Vers un développement plus efficace des systèmes de traduction statistique : un peu de vert dans un monde de BLEU

Li Gong[1,2] Aurélien Max[1,2] François Yvon[1]
(1) LIMSI-CNRS, Orsay, France (2) Univ. Paris-Sud, Orsay, France
{li.gong,aurelien.max,francois.yvon}@limsi.fr

Résumé. Dans cet article, nous montrons comment l'utilisation conjointe d'une technique d'alignement de phrases parallèles à la demande et d'estimation de modèles de traduction à la volée permet une réduction en temps très notable (jusqu'à 93% dans nos expériences) par rapport à un système à l'état de l'art, tout en offrant un compromis en termes de qualité très intéressant dans certaines configurations. En particulier, l'exploitation immédiate de documents traduits permet de compenser très rapidement l'absence d'un corpus de développement.

Abstract. In this article, we show how using both on-demand alignment of parallel sentences and on-the-fly estimation of translation models can yield massive reduction (up to 93% in our experiments) in development time as compared to a state-of-the-art system, while offering an interesting tradeoff as regards translation quality under some configurations. We show in particular that the absence of a development set can be quickly compensated by immediately using translated documents.

Mots-clés : traduction automatique statistique ; développement efficace ; temps de calcul.

Keywords: statistical machine translation ; efficient development ; computation time.

1 Introduction et motivations

Le développement de systèmes de traduction automatique statistiques repose sur des quantités de données parallèles en constante augmentation (voir par ex. (Smith *et al.*, 2013)). Son cout devient donc très important, en termes de temps de calcul et de moyens de calcul et de stockage. Une conséquence est que la disponibilité d'un système pour l'utilisateur final, qui peut parfois correspondre à un besoin pressant (Lewis, 2010), est soit impossible car trop couteuse, soit au mieux loin d'être immédiate. Pourtant, le développement standard d'un tel système produit d'énormes bases de connaissances de traduction dont seule une infime partie est effectivement utilisée.

Des travaux précédents (Callison-Burch *et al.*, 2005; Lopez, 2008) permettent d'améliorer cette situation, en ne construisant que la partie utile des ressources de traduction qui sont nécessaires à la traduction d'un texte particulier. Dans ces approches, les tables de traduction sont construites *à la demande* pour chaque nouveau texte, en s'appuyant sur un échantillonnage *à taille constante des exemples de traduction* permettant l'obtention de ressources à la fois compactes et compétitives sur le plan de la qualité des traductions produites. Cependant, cet échantillonnage repose sur un pré-traitement qui est lui même très couteux. En particulier, l'alignement automatique au niveau des mots du corpus bilingue est supposé pré-exister, et son calcul peut requérir un temps considérable (par exemple, plus de 40 jours de calculs pour un processeur dans les expériences décrites Section 3).

Dans cet article, nous allons montrer qu'il est possible d'obtenir des performances très proches d'un système de référence à l'état de l'art, en effectuant à la demande non seulement la sélection des exemples de traduction, mais également l'alignement au niveau des mots des phrases parallèles correspondantes. Une telle approche est rendue possible par les travaux récents sur l'alignement ciblé de Gong *et al.* (2013), eux-mêmes fondés sur les résultats en alignement par échantillonnage et segmentation récursive de Lardilleux *et al.* (2012). En outre, le cadre proposé ici offre des perspectives originales au niveau du développement incrémental de systèmes de traduction adaptés (toute nouvelle phrase parallèle peut-être immédiatement utilisée), ainsi que de la fusion du développement des systèmes et de leur utilisation dans un cadre interactif (quelques secondes seulement sont nécessaires pour traduire une phrase sans autre ressource qu'un corpus parallèle et un modèle de langue cible).

	finally	,	what	our	fellow	citizens	are	demanding	is	the	right	to	information	.
enfin	**0.607**	0.001	ϵ	ϵ	0	ϵ	ϵ	0	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
,	0.001	**0.445**	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0.001	ϵ	0.001	ϵ	0.001
c'	ϵ	ϵ	0.001	ϵ	ϵ	ϵ	ϵ	0	**0.036**	0.001	ϵ	ϵ	ϵ	ϵ
est	ϵ	ϵ	0.001	ϵ	ϵ	ϵ	ϵ	0	**0.223**	0.016	ϵ	0.001	ϵ	0.001
un	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0.005	**ϵ**	ϵ	ϵ	ϵ	ϵ
droit	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0	ϵ	ϵ	**0.084**	ϵ	ϵ	ϵ
à	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0.001	ϵ	0.001	0.003	0.001	**0.018**	ϵ	ϵ
l'	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0.002	0.009	ϵ	**0.002**	ϵ	ϵ
information	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	**0.499**	ϵ
que	ϵ	ϵ	0.002	ϵ	ϵ	ϵ	0.001	ϵ	0.002	0.001	ϵ	0.001	**ϵ**	ϵ
réclament	0	0	ϵ	ϵ	ϵ	ϵ	**ϵ**	**0.152**	ϵ	ϵ	0	0	0	ϵ
nos	ϵ	ϵ	**ϵ**	**0.171**	0.004	0.001	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
concitoyens	0	ϵ	ϵ	0.001	**0.323**	**0.009**	ϵ	ϵ	ϵ	ϵ	0	ϵ	0	ϵ
.	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	0.001	0.001	ϵ	ϵ	ϵ	**0.954**

FIGURE 1 – Illustration de la procédure d'alignement récursif (tiré de (Lardilleux *et al.*, 2012)). Les valeurs dans les cellules correspondent à des scores d'association ($0 < \epsilon \leq 0.001$). Les points d'alignement retenus par l'algorithme correspondent à ceux obtenus au niveau de récursion le plus profond (en gras).

Dans la section 2, nous décrivons le cadre proposé pour une construction efficace de systèmes reposant sur l'alignement de phrases à la demande (2.1) et l'estimation de modèles de traduction à la volée (2.2). Nous présentons dans la section 3 un ensemble d'expériences permettant de valider nos propositions : après la description du contexte expérimental (3.1), nous décrivons des configurations plus économes en temps dont les performances rejoignent, puis dépassent, celle d'un système de référence à l'état de l'art (3.2). Ces résultats sont discutés dans la section 4.

2 Développement efficace de systèmes de traduction automatique statistique

2.1 Alignement de phrases parallèles à la demande

Notre approche repose sur la capacité à aligner des phrases parallèles à la demande, afin d'en extraire des exemples de traduction pour des segments devant être traduits. Nous avons pour cela réutilisé l'approche de Gong *et al.* (2013), elle-même inspirée des travaux de Xu *et al.* (2006) et de Lardilleux *et al.* (2012), qui effectue un découpage binaire récursif des points d'alignement au niveau des mots possibles dans une paire de phrases parallèles (voir l'illustration Figure 1). Chaque point d'alignement entre paires de mots, correspondant à des cellules de la matrice d'alignement, est associé à un score d'association, obtenu ici par une procédure de ciblage de traduction par échantillonnage (voir (Gong *et al.*, 2013)). L'algorithme de segmentation commence au niveau d'un alignement complet entre les deux phrases, puis des points de coupe trouvés récursivement permettent de mettre en évidence des points d'alignement aux niveaux où plus aucune coupe n'est possible. Ce processus s'apparente à une analyse syntaxique descendante fondée sur les ITG (*Inversion Transduction Grammars*) (Wu, 1997), à la différence que le calcul de la segmentation s'effectue de façon gloutonne, en s'appuyant sur les scores d'association plutôt que sur un modèle probabiliste. Le travail de Gong *et al.* (2013) a montré que cette approche permettait d'obtenir des performances au niveau de l'état de l'art sur des tâches de traduction standard à petite échelle (avec alignement complet des corpus parallèles).

2.2 Estimation de modèles de traduction à la volée

Nous utilisons un tableau de suffixes (Manber & Myers, 1990) pour indexer la partie source des corpus, ce qui permet un accès rapide en $O(k)$ opérations pour des segments de k tokens. Un échantillon contenant des exemples d'un segment source à traduire $\bar{f}$ peut ainsi être obtenu efficacement et sa taille permet d'effectuer un compromis entre la précision de l'estimation et sa vitesse. Les approches précédentes utilisant cette technique (Callison-Burch *et al.*, 2005; Lopez, 2008) étaient fondées sur une échantillonnage aléatoire déterministe, garantissant que l'on choisissait toujours les mêmes exemples pour les mêmes segments. La probabilité de traduction d'un segment se calcule alors simplement par : $p(\bar{e}|\bar{f}) = count(\bar{f}, \bar{e}) / \sum_{\bar{e}'} count(\bar{f}, \bar{e}')$, avec $count(\cdot)$ le nombre d'occurrences d'un segment dans l'échantillon, incluant les occurrences où l'extraction d'une traduction est impossible (ce que Lopez (2008) décrit comme une estimation *cohérente*). L'échantillonage étant réalisé indépendamment pour chaque segment, il n'est pas possible d'estimer un modèle de traduction inverse, lequel peut néanmoins être approximé par : $p(\bar{f}|\bar{e}) = min(1.0, p(\bar{e}|\bar{f}) \times freq(\bar{f})/freq(\bar{e}))$, où $freq(\cdot)$ est la fréquence d'un segment dans le corpus entier. Les autres modèles utilisés sont les mêmes que dans la configuration par défaut d'un système `Moses` [1] fondé sur les segments (*phrase-based*).

1. `http://www.statmt.org/moses`

		# lignes	# tokens source	# tokens cible
apprentissage	WMT'13 + médical WMT'14	16,6M	396,9M	475,1M
développement	Cochrane	743	16,5K	21,4K
collection de test	Cochrane (100 documents)	1,8K	38,6K	49,3K

TABLE 1 – Description des corpus utilisés

3 Validation expérimentale

3.1 Contexte expérimental

Données Nous avons sélectionné la paire de langue anglais-français pour cette étude, car de très grandes quantités de données parallèles sont disponibles (plus de 400 millions de tokens anglais, soit plus de 15 fois la quantité utilisée dans l'étude de Lopez (2008)). Les données parallèles tout venant de l'atelier WMT ont été utilisées, ainsi que des données d'origines diverses du domaine médical [2]. Comme données d'évaluation, nous avons retenu les résumés systématiques de la littérature scientifique médicale produites par la collaboration Cochrane [3], qui présentent un certain nombre de caractéristiques souhaitables : les textes source en anglais sont de bonne qualité ; la langue d'origine des documents est toujours l'anglais, éliminant ainsi de possibles effets de traductions multiples ; les traductions de référence sont de bonne qualité, et la nature des documents tolère l'existence d'une seule traduction *normative* ; il est enfin possible de traiter ce corpus document par document. Le tableau 1 recense quelques statistiques utiles sur les différents corpus utilisés.

Systèmes Nous avons utilisé l'implémentation `mgiza++` [4] de l'aligneur statistique `giza++` (Och & Ney, 2003). Nous avons par ailleurs développé un aligneur à la volée tel que décrit section 2.1, en utilisant une taille d'échantillon de 100. Ce choix est motivé par les résultats en traduction fondée sur les segments donnés par Callison-Burch *et al.* (2005). Les heuristiques standard du système `Moses` pour l'extraction de traductions à partir de matrices d'alignement ont été utilisées, avec une taille de segment maximale de 6 tokens. Des modèles de langue 4-grammes pour le français ont été appris sur l'ensemble des données médicales en français de WMT'14. Le décodeur de `Moses` a été utilisé dans toutes les expériences et `KBMIRA` (Cherry & Foster, 2012) a été utilisé pour l'optimisation de certains systèmes.

Indicateurs de performance Nous avons utilisé les métriques standard BLEU (Papineni *et al.*, 2002) et TER (Snover *et al.*, 2006) et donnons des mesures correspondant à la moyenne des scores du test pour 3 optimisations indépendantes lorsqu'approprié. Comme indicateur principal du cout en ressources matérielles, le temps CPU utilisateur (*user CPU time*, tel que retourné par la commande `UNIX time`) a été utilisé comme focus sur la durée de calcul des processus mis en œuvre. Les programmes exécutés étant à différents niveaux de maturité et d'optimisation, ainsi qu'exécutés soit en code binaire (`mgiza++`) soit en dans une machine virtuelle Java (notre système), seules de larges différences pourront être considérées comme significatives.

3.2 Résultats expérimentaux

Cette section décrit les systèmes comparés dans nos expériences, dont les caractéristiques sont données dans la Table 2.

Système `moses` standard à très large échelle (`baseline-vanilla`) Afin de pouvoir comparer nos propositions à une approche bien connue et éprouvée, nous avons implémenté un système `Moses` utilisant l'intégralité des données. Le temps de développement correspond à l'exécution de `mgiza++` sur le corpus parallèle (1 006h), puis la construction des tables de traduction et de réordonnancement (206h), et enfin l'optimisation du système par `KBMIRA` (21h) après filtrage des tables pour les textes à traduire. Au total, 1 233h de temps processeur (plus de 51 jours) ont été nécessaires avant de pouvoir traduire la première phrase du corpus de test. En outre, les tables de traduction et de réordonnancement compressées occupent respectivement 20Go et 7,5Go, ce qui interdit leur installation sur la plupart des dispositifs mobiles.

2. Voir `http://www.statmt.org/wmt13` et `http://www.statmt.org/wmt14/medical-task`
3. `http://summaries.cochrane.org`
4. `http://www.kyloo.net/software/doku.php/mgiza:overview`

	BLEU	Δ	TER	Δ	temps CPU					
					TS	dev.	test	optim.	total	réduction
baseline-vanilla	34,12±0,10	-	48,59±0,22	-	-	1 212h		21h	1 233h	-
baseline-rnd	33,95±0,14	-0,17	48,79±0,13	+0,2	2h	1 006h		20h	1 028h	16%
config0	28,24	-5,88	49,92	+1,33	2h	0h	363h	0h	365h	70%
config1	28,87	-5,25	49,40	+0,81	2h	0h	111h	0h	113h	90%
config2	28,58	-5,54	49,54	+0,95	2h	0h	76h	0h	78h	93%
config2+spec	32,33	-1,79	46,42	-2,17	2h	0h	76h	0h	78h	93%
config2:tuned	33,38±0,06	-0,74	50,05±0,05	+1,46	2h	72h	76h	20h	170h	86%
config2:tuned+spec	37,63±0,05	+3,51	46,49±0,04	-2,10	2h	72h	76h	20h	170h	86%

TABLE 2 – Comparaison des performances de différents systèmes. Les résultats pour les systèmes optimisés (baseline, config2:tuned, config2:tuned+spec) sont décomposés en moyenne et écart type pour 3 optimisations.

Échantillonnage aléatoire des exemples de traduction (baseline-rnd) Nous avons également construit un système obtenu par sélection d'exemples d'apprentissage par échantillonnage aléatoire déterministe (Callison-Burch *et al.*, 2005; Lopez, 2008). Chaque segment d'un texte à traduire est cherché dans un tableau de suffixes du corpus source, et les alignements correspondant obtenus par mgiza++ sont utilisés pour extraire les traductions (ici encore, nous utilisons une taille d'échantillon de 100). Le temps de construction des tables de traduction et de réordonnancement est ici négligeable en regard du temps d'alignement, et le temps total avant traduction du corpus de test est de 1 028h (42 jours), une réduction de temps relativement modeste de 16% relativement à baseline-vanilla. Les gains en espace sont cependant bien évidemment beaucoup plus marquants, avec respectivement 32Mo et 8Mo pour les tables de traduction et de réordonnancement, lesquelles ne sont toutefois utiles que pour des documents particuliers. Finalement, la performance selon BLEU et TER est très proche de celle de baseline-vanilla. La réduction en temps de calcul et la construction à la demande des tables apparaissent déjà comme des modifications utiles.

Alignement des phrases à la demande, pas d'optimisation (config0) Nous considérons maintenant une configuration très simple : pour chaque document à traduire en séquence, les exemples de traduction sont choisis comme pour baseline-rnd et les phrases correspondantes sont alignées à la demande, puis les tables pour le document sont construites sur la base des alignements obtenus. Ces tables sont utilisées pour traduire chaque document en utilisant les paramètres par défaut du décodeur Moses. Le temps avant traduction du test (complet) n'inclut donc plus que le temps de construction du tableau de suffixes (2h) et l'estimation indépendante des tables pour l'ensemble des documents (363h), correspondant à 365h (15 jours), soit une réduction de temps sensible de 70% par rapport à baseline-vanilla. Cependant, le score BLEU se trouve très dégradé, avec une baisse de presque 6 points.

Mise en cache des tables de traduction et des alignements (config1) Nous avons implémenté un système de caches permettant de conserver (non sélectivement) d'une part les entrées des tables déjà construites pour des segments, et d'autre part les alignements de phrases parallèles déjà calculés par le système d'alignement à la demande. Si le premier type de cache n'apporte qu'un gain de temps marginal, le second s'avère plus performant, permettant de réduire le temps global à 113h (4,7 jours). Nous observons une légère amélioration dans les scores BLEU et TER.

Sélection d'exemples par réutilisation de phrases alignées (config2) Dans nos systèmes, le nombre de phrases à aligner à la demande a un impact direct sur le temps global. Nous avons donc étudié une technique simple qui réutilise préférentiellement des phrases déjà alignées. Ainsi, pour l'estimation des traductions d'un nouveau segment, les phrases déjà alignées qui le contiennent seront utilisées tout d'abord, puis éventuellement d'autres phrases seront alignées à la demande pour parvenir aux 100 exemples requis. Dans notre implémentation, l'échantillonnage aléatoire déterministe a d'abord lieu dans le sous-corpus déjà aligné, puis dans le corpus complémentaire. Cette technique mène à de faibles pertes en BLEU et TER par rapport à config1, mais permet pour un gain de temps sensible, désormais réduit à 78h (3,25 jours), soit une réduction de 93% par rapport à baseline-vanilla. Nous avons en outre mesuré une division par 5 du temps nécessaire (normalisé par la taille de chaque document) pour le calcul entre le premier et le dernier document de nos 100 documents de test, ce qui montre que notre approche parvient à exploiter très efficacement un cache qui grossit. Toutefois, l'écart en performance avec baseline-vanilla reste conséquent (environ 5,5 points BLEU).

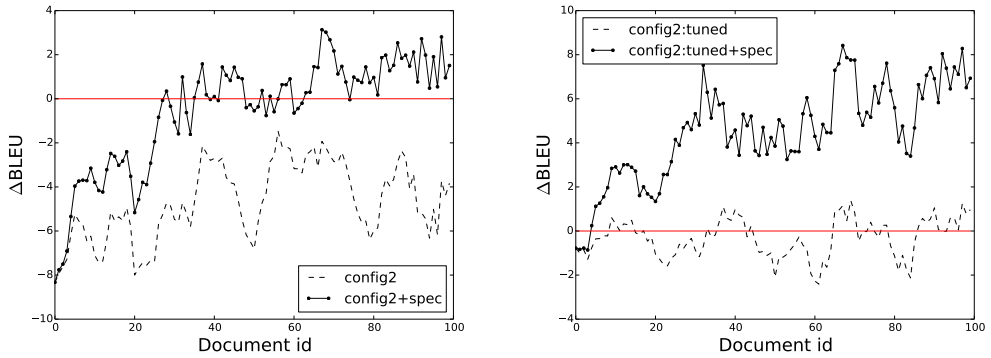

FIGURE 2 – Comparaison au niveau document relativement à `baseline-rnd`. En abscisses est représentée la séquence des documents traduits, et en ordonnées l'écart relatif en BLEU avec `baseline-rnd`.

Utilisation incrémentale de données du domaine (`config2+spec`) La configuration précédente est très en retrait par rapport à `baseline-vanilla`, cependant elle ne dépend pas de l'existence préalable d'un corpus de développement, lui-même très couteux à obtenir. Toutefois, notre aligneur à la demande nous permet trivialement d'ajouter aux données parallèles toute donnée nouvellement disponible. Nous considérons donc un scénario où chaque document complet devient immédiatement disponible après avoir été révisé. Nous alignons donc à la demande chaque document après sa traduction en utilisant sa traduction de référence (le faire pour l'ensemble des 99 premiers documents du test prend une demi-heure environ), puis utilisons les phrases parallèles alignées ainsi obtenues pour construire une nouvelle table de traduction contenant deux scores (modèle de traduction direct et pénalité de segment) dont les paramètres sont repris de la table principale (Hardt & Elming, 2010). Puisque cette table est estimée sur peu de données et qu'elle est de plus très sensible aux erreurs d'alignement, nous l'utilisons comme table secondaire (*back-off*) en complément de la table principale. Elle n'est donc utilisée que pour les longs segments non présents dans le grand corpus, ces segments correspondant possiblement à des termes et de la phraséologie propre à nos documents techniques.

Si le temps de développement reste sensiblement équivalent à `config2`, la performance en BLEU s'est très nettement rapprochée de celle de `baseline-vanilla`, puisque l'écart n'est plus que de 1,8 points. Il est intéressant de noter que cette configuration obtient un bien meilleur score TER que le système complet (baisse de 2,2 points), ce qui s'explique facilement par la réutilisation de grands segments spécifiques à la classe de documents et correspond donc à l'effet attendu d'une mémoire de traduction sous-phrastique. Il est particulièrement instructif de considérer comment la performance de traduction évolue pour chaque document i, qui utilise donc possiblement des exemples de traduction provenant des $i-1$ documents précédents. La partie gauche de la Figure 2 montre l'écart relatif en BLEU de nos deux variantes de `config2` par rapport à `baseline-vanilla`. Alors que, sans surprise, les documents sont systématiquement moins bien traduits par `config2`, `config2+spec` traduit moins bien les 30 premiers documents environ, puis traduit de mieux en mieux les documents. Ce résultat est particulièrement remarquable, puiqu'au bout d'environ 60 documents, `config2` traduit les documents systématiquement *mieux que le système de référence complet* ayant bénéficié d'une optimisation, tout en économisant 93% de temps de calcul.

Sélection d'exemples par réutilisation de phrases alignées et optimisation (`config2:tuned`) Nous nous ramenons maintenant à une situation plus usuelle, où un corpus de développement est utilisé pour optimiser le système ; la contrepartie en temps est la construction des tables pour la traduction de ce corpus, ramenant la réduction en temps par rapport à `baseline-vanilla` à 86% (soit 7 jours). La performance obtenue est alors très proche de celle de `baseline-vanilla`, avec une perte de seulement 0,7 points BLEU (métrique vers laquelle est effectuée l'optimisation). Ce résultat est là aussi particulièrement intéressant, car la performance obtenue est très proche de celle d'un système construit sur une très grande quantité de données.

Utilisation incrémentale de données du domaine et optimisation (`config2:tuned+spec`) L'étape naturelle suivante est de reprendre la configuration précédente `config2:tuned` et d'y adjoindre la possibilité étudiée précédemment d'exploiter de façon incrémentale un corpus additionnel constitué des documents précédemment traduits. Bien qu'il s'agisse ici d'une configuration sous-optimale (en particulier, parce que l'optimisation est faite une fois pour toutes et

ne prend pas en compte notre table additionnelle), les gains en traduction, toujours obtenus avec une réduction en temps d'environ 86% par rapport au système complet, sont relativement importants (+3,5 points BLEU et -2,1 points TER). On constate en outre (voir la partie droite de la Figure 2) une tendance à rapidement exploiter les nouvelles données, et donc à accroitre l'avantage vis-à-vis du système de référence tant que celui-ci n'a pas bénéficié d'un ré-entrainement (couteux) permettant d'ajouter les nouvelles données.

4 Discussion et perspectives

Nous avons décrit une série d'approches reposant sur le développement à la volée des ressources de systèmes de traduction statistique, ce qui permet un gain de temps considérable par rapport aux systèmes à l'état de l'art, tout en offrant, sous certaines conditions, des performances intéressantes en termes de qualité de traduction. Le résultat le plus marquant est la démonstration que des gains de traduction importants peuvent être obtenus lorsque les documents traduits sont immédiatement intégrés aux données d'apprentissage, ce qui permet même de se dispenser d'un corpus de développement. Il est ainsi possible d'aider un traducteur beaucoup plus rapidement que si celui-ci avait initialement à produire la traduction d'un corpus de développement.

L'exploitation incrémentale des données d'apprentissage pourrait aisément être menée au niveau de chaque phrase venant d'être révisée, laissant entrevoir des gains plus forts à l'intérieur des documents. Une autre perspective est la possibilité de réaliser une optimisation du système après chaque ajout significatif de données, plutôt qu'une fois pour toutes. Enfin, nous comptons étudier différentes manières d'améliorer encore les temps de calcul, en particulier en déterminant quels segments source ne sont pas utiles, voire correspondent à de mauvaises unités de traduction, ainsi qu'en stoppant l'échantillonnage des traductions pour un segment dès qu'une distribution de traductions stable est obtenue.

Références

CALLISON-BURCH C., BANNARD C. & SCHROEDER J. (2005). Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of ACL*, Ann Arbor, USA.

CHERRY C. & FOSTER G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of NAACL*, p. 427–436, Montréal, Canada.

GONG L., MAX A. & YVON F. (2013). Improving bilingual sub-sentential alignment by sampling-based transpotting. In *Proceedings of IWSLT*, Heidelberg, Germany.

HARDT D. & ELMING J. (2010). Incremental Re-training for Post-editing SMT. In *AMTA*, Denver, USA.

LARDILLEUX A., YVON F. & LEPAGE Y. (2012). Hierarchical sub-sentential alignment with Anymalign. In *Proceedings of the annual meeting of the European Association for Machine Translation*, p. 279–286, Trento, Italy.

LEWIS W. (2010). Haitian Creole : How to Build and Ship an MT Engine from Scratch in 4 days, 17 hours, & 30 minutes. In *Proceedings of EAMT*, Saint-Raphaël, France.

LOPEZ A. (2008). Tera-Scale Translation Models via Pattern Matching. In *Proceedings of COLING*, Manchester, UK.

MANBER U. & MYERS G. (1990). Suffix arrays : A new method for on-line string searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 319–327.

OCH F. J. & NEY H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics*, **29**(1), 19–51.

PAPINENI K., ROUKOS S., WARD T. & ZHU W.-J. (2002). BLEU : a method for automatic evaluation of machine translation. In *Proceedings of ACL*, p. 311–318.

SMITH J. R., SAINT-AMAND H., PLAMADA M., KOEHN P., CALLISON-BURCH C. & LOPEZ A. (2013). Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of ACL*, Sofia, Bulgaria.

SNOVER M., DORR B., SCHWARTZ R., MICCIULLA L. & MAKHOUL J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, p. 223–231, Cambridge, USA.

WU D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, **23**(3), 377–404.

XU J., ZENS R. & NEY H. (2006). Partitioning parallel documents using binary segmentation. In *Proceedings of WMT*, p. 78–85.