Intégration de l'alignement de mots dans le concordancier bilingue TransSearch

Stéphane Huet, Julien Bourdaillet, Philippe Langlais

TALN 2009, Senlis, 24–26 juin 2009

Intégration de l'alignement de mots dans le concordancier bilingue TransSearch

Stéphane Huet Julien Bourdaillet Philippe Langlais
DIRO - Université de Montréal
C.P. 6128, succursale centre-ville
H3C 3J7, Montréal, Québec, Canada
{huetstep,bourdaij,felipe}@iro.umontreal.ca

Résumé. Malgré les nombreuses études visant à améliorer la traduction automatique, la traduction assistée par ordinateur reste la solution préférée des traducteurs lorsqu'une sortie de qualité est recherchée. Dans cet article, nous présentons nos travaux menés dans le but d'améliorer le concordancier bilingue `TransSearch`. Ce service, accessible sur le Web, repose principalement sur un alignement au niveau des phrases. Dans cette étude, nous discutons et évaluons l'intégration d'un alignement statistique au niveau des mots. Nous présentons deux nouvelles problématiques essentielles au succès de notre nouveau prototype : la détection des traductions erronées et le regroupement des variantes de traduction similaires.

Abstract. Despite the impressive amount of recent studies devoted to improving the state of the art of machine translation, computer assisted translation tools remain the preferred solution of human translators when publication quality is of concern. In this paper, we present our ongoing efforts conducted within a project which aims at improving the commercial bilingual concordancer `TransSearch`. The core technology of this Web-based service mainly relies on sentence-level alignment. In this study, we discuss and evaluate the embedding of statistical word-level alignment. Two novel issues that are essential to the success of our new prototype are tackled: detecting erroneous translations and grouping together similar translations.

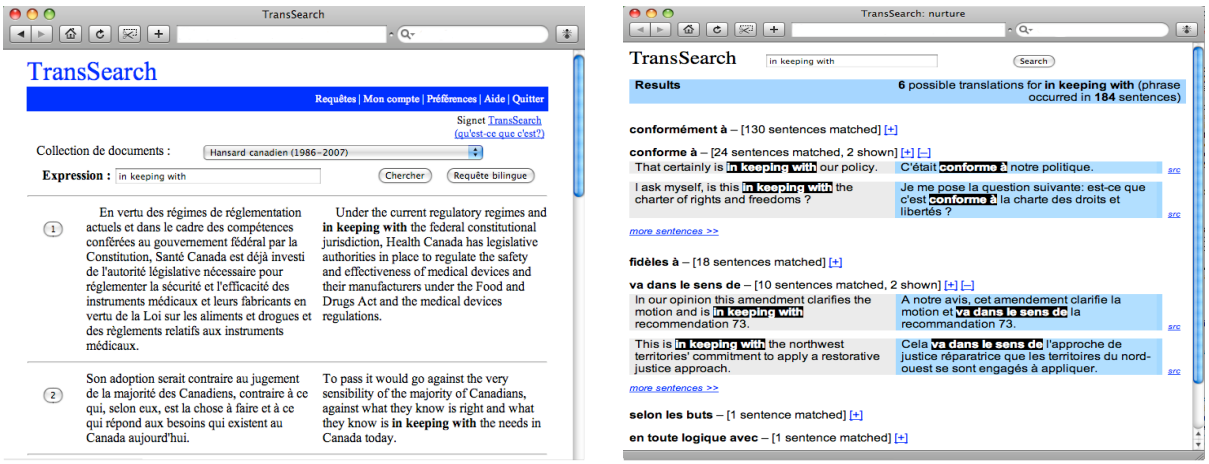
Mots-clés : alignement au niveau des mots, concordancier bilingue, traduction automatique.

Keywords: word-level alignment, bilingual concordancer, machine translation.

1 Introduction

Bien qu'un effort soutenu ait été consacré cette dernière décennie à l'amélioration des systèmes de traduction automatique, les traducteurs professionnels préfèrent encore aujourd'hui se tourner vers les outils de la traduction assistée par ordinateur (TAO), parmi lesquels figurent les systèmes à *mémoire de traduction* (MT) et les *concordanciers bilingues*. Ces deux types de systèmes exploitent une mémoire de traduction constituée d'un *bitexte*, *i.e.* un ensemble de paires d'unités (typiquement des phrases) qui sont la traduction l'une de l'autre. Alors qu'un système à MT est un dispositif de traduction (Planas & Furuse, 1999), un concordancier bilingue est conceptuellement plus simple puisque son objectif principal est de trouver au sein d'un bitexte les paires d'unités qui contiennent une requête (en général une séquence de mots) soumise par un utilisateur. C'est ensuite à l'utilisateur de localiser lui-même les parties intéressantes dans les réponses obtenues. Bien que ce type de système paraisse très simple, les concordanciers bilingues restent un outil très populaire en TAO. Macklovitch *et al.* (2008), indiquent ainsi que `TransSearch`[1], le concordancier commercial mis en ligne sur le Web et faisant l'objet de cet article, a reçu en moyenne 177 000 requêtes par mois sur une période d'un an (2006–2007).

La figure 1(a) présente une capture d'écran du concordancier `TransSearch` dans sa version actuelle. Dans cet exemple, suite à la requête `in keeping with` soumise par un utilisateur, le système retourne une page contenant les 25 premières paires de phrases de la mémoire de traduction qui contiennent une occurrence de la requête. Comme on peut le constater, rien n'est mis en valeur dans les phrases cibles retournées, ce qui contraint l'utilisateur à rechercher la traduction à l'intérieur de chaque phrase cible proposée.



(a) Version actuelle.

(b) Version en développement.

FIG. 1 – Résultats affichés par `TransSearch` pour la requête `in keeping with`.

L'objectif du projet `TS3` est d'identifier automatiquement dans les phrases retournées les différentes traductions d'une requête utilisateur. Bien que l'aspect du nouveau prototype ne soit pas encore définitif, la figure 1(b) montre une interface où l'utilisateur peut consulter les traductions automatiquement identifiées et considérées comme les plus pertinentes. S'il reste toujours possible de consulter les paires de phrases contenant la requête, l'utilisateur peut dorénavant cliquer sur une traduction donnée et visualiser son contexte d'emploi.

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

Dans la suite, nous présentons tout d'abord la technique mise en œuvre pour repérer les traductions au sein des phrases. Nous décrivons ensuite deux nouveaux problèmes importants que nous avons rencontrés: l'identification des alignements erronés (section 3) et le regroupement de variantes de traduction (section 4). Nous décrivons les données utilisées en section 5 et présentons nos résultats expérimentaux en section 6.

2 Repérage de traduction

Le *repérage de traduction* ou *transpotting* (diminutif de *translation spotting*), une partie essentielle du projet `TS3`, consiste à identifier dans du texte cible la traduction d'une requête en langue source (Véronis & Langlais, 2000). Nous appelons *transpot* l'ensemble des mots cibles automatiquement associés à une requête dans une paire de phrases. Dans l'exemple de la figure 1(b), `conformément à` et `va dans le sens de` sont deux des six transpots présentés à l'utilisateur pour la requête `in keeping with`.

2.1 Algorithme de transpotting

Le *transpotting* peut être vu comme un sous-problème de l'alignement au niveau des mots, comme suggéré dans (Simard, 2003). La traduction statistique est encore fortement basée sur des modèles d'alignement de mots (Brown *et al.*, 1993) que nous utilisons dans cette étude.

Formellement, à partir d'une phrase $S = s_1...s_n$ exprimée dans une langue dite source et de sa traduction $T = t_1...t_m$, un alignement $a = a_1...a_m$ de type IBM revient à connecter chaque mot de T à un mot de S ($a_j \in \{1, ..., n\}$) ou au mot vide ($a_j = 0$), ce dernier rendant compte des mots cibles non traduits. Plusieurs modèles proposés par Brown *et al.* (1993) décomposent la probabilité conjointe d'une phrase cible et de son alignement, étant donnée la phrase source. Pour des raisons calculatoires, nous nous concentrons dans cette étude sur la forme la plus simple, correspondant aux modèles IBM 1 et 2 :

$$p(t_1^m, a_1^m | s_1^n) = \prod_{j=1}^{m} \sum_{i \in [0,n]} p(t_j | s_i) \times p(i | j, m, n)$$

où le premier terme de la somme est la probabilité de transfert et le second la probabilité d'alignement. Avec cette décomposition, il est facile et efficace de calculer l'alignement le plus probable entre deux phrases, $\text{argmax}_{a_1^m}\, p(a_1^m | t_1^m, s_1^n)$, ce que nous appelons l'alignement de Viterbi par la suite. Toutefois, cette approche produit souvent des alignments discontinus, alors que ce type d'alignement n'est généralement pas nécessaire pour retrouver les bons transpots.

Afin de produire des transpots contigus, nous avons implémenté un algorithme de transpotting proposé initialement par Simard (2003). Pour chaque paire $\langle j_1, j_2 \rangle \in [1, m] \times [1, m]$, nous calculons deux alignements de Viterbi : l'un entre la suite de mots $t_{j_1}^{j_2}$ et la requête $s_{i_1}^{i_2}$, et l'autre entre les mots restants des phrases source et cible $\bar{s}_{i_1}^{i_2} \equiv s_1^{i_1-1} s_{i_2+1}^n$ et $\bar{t}_{j_1}^{j_2} \equiv t_1^{j_1-1} t_{j_2+1}^m$. Nous calculons alors :

$$\hat{t}_{\hat{j}_1}^{\hat{j}_2} = \underset{(j_1,j_2)}{\text{argmax}} \left\{ \max_{a_{j_1}^{j_2}} p(a_{j_1}^{j_2} | s_{i_1}^{i_2}, t_{j_1}^{j_2}) \times \max_{\bar{a}_{j_1}^{j_2}} p(\bar{a}_{j_1}^{j_2} | \bar{s}_{i_1}^{i_2}, \bar{t}_{j_1}^{j_2}) \right\}$$

Cette méthode, ayant une complexité en $O(nm^3)$, s'est avérée la plus performante de celles que nous avons testées (Bourdaillet *et al.*, 2009).

2.2 Intérêt du post-traitement

Avec la démarche précédemment décrite, les requêtes fréquentes dans la mémoire reçoivent un grand nombre de traductions. La figure 2 illustre ce phénomène en montrant 12 des nombreuses séquences de mots retournées par l'algorithme de transpotting pour la requête `in keeping with`. Dans cet exemple, certains transpots annotés d'une étoile sont clairement mauvais (*e.g.* `à`), tandis que d'autres en italique sont partiellement corrects (*e.g.* `conformément`). Il apparaît en outre que de nombreux transpots sont très proches (*e.g.* `conforme à` et `conformes à`).

`conforme à` (45)	`conformément à` (29)	`à`* (21)	`dans`* (20)	...
`conforme au` (12)	`conformes à` (11)	`avec`* (9)	*`conformément`* (9)	...
`correspond à` (1)	`respectent` (1)	`d'actualité`* (1)	`gestes en`* (1)	

FIG. 2 – Sous-ensemble des 273 transpots différents retournés pour la requête `in keeping with`, leur fréquence étant indiquée entre parenthèses.

Afin d'améliorer les performances du système et d'augmenter la diversité des traductions affichées, il est donc nécessaire d'identifier les erreurs d'alignement et de détecter les transpots similaires. Ce sont les deux problèmes que nous étudions dans les sections suivantes.

3 Filtrage des transpots

Distinguer les bons transpots des mauvais peut être vu comme un problème de classification. Nous avons considéré plusieurs classificateurs classiques[2] : l'algorithme du *voted-perceptron* (VP) (Freund & Schapire, 1999) qui s'est déjà montré performant dans plusieurs tâches de TALN (Collins, 2002), un *séparateur à vaste marge* (SVM) qui est souvent employé en apprentissage supervisé, un *arbre de décision à un niveau* (*decision stump*) pour sa simplicité, *AdaBoost* qui utilise un *decision stump* comme classificateur faible et un *classificateur à vote majoritaire* combinant un voted-perceptron, un SVM et AdaBoost.

Chaque classificateur a été entraîné de manière supervisée à partir d'un corpus annoté (*cf.* section 5). Nous avons calculé trois ensembles de caractéristiques pour chaque exemple, *i.e.* chaque paire requête/transpot (q, t). Le premier ensemble est constitué de caractéristiques relatives à la taille (comptée en nombre de mots) de q et t, avec l'intuition que les deux tailles sont corrélées. Le second ensemble regroupe plusieurs scores d'alignement obtenus avec des modèles IBM d'alignement de mots. Le dernier regroupe des indices plus linguistiques, parmi lesquels le pourcentage de mots grammaticaux dans q et t, ou le nombre de prépositions et d'articles. Au total, chaque exemple est associé au maximum à 40 caractéristiques numériques.

[2]Nous avons employé WEKA dans nos expériences `http://www.cs.waikato.ac.nz/ml/weka`.

4 Regroupement de variantes

Même après avoir identifié les transpots erronés, il reste souvent de nombreuses traductions pour une requête donnée. Par exemple, notre meilleur classificateur (voir la section 6.2) identifie 91 mauvais transpots parmi les 273 initialement proposés pour la requête `in keeping with`. Parmi les transpots restants, certains sont très similaires et sont donc redondants pour l'utilisateur (voir la figure 2)[3]. Nous estimons que parmi les 182 transpots restants, pas moins de 37 traductions *canoniques* sont intéressantes. Regrouper les transpots proches permet d'identifier plus facilement un sous-ensemble des traductions pertinentes.

Distance d'édition basée sur les mots. Nous avons développé une distance d'édition spécifique au niveau des mots pour regrouper les variantes. Différents coûts de substitution, de suppression et d'insertion ont été définis empiriquement selon les classes grammaticales ou les flexions possibles des mots ; ce paramétrage est donc dépendant de la langue. Nous avons utilisé un lexique développé au RALI, qui liste, pour le français et l'anglais, les lemmes de chaque forme fléchie et leurs différentes parties du discours.

Un coût minimal de substitution a été attribué empiriquement entre des formes fléchies d'un même lemme. De plus, un score a été fixé afin de pénaliser de manière croissante et dans l'ordre suivant les opérations d'édition impliquant des signes de ponctuation, des articles, des mots grammaticaux (prépositions, conjonctions et pronoms), des verbes auxiliaires et des mots lexicaux (verbes, noms, adjectifs et adverbes).

Regroupement des transpots. La comparaison de paires de transpots avec notre distance d'édition peut être vue comme un cas particulier de l'alignement multiple de séquences, un problème classique en bio-informatique (Chenna *et al.*, 2003). Nous avons adopté l'approche de construction progressive de l'alignement. Cette méthode commence par calculer les distances d'édition au niveau des mots entre chaque paire de transpots et sauvegarde les résultats dans une matrice de distances. Un algorithme de clustering ascendant appelé *neighbor-joining* (Saiou & Nei, 1987) est ensuite appliqué ; celui-ci construit un arbre en regroupant soit deux transpots, qui sont des feuilles de l'arbre, soit un transpot et un nœud de l'arbre représentant déjà plusieurs traductions, soit encore, deux nœuds. À chaque étape, la paire la plus similaire est regroupée et ajoutée à l'arbre, jusqu'à ce que tous les transpots soient alignés.

Au final, l'algorithme *neighbor-joining* fournit un arbre dont les feuilles sont les transpots ; les feuilles les plus proches dans cet arbre correspondent aux variantes les plus similaires, ce qui permet de construire des clusters de variantes en traversant l'arbre selon un parcours en profondeur. Les transpots associés à deux feuilles voisines et qui diffèrent uniquement selon des mots grammaticaux ou des flexions des mêmes formes simples, se retrouvent ainsi réunis dans un même cluster. Ce processus est répété jusqu'à ce qu'aucune variante ne soit identifiée comme similaire aux autres. Ce principe est illustré par la figure 3. Les deux transpots voisins `conforme à` et `conformes à` sont tout d'abord regroupés, de même que `conforme au` et `conforme aux`. Ces deux groupes sont ensuite fusionnés au sein d'un même cluster, le transpot `correspondant à`, jugé trop différent des autres, n'étant pas inclus dans le nouvel ensemble ainsi formé.

[3]Ce phénomème est particulièrement important en français du fait notamment des nombreuses formes conjuguées pour les verbes. De nombreux transpots diffèrent seulement par des signes de ponctuation ou des mots grammaticaux.

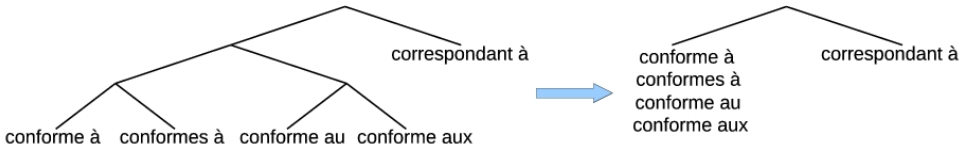


FIG. 3 – Regroupement de transpots proches.

5 Corpus

Mémoire de traduction. La mémoire de traduction sur laquelle s'appuie `TransSearch` est composée principalement du Hansard canadien, constitué de textes parallèles, en français et en anglais, issus des enregistrements officiels des sessions du parlement canadien. Pour les expériences détaillées ci-dessous, nous avons indexé avec Lucene[4] une mémoire comprenant 8,3 millions de paires de phrases français-anglais.

Corpus de référence automatique. Nous avons développé de façon automatisée un corpus de référence (REF) en croisant notre mémoire de traduction avec un lexique bilingue du RALI composé de près de 60 000 expressions (mots ou séquences de mots) ainsi qu'avec les 5 000 requêtes les plus fréquemment soumises au système par les utilisateurs. Notre référence est constituée de plus d'1,4 million de paires de phrases qui contiennent toutes une requête et une traduction validée par notre lexique.

Référence humaine. Afin d'entraîner les classificateurs décrits dans la section 3, quatre annotateurs humains ont été chargés d'identifier les mauvais transpots résultant de notre algorithme de transpotting. Nous avons décidé d'annoter hors-contexte un peu plus de 12 000 paires requête/traduction, ce qui permet une annotation rapide[5] mais laisse des cas difficiles à juger. Par exemple, dans notre exemple, `conforme à` est un transpot pouvant être facilement classé comme correct, mais d'autres ne sont pas aussi évidents, comme `dans le sens de` ou `tenir compte de` qui peuvent être valides en fonction du contexte. Un score kappa de 0,76 témoigne d'un haut degré d'accord inter-annotateurs.

6 Expériences

6.1 Transpotting

Pour chacune des 1 416 000 paires de phrases du corpus REF, nous avons évalué la capacité de notre algorithme de transpotting décrit dans la section 2.1 à identifier la traduction de référence $\hat{t}$ pour une requête q suivant les mesures de précision et rappel calculées comme suit :

$$\text{rappel} = |t \cap \hat{t}| \;/\; |\hat{t}| \qquad\qquad \text{précision} = |t \cap \hat{t}| \;/\; |t| \tag{1}$$

où t est le transpot identifié par l'algorithme et l'intersection retourne la plus longue séquence de mots commune à t et $\hat{t}$. De façon à calculer ces métriques sur l'intégralité du corpus de référence, nous moyennons dans un premier temps le rappel et la précision sur l'ensemble des

[4] `http://lucene.apache.org`
[5] De l'ordre de 40 secondes par requête.

paires correspondant à chaque requête q. Dans un second temps, nous moyennons les scores ainsi obtenus en accordant le même poids à chaque requête. La F-mesure est ensuite déduite de la précision et du rappel calculés sur l'ensemble du corpus.

	précision	rappel	F-mesure
transpotting	0,79	0,84	0,81
transpotting + filtrage	0,82	0,90	0,86

TAB. 1 – Résultat du transpotting avant et après le filtrage du classificateur sur le corpus REF. Voir la section 6.2 pour l'explication de la ligne 2.

Notre algorithme de transpotting (ligne 1 du tableau 1) obtient une précision de $0,79$ et un rappel de $0,84$, qui sont des résultats satisfaisants. La raison pour laquelle la précision est moins bonne que le rappel est liée au fait qu'assez souvent, la traduction de référence est une sous-séquence du transpot identifié, comme par exemple dans la figure 4.

`Je crois qu'il est tout à fait` **`conforme à`** `l'esprit du projet de loi.`

FIG. 4 – Transpot (souligné) et traduction de référence (en gras) pour la requête `in keeping with`.

6.2 Entraînement des classificateurs

Comme indiqué dans la section 3, nous avons entraîné plusieurs classificateurs à reconnaître les mauvais transpots. Ces classificateurs et plusieurs approches naïves (mais très compétitives) sont évalués suivant le taux d'exemples correctement classifiés (TECC). Puisque la tâche qui nous intéresse est celle de filtrer les mauvais transpots, nous présentons également les taux de précision et rappel relatifs à cette classe.

Le tableau 2 présente les résultats obtenus avec une validation croisée à 10 blocs. La première approche naïve utilisée (ligne 1) classifie tous les exemples comme bons. Elle obtient ainsi un TECC de $0,62$, mais n'est d'aucune utilité pour le filtrage. Une approche plus sensée — que nous avons découverte après avoir exploré l'utilité des différents ensembles de caractéristiques — classifie comme mauvais transpots ceux dont le taux de mots grammaticaux est supérieur à $0,75$. Cette approche obtient un bon TECC de $0,78$.

Nous avons commencé les expériences en étudiant la contribution de chaque ensemble de caractéristiques avec le voted-perceptron[6]. Quand le voted-perceptron est entraîné en utilisant un seul ensemble de caractéristiques, celui utilisant uniquement les caractéristiques grammaticales obtient le meilleur TECC de $0,79$ et une F-mesure de $0,65$. Bien que la configuration basée uniquement sur les caractéristiques issues des modèles d'alignement de mots IBM 2 obtienne un TECC légèrement inférieur ($0,78$), nous la considérons comme meilleure en raison de sa F-mesure de $0,73$ qui est plus élevée. La configuration utilisant toutes les caractéristiques pour représenter un exemple améliore clairement les résultats de l'approche naïve avec un TECC de $0,83$ et une F-mesure de $0,77$. On peut également remarquer qu'alors que la meilleure approche

[6]Des résultats similaires ont été observés pour les différents ensembles de caractéristiques avec les autres classificateurs et ne sont pas présentés ici.

			mauvais transpots		
Approche	Caractéristiques	TECC	préc.	rappel	F-mes.
naïve: tous bons		0,62	0,00	0,00	0,00
naïve: taux mots gram. $> 0,75$		**0,78**	0,88	0,49	**0,63**
Voted-Perceptron	taille	0,73	0,75	0,47	0,58
	IBM	0,78	0,69	0,78	0,73
	grammaticales	0,79	0,88	0,52	0,65
	toutes	0,83	0,81	0,73	0,77
Vote majoritaire		**0,84**	0,84	0,71	**0,77**

TAB. 2 – Performance des algorithmes de classification pour identifier les mauvais transpots.

naïve obtient une précision plus élevée que celle du meilleur voted-perceptron, ce dernier obtient un rappel et une précision plus équilibrés. Dans la mesure où il est difficile de savoir s'il vaut mieux privilégier la précision ou le rappel dans notre cas, nous préférons maximiser la F-mesure. En entraînant les autres classificateurs avec tous les ensembles de caractéristiques, aucun gain significatif n'a été observé. Néanmoins, avec le vote majoritaire combinant plusieurs méthodes, nous avons obtenu un meilleur TECC de $0,84$ et une F-mesure de $0,77$, comme reporté à la dernière ligne du tableau 2.

Avec le meilleur classificateur obtenu, le vote majoritaire, nous avons évalué l'impact du filtrage en utilisant ce classificateur pour filtrer les résultats du transpotting sur le corpus REF. Les résultats sont présentés dans le tableau 1 (ligne 2). En supprimant ainsi 7,9 % des transpots, on peut observer un gain significatif en F-mesure qui croît de $0,81$ à $0,86$. Le gain le plus important est en rappel qui passe de $0,84$ à $0,90$, ce qui signifie que le filtrage élimine principalement les transpots trop courts. En examinant manuellement les transpots filtrés, nous avons constaté que les mauvais transpots courts, comme les mots grammaticaux, étaient fréquemment identifiés comme mauvais par le classificateur. Bien que les méthodes de classification supervisées testées n'obtiennent pas d'améliorations majeures par rapport à une l'approche naïve basée sur l'observation des mots grammaticaux, les gains significatifs constatés par rapport à l'algorithme initial de transpotting démontrent l'intérêt de filtrer les mauvais transpots.

6.3 Regroupement de variantes

Si regrouper les variantes similaires est une fonctionnalité très intéressante d'un point de vue ergonomique, il n'est cependant pas facile de trouver le bon niveau de granularité du regroupement des transpots[7]. En conséquence, nous avons étudié deux approches. La première méthode regroupe les variantes qui ne diffèrent que par des signes de ponctuation ou qui sont des formes fléchies d'un même lemme. Elle est basée sur une distance d'édition, appelée D_1, qui utilise les mêmes coûts d'édition pour les mots grammaticaux et lexicaux. La seconde méthode est plus laxiste car elle est basée sur une distance d'édition, appelée D_2, qui attribue un coût d'édition moindre aux mots grammaticaux qu'aux mots lexicaux.

En regroupant les transpots obtenus à partir des 5 000 requêtes du corpus REF (et filtrés par notre meilleur classificateur), la première méthode obtient un nombre moyen de 136 clusters

[7]Cela nécessiterait probablement des tests avec des utilisateurs réels.

par requête avec D_1, alors qu'on a en moyenne 164 transpots par requête (sans regroupement). Comme attendu, en utilisant D_2, on réduit drastiquement le nombre de clusters à 86 par requête. En effet, contrairement à D_1, D_2 permet de regrouper des variantes similaires comme `sur des années` et `durant des années`. Toutefois cela conduit occasionnellement à des regroupements erronés comme `tout à fait` avec `fait tout`.

La figure 5 présente les 5 transpots les plus fréquents obtenus pour deux requêtes par l'algorithme de transpotting avec et sans regroupement. Sans le regroupement, on peut observer que la tendance est de proposer des formes fléchies d'une même traduction. Appliquer le regroupement conduit à plus de diversité, ce qui est préférable puisque le nombre de variantes pouvant être affichées à l'écran dans `TransSearch` est limité : nous estimons que l'affichage de 5 transpots sur une même page est un bon compromis (voir la figure 1(b)).

sans regroup.	décrits	décrite	décrit	tel que décrit	comme l'a
regroup. D_2	décrits	prévu	comme l'a	tel que prescrit	comme le propose
sans regroup.	s'est révélé	s'est avéré	s'est avérée	s'est révélée	a été
regroup. D_2	s'est révélé	s'est avéré	a été	s'est montré	a prouvé

FIG. 5 – Les 5 transpots les plus fréquents pour les requêtes `as described` et `has proven to be` avec ou sans regroupement.

Afin de simuler cela, nous avons mesuré la diversité des 5 transpots les plus fréquents[8] proposés en les considérant comme des sacs d'unigrammes. Pour cette évaluation, les mots sont lemmatisés et les mots grammaticaux sont supprimés. Nous utilisons les mesures de précision et rappel pour comparer les sacs de mots générés par les méthodes de regroupement à ceux de la référence, celle-ci étant formée à partir de la ressource humaine décrite dans la section 5. Suivant ces principes, nous avons obtenu une précision de $0,90$ et un rappel de $0,43$ sans recourir au regroupement. En construisant des clusters, le rappel a été significativement augmenté jusqu'à $0,47$ avec D_1 et à $0,54$ avec D_2, alors que la précision est restée sensiblement la même ($0,89$ avec D_1 et $0,86$ avec D_2). Ces résultats sont corrélés avec la plus grande diversité obtenue grâce au regroupement de variantes.

7 Discussion

Dans cette étude, nous avons étudié l'amélioration du concordancier bilingue `TransSearch` grâce à l'alignement de mots. Un algorithme de transpotting a été proposé et évalué. Nous avons présenté deux nouvelles problématiques essentielles au succès de notre nouveau prototype : la détection des transpots erronés et le regroupement des variantes similaires. Nous avons proposé des solutions à ces deux problèmes et évalué leur efficacité. En particulier, nous avons montré qu'il est possible de mieux détecter les mauvais transpots par rapport à une approche naïve compétitive, et que le regroupement de variantes améliore la diversité des transpots proposés.

Jusqu'à présent il nous a été difficile de comparer notre approche à d'autres de la communauté. Cela est dû principalement au caractère unique du système `TransSearch` qui archive une mémoire de traduction conséquente. Pour donner un point de comparaison, dans (Callisson-Burch *et al.*, 2005) les auteurs présentent les résultats d'alignement obtenus pour 120 requêtes

[8]Ces transpots correspondent à la variante la plus fréquente de chacun des 5 clusters les plus fréquents.

issues d'une mémoire de traduction de 50 000 paires de phrases. Cela reste plusieurs ordres de grandeur inférieur aux expériences présentées dans cet article.

Nous avons considéré des modèles d'alignements simples dans cette étude. Nous souhaitons étudier des modèles d'alignement plus précis, dont celui décrit dans (Vogel *et al.*, 1996)

Remerciements

Cette étude est financée par le Conseil National de Recherche du Canada, en collaboration avec l'entreprise canadienne Terminotix.

Références

BOURDAILLET J., HUET S., GOTTI F., LAPALME G. & LANGLAIS P. (2009). Enhancing the bilingual concordancer TransSearch with word-level alignment. In *22nd Conference of the Canadian Society for Computational Studies of Intelligence*, Kelowna, Canada.

BROWN P., DELLA PIETRA V., DELLA PIETRA S. & MERCER R. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, **19**(2), 263–311.

CALLISSON-BURCH C., BANNARD C. & SCHROEDER J. (2005). A compact data structure for searchable translation memories. In *10th European Conference of the Association for Machine Translation (EAMT)*, p. 59–65, Budapest, Hongrie.

CHENNA R., SUGAWARA H., KOIKE T., LOPEZ R., GIBSON T. J., HIGGINS D. G. & THOMPSON J. D. (2003). Multiple sequence alignment with the Clustal series of programs. *Nucleic Acids Research*, **31**(13), 3497–3500.

COLLINS M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP*, p. 1–8, Philadelphie, PA, USA.

FREUND Y. & SCHAPIRE R. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, **37**(3), 277–296.

MACKLOVITCH E., LAPALME G. & GOTTI F. (2008). Transsearch: What are translators looking for ? In *18th Conference of the Association for Machine Translation in the Americas (AMTA)*, p. 412–419, Waikiki, Hawai'i, USA.

PLANAS E. & FURUSE O. (1999). Formalizing translation memories. In *7th Machine Translation Summit*, p. 331–339, Singapour.

SAIOU N. & NEI M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, **4**(4), 406–425.

SIMARD M. (2003). Translation spotting for translation memories. In *HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond*, p. 65–72, Edmonton, Canada.

VÉRONIS J. & LANGLAIS P. (2000). *Evaluation of Parallel text Alignment Systems — The Arcade Project.*, chapter 19, p. 369–388. Kluwer Academic Publisher, Dordrecht, Pays-Bas.

VOGEL S., NEY H. & C. T. (1996). HMM-based word alignment in statistical translation. In *16th conference on Computational linguistics*, p. 836–841, Copenhague, Danemark.