèmes, mais dont la deuxième occurrence serait traduite par « un » par `moses` et « une » par `reranker`. Il y aurait donc un conflit entre `reranker` et `moses` sur la deuxième occurrence du mot « a », et au contraire accord sur la première. Ainsi, seule une différenciation de chaque token source peut nous permettre de prendre en compte un tel désaccord entre ces deux systèmes.

Pour cela, de façon identique à la technique proposée par Stroppa *et al.* (2007); Gimpel et Smith (2008), un identifiant unique est associé à chaque token : il s'agit simplement de sa position dans le corpus (par ex. « metro@10 »). En pratique, donc, les tokens des segments source de la table de traduction sont également *localisés* de la sorte, et chaque bisegment se trouve dupliqué lorsque nécessaire afin de couvrir chaque portion du texte à traduire avec des tokens localisés. Il devient donc possible pour un même token ou segment source d'avoir par exemple des occurrences à traduire qui apparaissent indépendamment dans nos tables de traduction `IN`, `OUT`, ou principale. [3]

Un exemple d'extraction des tables `IN` et `OUT`, avec des tokens localisés pour les segments en langue source, est présenté par la Figure 5.1.

5.1.3 Fonctionnement global du décodage à passes multiples

5.1.3.1 Accumulation des modèles et des hypothèses produites

Le partitionnement décrit dans la section 5.1.2.1 peut être effectué itérativement après chaque décodage suivi d'une passe de `reranker` sur la nouvelle liste d'hypothèses produites. Les différences entre les meilleures hypothèses de `moses` et `reranker` sont à chaque fois extraites afin de construire de nouvelles tables de traduction propres à chaque itération [4]. Le reclassement des hypothèses après chaque itération de décodage est effectué sur la concaténation des listes des n meilleures hypothèses produites par les itérations précédentes. De cette façon, ce reclassement s'effectue sur des listes qui sont potentiellement de plus en plus *diversifiées*, ce qui peut entraîner une amélioration de l'entraînement du système (Chatterjee et Cancedda, 2010; Gimpel *et al.*, 2013). Toutefois, pour que l'entraînement (ainsi que l'évaluation) de `reranker` soit possible sur un tel ensemble de listes, chaque hypothèse doit être associée à un ensemble commun de modèles. Or au fur et à mesure des itérations de nouveaux modèles, correspondant à ceux introduits par les tables `IN` et `OUT`, sont ajoutés. Afin de conserver un ensemble commun de modèles, ces modèles

3. L'adaptation que nous réalisons est donc locale, et en l'état ne permet donc pas une propagation de préférences au niveau d'un discours (voir par exemple les travaux de Hardmeier *et al.* (2012)).

4. De cette façon, si les deux types de tables de traduction, `IN` et `OUT`, sont extraites à chaque itération, trois itérations produiront six nouvelles tables de traduction en plus de la table originelle dont les bisegments extraits ont été supprimés. Il faut noter qu'il est possible qu'un bisegment localisé apparaisse dans plusieurs tables de traduction après au moins deux itérations.

IN et OUT sont donc inutilisés lors du reclassement des hypothèses par reranker.[5]

5.1.3.2 Optimisation du système

Les itérations du décodage à passes multiples sont effectuées jusqu'à ce que plus aucune amélioration du score BLEU ne soit obtenue par reranker sur un corpus de développement. Ce critère a été choisi afin d'éviter d'avoir à effectuer un trop grand nombre d'itérations tout en permettant une amélioration de la traduction. Lors de l'entraînement, les poids des modèles utilisés par le décodeur sont réoptimisés pour chaque itération afin de prendre en compte les nouvelles tables de traduction extraites ainsi que mettre à jour les poids des modèles précédents pour en tenir compte. Le système s'avère donc relativement coûteux à entraîner puisqu'il nécessite une réoptimisation complète du décodeur pour chaque itération. reranker doit lui aussi être entraîné pour chaque itération : ceci s'effectue sur le même corpus de développement en utilisant la concaténation des listes de n meilleures hypothèses produites par le décodeur au cours de l'ensemble des itérations précédentes pour augmenter la diversité des hypothèses. Lors de l'évaluation des performances du système, les poids obtenus après réoptimisation lors de la $i^{\text{ème}}$ itération sur le corpus de développement sont utilisés par le décodeur. De même, les poids obtenus lors de l'entraînement de reranker à la $i^{\text{ème}}$ itération sont utilisés sur le corpus d'évaluation à la $i^{\text{ème}}$ itération pour reclasser la concaténation des listes des n meilleures hypothèses produites par le décodeur. Le fonctionnement global de notre décodeur à passes multiples est présenté par la Figure 5.2.

5.2 Expériences

5.2.1 Systèmes de référence

Pour les expériences décrites dans ce chapitre, nous avons utilisé les tâches news et médical pour les deux directions de traduction fr→en et en→fr. Chaque itération effectuant un reclassement d'hypothèses qui implique le calcul de modèles complexes coûteux, nous nous sommes limité à n'utiliser que les 300 meilleures hypothèses produites par moses, contre 1000 qui avaient été utilisées par reranker dans les chapitres précédents.[6]

5. Nous avons pu constater empiriquement, dans des expériences non décrites dans ce manuscrit, qu'un système reranker entraîné et évalué sur la concaténation des listes d'hypothèses produites à chaque itération sans utiliser de modèles IN et OUT était légèrement plus performant qu'un système reranker entraîné et évalué sur la liste d'hypothèses produite lors de la dernière itération de décodage et en utilisant tous les modèles IN et OUT ajoutés.

6. La valeur de 300 a été choisie de façon empirique : des expériences effectuées sur le corpus de développement utilisé pour l'entraînement de reranker n'ont montré aucune amélioration des performances de reranker sur ce même corpus au-delà des 300 meilleures hypothèses produites par moses. Ce résultat se retrouve dans la lecture des résultats oracle de reclassements effectués sur des listes d'hypothèses de différentes tailles qui sont présentés dans la Figure 3.2. Nous pouvons en effet constater que la plupart des meilleures hypothèses, selon sBLEU, se situent dans les 300 meilleures hypothèses produites par moses parmi les 1 000 meilleures.

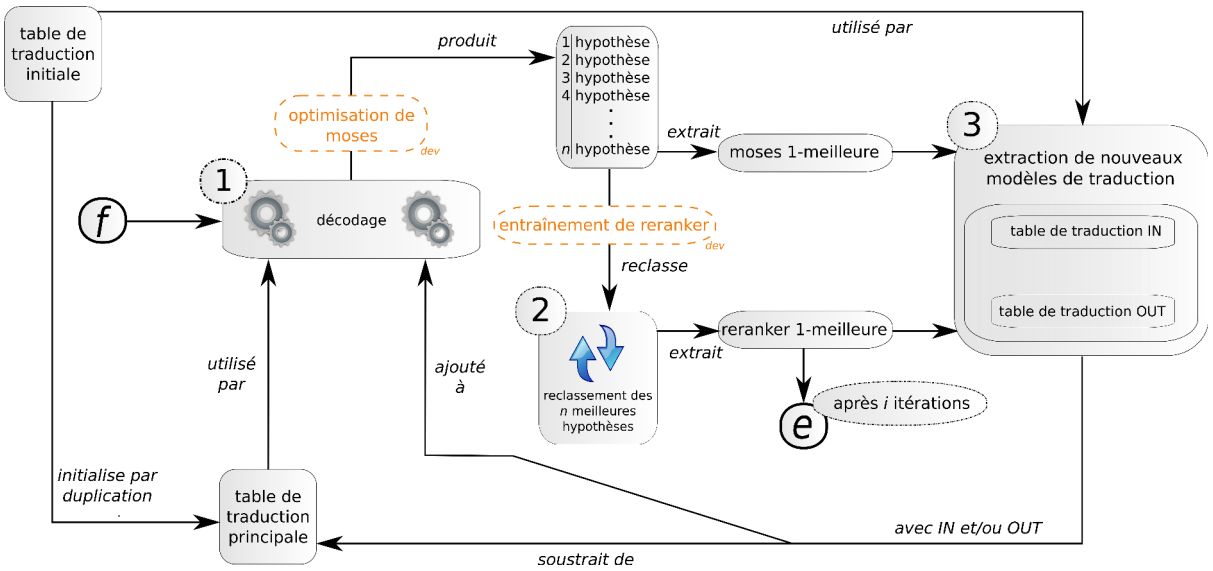


FIGURE 5.2 – Fonctionnement du décodeur à passes multiples. Une phrase source f est envoyée au décodeur (1) qui produit la liste des n meilleures hypothèses de traduction. Cette liste est reclassée à l'aide de modèles complexes (2). Sont ensuite extraits de nouvelles tables de traduction `IN` et `OUT` à partir de la comparaison entre la meilleure hypothèse du décodeur et celle donnée par le reclassement (3). Le contenu de ces nouvelles tables est retiré de la table de traduction principale (initialement complète). Un nouveau décodage est effectué avec ces nouvelles tables ainsi que la table principale mise à jour. Les étapes (1), (2) et (3) sont répétées pour un nombre d'itérations fixé à l'avance. Lors de la dernière itération, le reclassement d'hypothèses produit la traduction e de la phrase f. Les étapes entourées en pointillés (orange) sont effectuées avant l'évaluation sur un corpus de développement.

Les résultats d'expériences de contrôle où `reranker` reclasse les 300 meilleures hypothèses de `moses` sont présentés dans la Table 5.1. Ici, les modèles **IBM1**, **MosesNorm**, **SOUL**, **POSLM**, **Syntax**, **TagRatio** et **WPP**[7] (voir section 3.3.2.1) ont été utilisés en plus des modèles utilisés par le décodeur `moses`. Ces résultats confirment en évaluation qu'on obtient une performance similaire en utilisant les 300 ou 1 000 meilleures hypothèses produites par `moses`. Nous considérerons donc pour la suite de ce chapitre les systèmes `reranker` exploitant les 300 meilleures hypothèses de `moses` comme nos systèmes de référence.

Système	`médical`		`news`	
	fr→en	en→fr	fr→en	en→fr
`moses`	37,1	38,8	28,6	31,1
`reranker` 1 000 meilleures (*réf.*)	40,0	42,6	**30,0**	32,4
`reranker` 300 meilleures	**40,0**	**42,6**	$29{,}9_{(-0.1)}$	**32,4**

TABLE 5.1 – Résultats (scores BLEU) du reclassement par `reranker` pour les tâches `news` et `médical` sur des listes de 300 et 1 000 meilleures hypothèses produites par `moses`. L'entraînement et l'évaluation de `reranker` sont effectués sur des listes d'hypothèses de même taille. Les écarts inscrits entre parenthèses sont donnés par rapport à la configuration de `reranker`, dénotée (*réf.*), utilisant les 1 000 meilleures hypothèses produites par `moses`.

5.2.2 Résultats

Les résultats pour différentes configurations de partitionnement des tables de traduction sur les tâches `news` et `médical` sont présentés dans la Table 5.2. Pour chaque configuration, les résultats fournis sont ceux obtenus pour la dernière itération du décodage à passes multiples effectuée par `moses` et suivie par un reclassement par `reranker` sur l'accumulation des 300 meilleures hypothèses obtenues au fur et à mesure des itérations.

Nous pouvons observer que, pour toutes les configurations testées, la dernière itération de `moses`, avant le reclassement par `reranker` sur la concaténation des hypothèses collectées, améliore le score BLEU obtenu initialement par le système `moses` de référence. Ce résultat confirme l'intérêt d'extraire de la table de traduction principale des bisegments ayant été possiblement mal utilisés, au sens de `reranker`, permettant ainsi une meilleure adaptation des poids de la table de traduction principale et des nouvelles tables `IN` et `OUT`.

7. **PPP** n'est donc pas utilisé ici. Nous introduisons à la place **WPP**, plus utilisé en traduction automatique, et modélisant des informations directement au niveau des mots au contraire de **PPP** qui modélise des informations au niveau des bisegments et qui pouvait s'avérer plus approprié pour être utilisé dans un système tel que `rewriter` (voir le chapitre 4), réécrivant au niveau des bisegments.

Configuration		médical en→fr			médical fr→en			news en→fr			news fr→en		
		dev	test	# itér.	dev	test	# itér.	dev	test	# itér.	dev	test	# itér.
système de référence	moses	40,9	38,8	-	41,3	37,1	-	27,1	31,1	-	28,0	28,6	-
	reranker	43,7	42,6		44,2	40,0		28,5	32,4		29,0	29,9	
OUT	moses	43,4	41,6	7	43,0	38,8	5	27,8	31,9	4	28,5	29,2	2
	reranker	45,3	43,7		44,3	40,5		28,6	32,7		29,2	30,1	
IN	moses	45,4	43,8	7	43,6	39,9	4	28,4	32,4	4	28,6	29,3	2
	reranker	45,3	**44,3**		45,0	**40,9**		28,6	**33,2**		29,3	**30,5**	
IN et OUT	moses	44,2	41,5	7	42,8	38,7	5	28,2	32,2	2	28,8	29,1	3
	reranker	45,5	43,5		44,5	40,6		28,6	33,0		29,2	30,4	

TABLE 5.2 – Résultats (scores BLEU) pour différentes configurations de tables de traduction. OUT : configuration utilisant uniquement la table de traduction contenant les bisegments de la meilleure hypothèse moses absents de la meilleure hypothèse reranker. IN : configuration utilisant uniquement la table de traduction contenant les bisegments de la meilleure hypothèse reranker absents de la meilleure hypothèse moses. Pour toutes les configurations, la table de traduction principale est utilisée mais ne contient plus les bisegments déplacés dans l'une des nouvelles tables. Les résultats sont donnés pour la dernière itération du décodage à passes multiples pour moses et reranker.

De plus, comme illustré par la Figure 5.3, le décodage à passes multiples réduit rapidement l'écart entre les scores BLEU obtenus par `moses` et `reranker` sur la tâche `médical` en→fr. Ceci est particulièrement le cas pour la configuration n'utilisant qu'une table `IN`, pour laquelle `moses` obtient déjà un score supérieur au score initial de `reranker` après la troisième itération du décodage à passes multiples. En outre, on constate que la plupart des améliorations sont obtenues durant la première moitié des itérations du décodage à passes multiples ; en particulier, le score BLEU cesse d'augmenter après 4 itérations pour la configuration n'utilisant qu'une table `OUT`, voire même décroît pour la configuration utilisant les deux tables `IN` et `OUT`.

En dépit des gains significatifs obtenus par le décodage avec `moses`, celui-ci n'obtient pas, à l'issue de la dernière itération, un score BLEU meilleur que le score du système `reranker` de référence, à l'exception de la configuration n'utilisant qu'une table `IN` sur la tâche `médical` en→fr. Néanmoins, le décodage à passes multiples complet suivi d'un dernier reclassement par `reranker` améliore de façon notable le score BLEU relativement au système `reranker` de référence. Les améliorations apportées démontrent la capacité du décodage à passes multiples que nous avons proposé à aider le décodeur `moses` à produire des listes de n meilleures hypothèses qui s'avèrent plus adaptées pour l'expression des modèles complexes utilisés par `reranker`.

La configuration `IN`, qui isole à chaque itération l'ensemble des bisegments de la meilleure hypothèse de `reranker` qui n'apparaissent pas dans la meilleure hypothèse produite par `moses`, produit les meilleurs résultats pour toutes les tâches de traduction décrites. Les améliorations obtenues sont de 1,7 point BLEU par rapport au système `reranker` de référence sur `médical` en→fr, et de 0,8 point BLEU sur `news` en→fr. Elles sont de moindre ampleur pour la direction de traduction fr→en : 0,9 et 0,6 point BLEU respectivement pour les tâches `médical` et `news`.

La configuration `OUT`, qui isole à chaque itération l'ensemble des bisegments de la meilleure hypothèse de `moses` qui n'apparaissent pas dans la meilleure hypothèse identifiée par `reranker`, obtient des améliorations plus modestes que la configuration reposant sur `IN`. Ceci signifie que l'isolation des bisegments n'apparaissant pas dans la meilleure hypothèse obtenue par `reranker` mais qui apparaissant dans la meilleure hypothèse produite par `moses` semble moins utile pour produire de meilleures listes d'hypothèses à reclasser par `reranker`.

Combiner les deux tables `IN` et `OUT` permet d'obtenir une amélioration qui se situe entre les configurations utilisant les deux tables de traduction `IN` et `OUT` séparément. Ce dernier résultat peut sembler surprenant, ce système combinant plus de modèles nous pourrions le considérer comme mieux informé pour produire de nouvelles meilleures hypothèses. Pour tenter de mieux comprendre ces résultats, des analyses spécifiques seront effectuées dans la section 5.3.

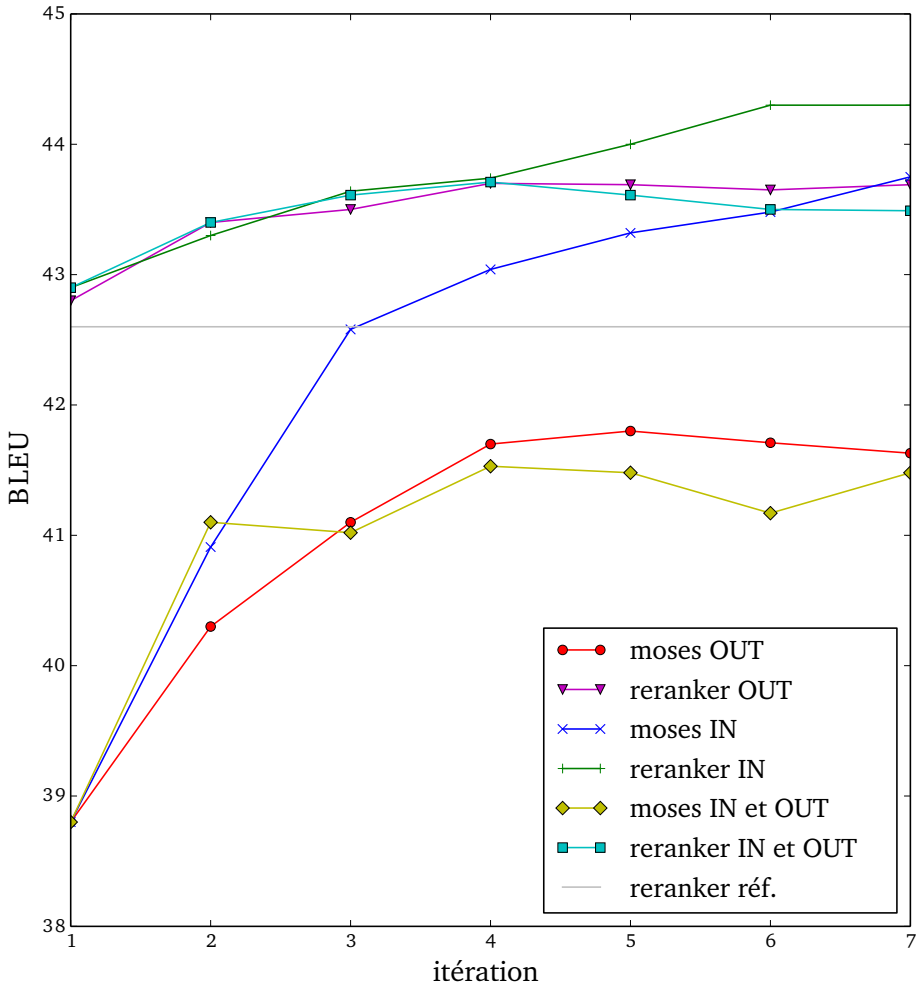


FIGURE 5.3 – Évolution du score BLEU en fonction des itérations du décodage à passes multiples pour toutes les configurations en évaluation sur la tâche `médical` en→fr.

Nous aurions également pu construire, de façon similaire aux tables `IN` et `OUT`, des modèles de langue `IN` et `OUT`, en apprenant par exemple ces modèles sur les n-grammes absents/présents de la meilleure hypothèse de `reranker`/`moses`. Cependant, nous avons pu constater empiriquement que l'ajout de tels modèles n'est pas bénéfique à notre approche. En effet, ajouter ces nouveaux modèles de langue à la fonction de score du décodeur apporte des informations qui sont trop fortement utilisée par le décodeur pour reproduire celle identifiée par `reranker`[8]. Une telle situation nuit donc à la génération de listes de n meilleures hypothèses contenant de la diversité utile. Avec de tels modèles de langue, des listes de 1 000 meilleures hypothèses `distinct`[9] ne contiennent plus qu'un nombre limité d'hypothèses (quelques dizaines en moyenne) qui diffèrent très peu entre elles, réduisant ainsi fortement le potentiel d'amélioration par un reclassement d'hypothèses effectué sur ce type de listes. Par conséquent, le score BLEU de l'oracle calculé sur ces 1 000 meilleures hypothèses est très proche, sinon identique, du score BLEU de la traduction trouvée par `reranker`.

5.3 Analyse

Dans cette section d'analyse, nous allons nous focaliser sur l'étude de la diversité des hypothèses produites par les décodages puis reclassées par `reranker`. L'étude sera faite dans un premier temps au niveau des hypothèses complètes dans la section 5.3.1.1, puis au niveau lexical dans la section 5.3.1.2. Enfin, nous verrons dans la section 5.3.1.3 l'impact d'une telle diversité sur l'entraînement de `reranker` lui-même. Comme dans le chapitre 4, nous nous appuierons principalement sur la tâche `médical` en→fr pour effectuer ces analyses.

5.3.1 Des listes d'hypothèses plus diversifiées

5.3.1.1 Analyse au niveau des hypothèses

Dans un premier temps, nous souhaitons évaluer la diversité et la qualité des listes d'hypothèses produites en comparant les configurations de décodage à passes multiples testées qui ont produit ces listes.

La Figure 5.4 décrit des résultats oracle calculés sur la concaténation des listes de meilleures hypothèses de taille k jusqu'à k=1000[10] pour différentes configurations de reclassement par `reranker` sur la dernière itération de notre décodage à passes multiples.

8. La traduction obtenue avec `reranker` ayant en moyenne un meilleur score BLEU que la meilleure traduction trouvée par le décodeur, durant la phase d'optimisation `KB-MIRA` affectera typiquement un poids positif très fort au modèle de langue `IN` et un poids négatif au modèle de langue `OUT`, guidant ainsi très facilement le décodeur vers la meilleure hypothèse de `reranker`.

9. générées donc à partir des 20 000 meilleures non-dédoublonnées

10. La valeur de 1 000 ayant été choisie ici arbitrairement, les listes d'hypothèses obtenues pour chaque phrase à la fin des décodages ont une taille variable : elles résultent de la concaténation des listes produites

On constate que les scores BLEU oracle croissent fortement avec le nombre d'hypothèses considérées. L'oracle calculé sur les 1 000 meilleures hypothèses pour la configuration n'utilisant qu'une table `IN` améliore de 2,2 points BLEU celui calculé sur les 1 000 meilleures hypothèses du système `reranker` de référence. Par ailleurs, les listes de k-meilleures hypothèses générées par `moses` obtiennent un score BLEU moyen [11] de plus en plus élevé en fonction du nombre d'hypothèses k. La Figure 5.5 montre également que la configuration utilisant les tables `IN` et `OUT` tend à générer des listes d'hypothèses avec un score BLEU moyen plus bas que pour les configurations utilisant ces deux tables séparément. Cette observation peut être une première piste d'explication pour la moins bonne performance de cette configuration de décodage à passes multiples par rapport à celle utilisant uniquement la table `IN` qui semble générer des hypothèses qui sont en moyenne de meilleure qualité.

Nous savons maintenant que la qualité des hypothèses produites par un décodage à passes multiples est globalement meilleure par rapport aux meilleures hypothèses produites par un premier décodage. Nous considérons à présent la diversité, potentiellement utile lors de l'entraînement de `reranker`, que ces listes contiennent.

La Figure 5.6 illustre la capacité du décodage à passes multiples à produire de nouvelles hypothèses de traduction qui étaient absentes des listes produites par les itérations précédentes. Quasiment tous les itérations produisent une liste de 300 hypothèses dont environ la moitié d'entre elles n'appartenaient pas à la liste d'hypothèses initialement produite par le système `moses` de référence pour les trois configurations de décodage à passes multiples testées. La part des nouvelles hypothèses produites, absentes des listes de l'ensemble des itérations précédentes, décroît sans surprise après plusieurs itérations pour la configuration n'utilisant qu'une table `IN`. En effet, dans cette configuration le décodage à passes multiples a tendance à renforcer les préférences exprimées par `reranker` en rang 1, encourageant ainsi `moses` à utiliser les bisegments préférés par `reranker`. Ceci ne se retrouve toutefois pas pour la configuration n'utilisant qu'une table `OUT`, pour laquelle la dernière itération parvient encore à générer une liste de 300 hypothèses dont 44,7% n'appartiennent pas aux listes d'hypothèses des itérations précédentes.

5.3.1.2 Analyse au niveau lexical

Après avoir considéré la diversité produite par notre approche au niveau des hypothèses complètes, nous allons maintenant nous concentrer sur le niveau des mots générés. Les résultats donnés par la Table 5.3 indiquent, pour chaque configuration, le pourcentage de nouveaux tokens en langue cible introduits par la liste d'hypothèses produites

par les décodages intermédiaires et sont dédoublonnées.

11. Afin d'obtenir le score BLEU moyen, la moyenne arithmétique des scores sBLEU des k meilleures hypothèses de traduction de chaque phrase est calculée. Pour chaque phrase, l'hypothèse ayant le score sBLEU le plus proche du sBLEU moyen est sélectionnée. Une fois qu'une hypothèse de traduction a été sélectionnée pour chaque phrase, un score BLEU standard est calculé sur l'ensemble des hypothèses sélectionnées. C'est ce score qui constitue le BLEU moyen présenté ici.

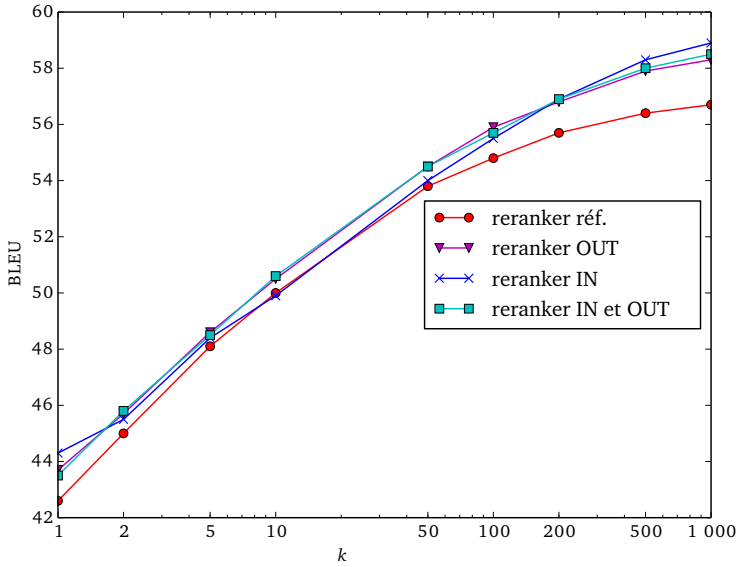


FIGURE 5.4 – Résultats oracle calculés sur les k meilleures hypothèses, c'est-à-dire la concaténation des n meilleures hypothèses produites par chaque décodage, obtenues lors du décodage à passes multiples sur la tâche `médical` en→fr.

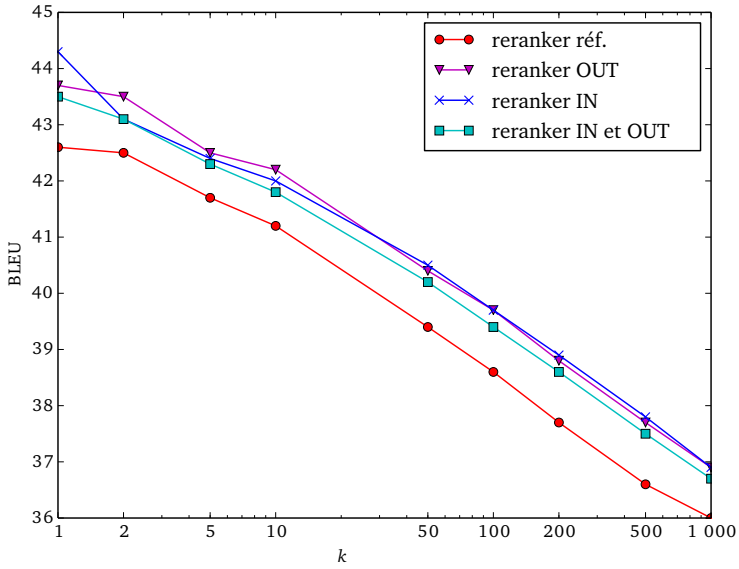


FIGURE 5.5 – Scores BLEU moyen calculés sur les k-meilleures hypothèses, c'est-à-dire la concaténation des n meilleures hypothèses produites par chaque décodage, obtenus lors du décodage à passes multiples sur la tâche `médical` en→fr.

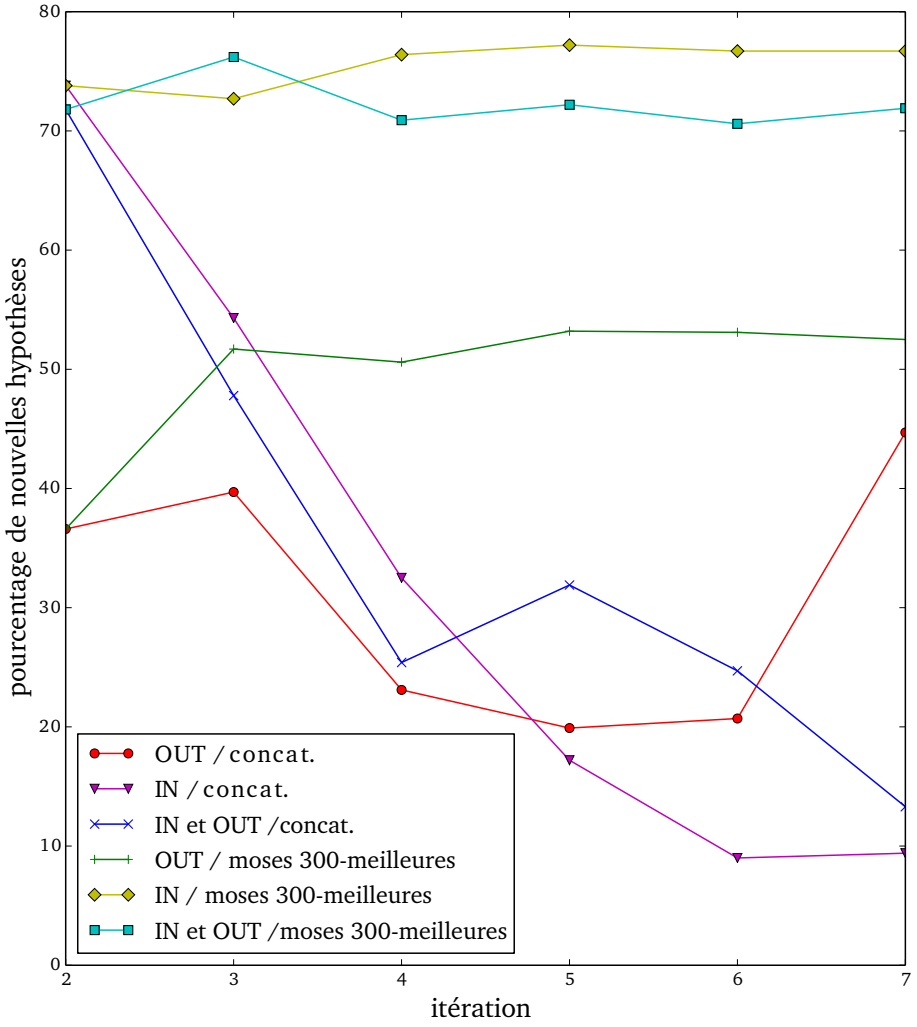


FIGURE 5.6 – Pourcentage de nouvelles hypothèses générées lors du décodage à passes multiples sur la tâche `médical` en→fr. `concat.` signifie que la liste des 300 meilleures hypothèses de l'itération courante est comparée à la concaténation des listes d'hypothèses produites par toutes les itérations précédentes au lieu d'être comparée aux 300 meilleures hypothèses initialement produites par le système `moses` de référence.

à la dernière itération et qui n'apparaissent pas dans les 300 meilleures hypothèses produites par le système `moses` de référence. Pour la tâche `news`, jusqu'à 33,2% des tokens de la liste produite à la dernière itération sont nouveaux. Pour la tâche `médical`, l'apparition de nouveaux tokens est bien plus modeste et se limite à 10,4% de nouveaux tokens. Pour cette même tâche, la configuration n'utilisant qu'une table `IN` produit en particulier une diversité lexicale deux fois plus grande que la configuration n'utilisant que la table `OUT`. Cette plus grande diversité peut expliquer pourquoi la configuration `IN` obtient de meilleurs scores BLEU. En effet, un plus grand nombre de traductions différentes sont présentes dans les listes reclassées par `reranker`, offrant au système plus de diversité pouvant améliorer les performances du reclassement d'hypothèses (Chatterjee et Cancedda, 2010; Gimpel *et al.*, 2013). La configuration `IN` renforce de cette façon les possibles meilleurs choix de traduction faits par `reranker` relativement à `moses` tout en permettant d'énumérer des alternatives de traduction pour les tokens source qui seraient par exemple plus difficiles à traduire. Cette diversité combinée à la qualité des hypothèses produites que nous avons étudiée dans la section précédente (5.3.1.1) semble donc être à l'origine de la meilleure performance de la configuration utilisant uniquement la table `IN` par rapport aux autres configurations.

Configuration	`médical`		`news`	
	en→fr	fr→en	en→fr	fr→en
`OUT`	6,0%	5,1%	26,4%	20,4%
`IN`	**9,9%**	**10,4%**	**33,2%**	**30,1%**
`IN et OUT`	9,0%	10,1%	29,0%	24,2%

TABLE 5.3 – Pourcentage de nouveaux tokens dans la liste d'hypothèses produites par la dernière itération du décodage à passes multiples qui n'appartiennent pas à la liste d'hypothèses produites par le système `moses` de référence.

5.3.1.3 Une meilleure diversité pour un reclassement d'hypothèses amélioré

Dans les sections précédentes 5.3.1.1 et 5.3.1.2, nous avons vu que le décodage à passes multiples a la capacité de produire des listes de meilleures hypothèses plus diversifiées et de meilleure qualité que la liste originellement produite par le système `moses` de référence. Les travaux de Gimpel *et al.* (2013) ont notamment montré l'importance de la diversité dans l'entraînement des systèmes de reclassement d'hypothèses. La Table 5.4 présente les résultats obtenus avec un système `reranker` entraîné en utilisant la concaténation des listes d'hypothèses produites par le décodage à passes multiples sur le corpus de développement et utilisées pour reclasser la liste des 300 meilleures hypothèses produites par un système `moses` standard. Différentes configurations sont également présentées à titre de comparaison.

Dans une première configuration, les 500 meilleures hypothèses produites par `moses` ont été mélangées avec 500 hypothèses extraites aléatoirement du treillis produit par

moses en suivant la procédure décrite par Chatterjee et Cancedda (2010)[12]. Ce mélange d'hypothèses est donc riche en diversité, la moitié de ces hypothèses ayant été extraites aléatoirement. Cependant, aucune amélioration n'a pu être observée en utilisant un tel ensemble d'hypothèses pour entraîner reranker relativement à un reclassement par reranker entraîné sur les 1 000 meilleures hypothèses produites par moses.

En revanche, utiliser l'ensemble des listes d'hypothèses produites au cours du décodage à passes multiples pour entraîner reranker permet d'obtenir de légères améliorations lors du reclassement des 1 000 meilleures hypothèses produites par moses, allant jusqu'à 0,3 point BLEU pour les systèmes médical et news en→fr en combinant les hypothèses produites par les 3 configurations de décodage à passes multiples OUT, IN, ainsi que la combinaison de IN et OUT (O+I+IO). Enfin, en utilisant cette configuration O+I+IO pour entraîner reranker et en reclassant la concaténation de toutes les listes d'hypothèses produites en évaluation par les 3 mêmes configurations, une amélioration supplémentaire a pu être obtenue. L'entraînement et l'évaluation sur le même type d'ensemble d'hypothèses O+I+IO permet ainsi à reranker d'obtenir une amélioration de 0,2 point BLEU pour médical et news en→fr par rapport au score obtenu à la dernière itération du décodage à passes multiples, culminant à une amélioration de respectivement 1,9 et 1,0 point BLEU relativement au système reranker de référence.

5.4 Combinaison du décodage à passes multiples et de la recherche locale

Dans le chapitre 4 nous avions introduit un système fondé sur un algorithme de recherche locale (rewriter), qui réécrit une hypothèse de traduction à l'aide d'un ensemble d'opérations de transformation. Ces opérations permettent la génération du voisinage de l'hypothèse dans lequel chaque hypothèse peut être évaluée avec une fonction de score exploitant des modèles complexes. Cette approche peut donc être considérée comme très différente de l'approche présentée dans le chapitre présent, qui elle regénère des hypothèses de traduction au moyen d'un décodage à passes multiples exploitant le résultat d'une passe de reclassement d'hypothèses. Dans cette section, nous allons donc étudier l'impact de l'ajout d'une passe supplémentaire effectuée par rewriter et dont le résultat sera directement utilisé pour guider le décodage. L'hypothèse faite ici est qu'étant donné que rewriter produit une meilleure traduction que reranker, utiliser le résultat de rewriter pour extraire de nouvelles tables de traduction, possiblement meilleures, pourrait mieux guider le décodeur. La section 5.4.1 décrit le fonctionnement de ce décodage à passes multiples intégrant un système de réécriture fondé sur un algorithme de recherche locale. La section 5.4.2 présente puis analyse ses résultats.

12. Pour effectuer cette extraction, l'option lattice-sample de moses a été utilisée.

Exemples d'entraînement	médical		news	
	en→fr	fr→en	en→fr	fr→en
reranker 300 meilleures (*réf.*)	42,6	40,0	32,4	29,9
reranker 1 000 meilleures	42,6	40,0	32,4	30,0 (+0.1)
reranker 500 meilleures+500-RS	42,6	40,0	32,4	29,9 (−0.1)
reranker OUT	42,8 (+0.2)	40,1 (+0.1)	32,6 (+0.2)	30,0 (+0.1)
reranker IN	42,9 (+0.3)	40,1 (+0.1)	32,6 (+0.2)	30,1 (+0.2)
reranker IN et OUT	42,9 (+0.3)	40,0	32,5 (+0.1)	29,9 (+0.0)
reranker O+I+IO	**42,9** (+0.3)	**40,1** (+0.1)	**32,7** (+0.3)	**30,1** (+0.2)
reranker O+I+IO évalué sur O+I+IO	**44,5** (+1.9)	**41,0** (+1.0)	**33,4** (+1.0)	**30,5** (+0.6)

TABLE 5.4 – Résultats (scores BLEU) obtenus par reranker pour différentes configurations d'entraînement. Le reclassement est effectué en évaluation sur les 300 meilleures hypothèses produites par moses, excepté pour la dernière ligne du tableau. La configuration O+I+IO utilise la concaténation des n meilleures hypothèses produites par moses à chaque itération du décodage à passes multiples de toutes les configurations (OUT (O), IN (I), IN et OUT (IO)). 500-meilleures+500-RS utilise elle la concaténation des 500 meilleures hypothèses produites par moses et de 500 hypothèses extraites aléatoirement (*Randomly Sampled* (RS)) du treillis produit par le décodeur en utilisant la procédure décrite par Chatterjee et Cancedda (2010). Les écarts inscrits entre parenthèses sont donnés par rapport à la configuration de reranker, dénotée (*réf.*), entraîné et évalué sur les 300 meilleures hypothèses produites par moses.

5.4.1 Ajout d'une nouvelle étape de réécriture après chaque décodage

Dans la section 5.2, nous avions constaté qu'un décodage à passes multiples utilisant uniquement la table IN produisait de meilleurs résultats que les configurations impliquant la table OUT. En conséquence, et tenant compte du fait que l'ajout d'une passe de recherche locale allonge significativement les temps de calcul, nous n'avons utilisé que des tables de type IN dans ces nouvelles expériences. Nous avons ici 3 types de tables IN que nous définissons comme suit :

— IN-reranker/moses : cette table est identique à la table IN que nous construisions dans la section 5.1.2, elle contient les bisegments **présents** dans la meilleure hypothèse de reranker mais **absents** de la meilleure hypothèse de moses.

— IN-rewriter/moses : cette table contient les bisegments **présents** dans la meilleure hypothèse de rewriter mais **absents** de la meilleure hypothèse de moses.

— IN-rewriter/reranker : cette table contient les bisegments **présents** dans la meilleure hypothèse de rewriter mais **absents** de la meilleure hypothèse de reranker.

Les étapes nécessaires à l'entraînement et l'évaluation de ce nouveau décodage à passes multiples, intégrant une passe de `rewriter`, sont les suivantes :

1. Optimisation de la fonction de score de `moses` avec `KB-MIRA` sur un corpus de développement.
2. Décodages du texte source du corpus de développement et d'évaluation avec les poids optimisés lors de l'étape précédente, pour produire les 300 meilleures hypothèses de traduction `distinct` pour chaque corpus.
3. Optimisation de la fonction de score de `reranker` avec `KB-MIRA` sur la concaténation des listes d'hypothèses produites à l'étape 2 et par les décodages du corpus de développement effectués lors des itérations précédentes.
4. Reclassement avec `reranker` de la concaténation des listes d'hypothèses produites par les décodages du corpus d'évaluation de l'étape 2 et des itérations précédentes.
5. Extraction de la table de réécriture `conf10k` sur les corpus de développement et d'évaluation et de la table `rpt10pef` (voir la section 4.2.3.1) sur le corpus de développement uniquement.
6. Production par une itération de `rewriter` des voisinages des 10 meilleures hypothèses de `reranker` avec la table `conf10k` et du voisinage de la meilleure hypothèse de `reranker` avec la table `rpt10pef`.
7. Optimisation de la fonction de score de `rewriter` avec `KB-MIRA` sur la concaténation des voisinages produits sur le corpus de développement à l'étape 6 et lors des itérations précédentes.
8. Réécriture de la meilleure hypothèse de `reranker` produite sur le corpus d'évaluation avec `rewriter` et sa fonction de score optimisée à l'étape précédente et la table de réécriture `conf10k` extraite à l'étape 5.
9. Extraction des tables de traduction `IN-reranker/moses`, `IN-rewriter/moses` et `IN-rewriter/reranker` en fonction des meilleures hypothèses de `moses`, `reranker` et `rewriter` obtenues respectivement aux étapes 2, 4 et 8.
10. Ajout des nouvelles tables aux décodages des itérations suivantes.
11. Répétition des étapes 1 à 10 aussi longtemps que le score BLEU obtenu par le reclassement avec la fonction de score de `rewriter` de la concaténation des voisinages produits sur le corpus de développement à l'étape 7 augmente.

Le système `rewriter` que nous utilisons est identique à celui décrit dans la section 4.2. Toutefois, il faut noter que, dans ce décodage à passes multiples, la fonction de score de `rewriter` est ici optimisée sur la concaténation des voisinages produits par `rewriter`, sur le corpus de développement, pour chaque itération. Nous concaténons les listes d'hypothèses produites à chaque itération pour l'optimisation de la fonction de score de `reranker`.

Avec l'extraction de la `IN-rewriter/moses` nous pouvons espérer isoler des bi-segments globalement meilleurs que ceux isolés par la table `IN-reranker/moses` que nous utilisions jusqu'à présent. Nous testerons donc une configuration de décodage à passes multiples qui utilise une table `IN-rewriter/moses` qui viendra remplacer la

table `IN-reranker/moses`. Nous pouvons également faire l'hypothèse que les tables `IN-reranker/moses` et `IN-rewriter/reranker` sont porteuses d'informations différentes qui peuvent venir accompagner utilement la table `IN-rewriter/moses` pour potentiellement guider le décodeur vers de meilleures traductions. Nous testerons donc également une deuxième configuration utilisant simultanément les trois tables `IN-reranker/moses`, `IN-rewriter/moses` et `IN-rewriter/reranker`. Les résultats de ces deux configurations sont décrits dans la section suivante.

5.4.2 Résultats et analyse

Les expériences ont été effectuées sur la tâche `news` en→fr. Les résultats sont présentés par la Table 5.5.

Dans cette expérience, le décodage à passes multiples est stoppé après seulement deux itérations du fait de l'absence d'amélioration du score BLEU calculé sur le corpus de développement lors d'une troisième itération. Comparé à un décodage à passes multiples effectué sans `rewriter`, cela représente deux itérations de moins, mais en contrepartie l'ajout de `rewriter` a pour conséquence une augmentation importante des temps de calcul nécessaires pour une itération [13].

Nous pouvons observer que nos deux nouvelles configurations, utilisant uniquement la table `IN-rewriter/moses` ou les trois tables `IN-reranker/moses`, `IN-rewriter/moses` et `IN-rewriter/reranker`, atteignent un score BLEU identique en évaluation de 33,4 points BLEU, soit 0,2 point BLEU de plus que la configuration de décodage à passes multiples de référence (dénotée *réf.* dans la Table 5.5) n'utilisant pas `rewriter`. Ce résultat est de plus atteint en seulement une itération contre 4 pour la configuration de référence *réf.*. L'ajout des tables `IN-reranker/moses` et `IN-rewriter/reranker` ne semble pas avoir d'impact sur le score BLEU qui reste identique en évaluation (avec une légère amélioration sur le corpus de développement de 0,1 point BLEU). En revanche, les traductions produites par ces deux nouvelles configurations sont différentes pour 19,3% des phrases (soit 580 phrases).

13. La phase la plus coûteuse est l'entraînement du système et notamment le calcul des modèles complexes sur les grands voisinages produits par une itération de `rewriter`. Sur les 2 525 phrases du corpus de développement de la tâche `news`, `rewriter` produit un total de 5,9 millions d'hypothèses. Le calcul des modèles complexes sur ces hypothèses dure ∼3 jours sur une machine utilisant 12 threads. Les modèles **SOUL** et **Syntax** sont de loin les plus coûteux à calculer. À cela vient principalement s'ajouter les temps de calcul nécessaires à l'optimisation de la fonction de score du décodeur (∼7 heures), la production des 300 meilleures hypothèses `distinct` (∼8 heures) puis le calcul des modèles complexes sur ces meilleures hypothèses (∼12 heures). En phase d'évaluation, en revanche, l'ajout d'une passe de recherche locale n'entraîne qu'une augmentation relativement faible des temps de calcul car, suivant les configurations, `rewriter` ne produit que de 300 000 à 400 000 hypothèses qui peuvent être évaluées en quelques heures avec nos modèles complexes.

Configuration		news en→fr dev	news en→fr test
Systèmes	moses	27,1	31,1
	reranker	28,5	32,4
	rewriter	28,6	33,0
multi-pass (4 itér.) (*réf.*)	moses	28,4	32,4
avec IN-reranker/moses (section 5.2.2)	reranker	28,6	33,2
multi-pass (+rewriter) (2 itér.)	moses	28,3	32,4
avec IN-rewriter/moses	reranker	28,7	33,0
	rewriter	**28,7**$_{(+0.1)}$	**33,4**$_{(+0.2)}$
multi-pass (2 itér.) (+rewriter)	moses	28,3	32,4
avec IN-reranker/moses IN-rewriter/moses IN-rewriter/reranker	reranker	28,6	33,0
	rewriter	**28,9**$_{(+0.2)}$	**33,4**$_{(+0.2)}$

TABLE 5.5 – Résultats (scores BLEU) de décodages à passes multiples (multi-pass), incluant une passe de recherche locale « (+rewriter) », pour différentes configurations de tables de traduction sur la tâche news en→fr. Les écarts inscrits entre parenthèses sont donnés par rapport au décodage à passes multiples (*réf.*) utilisant uniquement la table IN-reranker/moses.

Globalement, l'ajout d'une passe de recherche locale pour extraire de nouvelles tables de traduction semble permettre d'atteindre une meilleure traduction grâce notamment à la présence d'une ultime passe de rewriter en fin de séquence. Ce dernier produit une traduction meilleure de 0,4 point BLEU relativement à la traduction produite par rewriter avant redécodage. Nous pouvons également noter qu'utiliser les nouvelles tables IN au décodage permet à moses d'atteindre rapidement (en une itération) une traduction meilleure que celle produite par le système moses de référence, en atteignant un score BLEU similaire au système reranker de référence.

5.5 Analyse des différences entre les traductions produites par les systèmes

Nous avons dans ce manuscrit décrit des approches à l'état de l'art pour la traduction automatique statistique ainsi que des approches permettant d'exploiter efficacement des informations riches. Nous avons également introduit des approches qui, tel que montré par les métriques automatique standard, améliorent les performances des systèmes qui reposent sur les mêmes informations. Nous allons à présent analyser les différences entre les traductions produites par différents systèmes : moses (voir section 3.1), reranker (voir section 3.3.2), rewriter (voir section 4.2) et le décodage à passes multiples, effectuant une passe de reranker puis de rewriter à chaque itération, que nous appellerons multi-pass (+rewriter) (voir section 5.4). Cette analyse est effectuée sur la tâche news en→fr pour laquelle nous disposons des traductions ayant été générées par nos systèmes avec le même ensemble de modèles complexes. Des exemples de traductions produites par nos systèmes sur cette tâche sont donnés dans l'Annexe C.

Tout d'abord, la Figure 5.6 présente les écarts entre les scores sBLEU, ordonnés de façon croissante phrase à phrase, pour chaque couple de systèmes. La similarité globale entre les deux sorties est quant à elle mesurée avec BLEU (inter-BLEU). Nous pouvons observer en interprétant les scores inter-BLEU que la sortie de `rewriter` se différencie davantage de celle `moses`, avec un score inter-BLEU, calculé entre `moses` et `rewriter`, de 75,8, comparé à la sortie de `reranker` qui obtient un score de 82,4. `multi-pass` (+`rewriter`) s'éloigne encore plus de la traduction produite par `moses` avec un score de 74,4. Nos systèmes ont donc tendance à se différencier de la sortie initiale de `moses` tandis que `rewriter` et `multi-pass` (+`rewriter`) produisent une traduction assez proche, en effectuant un nombre similaire de dégradations et d'améliorations des traductions produites par `moses` au niveau des phrases, selon *s*BLEU, et avec un score inter-BLEU calculé entre eux de 94,4. Ceci est probablement la conséquence de l'influence de `rewriter` dans le décodage à passes multiples effectué par `multi-pass`.

En outre, comme le montre la Figure 5.7a, en partitionnant en quartiles les phrases traduites par `moses` selon leur score *s*BLEU, pour chaque système, nous pouvons constater une nette amélioration de la qualité des traductions, selon BLEU, pour les phrases les plus difficiles à traduire (quartile 1), avec `reranker` (+2,3 points BLEU) puis `rewriter` (+3,0 points BLEU), ce que nous observions déjà dans la section 4.2.3.2. En revanche, `rewriter` et `multi-pass` atteignent la même performance sur ce premier quartile. Ces deux systèmes ne se différencient que sur les deuxième et quatrième quartiles avec des améliorations respectives de 0,3 et 0,1 point BLEU pour `multi-pass` par rapport à `rewriter`. Cependant, si l'on effectue le partitionnement en quartiles en se basant sur la sortie de `multi-pass` (+`rewriter`), comme le montre la Figure 5.7d le premier quartile contient un ensemble de traductions produites par `multi-pass` (+`rewriter`) d'une qualité significativement moins bonne que celles produites initialement par `moses` avec une dégradation de 1,3 point BLEU. Sur ce quartile particulier, `rewriter` dégrade également la sortie de `moses`, mais de façon moins importante avec une perte 0,7 point BLEU. Au contraire, sur le quatrième quartile, l'amélioration obtenue avec `multi-pass` (+`rewriter`) est de 4,7 points BLEU par rapport à `moses` (+2,1 points BLEU rapport à `reranker`), et de 0,5 point BLEU par rapport à `rewriter`.

À la lumière de ces observations, et comme attendu, nous pouvons en déduire que `multi-pass` (+`rewriter`) semble donc renforcer les choix de `rewriter` en dégradant davantage les traductions que `rewriter` dégradait déjà, ou au contraire en améliorant davantage les traductions que `rewriter` améliorait déjà. Une manière d'éviter ces dégrations serait d'avoir recours à un classifieur utilisant notamment des mesures de confiance pour identifier en amont les phrases dont les traductions doivent être conservées en l'état, telles que produites par `moses` ou `rewriter`, et celles dont les traductions doivent être modifiées par nos systèmes, tel que nous le suggérions dans la section 4.2.3.2.

Un oracle qui choisirait la meilleure traduction pour chaque phrase parmi celles proposées par chaque système nous permettrait d'évaluer le gain maximal atteignable si plutôt

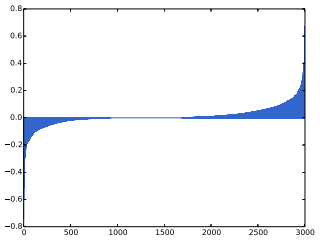
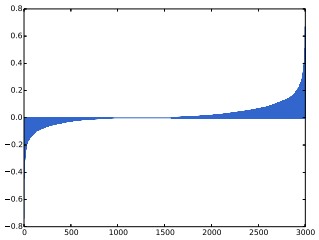
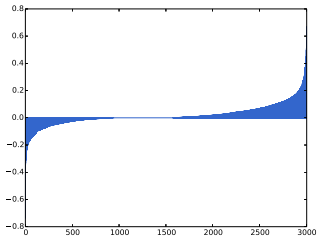
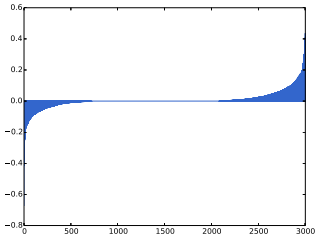
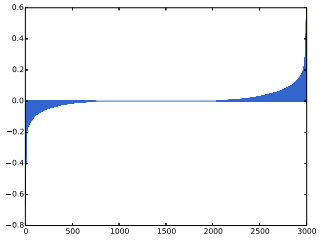
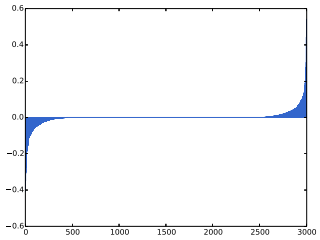
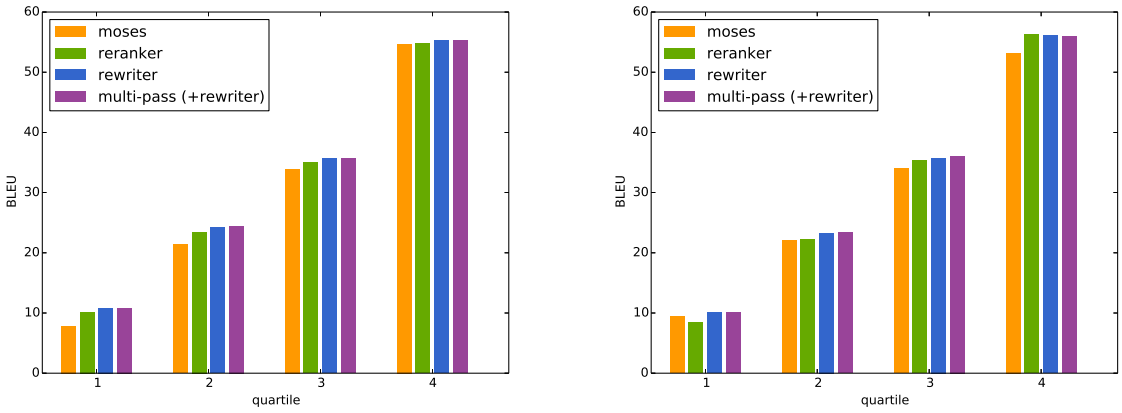
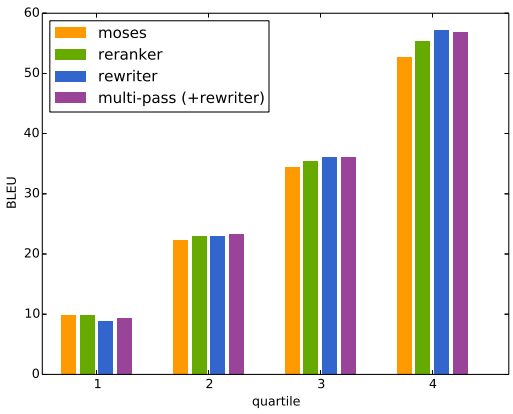
	reranker	rewriter	multi-pass (+rewriter)
moses	 inter-BLEU = 82,4	 inter-BLEU = 75,8	 inter-BLEU = 74,4
reranker		 inter-BLEU = 86,4	 inter-BLEU = 84,9
rewriter			 inter-BLEU = 94,4

TABLE 5.6 – Écarts entre scores *s*BLEU, ordonnés par ordre croissant, des traductions de chaque phrase (tâche `news` en→fr) obtenues avec les systèmes `reranker`, `rewriter`, `multi-pass` (+`rewriter`) (décodage à passes multiples incluant une passe de `rewriter` à chaque itération). Un écart est calculé en prenant le score *s*BLEU de la traduction donnée par le système de la colonne considérée auquel est soustrait le score *s*BLEU de la traduction donnée par le système de la ligne considérée. Un score inter-BLEU est également calculé entre les traductions produites par les deux systèmes.

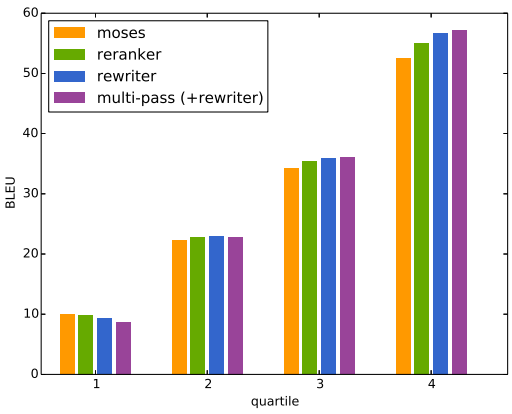


(a) `moses`

(b) `reranker`



(c) `rewriter`



(d) `multi-pass`

FIGURE 5.7 – Performance de la traduction de `moses`, `reranker`, `rewriter` et `multi-pass` (+`rewriter`) (tâche `news` en→fr) pour chaque quartile de phrases, où les quartiles sont définis de taille égale selon le score sBLEU de l'hypothèse de `moses` (5.7a), `reranker` (5.7b), `rewriter` (5.7c) ou `multi-pass` (+`rewriter`) (5.7d).

que de dégrader une traduction celle-ci était conservée en l'état. La Figure 5.7 présente les résultats d'un tel oracle pour toutes les configurations possibles. Nous pouvons observer que la meilleure paire de systèmes est la paire (`moses`, `multi-pass` (+`rewriter`)), avec un gain oracle de 0,9 point BLEU relativement à `multi-pass` (+`rewriter`) seul. De plus, les 3 triplets de systèmes impliquant `moses` sont légèrement meilleurs (34,7 points BLEU pour le meilleur) que le triplet ne l'impliquant pas (34,3 points BLEU). Ces résultats montrent ainsi que certaines traductions ont été dégradées par les différents systèmes appliqués sur les sorties de `moses`, et cela même dès la passe de reclassement avec `reranker`. En effet, la paire (`moses`, `reranker`) est meilleure de 0,9 point BLEU par rapport à `reranker` seul. La paire (`moses`, `rewriter`) est aussi intéressante puisque son score BLEU est 1,1 point BLEU supérieur à celui de `rewriter`. Identifier les traductions de `moses` amenées à être dégradées par `rewriter` permettrait donc un gain de 3,0 points BLEU par rapport à `moses`. Une détection automatique parfaite de la meilleure traduction parmi celles proposées par tous les systèmes permettrait de surcroît d'atteindre un gain de 3,8 points BLEU relativement à `moses`.

`moses`	`reranker`	`rewriter`	`multi-pass` (+`rewriter`)	oracle (BLEU)
✓				31,1
	✓			32,5
		✓		33,0
			✓	33,4
✓	✓			33,4
		✓	✓	33,6
	✓	✓		33,6
	✓		✓	34,0
✓		✓		34,1
✓			✓	34,3
	✓	✓	✓	34,3
✓	✓	✓		34,6
✓		✓	✓	34,6
✓	✓		✓	34,7
✓	✓	✓	✓	**34,9**

TABLE 5.7 – Score BLEU de l'oracle choisissant pour chaque phrase la meilleure traduction selon *s*BLEU parmi celles fournies par les systèmes sélectionnés (✓) pour la tâche `news` en→fr.

Résumé

Nous avons décrit dans ce chapitre une approche regénérant une traduction à l'aide de modèles complexes. Ceci est effectué au moyen d'un décodage à passes multiples qui exploite de nouveaux modèles appris sur les différences entre la meilleure hypothèse du décodeur et celle d'un système de reclassement ou d'un système de réécriture utilisant des modèles complexes. De tels modèles permettent de guider chaque redécodage afin que celui-ci produise une liste de n meilleures hypothèses proches d'un type d'hypothèses plus utile à l'optimisation de la fonction de score du système de reclassement. La concaténation des listes d'hypothèses produites par chaque décodage puis son reclassement permet l'obtention de meilleures traductions que celles produites initialement par le système reclassant les n meilleures hypothèses produites par un premier décodage. Nous avons pu montrer qu'une telle amélioration de la traduction obtenue par un système de reclassement est notamment rendue possible grâce à la diversification et à l'amélioration significative de la qualité des listes de n meilleures hypothèses produites à chaque redécodage. Nous avons également pu montrer que l'ajout d'une passe de recherche locale avant chaque redécodage pour extraire les nouvelles tables de traduction permet d'atteindre une traduction encore meilleure. Toutefois, l'ajout d'une passe de recherche locale augmente significativement la durée de l'entraînement du système ainsi que sa complexité d'ensemble.

Cependant, le système de réécriture utilisé dans ce décodage à passes multiples n'est qu'une réplication de celui que nous avions développé et évalué dans le chapitre 4 en dehors d'un décodage à passes multiples. Il paraît probable que son fonctionnement actuel soit mal adapté à son exploitation dans un tel cadre. Nous pouvons par exemple imaginer que les tables de réécriture ainsi que les exemples de voisinages utilisés pour l'optimisation de sa fonction de score puissent ici ne pas être optimaux pour l'entraînement d'un système de réécriture exécuté entre deux redécodages. Une nouvelle méthode d'extraction des tables de réécriture en fonction des redécodages précédents pourrait par exemple s'avérer plus adaptée dans un tel contexte que l'extraction d'une table de réécriture extraite uniquement sur le résultat du dernier décodage. En outre, il pourrait être intéressant d'étudier quel type d'hypothèses serait le plus adapté afin d'entraîner `rewriter` plus rapidement, en ayant moins d'exemples à évaluer avec des modèles complexes, et pour que celui-ci produise une traduction de meilleure qualité.

D'autres pistes d'améliorations concernent les tables `IN` et `OUT` utilisées à chaque décodage. Le système actuel n'utilise que des informations provenant spécifiquement des seules meilleures hypothèses des systèmes `moses`, `reranker` et `rewriter` pour extraire ces tables. Cette extraction pourrait, par exemple pour `reranker`, prendre en compte le reclassement des hypothèses dans sa globalité pour isoler plus précisément les bisegments utiles. L'ajout de nouveaux scores associés à chaque bisegment dans les tables de traduction extraites, pour par exemple mieux rendre compte de leur appréciation par les différents systèmes, constitue également une piste d'amélioration prometteuse.

Le fonctionnement du système de décodage à passes multiples incluant un système de reclassement d'hypothèses et de réécriture est résumé par la Figure 5.8.

Dans le chapitre 6, nous étudierons le potentiel d'un décodage à passes multiples guidé par des informations *parfaites*, qui seraient possiblement apportées par un intervenant humain. Cette étude nous permettra notamment de mettre en évidence dans quelle mesure il est possible d'améliorer une traduction automatique *via* un décodage à passes multiples guidé par une fonction de score que nous pourrons considérer encore mieux informée que celle utilisée dans ce chapitre.

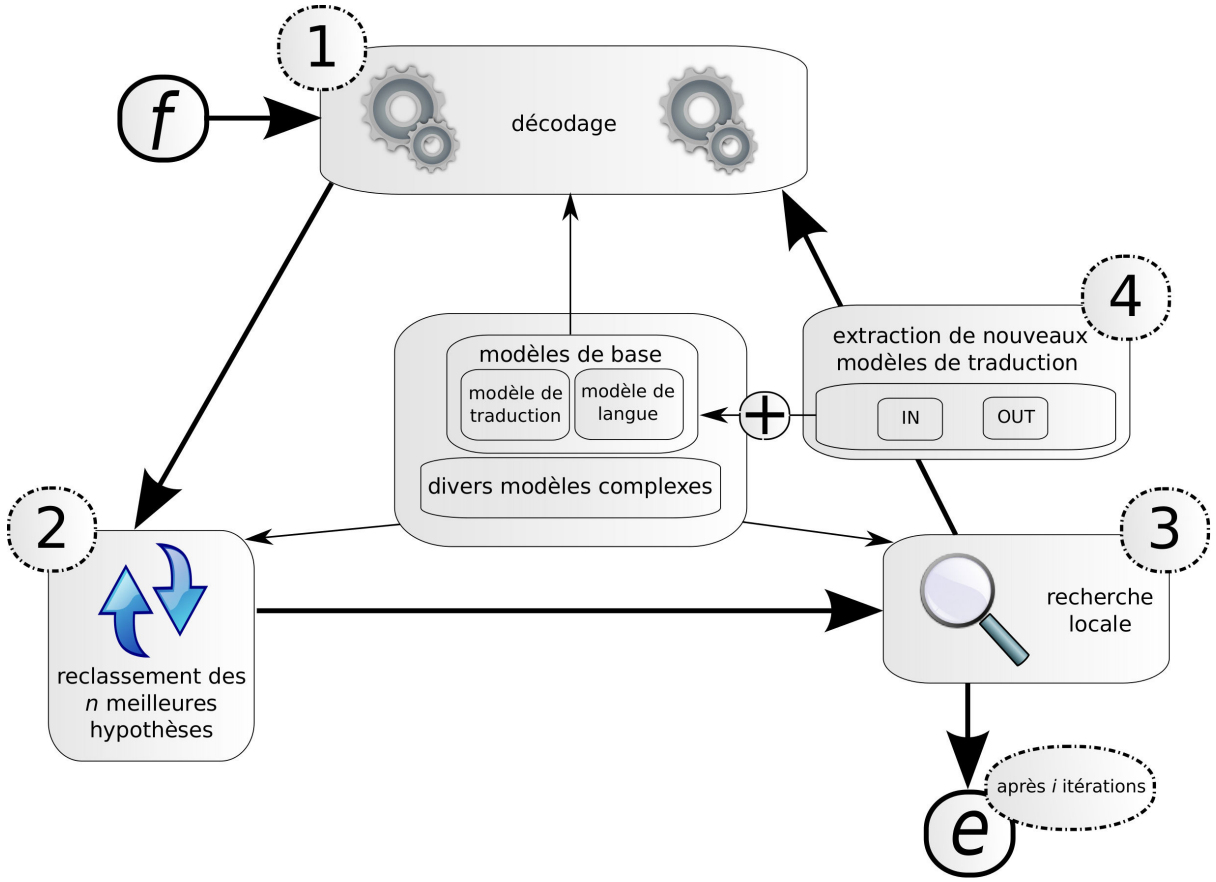


FIGURE 5.8 – Résumé du fonctionnement du décodage à passes multiples. Une phrase source f est décodée en utilisant les modèles de base du décodeur (1). Le décodeur produit la liste des meilleures hypothèses qui est reclassée à l'aide des modèles de base associés à des modèles complexes (2). Un système de réécriture utilisant un algorithme de recherche locale réécrit la meilleure hypothèse obtenue avec les mêmes modèles que ceux utilisés en reclassement (3). De nouveaux modèles sont extraits à partir du résultat de la réécriture et ajoutés parmi les modèles de base pour effectuer un redécodage (4) qui sera à nouveau suivie des étapes (2) et (3), voire même (4) s'il faut itérer une nouvelle fois. Enfin, après i itérations répétant les étapes (1), (2), (3) et (4) on obtient une traduction e.

6

Regénérer une traduction à l'aide de l'humain : la pré-post-édition

Sommaire

6.1 La pré-post-édition **119**
6.1.1 Description de l'approche 119
6.1.2 Partitionnement des modèles en fonction des annotations . . . 120
6.1.3 Fonctionnement global 123

6.2 Expériences **124**
6.2.1 Le paradigme de la post-édition simulée 124
6.2.2 TER_{PPE} : une métrique pour l'évaluation de la pré-post-édition 125
6.2.3 Systèmes de référence 126
6.2.4 Résultats 126

6.3 Analyse **128**
6.3.1 Évaluation de l'importance des modèles ajoutés 128
6.3.2 Ajout d'une passe de reclassement d'hypothèses après chaque itération 129

Résumé **132**

Dans les chapitres 4 et 5 nous avons décrit deux approches permettant l'amélioration d'une traduction de façon automatique. Bien que ces approches permettent d'obtenir des améliorations importantes, les traductions obtenues demeurent imparfaites, tel qu'illustré notamment par l'analyse d'erreurs effectuée sur la sortie du système de réécriture `rewriter` dans la section 4.2.3.4.

Ce dernier chapitre présente un nouveau cadre dans lequel nous pourrons analyser le potentiel d'amélioration des traductions, au moyen d'un décodage à passes multiples, dans une situation où l'on dispose d'informations bien plus riches que celles utilisées pour l'apprentissage des modèles complexes utilisés dans les chapitres précédents. Ces informations seront, dans le cadre de cette analyse, déduites d'une traduction de référence. Nous serons en mesure de décrire ce décodage à passes multiples parfaitement informé sous la forme d'une traduction *interactive*, qui serait réduite à sa forme la plus simple, et dans laquelle les annotations utilisées par le décodage à passes multiples seraient apportées directement par un humain entre chaque itération de décodage. Ce nouveau cadre pourrait permettre une amélioration significative à moindre coût des traductions avant leur possible post-édition (voir la section 3.6.2). C'est pourquoi nous appellerons cette nouvelle approche *pré-post-édition*, puisqu'elle viendra s'intercaler entre un décodeur et une possible étape de post-édition. Le principal défi de la pré-post-édition que nous n'aborderons pas dans ce travail réside dans le fait de pouvoir faire intervenir un humain, de la façon la plus simple possible, afin d'améliorer des traductions tout en diminuant le coût global de l'effort humain nécessaire pour les rendre exploitables.

Ce nouveau chapitre débutera par une présentation de la pré-post-édition dans la section 6.1 qui décrira son rôle et son fonctionnement reposant sur de nouveaux modèles générés à partir d'annotations apportées par un humain. La section 6.2 présentera des expériences de pré-post-édition *simulée*, exploitant une traduction de référence, et donnera leurs résultats, notamment par le biais d'une nouvelle métrique d'évaluation adaptée pour rendre compte du coût de la pré-post-édition, TER_{ppe}. Finalement, la section 6.3 présentera une analyse de l'impact des différents types de modèle utilisés ainsi que les conséquences de l'ajout d'une passe de reclassement d'hypothèses après chaque itération de pré-post-édition.

6.1 La pré-post-édition

6.1.1 Description de l'approche

Les études portant sur les meilleures performances atteignables par les systèmes de traduction statistiques (Turchi *et al.*, 2012; Wisniewski et Yvon, 2013), incluant nos propres résultats décrits dans la section 4.1.1, ont montré que ces systèmes produisent une traduction d'une qualité significativement plus basse que celle qu'il est possible de produire à partir des données d'entraînement utilisées pour les construire. L'amélioration significative d'une traduction, en vue de sa publication, est permise par l'intervention d'un humain, idéalement traducteur professionnel dans une étape appelée *post-édition* (voir section 3.6.1). Nous avons par ailleurs vu dans les chapitres précédents différentes approches qui permettent d'améliorer des traductions de façon entièrement automatique. En particulier, notre approche de décodage à passes multiples, introduite dans le chapitre précédent, utilise de nouveaux modèles complexes au cours de décodages successifs afin d'identifier des bisegments qui seraient possiblement à privilégier ou au contraire à éviter lors d'une nouvelle recherche. Dans ce chapitre, nous décrivons une approche permettant de guider le décodeur pour regénérer une nouvelle traduction en fonction d'annotations *parfaites* sur les séquences de tokens (ou de n-grammes) produits dans la meilleure hypothèse d'un système : celles-ci sont apportées par un intervenant humain. [1] Le processus de regénération proposé est très similaire à celui décrit dans la section 5, sa principale différence résidant dans le fait que les nouveaux modèles construits entre chaque itération utilisent des informations parfaites et non calculées de façon automatique par le biais d'un système de reclassement d'hypothèses reposant sur des modèles complexes. Il est donc attendu que les nouvelles informations obtenues après chaque itération de décodage permettent de guider de façon beaucoup plus efficace le décodeur pour atteindre des traductions de meilleure qualité.

Nous appelons cette nouvelle méthode de regénération d'hypothèses à l'aide d'un humain « pré-post-édition » (PPE). Ce nom provient du fait qu'elle opère à l'issue du décodage mais avant une éventuelle post-édition finale. Son principal intérêt est qu'elle permet de réduire le coût de la post-édition, activité complexe et coûteuse, de façon très importante comme nous allons le montrer. Pour cela, il est nécessaire de limiter autant que possible l'effort d'annotation demandé à l'humain durant la pré-post-édition. L'interaction entre le pré-post-éditeur et la traduction est donc conçue pour être très simple : le pré-post-éditeur indique simplement, par exemple à l'aide d'un surlignement au doigt sur un écran tactile, les n-grammes d'une hypothèse de traduction obtenue automatiquement qui lui semblent pouvoir être utilisés dans une traduction acceptable. La Figure 6.1 illustre quatre itérations de PPE effectuées sur une traduction initiale produite par `moses`.

Les séquences de tokens surlignées sont ensuite utilisées pour apprendre de nouveaux

1. Nous discuterons plus loin dans ce chapitre le caractère idéalisé de cette conception, et nous nous limiterons à la simulation du processus proposé par l'utilisation de traductions de référence déjà disponibles et devant être produites.

modèles qui seront utilisés lors d'une nouvelle passe de décodage. Il faut noter que seul le surlignagne des n-grammes corrects est demandé, et non celui des n-grammes qui semblent être incorrects. Ce choix a été fait afin de conserver l'extrême simplicité de l'interaction qui ne demande qu'un seul type d'annotation. De plus, un n-gramme peut être incorrect sans pour autant que les sous-n-grammes qui le composent le soient également. Ainsi, s'il pourrait sembler plus approprié de faire l'annotation des n-grammes incorrects, notamment à partir d'une traduction automatique de relativement bonne qualité, ce type d'annotation ne permettrait pas une interaction aussi simple que celle que nous proposons. La Figure 6.2 illustre une trace de pré-post-édition pour laquelle une annotation des n-grammes incorrects n'aurait pas été adaptée. Dans cette trace, les deux tokens « owing » et « largely » sont possiblement corrects étant donné la phrase à traduire, en revanche le bigramme « owing largely » ne l'est pas : ce bigramme n'est donc pas surligné, mais les tokens qui le composent le sont. Le travail du pré-post-éditeur peut donc être formulé comme une réponse à la question suivante : *quels sont les fragments de la traduction automatique présentée qui ont potentiellement leur place dans une traduction de bonne qualité ?*

6.1.2 Partitionnement des modèles en fonction des annotations

À partir des n-grammes annotés, que nous considérons donc comme pouvant faire partie de la traduction à atteindre, de nouveaux types de modèles sont appris : deux modèles de langue [2], et les modèles de deux tables de traduction.

6.1.2.1 Modèles de langue

Les n-grammes annotés sont dans un premier temps comptés, notamment afin de prendre en compte le fait qu'un même n-gramme peut apparaître plusieurs fois dans la traduction produite par le décodeur, et un modèle de langue, que nous qualifions de *positif* (`ml-positif`), est entraîné sur ces comptes. Un deuxième modèle de langue est par ailleurs entraîné à partir des n-grammes de la traduction contenant au moins un token n'ayant pas été annoté par le pré-post-éditeur. Nous pouvons considérer que, pour ces derniers n-grammes, le pré-post-éditeur n'est pas suffisamment sûr du fait qu'ils devraient apparaître dans une traduction correcte. Nous qualifions le modèle de langue appris avec leurs comptes de *négatif* (`ml-négatif`).

2. À la différence donc du décodage à passes multiples, nous avons pu constater que dans le cadre de la pré-post-édition l'ajout de nouveaux modèles de langue peut contribuer significativement à l'amélioration de la qualité de la traduction (voir section 6.3.1).

source	c' est la réponse à une nouvelle prise de conscience selon laquelle les entreprises chinoises sont indispensables à la survie économique de Taiwan
PPE#0	this is the answer to a new awareness that Chinese companies are essential to the economic survival of Taiwan
PPE#1	it is the response to a new awareness that Chinese firms are essential to Taiwan's economic survival .
PPE#2	it is the reply to a new awareness that Chinese enterprises is essential to Taiwan's economic survival .
PPE#3	it is responding to a new awareness that Chinese businesses is essential to Taiwan's economic survival .
PPE#4	it is responding to a new awareness that Chinese business is essential to Taiwan's economic survival .
référence	it is responding to a new awareness that Chinese business is essential to Taiwan's economic survival .

FIGURE 6.1 – Exemple d'une trace de pré-post-édition (PPE) pour une traduction du français vers l'anglais (produite par le système `news`). Chaque nouveau n-gramme annoté est indiqué sur fond vert. Les n-grammes sur fond gris sont ceux ayant été annotés au cours des itérations précédentes.

source	son impopularité semble être en grande partie due au chômage
PPE#0	his unpopularity seems to be | owing | largely | to unemployment
PPE#1	his unpopularity seems to be largely owing to unemployment
référence	his unpopularity seems to be largely owing to unemployment

FIGURE 6.2 – Exemple d'une annotation de deux tokens corrects (« owing » et « largely ») formant un bigramme incorrect. Durant la première itération de PPE un réordonnancement des deux tokens est effectué afin de rendre la traduction correcte.

Pour le système de pré-post-édition présenté dans ce chapitre, les nouveaux modèles de langue sont des modèles 6-grammes appris avec un lissage Witten-Bell [3]. L'entraînement des modèles de langue a été réalisé de la même manière que dans nos autres expériences avec l'outil SRILM. [4]

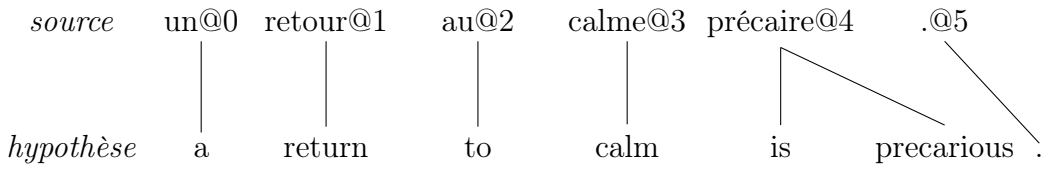
6.1.2.2 Tables de traduction

Les bisegments de la table de traduction utilisée par le décodeur pour produire la traduction et qui sont *compatibles* avec les n-grammes surlignés sont identifiés en tenant compte des alignements au niveau des mots fournis par le décodeur entre la phrase source et la traduction annotée. Ces bisegments sont ensuite supprimés de cette table et stockés dans une nouvelle table de traduction *positive* (`tt-positive`). Afin d'éviter que le décodeur ne modifie la traduction d'un token déjà annoté comme correct au cours des itérations suivantes de pré-post-édition, tous les bisegments contenant une traduction alternative dans la table de traduction principale sont supprimés. [5] Les bisegments compatibles avec les n-grammes contenant au moins un token non annoté sont également supprimés de la table de traduction principale et stockés dans une nouvelle table de traduction *négative* (`tt-négative`). Avec un tel partitionnement, un bisegment ne peut se trouver que dans une seule table de traduction parmi trois : la table principale ou l'une des nouvelles tables `tt-positive` et `tt-négative`. Par ailleurs, puisqu'un token peut apparaître plusieurs fois dans un même texte à traduire, nous utilisons ici encore des tokens localisés tels que décrits dans la section 5.1.2.2.

3. Nous utilisons ici un lissage Witten-Bell car celui-ci est plus adapté pour des estimations obtenues à partir de peu de données que le lissage Kneser-Ney, surtout lorsqu'il s'agit d'estimer de longs n-grammes.

4. SRILM redécompose les n-grammes et leurs sous-n-grammes qui les composent durant l'entraînement du modèle de langue. En conséquence, pour un bigramme tel que « rouge pomme » incorrect, mais qui contiendrait les tokens corrects « rouge » et « pomme », le bigramme sera décomposé en ses unigrammes corrects. Ceux-ci, par ailleurs déjà utilisés pour l'entraînement du modèle de langue positif `ml-positif`, seront donc également pris en compte pour l'entraînement du modèle de langue négatif `ml-négatif`.

5. Ce choix est discutable, notamment si, au fur et à mesure des itérations, un token précédemment surligné ne semble plus approprié dans son nouveau contexte obtenu après redécodage. Pour contrer un tel effet, il pourrait être nécessaire de créer une nouvelle possibilité d'interaction qui donnerait la possibilité à l'humain d'invalider un n-gramme précédemment surligné et réintroduirait donc dans la table de traduction principale les alternatives de traduction qui en avaient été supprimées.



référence return to precarious calm .

`tt-positive`		`tt-négative`	
source	cible	source	cible
retour@1 au@2	return to	précaire@4	is precarious
précaire@4	precarious	calme@3 précaire@4	calm is precarious
.@5	.	précaire@4 .@5	is precarious .

`ml-positif`		`ml-négatif`	
n-gramme	# occurrence	n-gramme	# occurrence
return	1	a	1
return to	1	a return to	1
to	1	to precarious	1
calm	1	to calm is precarious .	1

FIGURE 6.3 – Exemple de bisegments et de n-grammes extraits d'une traduction automatique effectuée par `moses` sur la tâche `news` fr→en. Les bisegments extraits sont répartis dans deux tables de traduction, `tt-positive` et `tt-négative`, tandis que les n-grammes et leurs comptes sont utilisés pour entraîner deux modèles de langue `ml-positif` et `ml-négatif`.

La Figure 6.3 présente un exemple de bisegments et de n-grammes extraits d'une traduction automatique qui sont utilisés pour construire des tables de traduction `tt-positive` et `tt-négative` ainsi qu'entraîner des modèles de langue `ml-positif` et `ml-négatif`.

6.1.3 Fonctionnement global

6.1.3.1 Utilisation des nouveaux modèles

Les nouveaux modèles utilisés à chaque itération de PPE sont spécifiques à chaque phrase à traduire, et l'utilisation de tokens localisés nous permet d'avoir des bisegments propres pour la traduction de chaque séquence de tokens individuelle. Les modèles de langue `ml-positif` et `ml-négatif` sont également appris spécifiquement pour chaque phrase à partir des annotations d'une hypothèse de traduction. Afin d'éviter que ces modèles ne soient appris sur une quantité de données de plus en plus réduite au fil des

itérations de PPE[6], les modèles de langue et les tables de traduction sont mis à jour en cumulant l'ensemble des données collectées depuis le début de la pré-post-édition. Cela signifie donc, en particulier, que les bisegments qui n'apparaissaient précédemment pas dans les tables `tt-positive` ou `tt-négative` sont ajoutés à la table existante correspondante, tandis que les comptes des n-grammes utilisés pour l'entraînement des modèles de langue sont cumulés sur l'ensemble des itérations pour apprendre de nouveaux modèles de langue.

6.1.3.2 Optimisation du système

Comme cela est fait pour notre décodage à passes multiples (voir la section 5.1.3.2), les poids des modèles utilisés par la fonction de score du décodeur sont optimisés sur le même corpus de développement que celui utilisé pour l'optimisation des poids de la fonction de score originelle du décodeur. Ces poids sont spécifiques à chaque itération. Ainsi, par exemple, les poids optimisés obtenus lors de la troisième itération de PPE effectuée sur le corpus de développement s'appliqueront lors de la troisième itération de PPE effectuée sur un corpus de test. L'optimisation s'arrête après un nombre d'itérations i fixé à l'avance. Le point essentiel ici est de trouver un bon compromis entre les gains apportés par plusieurs itérations d'une part, et les difficultés pratiques de mise en œuvre de la pré-post-édition d'autre part, laquelle repose sur une réanalyse par l'humain d'hypothèses de traduction mises à jour après chaque itération. Pour que cette approche soit suffisamment efficace, il est nécessaire que des améliorations soient observables dès les premières itérations. C'est pourquoi dans la partie expérimentale qui suit le nombre d'itérations a été fixé à 5. Toutefois, ce nombre est ici fixé de manière globale, un nombre d'itérations adapté à chaque hypothèse à annoter, en fonction de sa qualité par exemple, serait sans doute plus efficace.

6.2 Expériences

6.2.1 Le paradigme de la post-édition simulée

L'évaluation de l'approche de traduction interactive que nous proposons, en apparence très simple, repose idéalement sur la mise en situation de personnes compétentes pour produire des annotations au niveau des segments sur des hypothèses de traduction qui évoluent. Or cela revêt de nombreuses difficultés pratiques, la principale d'entre elles étant l'accès à des annotateurs entraînés pour cette tâche. Dans ce travail, nous avons dû nous limiter à un protocole nous permettant d'évaluer la pré-post-édition sans recours à des annotateurs humains. Nous avons repris et adapté le paradigme de *post-édition simulée* (Carl *et al.*, 2011; Denkowski *et al.*, 2014). Dans ce paradigme, l'évaluation repose sur des traductions de référence qui ne sont pas obtenues par post-édition de traductions automatiques. Ces traductions de référence étant déjà disponibles, cela nous permet d'évaluer

6. Par exemple, pour une hypothèse de traduction dont tous les n-grammes sont annotés comme corrects, le modèle `ml-négatif` ne contiendrait aucune information.

la pré-post-édition à moindre coût, de façon répétée lors de l'optimisation et de l'évaluation des systèmes, ainsi que sur plusieurs tâches de traduction. Toutefois, il faut noter qu'un des principaux défauts du recours à des traductions de référence existantes est que cela contraint durement les traductions qui seront récompensées, qui peuvent ne pas toujours correspondre aux choix qu'aurait fait un pré-post-éditeur humain en situation réelle. En outre, les difficultés inhérentes à l'annotation, qui consiste donc à repérer des segments qui pourraient participer à une traduction acceptable, ne peuvent pas être prises en compte de manière réaliste.

Aucun humain n'est donc directement intervenu dans les expériences de ce chapitre. Les annotations des n-grammes[7] sont effectuées en comparant une traduction de référence à la traduction automatique produite par le décodeur. Si un n-gramme appartient à la traduction de référence, celui-ci est simplement considéré correct. Cela revient à imaginer qu'une telle traduction de référence serait la traduction que le pré-post-éditeur souhaiterait atteindre. Nous pouvons donc percevoir l'évaluation de la pré-post-édition que nous faisons ici comme un moyen de mesurer le potentiel d'amélioration des traductions pouvant être obtenues par le biais d'un décodage à passes multiples, correspondant à notre proposition du chapitre précédant, qui disposerait d'informations parfaites.

6.2.2 TER_{PPE} : une métrique pour l'évaluation de la pré-post-édition

Les métriques d'évaluation en traduction automatique ne tiennent pas compte, en l'état, du coût des interactions à effectuer lors de la pré-post-édition. Pour évaluer un tel coût, nous proposons une adaptation de la métrique TER (voir la section 2.4.3) que nous appellerons TER_{PPE}. La métrique TER a typiquement recours à 4 opérations d'édition agissant sur des tokens de l'hypothèse de traduction évaluée : `substitution` (sub), `insertion` (i), `suppression` (sup) et `déplacement` (d). Chacune de ces opérations a un coût de 1.

Nous introduisons dans TER_{ppe} une nouvelle opération : `validation` (v). Cette nouvelle opération correspond à ce que réaliserait un pré-post-éditeur lorsqu'il indique par annotation qu'un token est correct. Les autres opérations classiques de TER sont celles qu'il effectuerait pour transformer l'hypothèse de traduction en la traduction de référence. Cependant, si une opération telle que `validation` n'apparaît pas dans la mesure de TER, nous pouvons raisonnablement affirmer que celle-ci est pourtant bien effectuée par un post-éditeur : en effet, le post-éditeur a à déterminer tôt ou tard si chaque token d'une hypothèse de traduction doit être édité ou non. L'opération `validation` reçoit elle aussi un poids dans TER_{ppe}, α ; bien qu'on pourrait supposer que cette opération, en particulier si elle est réalisée à l'aide du doigt, est bien moins coûteuse que les autres opérations reposant sur une édition au clavier, nous donnerons dans ce chapitre des valeurs

7. Dans les expériences de pré-post-édition présentées ici, les n-grammes extraits ont une taille pouvant aller jusqu'à 6 tokens.

de TER_{ppe} pour α variant sur $[0, 1]$. Notre mesure TER_{ppe} se calcule ainsi :

$$\text{TER}_{\text{PPE}} = \frac{\#sub + \#i + \#sup + \#d + \alpha\#v}{r + \alpha r} \tag{6.1}$$

où r est le nombre de tokens de la traduction de référence. En particulier, pour $\alpha = 0$, la mesure TER_{PPE} correspond à celle de TER, ce qui signifierait un coût de pré-post-édition précédant une post-édition comme nulle. À l'opposé, pour $\alpha = 1$, l'annotation d'un token lors de la pré-post-édition aurait le même coût que n'importe quelle autre opération telle que la substitution (*sub*).

6.2.3 Systèmes de référence

Les expériences de ce chapitre ont été menées sur les tâches `news` et `médical` pour les deux directions de traduction fr→en et en→fr. Les systèmes de traduction de référence utilisés dans les expériences suivantes sont ceux présentés dans la section 5.2.1.

6.2.4 Résultats

Les résultats de 5 itérations de pré-post-édition sont présentés dans les Tables 6.1 et 6.2. Pour ces premiers résultats, l'évaluation a été faite à l'aide des métriques TER et BLEU, en utilisant chacune de ces métriques pour l'optimisation du système `moses` de référence et des systèmes correspondant à chaque itération de PPE.

Le premier résultat marquant est que la pré-post-édition améliore significativement la traduction initiale effectuée par `moses` dès la première itération, avec une amélioration allant jusqu'à 9,8 points BLEU et -8,2 points TER pour la tâche `news` fr→en. Comme attendu, optimiser sur une métrique particulière permet d'obtenir de meilleurs résultats avec cette même métrique en évaluation (Cer *et al.*, 2010) pour la première itération de PPE.[8] Plus généralement, l'optimisation avec TER donne des améliorations du score BLEU qui sont nettement moins importantes qu'une optimisation avec BLEU pour la tâche `news` ; en outre, ces améliorations sont bien plus fortes durant les premières itérations. Compte tenu de ces observations, seule une optimisation avec BLEU a été effectuée pour la tâche `médical`.[9]

L'amélioration de la traduction s'accroît de façon importante au fur et à mesure des itérations de pré-post-édition. Le score BLEU est ainsi amélioré de 21,1 points BLEU (correspondant à -12,3 points TER) par rapport à la traduction initiale donnée par `moses`

8. Cela n'est toutefois pas toujours le cas pour les dernières itérations (cf. `news` en→fr) pour lesquelles une optimisation avec BLEU donne de meilleurs résultats avec TER en évaluation.

9. Les systèmes optimisés avec TER ont de plus tendance à réduire fortement la taille des hypothèses produites, accordant en conséquence une plus grande importance à la pénalité de concision utilisée lors du calcul du score BLEU et favorisant un plus grand nombre d'insertions à effectuer lors du calcul du score TER.

après 5 itérations de pré-post-édition sur la tâche `news` fr→en. Les résultats suivent la même tendance pour les deux directions de traduction et domaines étudiés ici, avec par exemple un gain de 12,1 points BLEU (-9,7 points TER) obtenu pour le système `médical` fr→en. Pour les systèmes `médical`, l'amplitude moins grande de l'amélioration est peut-être due aux scores initiaux obtenus par la traduction `moses` qui sont meilleurs que sur la tâche `news` (par exemple 37,1 points BLEU contre 28,6 points BLEU respectivement pour les tâches `médical` et `news` fr→en), rendant ainsi son amélioration possiblement plus difficile.

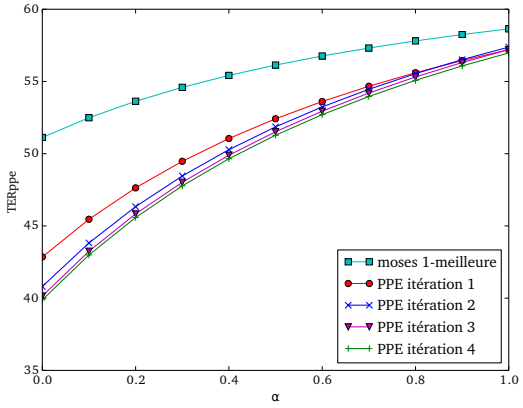
Itération	`news` fr→en				`news` en→fr			
	optimisé avec TER		optimisé avec BLEU		optimisé avec TER		optimisé avec BLEU	
	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU
`moses`	51,1	28,2	52,7	28,6	52,3	29,7	51,8	31,1
PPE itération 1	42,9	35,4	46,7	38,4	44,4	35,0	47,3	39,6
PPE itération 2	40,8	37,3	43,7	43,4	43,0	36,3	44,6	43,9
PPE itération 3	40,8	37,8	42,2	46,2	42,5	36,4	43,5	46,6
PPE itération 4	39,9	37,9	40,9	48,3	42,3	36,5	42,3	48,2
PPE itération 5	**39,9**	**37,9**	**40,4**	**49,7**	**42,2**	**36,6**	**41,0**	**49,5**

TABLE 6.1 – Résultats (scores TER et BLEU) de pré-post-édition par itération sur la tâche `news` fr→en et en→fr (optimisée avec les métriques TER ou BLEU).

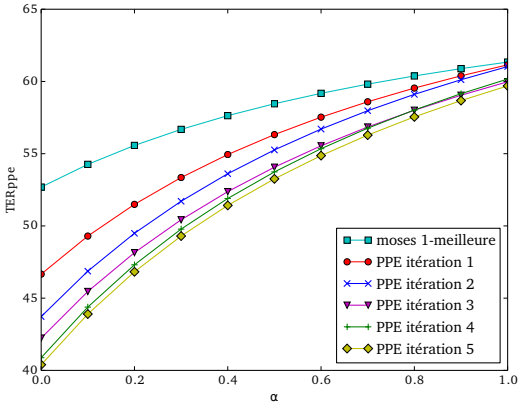
Itération	`médical` fr→en		`médical` en→fr	
	optimisé avec BLEU		optimisé avec BLEU	
	TER	BLEU	TER	BLEU
`moses`	42,2	37,1	44,0	38,8
PPE itération 1	36,9	44,9	37,2	48,3
PPE itération 2	34,8	47,5	35,3	51,1
PPE itération 3	34,1	48,5	33,5	52,9
PPE itération 4	32,9	49,2	32,4	54,0
PPE itération 5	**32,5**	**49,2**	**32,1**	**54,8**

TABLE 6.2 – Résultats (scores TER et BLEU) de pré-post-édition par itération sur la tâche `médical` fr→en et en→fr (optimisée avec la métrique BLEU).

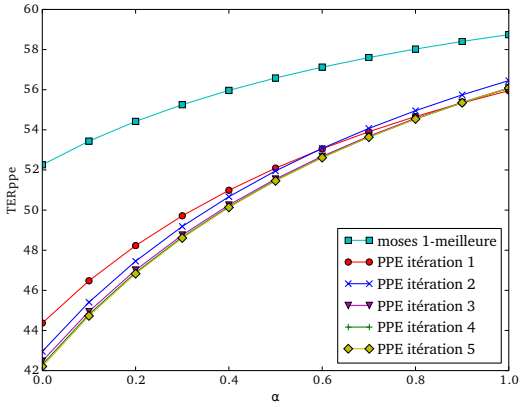
Les Figures 6.4 et 6.5 montrent comment les scores obtenus avec la métrique TER_{PPE} varient pour différentes valeurs de α pour les tâches `news` et `médical`. Nous observons que, pour toutes les valeurs de α, toutes les itérations de pré-post-édition obtiennent un meilleur score que la situation où la post-édition est réalisée directement à partir de la traduction initialement produite par `moses`. Cette tendance se réduit néanmoins au fur et à mesure des itérations de pré-post-édition pour des valeurs de α proches de 1.



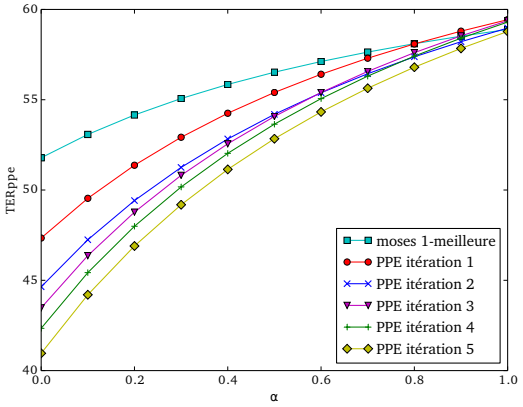
(a) `news` fr→en optimisé avec TER.



(b) `news` fr→en optimisé avec BLEU.



(c) `news` en→fr optimisé avec TER.



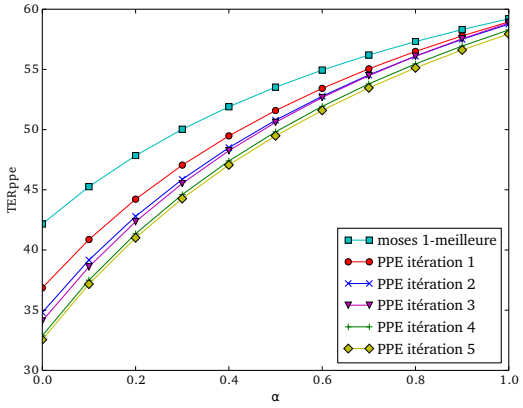
(d) `news` en→fr optimisé avec BLEU.

FIGURE 6.4 – Résultats de pré-post-édition mesurés avec TER_{PPE} pour différentes valeurs de α entre 0 et 1 sur la tâche `news`, avec un système optimisé avec TER ou BLEU. Pour gagner en lisibilité, les courbes correspondant à la $5^{\text{ème}}$ itération ont été masquées sur les Figures 6.4a et 6.4c.

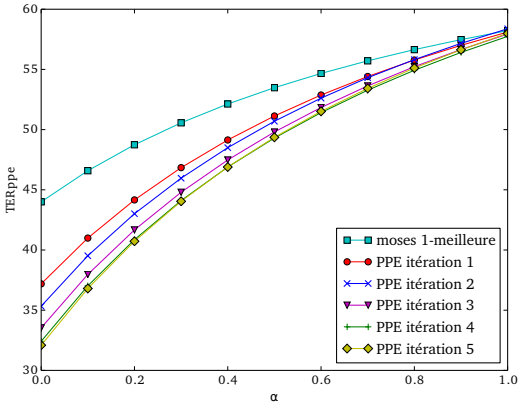
6.3 Analyse

6.3.1 Évaluation de l'importance des modèles ajoutés

Nous avons évalué l'importance des modèles ajoutés lors de la pré-post-édition à ceux utilisés par le décodeur. La Table 6.3 met en évidence une certaine complémentarité entre les modèles dits « négatifs » (`ml-négatif` et `tt-négative`) et les modèles dits « positifs » (`ml-positif` et `tt-positive`) : on observe une chute de presque 10 points BLEU, sur les tâches `news` fr→en et en→fr, lorsque l'un des deux types de modèle n'est pas utilisé. De plus, les modèles de langue (`ml-positif` et `ml-négatif`) contribuent plus largement, selon BLEU, à l'amélioration des traductions lors de la pré-post-édition que l'utilisation de tables de traduction (`tt-positive` et `tt-négative`) : on observe une chute de 9,6 points BLEU sur la tâche `news` fr→en lorsque les modèles de langue ne sont pas utilisés, contre



(a) `médical` fr→en optimisé avec BLEU.



(b) `médical` en→fr optimisé avec BLEU.

FIGURE 6.5 – Résultats de pré-post-édition mesurés avec TER_{PPE} pour différentes valeurs de α entre 0 et 1 sur la tâche `médical`, avec un système optimisé avec BLEU.

une chute bien plus modeste de 4,4 points BLEU lorsque ce sont les tables de traduction qui ne sont pas utilisées. L'inverse peut être constaté pour les scores TER : nous avons une augmentation du score TER de 4,6 points avec les modèles de langue (`ml-positif` et `ml-négatif`) contre une augmentation deux fois moins grande de 2,6 points sans ces mêmes modèles de langue pour le sens de traduction fr→en. La même observation peut être effectuée pour le sens en→fr.

La configuration de PPE sans `ml-négatif` et `ml-positif` est particulièrement intéressante : elle est très proche de la configuration d'un décodage à passes multiples utilisant les tables de traduction `IN` et `OUT` dont nous avons présenté les résultats dans la section 5.2.2. Elle simule approximativement [10] la situation où `reranker` donnerait la traduction de référence comme sa meilleure traduction pour l'extraction des tables `IN` et `OUT`. Par conséquent, nous pouvons en déduire qu'il existe un important potentiel d'amélioration des traductions si nous disposions de modèles mieux informés qui permettraient un meilleur guidage du décodeur au fur et à mesure des itérations d'un décodage à passes multiples.

6.3.2 Ajout d'une passe de reclassement d'hypothèses après chaque itération

Nous avons montré dans les sections précédentes qu'une pré-post-édition fondée sur l'utilisation d'annotations obtenues par connaissance d'une traduction à atteindre et une

10. En effet, dans cette simulation nous n'utilisons pas d'alignements entre la phrase source et la référence correspondante pour extraire les tables de traduction. Au contraire, dans le décodage à passes multiples, les bisegments sont extraits en tenant compte des alignements entre la phrase source et la traduction donnée par `reranker`.

Configuration	fr→en optimisé avec BLEU TER	fr→en optimisé avec BLEU BLEU	en→fr optimisé avec BLEU TER	en→fr optimisé avec BLEU BLEU
`moses`	52,7	28,6	51,8	31,1
PPE avec tous les modèles (*réf.*)	40,4	49,7	41,0	49,5
PPE sans `tt-négative` et `ml-négatif`	**45,2**$_{(+4,8)}$	39,4$_{(-10,3)}$	**47,1**$_{(+6,1)}$	39,0$_{(-10,5)}$
PPE sans `tt-positive` et `ml-positif`	46,7$_{(+6,3)}$	**39,8**$_{(-9,9)}$	48,3$_{(+7,3)}$	**39,8**$_{(-9,7)}$
PPE sans `tt-négative` et `tt-positive`	45,0$_{(+4,6)}$	**45,3**$_{(-4,4)}$	46,5$_{(+5,5)}$	**44,9**$_{(-4,6)}$
PPE sans `ml-négatif` et `ml-positif`	**42,7**$_{(+2,3)}$	40,1$_{(-9,6)}$	**43,2**$_{(+2,2)}$	42,0$_{(-7,5)}$

TABLE 6.3 – Résultats (scores TER et BLEU) de la pré-post-édition sur la tâche `news`, après 5 itérations, pour différentes configurations. Les écarts inscrits entre parenthèses sont donnés par rapport à la configuration de PPE dénotée (*réf.*) qui utilise tous les modèles.

réoptimisation d'un système de traduction permettait d'obtenir des améliorations importantes de la qualité de traduction. Or les chapitres précédents ont permis de décrire et d'évaluer des approches exploitant des modèles complexes qui obtiennent de meilleures performances que des systèmes privés de ces modèles. Il est donc intéressant de considérer si l'utilisation de tels modèles permettrait de mieux tirer profit des annotations de pré-post-édition. Nous avons effectué des reclassements des meilleures hypothèses produites par le décodeur après chaque itération de pré-post-édition. Les modèles complexes utilisés sont identiques à ceux que nous avions utilisés dans nos expériences de décodages à passes multiples (voir section 5.2). Chaque itération de pré-post-édition produit une liste de 300-meilleures hypothèses qui est concaténée aux listes produites lors des itérations précédentes. La liste concaténée obtenue est reclassée à chaque itération par un système `reranker`. De la même manière que dans le décodage à passes multiples, les poids utilisés en évaluation à l'itération i par la fonction de score de `reranker` sont ceux obtenus après une optimisation avec `KB-MIRA` sur la concaténation des listes d'hypothèses produites sur le corpus de développement après i itérations. Les modèles `tt-positive`, `tt-négative`, `ml-positif` et `ml-négatif` sont construits à partir des n-grammes validés de la traduction donnée par `reranker` à chaque itération de pré-post-édition.

La table 6.4 présente les résultats de la pré-post-édition obtenus sur la tâche `news` fr→en et en→fr, qui sont à comparer à ceux donnés dans la Table 6.1, obtenus par un système de PPE ne faisant intervenir aucun système `reranker`. Nous pouvons constater que dès la première itération de PPE, l'utilisation de `reranker` permet d'obtenir, sur la tâche `news` en→fr, un gain supplémentaire de 1,9 point BLEU par rapport à une configuration de PPE n'utilisant aucun reclassement d'hypothèses. L'amélioration obtenue est en revanche plus limitée pour la direction fr→en, avec un gain de 1,0 point BLEU. En utilisant `reranker` nous obtenons donc, dès la première itération suivie de `reranker`, un gain de 10,4 et 10,8 points BLEU respectivement pour fr→en et en→fr relativement au système `moses` de référence. Pour la direction en→fr, l'amélioration après 5 itérations de PPE atteint alors 19,5 points BLEU, contre 18,4 points BLEU en l'absence de re-

Itération	Système	news fr→en optimisé avec BLEU TER	news fr→en optimisé avec BLEU BLEU	news en→fr optimisé avec BLEU TER	news en→fr optimisé avec BLEU BLEU
moses		52,7	28,6	51,8	31,1
reranker		$51{,}5_{(-1{,}2)}$	$29{,}4_{(+0{,}8)}$	$50{,}8_{(-1{,}0)}$	$32{,}5_{(+1{,}4)}$
PPE itération 1	moses	$46{,}7_{(0{,}0)}$	$38{,}6_{(+0{,}2)}$	$46{,}7_{(-0{,}6)}$	$40{,}3_{(+0{,}7)}$
	reranker	$45{,}8_{(-0{,}9)}$	$39{,}4_{(+1{,}0)}$	$45{,}6_{(-1{,}7)}$	$41{,}5_{(+1{,}9)}$
PPE itération 2	moses	$42{,}8_{(-1{,}1)}$	$44{,}5_{(+1{,}1)}$	$43{,}8_{(-0{,}8)}$	$45{,}0_{(+1{,}1)}$
	reranker	$42{,}7_{(-1{,}2)}$	$44{,}6_{(+1{,}2)}$	$43{,}1_{(-1{,}5)}$	$45{,}8_{(+1{,}9)}$
PPE itération 3	moses	$40{,}9_{(-1{,}3)}$	$47{,}7_{(+1{,}5)}$	$42{,}7_{(-0{,}8)}$	$47{,}6_{(+1{,}0)}$
	reranker	$40{,}9_{(-1{,}3)}$	$47{,}7_{(+1{,}5)}$	$41{,}8_{(-1{,}7)}$	$48{,}2_{(+1{,}6)}$
PPE itération 4	moses	$40{,}3_{(-0{,}7)}$	$49{,}4_{(+1{,}1)}$	$41{,}3_{(-1{,}0)}$	$49{,}4_{(+1{,}2)}$
	reranker	$39{,}6_{(-1{,}4)}$	$49{,}8_{(+1{,}5)}$	$41{,}0_{(-1{,}3)}$	$49{,}7_{(+1{,}5)}$
PPE itération 5	moses	$39{,}1_{(-1{,}3)}$	$51{,}0_{(+1{,}3)}$	$40{,}7_{(-0{,}3)}$	$50{,}4_{(+0{,}9)}$
	reranker	$38{,}8_{(-1{,}6)}$	$51{,}4_{(+1{,}7)}$	$40{,}7_{(-0{,}3)}$	$50{,}6_{(+1{,}1)}$

TABLE 6.4 – Résultats (scores TER et BLEU) de pré-post-édition, incluant une passe de `reranker` entre chaque itération, sur la tâche `news` fr→en et en→fr (optimisé avec BLEU). Les résultats donnés à chaque itération sont calculés pour `moses` puis `reranker`. Les scores entre parenthèses donnent la différence avec les scores présentés par la table 6.1 pour la même itération de PPE, effectuée donc sans aucune passe de `reranker`.

classements intermédiaires. La pré-post-édition bénéficie donc de la disponibilité d'une meilleure hypothèse de traduction avant chaque passe de PPE tel que permis par un reclassement d'hypothèses. Ces résultats suggèrent que toute amélioration de la qualité des amorces utilisées est souhaitable : nous pourrions en particulier utiliser pour cela un système tel que `rewriter` (voir section 4) [11]. En revanche, l'impact de `reranker` diminue au fil des itérations, au moins pour la direction en→fr, les améliorations qu'il apporte passant de 1,9 point BLEU, ou 1,7 point TER, à la première itération à 1,1 point BLEU, ou 0,3 point TER, à la cinquième itération. Tandis que, pour la direction fr→en, les scores sont toujours fortement améliorés avec des gains de 1,7 point BLEU et 1,6 point TER.

Nous pouvons également constater une réduction de l'écart entre les scores obtenus par `moses` et ceux obtenus par `reranker` au fur et à mesure des itérations, par exemple sur la tâche `news` en→fr : de 1,1 point TER et 1,2 point BLEU à la première itération nous passons à la cinquième itération à un écart nul en TER et de 0,2 point BLEU entre les deux systèmes.

11. Au prix toutefois d'une forte augmentation des temps de calcul, l'exécution de `rewriter` avec les modèles complexes utilisés ici étant particulièrement coûteuse.

Résumé

Nous avons décrit dans ce chapitre une méthode permettant l'amélioration d'une traduction, dans le cadre d'un décodage à passes multiples, *via* des informations idéales déduites d'une traduction de référence et qui pourraient être obtenues à partir d'annotations effectuées par un humain. Ces annotations seraient collectées au travers d'une interaction qui se veut simplifiée à l'extrême, correspondant à la simple validation, par exemple au doigt sur un écran tactile, des séquences de tokens de la traduction qui auraient leur place dans une bonne traduction. À partir de ces annotations, de nouveaux modèles sont appris et utilisés par le décodeur pour regénérer une nouvelle traduction. De plus, en faisant l'hypothèse que l'annotation effectuée par l'humain a un coût très faible, notre approche de pré-post-édition permet une réduction du coût de la post-édition d'une traduction en proposant au post-éditeur une traduction de meilleure qualité et cela dès une première itération de pré-post-édition. Cette observation permet d'envisager que des améliorations de la qualité des traductions encore plus importantes pourraient être obtenues si nous disposions de modèles complexes mieux informés.

L'approche présentée dans ce chapitre n'a pu être évaluée que dans un cadre de pré-post-édition simulée considérant des traductions de référence comme des post-éditions permettant ainsi de déduire automatiquement les n-grammes à annoter durant la pré-post-édition. Une évaluation impliquant des pré-post-éditeurs et des post-éditeurs humains s'avérera nécessaire pour confirmer la possible utilisation de cette approche dans un cadre interactif réel. De plus, une réflexion devra être menée pour développer une interface suffisamment ergonomique pour faciliter au mieux la pré-post-édition et notamment rendre l'enchaînement des itérations le moins perturbant possible, en mettant par exemple en évidence les parties de la traduction modifiée par la dernière itération du décodeur.

Si une telle approche s'avérait efficace dans un cadre interactif réel, il pourrait être envisageable de confier la pré-post-édition à une personne monolingue ne connaissant pas la langue source, dont le travail se limiterait à l'identification des n-grammes grammaticalement corrects et plausibles. Cela mènerait toutefois à des situations où des traductions grammaticalement correctes mais sémantiquement incorrectes pourraient être validées. Afin de limiter l'impact négatif de telles annotations erronées sur les performances de la pré-post-édition, une solution pourrait être de désactiver l'extraction de nouvelles tables de traduction. Les annotations ne seraient donc plus modélisées que par des modèles de langue dont l'un des principaux objectif est précisément de contrôler la grammaticalité des hypothèses produites. Malgré la désactivation des tables de traduction, de très nettes améliorations demeureraient envisageables par le biais de la pré-post-édition comme le montrent les expériences décrites dans la section 6.3.1.

Dans ce travail, nous nous sommes attaché à conserver une forme d'interaction la plus simple possible, ce qui limite en conséquence le type d'informations que notre approche peut exploiter. Il pourrait donc être envisageable de complexifier le mode d'interaction

offert pour donner plus de possibilités au pré-post-éditeur, ce qui pourrait inclure la sélection explicite des tokens incorrects, l'indication du type d'opération d'édition à effectuer (token à supprimer, remplacer, déplacer), la possibilité d'opérer certains déplacements des séquences de tokens, ou encore la capacité à invalider une séquence de tokens validés au cours d'une itération précédente.

La Figure 6.6 résume le fonctionnement général de la pré-post-édition.

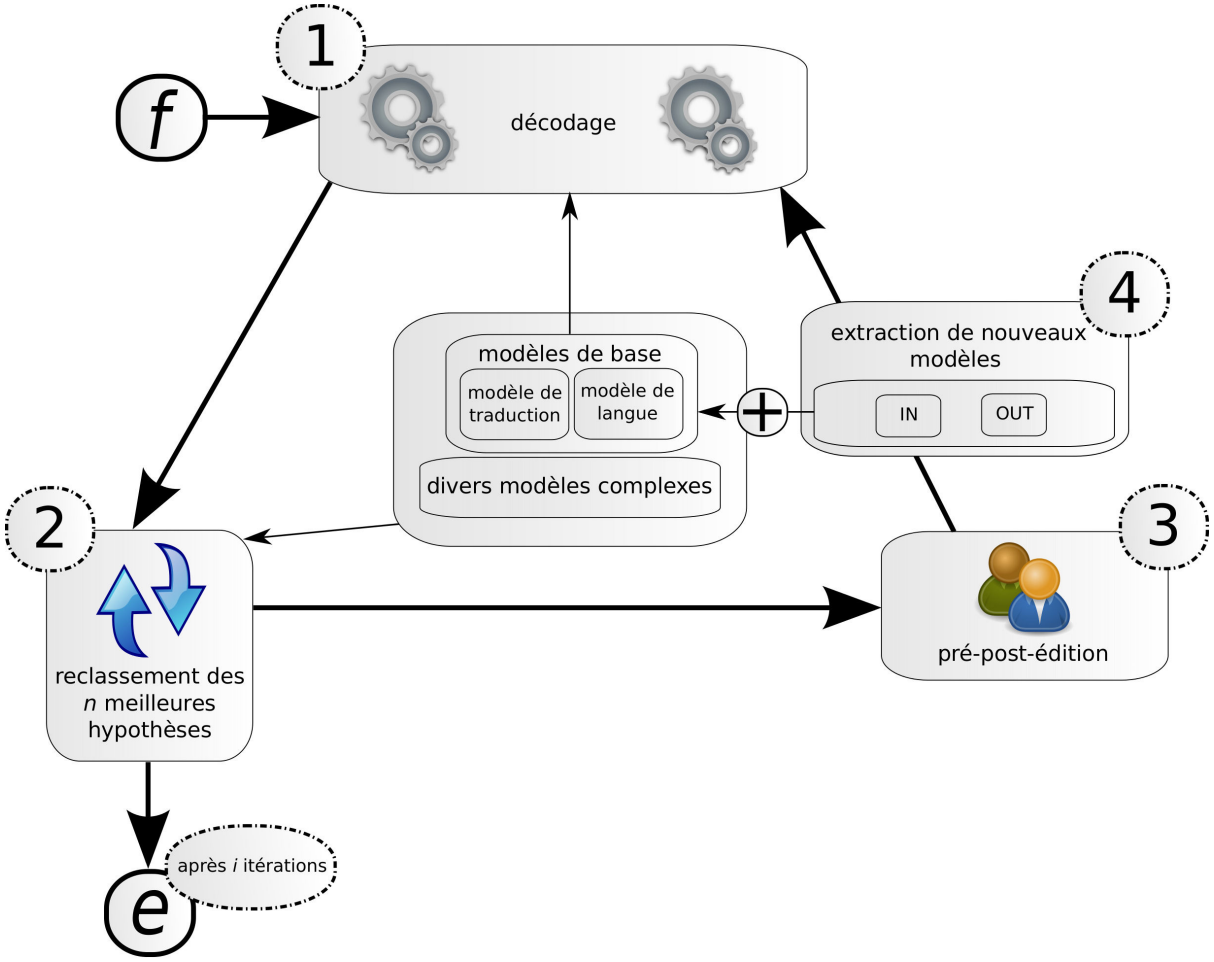


FIGURE 6.6 – Résumé du fonctionnement d'une chaîne de traduction incluant une étape de pré-post-édition. Une phrase source f est décodée en utilisant les modèles de base du décodeur (1). Le décodeur produit la liste des meilleures hypothèses qui est reclassée à l'aide des modèles de base associés à des modèles complexes (2). Une première pré-post-édition est effectuée par des humains indiquant les séquences de tokens valides de la traduction donnée par le système de reclassement (3). De nouveaux modèles sont extraits à partir des annotations ainsi obtenues (4) et sont utilisés lors d'un nouveau décodage de la phrase f. Les étapes (1), (2), (3) et (4) sont répétées pour un nombre d'itérations fixé à l'avance. À chaque itération l'étape de reclassement (3) est effectuée sur la concaténation de toutes les listes de n-meilleures hypothèses produites lors des itérations précédentes. Le traduction e est produite par un dernier reclassement d'hypothèses.

7

Conclusion

Dans ce dernier chapitre, nous allons brièvement résumer l'ensemble des contributions présentées dans ce manuscrit et discuter leur impact sur la traduction automatique statistique. Pour aller plus loin, nous présenterons également un ensemble de perspectives et d'améliorations possibles à nos travaux.

Contributions

Les traductions automatiques produites par des systèmes statistiques à l'état de l'art sont encore aujourd'hui très imparfaites. L'une des raisons de cette imperfection trouve son origine dans le décodage au cours duquel seules des informations relativement simples sont utilisées pour modéliser et produire une traduction en un temps raisonnable. Des informations plus riches, par exemple de type grammatical ou sémantique, mais également plus complexes à calculer pourraient permettre une meilleure modélisation des traductions et par conséquent améliorer la qualité des traductions identifiées par un système. Cependant, de telles informations riches ne peuvent souvent pas être utilisées au cours du décodage sans une modification profonde des heuristiques de parcours de l'espace de recherche du décodeur et un allongement significatif des temps de calcul nécessaires à la production d'une traduction. Parmi les types de modèles qui s'avèrent trop complexes à exploiter au cours du décodage, nous nous sommes focalisé sur l'utilisation de modèles (a) nécessitant une hypothèse complète pour être calculés, (b) trop coûteux pour être calculés sur les nombreuses hypothèses partielles évaluées au cours du décodage, ou encore (c) construits à partir d'informations produites après un premier décodage.

Ces modèles complexes, puisqu'ils ne peuvent pas être utilisés au cours du décodage,

sont donc utilisés *a posteriori*, typiquement pour reclasser la liste des n meilleures hypothèses de traduction produites par le décodeur. Cette approche de reclassement d'hypothèses est aujourd'hui très employée en TAS pour utiliser des modèles complexes. Pour autant, cette approche possède des limites que nous avons pu nous-même mettre en évidence dans ce travail. Elle présente toutefois de bonnes garanties d'amélioration et surtout permet l'utilisation de modèles complexes de différents types, au contraire d'autres approches telles que celles proposées par Chen *et al.* (2008a,b, 2010) et qui ont recours à de multiples redécodages qui ne peuvent bénéficier que d'informations riches très particulières calculées *a posteriori* sur l'espace de recherche du décodeur.

Dans ce manuscrit, afin de contourner les limites de ces solutions, nous avons introduit et évalué plusieurs approches qui permettent l'amélioration de traductions *a posteriori*. Nos différentes propositions ouvrent la voie à une meilleure utilisation de modèles complexes exploitant des informations riches de différents types. Nous pouvons en tirer, principalement, les contributions résumées dans les paragraphes suivants :

Mise en évidence du potentiel d'amélioration de la traduction par un algorithme de recherche locale
Une recherche locale oracle, effectuée à partir de la meilleure traduction produite par un décodeur et guidée par une fonction de score parfaite, permet une amélioration très importante de la qualité de la traduction réécrite. La recherche locale, lorsqu'elle est guidée par une fonction de score qui est meilleure que celle du décodeur, a donc le potentiel d'améliorer une traduction.

Recherche locale exploitant des modèles complexes pour améliorer la traduction produite par un système de reclassement d'hypothèses exploitant ces mêmes modèles complexes
Une recherche locale guidée par une fonction de score utilisant les mêmes modèles complexes que le système de reclassement ayant servi à produire sa traduction « amorce » permet une amélioration significative de cette traduction « amorce ». Les meilleures traductions ainsi produites sont dans le voisinage de la meilleure hypothèse du système de reclassement mais souvent absentes des n meilleures hypothèses reclassées. Utiliser une table de réécriture largement réduite issue de l'extraction des bisegments utilisés par les n meilleures hypothèses produites par le décodeur permet à la recherche locale de produire de petits voisinages qui peuvent être évalués raisonnablement rapidement en dépit de la complexité des modèles utilisés, tout en offrant de bonnes garanties d'amélioration. Simple à implémenter, améliorant la traduction et relativement rapide, une telle recherche locale donne la possibilité de contourner les limites d'un système de reclassement d'hypothèses, tout en ne requérant pas l'accès à de nouveaux modèles, et pourrait donc systématiquement venir le compléter lorsqu'il s'agit de construire un système de référence de TAS fondée sur les segments.

Décodage à passes multiples guidé par des modèles complexes et une recherche locale

L'utilisation lors du décodage de tables de traduction calculées sur la simple différence entre la meilleure hypothèse du décodeur et celle d'un système de reclassement d'hypothèses exploitant des modèles complexes permet au décodeur de produire une liste de meilleures hypothèses améliorée qui s'avère plus adaptée au reclassement à l'aide de ces mêmes modèles complexes. Un reclassement de l'ensemble des hypothèses produites à l'issue de plusieurs itérations d'un décodage à passes multiples, expoitant de nouvelles tables de traduction de ce type, permet d'obtenir une meilleure traduction que celle obtenue lors du reclassement initial. De surcroît , calculer, puis utiliser au décodage, des tables de traduction sur la différence entre la meilleure hypothèse produite par le décodeur et celle obtenue après réécriture par recherche locale de la meilleure hypothèse donnée par un système de reclassement, permet au décodage à passes multiples de produire une traduction encore meilleure, en contre-partie d'un temps de calcul nettement plus important.

Pré-post-édition réduisant le coût d'une post-édition
Nous avons également montré qu'un décodage à passes multiples mieux informé, par exemple au moyen d'annotations fournies par un humain, peut améliorer les traductions de manière très importante. Cette observation permet d'envisager que des modèles complexes mieux informés que ceux que nous avons exploités dans ce manuscrit pourraient donc permettre d'obtenir une amélioration de la traduction plus importante encore. De plus, dans le cadre expérimental que nous avons proposé nous avons pu décrire un nouveau mode d'interaction entre un humain et le décodeur. Cette interaction, simplifiée à l'extrême, consiste pour l'humain à valider au doigt, par exemple sur un écran tactile, les séquences de tokens qu'il juge comme probablement correctes. Nous avons appelé ce mode d'intéraction *pré-post-édition* car il permettrait d'améliorer la traduction en amont d'une possible post-édition et aurait donc comme effet majeur le fait d'en réduire significativement le coût. À large échelle, des gains de temps significatifs pour la traduction de nombreux documents permettraient, à coût constant, de rendre accessibles de nombreux documents dans de nouvelles langues.

Perspectives

Les travaux présentés dans ce manuscrit ont permis de développer de nouvelles méthodes réalisant une meilleure exploitation de modèles complexes qui n'étaient pas au cours d'un décodage initial. Les analyses de nos différents résultats nous ont permis de mettre en évidence les limites de nos propositions, mais également de faire émerger des pistes d'amélioration possibles. Le reste de ce chapitre de conclusion est consacré au développement des principales pistes d'amélioration qui, bien que prometteuses, n'ont pu être traitées dans ce travail.

Utiliser des mesures de confiance pour mieux guider la recherche locale
Dans la section 4.2.3.3, nous avions vu par l'intermédiaire d'expériences *semi-oracle*

qu'une mesure de confiance parfaite au niveau des bisegments permet à la recherche locale guidée par des modèles complexes de produire une meilleure traduction. La recherche locale était alors empêchée de réécrire les segments indiqués comme corrects par de telles mesures, mais fonctionnait par ailleurs sans indications supplémentaires pour les bisegments restants. Une telle mesure de confiance pourrait être calculée automatiquement en utilisant les résultats de travaux sur l'estimation de confiance au niveau des mots [1]. À partir de ces estimations de confiance, il pourrait en effet être envisageable d'en déduire une estimation de confiance au niveau des segments pour indiquer à la recherche locale quels bisegments doivent être réécrits. Une autre manière de procéder consisterait à intégrer ces estimations de confiance directement dans la fonction de score de la recherche locale en tant que nouveau modèle complexe. En plus d'être un piste d'amélioration prometteuse pour permettre à la recherche locale d'améliorer davantage la traduction, empêcher la réécriture de certains bisegments peut potentiellement réduire fortement la taille de l'espace de recherche. Il deviendrait donc moins coûteux d'augmenter la taille du faisceau de la recherche, que nous avions toujours conservée inférieure à 10, ainsi que d'activer de nouveaux types d'opérations.

Activer de nouveaux types d'opérations au cours d'une recherche locale guidée par des modèles complexes

Dans nos expériences présentées dans la section 4.2 nous avions uniquement utilisé les opérations de réécriture `replace`, `split` et `merge`, notamment afin de limiter la taille de l'espace de recherche. En limitant par d'autres biais la taille de l'espace de recherche (par exemple à l'aide de mesures de confiance comme nous l'avons décrit dans le paragraphe précédent), ou en simplifiant, ponctuellement, la fonction de score de la recherche locale, il deviendrait possible d'activer de nouvelles opérations telles que `move` ou `paraphrase` tout en conservant des temps de calcul raisonnables. L'activation d'opérations conjointes telles que l'opération `bi-replace` pourrait également devenir envisageable en vue d'effectuer des réécritures moins locales tout en réduisant les risques de parvenir rapidement à un maximum local de la fonction de score. En outre, il pourrait être possible de décider quel type d'opération doit être appliqué pour chaque bisegment, par exemple en utilisant un classifieur comparable à celui décrit par Bach *et al.* (2011).

Sélectionner les hypothèses de traduction amorces les plus adaptées pour une recherche locale

Nos expériences de recherche locale ont utilisé comme traductions « amorces » les k meilleures hypothèses produites par le décodeur pour alimenter le faisceau de recherche. Nous nous sommes limité dans notre travail à l'utilisation de la seule meilleure hypothèse produite par la décodeur. Cependant, ce choix peut ne pas être le plus adapté pour une recherche locale effectuée au niveau des bisegments. En effet, la segmentation de cette hypothèse peut s'avérer être très particulière et mener rapidement la recherche locale vers un maximum local de la fonction de score. Choisir la ou les traductions amorces possédant les segmentations en bisegments les plus probables *a posteriori* pourrait produire de meilleurs résultats. Une autre possibilité serait au contraire de favoriser la diversité des

1. Pour cela nous pourrions par exemple utiliser les outils d'estimation de confiance issues des travaux de Specia *et al.* (2015); Servan *et al.* (2015).

segmentations des amorces fournies à la recherche locale qui seraient sélectionnées parmi les n meilleures hypothèses produites par le décodeur.

Adapter le système de réécriture fondé sur un algorithme de recherche locale pour un décodage à passes multiples

Comme nous l'indiquions dans la section 5.4, l'ajout d'une passe de recherche locale au cours d'un décodage à passes multiples permet d'atteindre une meilleure traduction mais au prix d'une forte augmentation des temps de calcul. Le système de réécriture utilisé pour réaliser cette recherche locale n'a pas été adapté à ce nouveau cadre d'utilisation, et donc sa configuration est identique à la configuration optimale décrite dans la section 4.2.3. La table de réécriture ou les exemples utilisés pour optimiser la fonction de score de la recherche locale pourraient être adaptés en vue notamment de réduire les temps d'entraînement du système tout en permettant une amélioration plus forte de la traduction. Par exemple, la table de réécriture pourrait être extraite sur la base de l'ensemble des décodages des itérations précédentes et non plus uniquement sur la base du dernier décodage effectué.

Exploiter des informations plus complexes lors de la construction des nouvelles tables de traduction pour le décodage à passes multiples

Notre approche de décodage à passes multiples n'exploite que la différence entre la meilleure hypothèse produite par le décodeur et la meilleure hypothèse donnée par un système de reclassement d'hypothèses (ou une recherche locale). De plus, cette différence n'est utilisée que pour réaliser un partitionnement des tables de traduction sans modifier ou ajouter de scores associés à chaque bisegment. Cette partition pourrait être effectuée en profitant d'informations relatives au reclassement global de la liste d'hypothèses. Par exemple, nous pourrions identifier les bisegments plus ou moins préférés globalement par le système de reclassement et ne plus nous focaliser particulièrement sur ceux se trouvant dans les seules meilleures hypothèses de chaque système. De nouveaux scores à associer à chaque bisegments pourraient par exemple être déduits du reclassement des hypothèses.

Guider le décodeur lors d'un décodage à passes multiples pour produire des listes d'hypothèses plus diversifiées

Lors de nos expériences de décodage à passes multiples nous avions pu observer une diminution dans la production de diversité parmi les n meilleures hypothèses produites par le décodeur au fur et à mesure des itérations (voir section 5.3.1.1. Cette baisse était particulièrement sensible lorsque nous utilisions les deux tables `IN` et `OUT` dans un même décodage. Cette diminution de la diversité peut être attribuée à l'optimisation du décodeur qui apprend trop facilement à reproduire la meilleure hypothèse donnée par le système de reclassement, cette dernière étant meilleure que celle qu'il aurait naturellement produite sans les nouvelles tables de traduction. De façon générale, la table de traduction `IN` est associée à des poids positifs relativement forts, au contraire de la table `OUT` qui obtient des poids négatifs. Le décodeur est donc guidé vers la partie de l'espace de recherche privilégiant les bisegments de la table `IN` et évitant ceux de la table `OUT`. Ainsi, d'itération en itération, la qualité moyenne des hypothèses s'améliore mais celles-ci

deviennent de plus en plus similaires, ce qui n'est pas souhaitable pour l'entraînement discriminant du système de reclassement d'hypothèses. Ce système est entraîné après chaque rédecodage sur des listes d'hypothèses gagnant de moins en moins de diversité, et cela malgré la concaténation des listes d'hypothèses d'une itération à l'autre. Une solution consisterait à forcer la diversification des listes d'hypothèses, notamment pour produire des hypothèses plus éloignées des hypothèses préférées par le système de reclassement. Ces hypothèses pourraient faire office d'exemples *négatifs* et ainsi renforcer le pouvoir discriminant du système de reclassement.

Exploiter de nouveaux modèles complexes

Les améliorations effectuées par les différents systèmes que nous avons proposés sont obtenues grâce à l'utilisation de modèles complexes. Il existe bien d'autres types de modèles non utilisables au cours du décodage et prometteurs que nous aurions pu utiliser dans notre travail. Citons par exemple l'exploitation d'informations au niveau discursif (Carpuat, 2009; Hardmeier *et al.*, 2012), d'analyses sémantiques (Wu et Fung, 2009) ou encore de modèles syntaxiques bilingues (Garmash et Monz, 2015). Nos systèmes permettent une meilleure exploitation de tous ces types de modèles, et ouvrent même possiblement la voie au développement de nouveaux modèles qui seraient spécialement conçus pour une exploitation au cours d'une recherche locale et/ou d'un décodage à passes multiples.

Mettre en pratique la pré-post-édition avec des humains

La pré-post-édition est une approche prometteuse pour améliorer simplement une traduction à l'aide de l'humain. Cependant, nous ne l'avons étudiée ici que dans le cadre d'une simulation où une traduction de référence indique les séquences de tokens correctes. Malgré les très fortes améliorations que nous avons pu observer, la pré-post-édition pose plusieurs défis que nous n'avons pas abordés. Une interface d'annotation des traductions suffisamment ergonomique devrait être développée. Le principal défi identifié serait de présenter les évolutions des traductions d'une itération à l'autre de la manière la plus utile possible. Une étude devrait également être menée afin de mesurer la tolérance aux erreurs d'annotation commises par le pré-post-éditeur qui aurait par exemple considéré un token correct alors que celui-ci ne devrait pas figurer dans la traduction. Sur ce plan, il serait également envisageable de donner la possibilité au pré-post-éditeur de modifier les annotations validées au cours des itérations. Enfin, une traduction pré-post-éditée comportera très probablement des erreurs de traduction de natures différentes relativement à celles contenues dans la traduction avant pré-post-édition. Il sera intéressant d'analyser le comportement du post-éditeur face à ce nouveau type de traductions ainsi que ses effets sur le temps nécessaire à la post-édition.

Appendices

A

Extraits de bicorpus

Dans ces extraits de corpus parallèles, la phrase source est en anglais et précéde la phrase cible en français. Les 15 phrases extraites apparaissent consécutivement dans chaque corpus. Les biphrases sont déjà segmentées en tokens et sont séparées d'une barre horizontale.

A.1 Extrait du bicorpus d'évaluation utilisé pour la tâche `news`

whether you the tracking devices develop into bracelets or pendants , they are still using the same technology as that used for monitoring vehicles .
que l' appareil de traçage soit développé sous la forme de bracelet ou de pendentif , il utilisiera toujours la même technologie que pour tracer des voitures .

---

it is a combination of location detection via GPS (used by regular car navigation) and the functions of a mobile phone , which reports the current vehicle position .
il s' agit d' une combinaison de systèmes d' acquisition d' une position au moyen du GPS (qui est utilisé couramment en autonavigation) et d' une fonction d' un téléphone portable qui alerte la position actuelle d' un véhicule .

---

the army has enough igniter cord to wind the whole of Czech Republic . but they will expire in only two years .
l' armée a tant de cordeaux détonants qu' elle pourrait faire le tour de la République tchèque avec . mais elle ne peut tenir que deux ans .

Czech army has purchased 445 kilometers of cords , which detonate explosives .
l' armée a acheté pour 445 km de cordeaux utilisés pour les explosifs .

---

already last year the Ministry of Defence delivered 224 kilometers of the cords lines into the warehouses .
déjà l' année dernière le Ministère de la Défense en avait acheté 224 km.

---

the army thus has reserves for 225 years .
l' armée a donc des réserves pour 225 ans .

---

the problem is that life of the lines is two to four years .
le problème est que la durée de vie des cordeaux est de deux à quatre ans maximum .

---

the Army purchased the detonating cords from STV Group , Inc. for CZK 40 million .
l' armée a acheté des cordeaux détonants à la société anonyme STV Group pour 40 millions de couronnes .

---

the General Staff provided the reasoning for buying such a huge quantity of igniter cords to MF Dnes stating that soldiers use 60 kilometers of igniter cords in training per year .
l' état-major général a donné comme raison de l' achât d' une énorme quantité de cordeaux détonants à MF Dnes que les soldats en utilisent 60 km par an .

---

calculations of army bomb disposal experts that the MF Dnes approached imply that it would be a very intense pyrotechnic training that would be undergone also by all the generals every year , along with the officers of the army , secretaries and cleaners .
selon les calculs de réserve à détenir faits par les pyrotechniciens de l' armée , que MF Dnes a contacté , il en ressort que même les généraux , les officiers ayant un commandement , secrétaires ou bien les personnels de ménage devraient avoir chaque année un exercice aux explosifs .

Defense Minister Alexandr Vondra and army commander Vlastimil Pick included .
Alexandre Vondra , Ministre de la défense et Vlastimil Picek , commandant de l' armée ne font pas exception .

---

only thus could be the army annual consumption of sixty kilometers of igniter cords justified .
seulement pour argumenter la consommation annuelle de l' armée de 60 km de cordeaux détonants .

---

this period corresponds to 25,000 explosions .
à cette distance correspond 25 000 explosions .

---

and even if it were true , the army reserves would be enough for more than ten years .
et même si c' était vrai , les réserves de l' armée pourraient y subvenir pour 10 ans .

---

however , if the basic training of any new professional actually needed just two meters of the igniter cord , as the experts say , then the army has ensured by the purchase - even with increased consumption by engineer or special units - the stock for the above 225 years .
si au contraire il serait vrai que pour l' entraînement de base de chacun des nouveaux professionnels il suffit au plus de 2 m de cordeaux détonants , comme le disent les experts , cela signifie que l' armée avec son nouvel achat a assuré - avec également la consommation des unités spéciales et du génie - des réserves pour les 225 ans mentionnés .

A.2 Extrait du bicorpus d'évaluation utilisé pour la tâche médical

this trial compares a treated group of patients with an untreated group of patients .
cet essai compare un groupe de patients traités avec un groupe de patients non traités .

---

people with TSC1-related tuberous sclerosis complex are born with one mutated copy of the TSC1 gene in each cell .
les personnes atteintes de sclérose tubéreuse complexe liée à TSC1 naissent avec une copie mutée du gène TSC1 dans chaque cellule .

---

to assess the qualitative and quantitative toxicities of this regimen .
évaluer la toxicité qualitative et quantitative de ce régime .

---

monoclonal antibodies , such as cetuximab , can block cancer growth in different ways .
les anticorps monoclonaux , comme le cétuximab , peuvent bloquer la croissance du cancer de différentes manières .

---

the investigators want to find out what effects , good or bad , the drugs have on the patient's cancer .
les chercheurs veulent savoir quels sont les effets , positifs ou négatifs , des médicaments sur le cancer du patient .

---

the prevalence of intestinal metaplasia and dysplasia was much higher in areas with high risk for gastric cancer .
la prévalence de la métaplasie intestinale et de la dysplasie était beaucoup plus élevée dans les zones à haut risque de cancer gastrique .

---

patient is , in the opinion of the investigator , willing and able to comply with the study medication regimen and all other study requirements .
selon l' enquêteur , le patient est désireux et capable de se conformer au régime de traitement de l' étude et à toutes les autres exigences de l' étude .

---

prior chemotherapy or radiotherapy within 4 weeks prior to study entry (6 weeks for nitrosoureas and mitomycin C) .
la chimiothérapie ou la radiothérapie dans les 4 semaines avant d' entrer dans l' étude (6 semaines pour les nitrosourées et la mitomycine C) .

---

the patients were followed median 38 (25 - 61) months for the recurrences and had a follow up visit at two years to investigate trends in the alcohol consumption .
les patients ont été suivis pendant une durée médiane de 38 (25 - 61) mois pour les récidives et ils ont eu une visite de suivi après deux ans pour étudier les tendances de la consommation d' alcool .

---

be aware of any risk factors you have , so that you can get prompt diagnosis and treatment if a bile duct becomes blocked .
soyez conscient des facteurs de risque que vous avez de sorte que vous pouvez obtenir un diagnostic et un traitement rapide si un canal biliaire est bloqué .

---

all these patients will be advised , i. e. undergo " behavioural intervention " , against alcohol use in the standard fashion as those in the standard group .
tous ces patients seront informés , c'est-à-dire subiront " une intervention comportementale " , contre la consommation d' alcool dans le mode standard comme ceux dans le groupe standard .

if this happens , a member of the treatment team will provide information about ways of relieving the symptoms and removing the fluid .
dans ce cas , un membre de l' équipe de traitement fournira des informations sur les moyens de soulager les symptômes et d' éliminer le fluide .

---

drugs called anti-emetics can be prescribed to help control nausea and vomiting .
les médicaments appelés anti- émétiques peuvent être prescrits pour aider à traiter les nausées et les vomissements .

---

gastroparesis is one of the most underdiagnosed problems in cancer patients , and often overlooked as a potential etiology of chronic nausea and vomiting .
la gastroparésie est l'un des problèmes les plus sous - diagnostiqués chez les patients souffrant du cancer et on la néglige souvent en tant qu' une étiologie potentielle de nausées et vomissements chroniques .

---

pain severity rating scale : 0 (no pain) , 1 (mild pain) , 2 (moderate pain) , or 3 (severe pain) .
échelle de douleur : 0 (absence de douleur) , 1 (douleur légère) , 2 (douleur modérée) ou 3 (douleur sévère) .

A.3 Extrait du bicorpus d'évaluation utilisé pour la tâche TED Talks

it's separating students from A , B , C , D and so on .
ça consiste à séparer les étudiants par note A , B , C , D etc .

---

and the A students get the tougher curriculum , the best teachers , etc .
et les étudiants A ont le programme le plus difficile , les meilleurs professeurs , etc .

---

well , they took , over a three month period , D level students , gave them A's , told them they were A's , told them they were bright .
alors , ils ont pris , sur une période de trois mois , des étudiants niveau D , leur ont donné des A , leur ont dit qu' ils étaient des A , qu' ils étaient intelligents .

---

and at the end of this three month period , they were performing at A level .
et à la fin de la période de trois mois , ils avaient des résultats de niveau A .

---

and , of course , the heartbreaking , flip side of this study , is that they took the A students and told them they were D's .
et , bien sûr , le côté navrant , le revers de cette étude , c' est qu' ils avaient pris les étudiants A et leur avaient dit qu' ils étaient de niveau D .

---

and that's what happened at the end of that three month period .
et c' est ce qui s' est passé à la fin de ces trois mois .

---

those who were still around in school , besides the people who had dropped out .
ceux qui étaient dans le coin , à l' école , en plus de ceux qui avaient abandonné .

---

a crucial part of this case study was that the teachers were duped too .
un point essentiel de cette étude de cas est que les professeurs étaient également dupes .

---

the teachers didn't know a switch had been made .
les professeurs ne savaient pas qu' on avait fait un échange .

---

they were simply told these are the A students , these are the D students .
on leur disait seulement voici les étudiants A , voici les étudiants D .

---

and that's how they went about teaching them and treating them .
et c' est la manière dont ils se sont mis à leur enseigner et à les traiter .

---

so , I think that the only true disability is a crushed spirit , a spirit that's been crushed doesn't have hope .
alors , je pense que la seule véritable infirmité c' est un esprit écrasé , un esprit qui est écrasé n' a pas d' espoir .

---

it doesn't see beauty .
il ne voit pas la beauté .

---

it no longer has our natural , childlike curiosity and our innate ability to imagine .
il n' a plus notre curiosité naturelle , enfantine et notre capacité innée à imaginer .

---

if instead , we can bolster a human spirit to keep hope , to see beauty in themselves and others , to be curious and imaginative , then we are truly using our power well .
si à la place , on peut soutenir un esprit humain pour qu' il garde espoir , qu' il voit la beauté en lui-même et dans les autres , qu' il soit curieux et imaginatif , alors nous utilisons notre pouvoir à bon escient .

B

Exemples de tokens hors-vocabulaire

Dans cette partie des Annexes, 100 tokens hors-vocabulaire (i.e. absents des corpus parallèles utilisés pour l'apprentissage du modèle de traduction, ou pour lesquels aucune traduction n'apparaît dans les modèles de traduction) sont extraits aléatoirement parmi ceux présents dans les corpus d'évaluation de chaque tâche et pour les deux langues anglais et français. Les tokens listés sont séparés par des espaces.

B.1 Tokens hors-vocabulaire du corpus d'évaluation de la tâche `news`

anglais : milions Indem Bartholdy Celestine Carod-Rovira BANESTO Tanveer challanges Túñez BytyOKD.cz Allegrone anti-China classy ecoactivist phylo-sovereignty Janousek's Guereini Beyrle 2018-2022 Psrty amd recompose Mansfield-et-Pontefract Pravo Taschenfederkern Alliot-Marie. inching Xabi single-L-159 Twinsaver D ?cký 14-inch bicycle-drawn Strapá ? hundred-million Gortari Villas Esquerra Adelaida gelatiere Hasse's castor-oil Noia fernet Rayados Ing-wen Avner INDEC Anke Belgium-The Kaucky Aubagne Karabadjakian Mandolin recetly eveneing Hilbeck Prokopova Fernández's mayoralties Astro Vetell Farmet Banteay Dair Kakavand quips Veng Sakozy Absurdities previosu bristle-end cinemaphile avenger inclduing micro-robots L.A.-based Tepatlaxco Fernet Arch.Design pointsover houshould ?erná Walhalla Khakavgand whist filtrations WikiLeaks. DISCARDED Balthazar Rosicrucians Bakira dreamy artists'sketch sombrely side-walks Robrt existece rioted XM-25

français : trafique Catalana Playskool déguerpissiez Dávalos kirchnerisme Actuel Raffin Youk octet's Castelldefels étaitent Krol désintoxciation roulables Appassionatamente Hulda macroportions edging térébrante épuisez happening acool Muki calcules cordeux proliférant cheque Vetell aèrent Sterboholska iXL Klimes vocalistes perçantes Havirovaska Gomà shou INDEC Motol dansdes découpant Barrientos erraient Hlinsky épigraphe Bieniawski Papy Mages Cleopatra Hayatou consumus antiprix Sleng CIU basically Gracia vieile Axor Externe Esquerra Cervantes' Efsa Webkinz Efficiente Sor IPAD

Cerna décison chalco Monteagudo grâve Lyoner cinéphilique Adelaida Ludek akc fils' 295M figeaient élégir municipe Lubomira Jumapam Puffle Milesovice carodistes questionnaient Gortari combiens Bartoli Junaid Mads eXcentris Marienbad prologiciel remuera Coronado sorbetières débarassera

B.2 Tokens hors-vocabulaire du corpus de test de la tâche `médical`

anglais : consented pre-shock alpha-globin LHON-causing fluoroscopy-guided cyst-enteric COL9A1 furious SL-SAT Meytiv Dermatologists Desk OrthoPilot perkier RECIST Triage intralipid fiber-rich sprinting dysphagic Trifecta HNRCA terrible HGG CRS-100 pregestational Assaf Australians witnessed volume-overload dumb Harofeh EMBASE misdistribution Conrad MOST non-shockable isn't multi-country surveyed NEFA pharyngocise P5'N cancer-killing symptom-triggered TIWNB Allina's Marshfield fruste GORD Healthy-Skin-Today. polytherapy non-khat Diabetiques Allina cTn physician-performed Profiler Fig3a Britton NQMC nonvalvular Navigation CT327 Columbus caseload fixed-schedule videofluoroscopy model-independent nurse-run lipophagia virus-inducted Mikes interrelations -defined Mirna double-duct Limberg okay tolvaptan attender liver-based scant anticoagulate NKF-DOQI PVE misnomer CC-IFA Researchers NetWellness BestBets standard-of-care neo-aortic Help diabetesaustralia fibrillation/ MCE Cipro protocol-required nonrandomized

français : Programming combinera bicoronales autolimitante PIMS Free COL9A1 PIIINP allé Australien 115x stressante sialurie paraissait minent Q16OZ diabetesaustralia infructueuse discordants aurais nocosomiale soutiendrait périnéphrétiques Mikes Guided glutéal PLUS exigeait aidez albiglutide COL9A3 cTn aveuglé tolvaptan gratuite plasticiens offrons DMOP finiront prégestationnel Abstracts Tufts naissent testent améliorerons adénodétecteur évalueront Panel RandyAmy SLD check schizocytes Diabétiques fournissons réservent nuits croient Help Today SEC23B faisons nuirait discorde étudions check-up néerlandais chlordiazepoxide Clinic Cipro ofatumumab lipophagie facelifting voyait Homecare CMT1 Retinopathy Nurse vidéofluoroscopie Maha confirmerait Serevent essaient comparons TAVI gold DKA muette Pediatric simulacre MAHA tronçonneuse évaluons erythrocytaphérèse VBH résumera mentionnera MEDLINE retest HGG a-t-il

B.3 Tokens hors-vocabulaire du corpus de test de la tâche `TED Talks`

anglais : methodologically Nathaniel's Knows procured handicapped self-motivation Innovator Thorazine post-human disgraceful Skillman Cassava's monoculture idea's devaluing repurposed tar-like Krispies chemosynthesis Chez Dembois farm's Awareness laurels Delaware Choice world-saving antioxidant game-play aspired jockey bots adjudicate Kahneman burdened gimmicky seared soft-surface Malcom busted panting Factor Halloween Offit Miguel's jiggling affliction 1900's astrobiologists foodie unmarked endowed Vinod spooled bushel pitfalls synopsis enemas goody-two ploy vu 10-12 Winston mosquitos Black-Shouldered purifies salmons ILM mid-area redemptive attentuation erudite Bruckner non-economic Herodotus aspirin sprig algaes beetroot southwestern Salonen pre-cosmonaut self-righteous insisting deploying raggedy castrated Scout deadening Beethoven's planetary-scale evaluation kingdom-wide sixty textured counted-out monocultures Inc. marshmallowchallenge.com halibuts

français : +20 changeront poursuivent +1 Hershey handicapant saignerait célébrée retapé antirétroviraux réceptif gélule émette boiteuses A92 renvoient nettoierais Crocodile YMCA prévisionnel choper loques CM2 Esa Orchestre adminis-

trez brevettent stériliser Terrapower contrebassiste domine virtuose filmées marcheriez charnu encyclopédique Tchaïkovski ultérieures gélules Pivot Bjorn quatorzième assistaient usé festoyer pêché Salonen planétologistes navrant intransigeant assouvi Spaghetti prouvait inspirons accidenté monoculture hypnotisé délires Bruckner divisèrent tissons daaa célébrait Innovateur redire intermittence post- diphtérie flamants confrontons paumés Sports oublieraient Lydie contaminants cyborgs possédiez tromperie enlacer filtrant aggravation complaire résumés goûtée arpèges Azeroth inscrivez Guadalquivir catch Delaware Essence Offit spaghettis tronçonneuses népotiques améliorera nichent optimum Mondiales educe

C

Exemples de traductions produites

Dans cette partie des Annexes, nous présentons un ensemble de traductions produites sur le corpus d'évaluation de la tâche `news` en→fr par les systèmes suivants :

- `moses` : la meilleure hypothèse produite par le décodeur utilisé (voir section 3.1)
- `reranker` : la meilleure hypothèse obtenue après reclassement des 1 000 meilleures hypothèses produites par `moses` (voir section 3.3.2)
- `rewriter` : la meilleure hypothèse produite par la meilleure configuration du système de réécriture basé sur un algorithme de recherche locale (voir section 4.2)
- `multi-pass` (+`rewriter`) : la meilleure hypothèse produite par le décodage à passes multiples effectuant une passe de `reranker` puis `rewriter` après chaque décodage (voir section 5.4)
- PPE (+`reranker`) : la meilleure hypothèse produite après 5 itérations de pré-post-édition simulée effectuant une passe de `reranker` après chaque décodage (voir section 6.3.2)
- `GOS` : la meilleure hypothèse produite par une recherche locale oracle, avec toutes les opérations activées, dont l'hypothèse amorce est la meilleure hypothèse produite par `moses` (voir section 4.1)

Parmi ces systèmes, `reranker`, `rewriter`, `multi-pass` (+`rewriter`) et PPE (+`reranker`) utilisent le même ensemble de modèles complexes que celui présenté dans la section 5.2.1. Les exemples sont présentés selon leur ordre d'apparition dans le corpus et ont été selectionnés manuellement (notamment en fonction de la longueur des hypothèses) afin d'illustrer les améliorations et dégradations qu'il est possible d'obtenir avec ces systèmes, parfois sans corrélation avec l'évolution du score sBLEU (également indiqué ici).

	*s*BLEU	
source		*also railways and airports must are facing the problems .*
`moses`	12,52	aussi les chemins de fer et les aéroports sont confrontés aux problèmes .
`reranker`	11,57	les chemins de fer et les aéroports doivent aussi sont confrontés aux problèmes .
`rewriter`	13,13	les chemins de fer et les aéroports sont également confrontés aux problèmes .
`multi-pass` (+`rewriter`)	16,59	aussi les chemins de fer et les aéroports sont confrontés à des problèmes .
PPE (+`reranker`)	34,24	les voies ferrées et les aéroports , il y a des problèmes .
`GOS`	70,37	également des les voies ferrées et l' aéroport sont .
référence		les voies ferrées et l' aéroport rencontrent également des problèmes .
source		*fortunately , tables make the calculations easier .*
`moses`	51,00	heureusement , les tableaux de faire les calculs .
`reranker`	26,75	heureusement , les calculs des tableaux rendent plus faciles .
`rewriter`	40,15	heureusement , les tableaux rendent les calculs plus faciles .
`multi-pass` (+`rewriter`)	40,15	heureusement , les tableaux rendent les calculs plus faciles .
PPE (+`reranker`)	40,96	heureusement , les tableaux , les facilite , les calculs .
`GOS`	58,15	heureusement , les tableaux faire les calculs .
référence		heureusement , les tableaux facilitent les calculs .
source		*Nigeria has just one refinery of its own .*
`moses`	20,24	le Nigeria vient d' une raffinerie de ses propres .
`reranker`	18,25	le Nigeria vient d' une raffinerie de son propre pays .
`rewriter`	20,16	le Nigeria n' a qu' une seule raffinerie de son propre pays .
`multi-pass` (+`rewriter`)	18,25	le Nigeria vient d' une raffinerie de son propre pays .
PPE (+`reranker`)	27,81	le Nigeria une seule raffinerie d' un seul .
`GOS`	50,97	le Nigeria vient d' une seule raffinerie . d'
référence		le Nigeria ne dispose que d' une seule raffinerie .
source		*Singapore Airlines temporarily kept about three of its eleven A380 on the ground .*
`moses`	37,42	Singapore Airlines temporairement conservés sur trois de ses onze A380 sur le terrain .
`reranker`	37,42	Singapore Airlines temporairement conservées environ trois de ses onze A380 sur le terrain .
`rewriter`	46,04	Singapore Airlines a temporairement maintenu environ trois de ses onze A380 sur le terrain .
`multi-pass` (+`rewriter`)	46,04	Singapore Airlines a temporairement maintenu environ trois de ses onze A380 sur le terrain .
PPE (+`reranker`)	72,89	Singapore Airlines a temporairement tenues trois de ses onze A380 .
`GOS`	77,68	Singapore Airlines a temporairement sur trois de ses onze A380
référence		Singapore Airlines a temporairement immobilisé trois de ses onze A380 .

	*s*BLEU	
source		*neither version is compatible with older cartridges .*
`moses`	24,12	aucune des deux versions est compatible avec plus de cartouches .
`reranker`	42,93	aucune des deux versions n' est compatible avec les vieux cartouches .
`rewriter`	65,90	aucune des deux versions n' est compatible avec les anciennes cartouches .
`multi-pass` (+`rewriter`)	65,90	aucune des deux versions n' est compatible avec les anciennes cartouches .
PPE (+`reranker`)	100	aucune version n' est compatible avec les anciennes cartouches .
`GOS`	100	aucune version n' est compatible avec les anciennes cartouches .
référence		aucune version n' est compatible avec les anciennes cartouches .
source		*one squeeze equals one note , and a second squeeze starts building a song .*
`moses`	17,64	une pression égale un billet , et un deuxième resserrement commence la construction d' une chanson .
`reranker`	19,20	un rétrécissement équivaut à une note et un deuxième resserrement commence à construire une chanson .
`rewriter`	19,72	une compression équivaut à une note et un deuxième resserrement commence à construire une chanson .
`multi-pass` (+`rewriter`)	40,37	une compression équivaut à une note , et une seconde compression commence à construire une chanson .
PPE (+`reranker`)	71,31	une pression est égale à une note , et une deuxième rétrécissement commencera une chanson .
`GOS`	79,18	une pression est égale à une note , et une deuxième la construction d' une chanson .
référence		une pression est égale à une note , et une deuxième permet de composer une chanson .
source		*instead , he said , a period of disagreement is likely , even inevitable .*
`moses`	56,27	au lieu de cela , a -t-il dit , une période de désaccord est probable et même inévitable .
`reranker`	80,64	au lieu de cela , dit -il , une période de désaccord est probable et même inévitable .
`rewriter`	100	au lieu de cela , dit -il , une période de désaccord est probable , voire inévitable .
`multi-pass` (+`rewriter`)	100	au lieu de cela , dit -il , une période de désaccord est probable , voire inévitable .
PPE (+`reranker`)	100	au lieu de cela , dit -il , une période de désaccord est probable , voire inévitable .
`GOS`	100	au lieu de cela , dit -il , une période de désaccord est probable , voire inévitable .
référence		au lieu de cela , dit -il , une période de désaccord est probable , voire inévitable .

D

Publications de l'auteur

Publications dans le cadre du travail de thèse

Marie, B. and Max, A. (2015). Touch-Based Pre-Post-Editing of Machine Translation Output. In EMNLP 2015, Lisbon, Portugal.

Marie, B. and Max, A. (2015). Multi-Pass Decoding With Complex Feature Guidance for Statistical Machine Translation. In ACL-IJCNLP 2015, Beijing, China.

Marie, B., Max, A. (2014). Confidence-based Rewriting of Machine Translation Output. In EMNLP 2014, Doha, Qatar.

Marie, B. and Max, A. (2013). A Study in Greedy Oracle Improvement of Translation Hypotheses. In IWSLT 13, Heidelberg, Germany.

Autres publications

Marie, B., Allauzen, A., Burlot, F., Do, Q. K., Ive, J., Knyazeva, E., Labeau, M., Lavergne, T., Löser, K., Pécheux, N., Yvon, F. (2015). LIMSI@WMT'15 : Translation Task. In WMT'15, Lisbon, Portugal.

Marie, B. and Apidianaki, M. (2015). Alignment-based sense selection in METEOR and the RATATOUILLE recipe. In WMT'15, Lisbon, Portugal.

Apidianaki, M., Marie, B. (2015). METEOR-WSD : Improved Sense Matching in MT Evaluation. In SSST-9, Denver, US.

Pécheux, N., Gong, L., Do, Q. K., Marie, B., Ivanishcheva, Y., Allauzen, A., Lavergne, T., Niehues, J., Max, A., Yvon, F. (2014). LIMSI @ WMT'14 Medical Translation Task. In WMT'14, Baltimore, US.

Bibliographie

Allauzen, A., Pécheux, N., Do, Q. K., Dinarelli, M., Lavergne, T., Max, A., Le, H.-s. et Yvon, F. (2013). LIMSI @ WMT13. Dans *Proceedings of WMT*, Sofia, Bulgaria.

Allauzen, A. et Yvon, F. (2011). Méthodes statistiques pour la traduction automatique. Dans Gaussier, E. et Yvon, F., éditeurs : *Modèles statistiques pour l'accès à l'information textuelle*, chapitre 7. Hermès, Paris.

Apidianaki, M. et Marie, B. (2015). METEOR-WSD : Improved Sense Matching in MT Evaluation. Dans *Proceedings of SSST-9*, Denver, USA.

Arun, A., Dyer, C., Haddow, B., Blunsom, P., Lopez, A. et Koehn, P. (2009). Monte carlo inference and maximization for phrase-based translation. Dans *Proceedings of CoNLL*, Boulder, Colorado.

Axelrod, A., He, X. et Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. Dans *Proceedings of EMNLP*, Edinburgh, UK.

Aziz, W., Dymetman, M. et Specia, L. (2014). Exact decoding for phrase-based statistical machine translation. Dans *Proceedings of EMNLP*, Doha, Qatar.

Bach, N., Huang, F. et Al-Onaizan, Y. (2011). Goodness : a method for measuring machine translation confidence. Dans *Proceedings of ACL-HLT*, Portland, Oregon.

Bangalore, S. (2001). Computing consensus translation from multiple machine translation systems. Dans *Proceedings of ASRU*, Madonna di Campiglio, Italy.

Birch, A., Osborne, M. et Koehn, P. (2007). Ccg supertags in factored statistical machine translation. Dans *Proceedings of WMT*, Prague, Czech Republic.

Bisazza, A., Ruiz, N., Federico, M. et Kessler, F.-F. B. (2011). Fill-up versus interpolation methods for phrase-based smt adaptation. Dans *Proceedings of IWSLT*, San Francisco, USA.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A. et Ueffing, N. (2004). Confidence estimation for machine translation. Dans *Proceedings of COLING*, Geneva, Switzerland.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L. et Turchi, M. (2015). Findings of the 2015 workshop on statistical machine translation. Dans *Proceedings of WMT*.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L. et Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2).

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D. et Mercer, R. L. (1993). The mathematics of statistical machine translation : Parameter estimation. *Computational Linguistics*, 19(2).

Carl, M., Barbara, D., Elming, J., Daniel, H. et Lykke, J. A. (2011). The Process of Post-Editing : A pilot study. *Copenhagen Studies in Language*.

Carl, M. et Way, A. (2003). *Recent Advances in Example-Based Machine Translation*. Text, Speech and Language Technology. Springer Netherlands.

Carpuat, M. (2009). One translation per discourse. Dans *Proceedings of the Workshop on Semantic Evaluations : Recent Achievements and Future Directions*, Boulder, Colorado.

Carter, S. et Monz, C. (2011). Syntactic discriminative language model rerankers for statistical machine translation. *Machine Translation*, 25(4).

Cer, D., Manning, C. D. et Jurafsky, D. (2010). The best lexical metric for phrase-based statistical mt system optimization. Dans *Proceedings of NAACL*, Los Angeles, USA.

Cer, D., Manning, C. D. et Jurafsky, D. (2013). Positive diversity tuning for machine translation system combination. Dans *Proceedings of WMT*, Sofia, Bulgaria.

Chatterjee, R., Weller, M., Negri, M. et Turchi, M. (2015). Exploring the planet of the apes : a comparative study of state-of-the-art methods for mt automatic post-editing. Dans *Proceedings of ACL*, Beijing, China.

Chatterjee, S. et Cancedda, N. (2010). Minimum error rate training by sampling the translation lattice. Dans *Proceedings of EMNLP*, Cambridge, USA.

Chen, B. et Cherry, C. (2014). A systematic comparison of smoothing techniques for sentence-level bleu. Dans *Proceedings of WMT*, Baltimore, USA.

Chen, B., Federico, M. et Cettolo, M. (2007). Better n-best translation through generative n-gram language model. Dans *Proceeding of MT Summit*, Copenhagen, Denmark.

Chen, B., Foster, G. et Kuhn, R. (2010). Fast Consensus Hypothesis Regeneration for Machine Translation. Dans *Proceedings of WMT*, Uppsala, Sweden.

Chen, B., Zhang, M., Aw, A. et Li, H. (2008a). Exploiting N-best Hypotheses for SMT Self-Enhancement. Dans *Proceedings of ACL*, Columbus, USA.

Chen, B., Zhang, M., Aw, A. et Li, H. (2008b). Regenerating Hypotheses for Statistical Machine Translation. Dans *Proceedings of COLING*, Manchester, UK.

Chen, S. F. et Goodman, J. T. (1998). An empirical study of smoothing techniques for language modeling. Rapport technique TR-10-98, Computer Science Group, Harvard University.

Cherry, C. et Foster, G. (2012). Batch tuning strategies for statistical machine translation. Dans *Proceedings of NAACL-HLT*, Montreal, Canada.

Chevalier, M., Dansereau, J., Poulin, G., T.A.U.M. et de Montréal. Groupe de recherches pour la traduction automatique, U. (1978). *TAUM-MÉTÉO : description du système*. TAUM/Université de Montréal.

Chiang, D., Marton, Y. et Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. Dans *Proceedings of EMNLP*, Honolulu, USA.

Collins, M., Koehn, P. et Kučerová, I. (2005). Clause restructuring for statistical machine translation. Dans *Proceedings of ACL*, Ann Arbor, Michigan.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. et Singer, Y. (2006). Online passive-aggressive algorithms. *JMLR*, 7.

Crego, J. M., Max, A. et Yvon, F. (2010). Local lexical adaptation in Machine Translation through triangulation : SMT helping SMT. Dans *Proceedings of COLING*, Beijing, China.

Déchelotte, D., Adda, G., Allauzen, A., Bonneau-Maynard, H., Galibert, O., Gauvain, J.-L., Langlais, P. et Yvon, F. (2008). Limsi's statistical translation systems for WMT'08. Dans *Proceedings of WMT*, Columbus, USA.

Denkowski, M. et Lavie, A. (2014). Meteor universal : Language specific translation evaluation for any target language. Dans *Proceedings of EACL*, Gothenburg, Sweden.

Denkowski, M. J., Dyer, C. et Lavie, A. (2014). Learning from Post-Editing : Online Model Adaptation for Statistical Machine Translation. Dans *Proceedings of EACL*, Gothenburg, Sweden.

Devlin, J. et Matsoukas, S. (2012). Trait-based Hypothesis Selection for Machine Translation. Dans *Proceedings of NAACL-HLT*, Montréal, Canada.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R. et Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. Dans *Proceedings of ACL*, Baltimore, USA.

Do, Q.-K., Allauzen, A. et Yvon, F. (2015). A discriminative training procedure for continuous translation models. Dans *Proceedings of EMNLP*, Lisbon, Portugal.

Dyer, C., Lopez, A., Ganitkevitch, J., Weese, J., Ture, F., Blunsom, P., Setiawan, H., Eidelman, V. et Resnik, P. (2010). cdec : A decoder, alignment, and learning framework for finite-state and context-free translation models. Dans *Proceedings of ACL*, Uppsala, Sweden.

Federmann, C. (2010). Appraise : An open-source toolkit for manual phrase-based evaluation of translations. Dans *Proceedings of LREC*, Valetta, Malta.

Fellbaum, C. (1998). *WordNet : An Electronic Lexical Database*. Bradford Books.

Foster, G., Gandrabur, S., Langlais, P., Plamondon, P., Russell, G. et Simard, M. (2003). Statistical machine translation : Rapid development with limited resources. Dans *Proceedings of MT Summit*, New Orleans, USA.

Foster, G., Langlais, P. et Lapalme, G. (2002). User-Friendly Text Prediction for Translators. Dans *Proceedings of EMNLP*, Philadelphia, USA.

Galley, M. et Manning, C. D. (2008). A simple and effective hierarchical phrase reordering model. Dans *Proceedings of EMNLP*, Honolulu, USA.

Gao, Q. et Vogel, S. (2008). Parallel implementations of word alignment tool. Dans *Proceedings of SETQA-NLP*, Columbus, USA.

Garmash, E. et Monz, C. (2015). Bilingual structured language models for statistical machine translation. Dans *Proceedings of EMNLP*, Lisbon, Portugal.

Gerlach, J., Porro Rodriguez, V., Bouillon, P. et Lehmann, S. (2013a). Combining pre-editing and post-editing to improve smt of user-generated content. Dans *Proceedings of MT Summit Workshop on Post-editing Technology and Practice*, Nice, France.

Gerlach, J., Porro Rodriguez, V., Bouillon, P. et Lehmann, S. (2013b). La préédition avec des règles peu coûteuses, utile pour la ta statistique des forums ? Dans *Proceedings of TALN*, Les Sables d'Olonne, France.

Germann, U. (2003a). Greedy decoding for statistical machine translation in almost linear time. Dans *Proceedings of HLT-NAACL*, Edmonton, Canada.

Germann, U. (2003b). Greedy decoding for statistical machine translation in almost linear time. Dans *Proceedings of NAACL-HLT*, Edmonton, Canada.

Germann, U., Jahr, M., Knight, K., Marcu, D. et Yamada, K. (2001a). Fast decoding and optimal decoding for machine translation. Dans *Proceedings of ACL*, Toulouse, France.

Germann, U., Jahr, M., Knight, K., Marcu, D. et Yamada, K. (2001b). Fast decoding and optimal decoding for machine translation. Dans *Proceedings of ACL*, Toulouse, France.

Gimpel, K., Batra, D., Dyer, C., Shakhnarovich, G. et Tech, V. (2013). A Systematic Exploration of Diversity in Machine Translation. Dans *Proceedings of EMNLP*, Seatlle, USA.

Gimpel, K. et Smith, N. A. (2008). Rich source-side context for statistical machine translation. Dans *Proceedings of WMT*, Columbus, Ohio.

Gispert, A., Blackwood, G., Iglesias, G. et Byrne, W. (2012). N-gram posterior probability confidence measures for statistical machine translation : an empirical study. *Machine Translation*.

Gong, L., Max, A. et Yvon, F. (2014). Incremental Development of Statistical Machine Translation Systems. Dans *Proceedings of IWSLT*, Lake Tahoe, USA.

Goutte, C., Carpuat, M. et Foster, G. (2012). The impact of sentence alignment errors on phrase-based machine translation performance. Dans *Proceedings of AMTA*, San Diego, USA.

Graham, Y., Baldwin, T. et Mathur, N. (2015). Accurate evaluation of segment-level machine translation metrics. Dans *Proceedings of NAACL-HLT*, Denver, USA.

Graham, Y., Baldwin, T., Moffat, A. et Zobel, J. (2014). Is machine translation getting better over time ? Dans *Proceedings of EACL*, Gothenburg, Sweden.

Green, S., Cer, D. et Manning, C. D. (2014a). Phrasal : A toolkit for new directions in statistical machine translation. Dans *In Proceddings of WMT*, Baltimore, USA.

Green, S., Heer, J. et Manning, C. D. (2013a). The Efficacy of Human Post-Editing. Dans *Proceedings of CHI*, Paris, France.

Green, S., Wang, S., Cer, D. et Manning, C. D. (2013b). Fast and adaptive online training of feature-rich translation models. Dans *Proceedings of ACL*, Sofia, Bulgaria.

Green, S., Wang, S., Chuang, J., Heer, J., Schuster, S. et Manning, C. D. (2014b). Human Effort and Machine Learnability in Computer Aided Translation. Dans *Proceedings of EMNLP*, Doha, Qatar.

Hardmeier, C., Nivre, J. et Tiedemann, J. (2012). Document-wide decoding for phrase-based statistical machine translation. Dans *Proceedings of EMNLP-CoNLL*, Jeju Island, Republic of Korea.

Hasler, E., Haddow, B. et Koehn, P. (2011). Margin infused relaxed algorithm for moses. Dans *Prague Bull. Math. Linguistics*, Prague, Czech Republic.

Hildebrand, A. S. et Vogel, S. (2008). Combination of machine translation systems via hypothesis selection from combined n-best lists. Dans *Proceedings of AMTA*, Honolulu, USA.

Hopkins, M. et May, J. (2011). Tuning as ranking. Dans *Proceedings of EMNLP*, Edinburgh, UK.

Huang, L. et Chiang, D. (2007). Forest rescoring : Faster decoding with integrated language models. Dans *Proceedings of ACL*, Prague, Czech Republic.

Hutchins, W. (1997). From first conception to first demonstration : the nascent years of machine translation, 1947–1954. a chronology. *Machine Translation*, 12(3).

Hutchins, W. et Somers, H. (1992). *An introduction to machine translation*. Academic Press.

Johnson, H., Martin, J., Foster, G. et Kuhn, R. (2007). Improving Translation Quality by Discarding Most of the Phrasetable. Dans *Proceedings of EMNLP*, Prague, Czech Republic.

Knight, K. (1999). Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4).

Knight, K. et Chander, I. (1994). Automated postediting of documents. *CoRR*, abs/cmp-lg/9407028.

Koehn, P. (2004). Pharaoh : A beam search decoder for phrase-based statistical machine translation models. Dans *Proceedings of AMTA*, Washington DC, USA.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press.

Koehn, P., Axelrod, A., Birch, A., Callison-Burch, C., Osborne, M. et Talbot, D. (2005). Edinburgh system description for the 2005 IWSLT speech translation evaluation. Dans *Proceedings of IWSLT*, Pittsburgh, USA.

Koehn, P., Birch, A. et Steinberger, R. (2009). 462 machine translation systems for Europe. Dans *Proceedings of MT Summit*, Ottawa, Canada.

Koehn, P. et Haddow, B. (2009). Interactive assistance to human translators using statistical machine translation methods. Dans *Proceedings of MT Summit*, Ottawa, Canada.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A. et Herbst, E. (2007). Moses : Open Source Toolkit for Statistical Machine Translation. Dans *Proceedings of ACL*, Prague, Czech Republic.

Koehn, P., Och, F. J. et Marcu, D. (2003). Statistical phrase-based translation. Dans *Proceedings of NAACL*, Edmonton, Canada.

Koehn, P. et Schroeder, J. (2007). Experiments in domain adaptation for statistical machine translation. Dans *Proceedings of WMT*, Prague, Czech Republic.

Kolachina, P., Cancedda, N., Dymetman, M. et Venkatapathy, S. (2012). Prediction of Learning Curves in Machine Translation. Dans *Proceedings of ACL*, Jeju Island, Republic of Korea.

Kurokawa, D., Goutte, C. et Isabelle, P. (2009). Automatic detection of translated text and its impact on machine translation. Dans *Proceedings of MT Summit*, Ottawa, Canada.

Langlais, P., Foster, G. et Lapalme, G. (2000). Transtype : A computer-aided translation typing system. Dans *Proceedings of NAACL-ANLP Workshop on Embedded Machine Translation Systems*, Seattle, USA.

Langlais, P., Gotti, F. et Patry, A. (2007). A Greedy Decoder for Phrase-Based Statistical Machine Translation. Dans *Proceedings of TMI*, Skovde, Sweden.

Le, H.-S., Allauzen, A. et Yvon, F. (2012). Continuous Space Translation Models with Neural Networks. Dans *Proceedings of NAACL*, Montréal, Canada.

Le, H.-S., Oparin, I., Allauzen, A., Gauvain, J.-L. et Yvon, F. (2011). Structured Output Layer Neural Network Language Model. Dans *Proceedings of ICASSP*, Prague, Czech Republic.

Lembersky, G., Ordan, N. et Wintner, S. (2012). Language Models for Machine Translation : Original vs. Translated Texts. *Computational Linguistics*, 38(4).

Lin, C.-Y. et Och, F. J. (2004). Orange : A method for evaluating automatic evaluation metrics for machine translation. Dans *Proceedings of COLING*, Geneva, Switzerland.

Luong, N.-Q., Besacier, L. et Lecouteux, B. (2014a). An Efficient Two-Pass Decoder for SMT Using Word Confidence Estimation. Dans *Proceedings of EAMT*, Dubrovnik, Croatia.

Luong, N.-Q., Besacier, L. et Lecouteux, B. (2014b). Word Confidence Estimation for SMT N -best List Re-ranking. Dans *Proceedings of the Workshop HaCaT*, Gothenburg, Sweden.

Machacek, M. et Bojar, O. (2014). Results of the wmt14 metrics shared task. Dans *Proceedings of WMT*, Baltimore, USA.

Marie, B., Allauzen, A., Burlot, F., Do, Q.-K., Ive, J., knyazeva, e., Labeau, M., Lavergne, T., Löser, K., Pécheux, N. et Yvon, F. (2015). Limsi@wmt'15 : Translation task. Dans *Proceedings of WMT*, Lisbon, Portugal.

McDonald, R., Hall, K. et Mann, G. (2010). Distributed training strategies for the structured perceptron. Dans *Proceedings of HTL-NAACL*, Los Angeles, California.

Mirkin, S., Venkatapathy, S. et Dymetman, M. (2013). Confidence-driven rewriting for improved translation. Dans *Proceedings of MT Summit*, Nice, France.

Nakov, P., Guzmán, F. et Vogel, S. (2012). Optimizing for sentence-level BLEU+1 yields short translations. Dans *Proceedings of COLING*, Mumbai, India.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. Dans *Proceedings of ACL*, Sapporo, Japan.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z. et Radev, D. (2004). A smorgasbord of features for statistical machine translation. Dans *Proceedings of HLT-NAACL*, Boston, USA.

Och, F. J. et Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. Dans *Proceedings of ACL*, Philadelphia, USA.

Och, F. J. et Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Och, F. J., Tillmann, C., Ney, H. et Informatik, L. F. (1999). Improved alignment models for statistical machine translation. Dans *Proceedings of EMNLP-VLC*, College Park, USA.

Och, F. J., Ueffing, N. et Ney, H. (2001). An efficient a* search algorithm for statistical machine translation. Dans *Proceedings of the ACL Workshop on Data-driven Methods in Machine Translation*, Toulouse, France.

Onishi, T., Utiyama, M. et Sumita, E. (2010). Paraphrase Lattice for Statistical Machine Translation. Dans *Proceedings of ACL*, Uppsala, Sweden.

Papineni, K., Roukos, S., Ward, T. et Zhu, W.-J. (2002). Bleu : a method for automatic evaluation of machine translation. Dans *Proceedings of ACL*, Philadelphia, USA.

Parton, K., Habash, N., McKeown, K., Iglesias, G. et de Gispert, A. (2012). Can Automatic Post-Editing Make MT More Meaningful ? Dans *Proceedings of EAMT*, Trento, Italy.

Pécheux, N., Gong, L., Do, Q. K., Marie, B., Ivanishcheva, Y., Allauzen, A., Lavergne, T., Niehues, J., Max, A. et Yvon, F. (2014). LIMSI @ WMT'14 Medical Translation Task. Dans *Proceedings of WMT*, Baltimore, USA.

Post, M., Ganitkevitch, J., Orland, L., Weese, J., Cao, Y. et Callison-Burch, C. (2013). Joshua 5.0 : Sparser, better, faster, server. Dans *Proceedings of WMT*, Sofia, Bulgaria.

Rosa, R., Mareček, D. et Dušek, O. (2012). Depfix : A system for automatic correction of czech mt outputs. Dans *Proceedings of WMT*, Montreal, Canada.

Rosti, A.-V. I., Zhang, B., Matsoukas, S. et Schwartz, R. (2008). Incremental hypothesis alignment for building confusion networks with application to machine translation system combination. Dans *Proceedings of WMT*, Columbus, Ohio.

Schroeder, J., Cohn, T. et Koehn, P. (2009). Word Lattices for Multi-Source Translation. Dans *Proceedings of EACL*, Athens, Greece.

Schwartz, L., Callison-Burch, C., Schuler, W. et Wu, S. (2011). Incremental syntactic language models for phrase-based translation. Dans *Proceedings of ACL-HLT*, Portland, USA.

Servan, C., Ngoc, T. L., Ngoc, Q. L., Lecouteux, B. et Besacier, L. (2015). An open source toolkit for word-level confidence estimation in machine translation. Dans *Proceedings of IWSLT*, Da Nang, Vietnam.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., et Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. Dans *Proceedings of AMTA*, Cambridge, USA.

Snover, M., Madnani, N., Dorr, B. et Schwartz, R. (2009). Ter-plus : paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3).

Specia, L. et Farzindar, A. (2010). Estimating machine translation post-editing effort with hter. Dans *Proceedings of the AMTA Workshop Bringing MT to the User : MT Research and the Translation Industry*, Denver, USA.

Specia, L., Paetzold, G. et Scarton, C. (2015). Multi-level translation quality prediction with quest++. Dans *Proceedings of ACL-IJCNLP*, Beijing, China.

Specia, L., Shah, K., de Souza, J. G. et Cohn, T. (2013). QuEst - A Translation Quality Estimation Framework. Dans *Proceedings of ACL*, Sofia, Bulgaria.

Stanojević, M., Kamran, A., Koehn, P. et Bojar, O. (2015). Results of the wmt15 metrics shared task. Dans *Proceedings of WMT*, Lisbon, Portugal.

Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. Dans *Proceedings of ICSLP*, Denver, USA.

Stroppa, N., Van den Bosch, A. et Way, A. (2007). Exploiting source similarity for SMT using context-informed features. Dans *Proceedings of TMI*, Skövde, Sweden.

Takezawa, T., Sumita, E., Sugaya, F., Yamamoto, H. et Yamamoto, S. (2002). Toward a Broad-coverage Bilingual Corpus for Speech Translation of Travel Conversations in the Real World. Dans *Proceedings of LREC*, Las Palmas, Spain.

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. Dans *Proceedings of HLT-NAACL*, Boston, Massachusetts.

Toutanova, K. et Manning, C. D. (2000). Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger. Dans *Proceedings of EMNLP*, Hong Kong.

Turchi, M., De Bie, T., Goutte, C. et Cristianini, N. (2012). Learning to Translate : A Statistical and Computational Analysis. *Advances in Artificial Intelligence*.

Ueffing, N. et Ney, H. (2007). Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1).

Vaswani, A., Zhao, Y., Fossum, V. et Chiang, D. (2013). Decoding with large-scale neural language models improves translation. Dans *Proceedings of EMNLP*, Seattle, USA.

Vilar, D., Xu, J., D'Haro, L. F. et Ney, H. (2006). Error Analysis of Statistical Machine Translation Output. Dans *Proceedings of LREC*, Genoa, Italy.

Vogel, S., Ney, H. et Tillmann, C. (1996). Hmm-based word alignment in statistical translation. Dans *Proceedings of COLING*, Copenhagen, Denmark.

Watanabe, T., Suzuki, J., Tsukada, H. et Isozaki, H. (2007). Online large-margin training for statistical machine translation. Dans *Proceedings of EMNLP-CoNLL*, Prague, Czech Republic.

Wisniewski, G., Allauzen, A. et Yvon, F. (2010). Assessing phrase-based translation models with oracle decoding. Dans *Proceedings of EMNLP*.

Wisniewski, G. et Yvon, F. (2013). Oracle decoding as a new way to analyze phrase-based machine translation. *Machine Translation*, 27(2).

Wu, D. et Fung, P. (2009). Semantic roles for smt : A hybrid two-pass model. Dans *Proceedings of NAACL*, Boulder, Colorado.

Xia, F. et McCord, M. (2004). Improving a statistical mt system with automatically learned rewrite patterns. Dans *Proceedings of COLING*, Geneva, Switzerland.

Zens, R. et Ney, H. (2006). N-Gram Posterior Probabilities for Statistical Machine Translation. Dans *Proceedings of WMT*, New York City, USA.

Zhu, M., Zhang, Y., Chen, W., Zhang, M. et Zhu, J. (2013). Fast and Accurate Shift-Reduce Constituent Parsing. Dans *Proceedings of ACL*, Sofia, Bulgaria.

Titre Exploitation d'informations riches pour guider la traduction automatique statistique

Mots-clés traduction automatique statistique, modèle complexe, reclassement d'hypothèses, recherche locale, décodage à passes multiples, post-édition

Résumé S'il est indéniable que de nos jours la traduction automatique (TA) facilite la communication entre langues, et plus encore depuis les récents progrès des systèmes de TA statistiques, ses résultats sont encore loin du niveau de qualité des traductions obtenues avec des traducteurs humains.

Ce constat résulte en partie du mode de fonctionnement d'un système de TA statistique, très contraint sur la nature des modèles qu'il peut utiliser pour construire et évaluer de nombreuses hypothèses de traduction partielles avant de parvenir à une hypothèse de traduction complète. Il existe cependant des types de modèles, que nous qualifions de « complexes », qui sont appris à partir d'informations riches. Si un enjeu pour les développeurs de systèmes de TA consiste à les intégrer lors de la construction initiale des hypothèses de traduction, cela n'est pas toujours possible, car elles peuvent notamment nécessiter des hypothèses complètes ou impliquer un coût de calcul très important. En conséquence, de tels modèles complexes sont typiquement uniquement utilisés en TA pour effectuer le reclassement de listes de meilleures hypothèses complètes. Bien que ceci permette dans les faits de tirer profit d'une meilleure modélisation de certains aspects des traductions, cette approche reste par nature limitée : en effet, les listes d'hypothèses reclassées ne représentent qu'une infime partie de l'espace de recherche du décodeur, contiennent des hypothèses peu diversifiées, et ont été obtenues à l'aide de modèles dont la nature peut être très différente des modèles complexes utilisés en reclassement.

Nous formulons donc l'hypothèse que de telles listes d'hypothèses de traduction sont mal adaptées afin de faire s'exprimer au mieux les modèles complexes utilisés. Les travaux que nous présentons dans cette thèse ont pour objectif de permettre une meilleure exploitation d'informations riches pour l'amélioration des traductions obtenues à l'aide de systèmes de TA statistique.

Notre première contribution s'articule autour d'un système de réécriture guidé par des informations riches. Des réécritures successives, appliquées aux meilleures hypothèses de traduction obtenues avec un système de reclassement ayant accès aux mêmes informations riches, permettent à notre système d'améliorer la qualité de la traduction.

L'originalité de notre seconde contribution consiste à faire une construction de listes d'hypothèses par passes multiples qui exploitent des informations dérivées de l'évaluation des hypothèses de traduction produites antérieurement à l'aide de notre ensemble d'informations riches. Notre système produit ainsi des listes d'hypothèses plus diversifiées et de meilleure qualité, qui s'avèrent donc plus intéressantes pour un reclassement fondé sur des informations riches. De surcroît, notre système de réécriture précédent permet d'améliorer les hypothèses produites par cette deuxième approche à passes multiples.

Notre troisième contribution repose sur la simulation d'un type d'information idéalisé parfait qui permet de déterminer quelles parties d'une hypothèse de traduction sont correctes. Cette idéalisation nous permet d'apporter une indication de la meilleure performance atteignable avec les approches introduites précédemment si les informations riches disponibles décrivaient parfaitement ce qui constitue une bonne traduction. Cette approche est en outre présentée sous la forme d'une traduction interactive, baptisée « pré-post-édition », qui serait réduite à sa forme la plus simple : un système de TA statistique produit sa meilleure hypothèse de traduction, puis un humain apporte la connaissance des parties qui sont correctes, et cette information est exploitée au cours d'une nouvelle recherche pour identifier une meilleure traduction.

Title Complex Feature Guidance for Statistical Machine Translation

Keywords statistical machine translation, complex feature, hypotheses reranking, greedy search, multi-pass decoding, post-editing

Abstract Although communication between languages has without question been made easier thanks to Machine Translation (MT), especially given the recent advances in statistical MT systems, the quality of the translations produced by MT systems is still well below the translation quality that can be obtained through human translation. This gap is partly due to the way in which statistical MT systems operate ; the types of models that can be used are limited because of the need to construct and evaluate a great number of partial hypotheses to produce a complete translation hypothesis. While more "complex" models learnt from richer information do exist, in practice, their integration into the system is not always possible, would necessitate a complete hypothesis to be computed or would be too computationally expensive. Such features are therefore typically used in a reranking step applied to the list of the best complete hypotheses produced by the MT system.

Using these features in a reranking framework does often provide a better modelization of certain aspects of the translation. However, this approach is inherently limited : reranked hypothesis lists represent only a small portion of the decoder's search space, tend to contain hypotheses that vary little between each other and which were obtained with features that may be very different from the complex features to be used during reranking.

In this work, we put forward the hypothesis that such translation hypothesis lists are poorly adapted for exploiting the full potential of complex features. The aim of this thesis is to establish new and better methods of exploiting such features to improve translations produced by statistical MT systems.

Our first contribution is a rewriting system guided by complex features. Sequences of rewriting operations, applied to hypotheses obtained by a reranking framework that uses the same features, allow us to obtain a substantial improvement in translation quality.

The originality of our second contribution lies in the construction of hypothesis lists with a multi-pass decoding that exploits information derived from the evaluation of previously translated hypotheses, using a set of complex features. Our system is therefore capable of producing more diverse hypothesis lists, which are globally of a better quality and which are better adapted to a reranking step with complex features. What is more, our forementioned rewriting system enables us to further improve the hypotheses produced with our multi-pass decoding approach.

Our third contribution is based on the simulation of an ideal information type, designed to perfectly identify the correct fragments of a translation hypothesis. This perfect information gives us an indication of the best attainable performance with the systems described in our first two contributions, in the case where the complex features are able to modelize the translation perfectly. Through this approach, we also introduce a novel form of interactive translation, coined "pre-post-editing", under a very simplified form : a statistical MT system produces its best translation hypothesis, then a human indicates which fragments of the hypothesis are correct, and this new information is then used during a new decoding pass to find a new best translation.