Application d'un algorithme de traduction statistique à la normalisation de textos

Gabriel Bernier-Colborne[1]
(1) Observatoire de linguistique Sens-Texte
Université de Montréal
gabriel.bernier-colborne@umontreal.ca

RÉSUMÉ

Ce travail porte sur l'application d'une technique de traduction statistique au problème de la normalisation de textos. La méthode est basée sur l'algorithme de recherche vorace décrit dans (Langlais *et al.*, 2007). Une première normalisation est générée, puis nous appliquons itérativement une fonction qui génère des nouvelles hypothèses à partir de la normalisation courante, et maximisons une fonction de score. Cette méthode fournit une réduction du taux d'erreurs moyen par phrase de 33 % sur le corpus de test, et une augmentation du score BLEU de plus de 30 %. Nous mettons l'accent sur les fonctions qui génèrent la normalisation initiale et sur les opérations permettant de générer des nouvelles hypothèses.

ABSTRACT

Applying a Statistical Machine Translation Algorithm to SMS Text Message Normalization

We report on the application of a statistical machine translation algorithm to the problem of SMS text message normalization. The technique is based on a greedy search algorithm described in (Langlais *et al.*, 2007). A first normalization is generated, then a function that generates new hypotheses is applied iteratively to a current best guess, while maximizing a scoring function. This method leads to a drop in word error rate of 33% on a held-out test set, and a BLEU score gain of over 30%. We focus on the methods of generating the initial normalization and the operations that allow us to generate new hypotheses.

MOTS-CLÉS : Traduction statistique, normalisation de textos, algorithme de recherche vorace, modèle de langue.

KEYWORDS: Machine translation, SMS, text message, normalization, greedy search algorithm, language model.

1 Introduction

Les messages textes (SMS ou textos) contiennent fréquemment des formes qui ne sont pas conformes à l'orthographe ordinaire, ce qui rend leur traitement par des systèmes de traitement automatique de la langue problématique. La normalisation des textos consiste à « réécrire les textos au moyen d'une orthographe plus classique afin de les rendre plus facilement lisibles par un humain ou un ordinateur » (Yvon, 2008, p. 5) [1]. Par exemple, si on rencontre la forme « stai comment le ... », l'objectif est de produire une normalisation telle que « comment était le ... ».

Étant donné la popularité énorme des messages textes et des formes de communication apparentées, l'intérêt que pose la normalisation de ces messages a augmenté, ainsi le problème a-t-il inspiré de nombreux travaux depuis quelques années. Les différentes approches proposées font appel aux techniques de la correction orthographique, de la traduction statistique et de la reconnaissance automatique de la parole (Yvon, 2008). Par exemple, (Aw *et al.*, 2006) traitent le problème comme une tâche de traduction, où on vise à traduire l'anglais des textos en anglais standard. (Yvon, 2008) traite le problème comme une tâche de reconnaissance automatique de la parole (RAP), mais utilise également des techniques de correction orthographique ; une représentation phonétique des textos joue le rôle du modèle acoustique, et un modèle de langue est utilisé pour convertir les séquences de phones en séquences de mots. (Beaufort *et al.*, 2010) proposent pour leur part un système qui combine des techniques de correction automatique et de traduction statistique.

Ce travail porte sur l'application d'une technique de traduction statistique au problème de la normalisation de textos. Le problème consiste donc à « traduire » un texto en français standard. Ainsi, l'objectif de ce travail est de maximiser $p(f|e)$, où e désigne un texto et f, sa normalisation. On peut reformuler le problème ainsi en appliquant la loi de Bayes : $p(f|e) = p(f) \cdot p(e|f)$, ces deux termes étant déterminés par des modèles de langue et de traduction respectivement.

Une remarque concernant l'évaluation des techniques de normalisation de textos s'impose. Deux métriques sont souvent utilisées pour cette évaluation : certains auteurs utilisent le score BLEU (Papineni *et al.*, 2001), d'autres utilisent le taux d'erreur moyen par phrase (word error rate ou WER). Les deux métriques sont utilisées dans nos évaluations (ainsi que le taux de phrases erronées ou SER), et nous proposons qu'il est plus pertinent d'observer la réduction du WER, plutôt que le WER final, étant donné que les corpus de textos contiennent différentes quantités de formes à normaliser.

Les résultats présentés dans la littérature divergent beaucoup, et il est très délicat d'établir des comparaisons, notamment en raison des différences quant à la langue et la taille des corpus utilisés (en plus de l'utilisation de différentes métriques). (Aw *et al.*, 2006), qui travaillent sur la langue anglaise, obtiennent un score BLEU de 0,81. (Beaufort *et al.*, 2010) affirment que les systèmes à l'état de l'art obtiennent un WER de 11 %, et le système qu'ils proposent, qui exploite le corpus *SMS pour la science*, obtient un WER de 9,3 % et un score BLEU de 0,83. (Yvon, 2008) obtient un WER de 17,8 %, un résultat semblable à ce qu'on obtiendrait en utilisant un système générique de traduction statistique pour traiter ce problème. (Kobus *et al.*, 2008) obtiennent un WER de 16,5 % avec un système basé sur la métaphore de la RAP, de 12,3 % avec un système de traduction statistique, et d'environ 10,8% en combinant les deux systèmes.

1. Nous traduisons.

Le reste de cet article sera organisé de la façon suivante. Dans la section 2, nous décrirons les ressources utilisées dans le cadre de ce travail. La section 3 portera sur l'algorithme de recherche vorace que nous avons implémenté ; l'accent sera placé sur la fonction qui génère la normalisation initiale et la fonction qui génère de nouvelles hypothèses. Enfin, dans la section 4, nous analyserons les résultats obtenus.

2 Ressources

Trois ressources sont utilisées pour mettre en application l'algorithme vorace de recherche : un modèle de langue, un modèle de traduction et un corpus de textos annotés. Ce corpus est constitué de textos en français recueillis et annotés dans le cadre du projet Texto4Science (Langlais *et al.*, 2012). Chaque texto est accompagné d'une normalisation produite par un annotateur humain. Nous utilisons un corpus d'entraînement totalisant 11 000 textos alignés avec leur normalisation, un corpus de développement de 1135 paires et un corpus de test de 1000 textos non vus à l'entraînement, utilisé pour l'évaluation finale. Ce test est effectué seulement une fois, sur la meilleure version de notre système. Les autres résultats présentés proviennent tous d'évaluations sur le corpus de développement.

Le modèle de langue est un modèle trigramme avec lissage Kneser-Ney entraîné sur un corpus de français totalisant 673 000 phrases et 8, 6 millions de mots, qui comprend les textos normalisés du corpus d'entraînement.

Quant au modèle de traduction, nous utilisons un modèle probabiliste appris sur le corpus d'entraînement, de la forme $p(f|e)$ où e sont des mots de la langue des textos et f des mots du français normalisé. Le modèle est basé sur un alignement mot-à-mot entre f et e. Dans l'algorithme de recherche vorace décrit ci-dessous, la fonction qui génère de nouvelles hypothèses comprend une opération d'insertion de mots qui vise à combler les lacunes de ce modèle mot-à-mot. La simplicité de ce modèle, et de la fonction de score utilisée (voir section 3), est cohérente avec une approche par recherche vorace.

3 Algorithme

La technique mise en application ici est basée sur l'algorithme vorace de recherche décrit dans (Langlais *et al.*, 2007). Cet algorithme fait appel à trois fonctions : la première (*Seed*) génère une traduction initiale, la deuxième (*Score*) attribue aux traductions un score que l'on tente de maximiser, et la troisième (*Neighborhood*) génère, au moyen de différentes transformations, un ensemble d'hypothèses à tester à la prochaine itération, jusqu'à ce que le score plafonne. Dans (Langlais *et al.*, 2007), la fonction Seed choisit simplement la traduction la plus probable selon un modèle de traduction à segments ; la fonction Score est une combinaison log-linéaire de

modèles :

$$\begin{aligned} Score(e,f) = & \lambda_{lm} \log\, p_{lm}(f) & + \\ & \sum_i \lambda^i_{tm} \log\, p^i_{tm}(f|e) & - \\ & \lambda_w |f| & - \\ & \lambda_d p_d(e,f) \end{aligned}$$

où les λ sont des coefficients, p_{lm} est un modèle de langue, p^i_{tm} sont les différents modèles de traduction, $|f|$ est la longueur de la traduction et $p_d(e,f)$ est un modèle de distorsion.

L'algorithme vorace applique itérativement la fonction Neighborhood à une traduction courante et maximise le score jusqu'à ce qu'il plafonne.

Nous appliquons ici l'algorithme vorace au problème de la normalisation de textos. L'approche consiste globalement à :
– Générer une première normalisation plausible (Seed)
– Attribuer un score à cette normalisation (Score)
– Générer des nouvelles hypothèses au moyen de transformations (Neighborhood)
– Boucler les deux étapes précédentes jusqu'à ce que le score plafonne

3.1 Fonction Seed

Pour générer la normalisation initiale, deux méthodes sont comparées : recherche locale de la normalisation la plus probable pour chaque mot ; et identification de la meilleure normalisation par décodage de type Viterbi.

En ce qui concerne le décodage de type Viterbi, il est effectué à l'aide de la commande *Disambig* de SRILM (Stolcke, 2002), que nous utilisons pour produire la normalisation la plus probable étant donné une phrase source et un modèle de traduction. On peut également fournir à ce programme un modèle de langue afin qu'il maximise $p(e|f) \cdot p(f)$ plutôt que seulement $p(e|f)$.

3.2 Fonction Score

Nous simplifions la fonction de score de la façon suivante :

$$Score(e,f) = \lambda_{lm} \log\, p_{lm}(f) + \lambda_{tm} \log\, p_{tm}(e|f)$$

Le score utilisé maximise donc $p(e|f) \cdot p(f)$, ces deux probabilités étant déterminées au moyen des modèles de traduction et de langue. En ce qui concerne $p_{tm}(e|f)$, ce terme est calculé suivant la méthode IBM1 :

$$p(e_1^J|f_1^I) = \prod_{j=1}^{J} \left(\frac{1}{I} \sum_{i=0}^{I} p(e_j|f_i) \right)$$

Quant à $p_{lm}(f)$, nous calculons le produit des probabilités des trigrammes d'une phrase [2] (des tokens de début et de fin de phrase sont ajoutés). Ces probabilités sont tirées du modèle de langue.

3.3 Fonction Neighborhood

(Langlais *et al.*, 2007) décrivent six opérations mises en application dans la fonction Neighborhood, dont quelques-unes sont propres aux modèles à segments utilisés dans ce travail, alors que l'approche utilisée ici traduit (normalise) mot à mot. En revanche, les opérations *Swap*, qui intervertit deux mots adjacents, et *Replace*, qui remplace un segment dans la traduction par d'autres segments présents dans les modèles de traduction, s'appliquent très bien au modèle de traduction mot-à-mot. Nous appliquons aussi une opération que les auteurs ont suggérée, c'est-à-dire l'insertion de mots.

Celle-ci consiste à insérer des mots à n'importe quelle position dans une phrase, le vocabulaire des mots à insérer pouvant être déterminé de différentes façons. Nous mettons à l'épreuve deux variantes. L'opération *Insert_sp* insère seulement des mots que (Brown *et al.*, 1993) qualifient de *spurious*, c'est-à-dire des mots de la phrase cible qui ne sont alignés avec aucun mot dans la phrase source. Ceux-ci sont identifiés automatiquement à partir du modèle de traduction, en repérant tous les mots qui sont associés au mot vide. La deuxième opération, que nous appelons *Insert_tr*, insère d'autres traductions présentes dans le modèle de traduction pour les mots de la phrase source, l'objectif étant de combler les lacunes du modèle mot-à-mot, qui risque de proposer une traduction incorrecte dans les cas où un mot source doit être traduit par plus d'un mot cible.

En somme, la fonction Neighborhood fait appel à quatre opérations :
– Swap : intervertir deux mots adjacents
– Replace : remplacer un mot cible par d'autres équivalents potentiels
– Insert_tr : insérer d'autres équivalents potentiels d'un mot source
– Insert_sp : insérer des mots *spurious*
Les opérations Insert_sp et Swap seront utilisées dans toutes les versions évaluées ici sauf indication contraire, tandis que Replace et Insert_tr feront l'objet d'évaluation distinctes.

4 Analyse des résultats

4.1 Seed et Neighborhood

L'objectif principal de cette évaluation est de mettre à l'épreuve différentes façons d'obtenir la normalisation initiale (fonction Seed) et de générer des nouvelles hypothèses (Neighborhood). Avant de procéder à ces tests, nous avons d'abord enrichi manuellement la liste de mots *spurious* exploitée par l'opération Insert_sp. Une analyse rapide des mots extraits du modèle de traduction a montré que plus de la moitié étaient des mots de classes fermées. Nous avons complété les

2. Notre programme exploite un wrapper pour Python qui permet d'interroger SRILM (Madnani, 2009). Voir `http://www.desilinguist.org`.

listes d'articles, de déterminants démonstratifs et possessifs et de pronoms, ajoutant 32 mots à la liste. Une légère diminution du WER a été observée, à très faible coût.

Seed	IT	WER (%)	SER (%)	BLEU
Baseline		21,01	62,29	0,5683
Topword	Non	31,87	75,42	0,4202
	Oui	29,37	74,63	0,4382
Dis	Non	31,45	74,98	0,4237
	Oui	28,92	74,36	0,4456
Dis2	Non	14,05	53,92	0,7169
	Oui	12,22	48,63	0,7468
Dis3	Non	12,78	49,96	0,7394
	Oui	**11,05**	**43,88**	**0,7674**

TABLE 1 – Influence de Seed et de Insert_tr

Les scores qu'offrent différentes variantes de la fonction Seed sont présentées dans la table 1. Pour chacune des techniques, deux variantes de la fonction Neighborhood sont évaluées. Chacune comprend les opérations Swap et Insert_sp, mais nous activons et désactivons l'opération Insert_tr (indiqué dans la colonne *IT*). En ce qui concerne les variantes de Seed, *Topword* choisit simplement le mot cible le plus probable pour chaque mot source. *Dis* utilise le décodage Viterbi au moyen de Disambig, mais n'exploite aucun modèle de langue, seulement un modèle de traduction. *Dis2* exploite un modèle de langue bigramme et *Dis3*, un modèle trigramme. Enfin, pour déterminer le *baseline*, nous conservons simplement le texto de départ.

Les résultats montrent que les techniques naïves de génération de la normalisation initiale offrent des scores très pauvres, Topword et Dis obtenant des résultats à peu près équivalents. Or, lorsqu'on fournit un modèle de langue à Disambig, les scores deviennent nettement meilleurs. Cela suggère que cette implémentation de l'algorithme nécessite une normalisation initiale d'une certaine qualité.

Nous avons également évalué la fonction Replace, qui parcourt les mots de la source, extrait tous les équivalents du modèle de traduction, cherche la traduction du mot source dans la traduction courante, et la remplace par chacun des équivalents. Nous l'avons implémentée dans la version du programme qui obtient les meilleurs résultats, c'est-à-dire Dis3 avec Insert_tr, et le taux d'erreurs moyen par phrase ne diminue pas ; au contraire, il augmente d'environ 4 %, et le score BLEU diminue de 2 %. Il semble donc que l'opération Replace n'est pas bénéfique, du moins lorsque les normalisations initiales sont de bonne qualité. Nous montrerons dans la section suivante que le contraire est vrai lorsque celles-ci sont moins bonnes.

4.2 Amélioration des normalisations générées naïvement

Ayant identifié une combinaison de fonctions qui produit des résultats satisfaisants, nous cherchons à vérifier dans quelle mesure l'algorithme vorace de recherche améliore la qualité des normalisations fournies par la fonction Seed la plus naïve, c'est-à-dire Topword.

La table 2 présente les résultats de cette évaluation. *Dis3* indique les résultats qu'on obtient simplement en laissant à Disambig le soin de choisir la meilleure normalisation étant donné un modèle de traduction et un modèle de langue trigramme. *Greedy_search* désigne l'implémentation de l'algorithme qui obtient les meilleurs résultats : Dis3 est utilisé pour la traduction initiale, et la fonction Neighborhood comprend les opérations Swap, Insert_sp et Insert_tr. *TW* indique les résultats qu'on obtient par la méthode Topword, sans application de l'algorithme vorace. Par la suite, on montre comment la performance de l'algorithme vorace varie à mesure qu'on ajoute des opérations à la fonction Neighborhood : on désigne Swap par *SW*, Insert_sp par *IS*, Insert_tr par *IT* et Replace par *RE*.

Les résultats montrent que l'algorithme vorace n'améliore pas énormément la qualité des normalisations produites par Dis3, qui sont déjà beaucoup plus proches des normalisations de référence. Or, nous arrivons tout de même à réduire le taux d'erreurs moyen par phrase (WER) de presque moitié et à augmenter le score BLEU d'environ 35 % par rapport au baseline.

Si l'apport de l'algorithme vorace n'est pas énorme lorsque les normalisations initiales sont bonnes, il devient considérable lorsque celles-ci sont générées grossièrement. Les normalisations générées par Topword s'éloignent nettement des normalisations de référence, et Swap et Insert_sp ne les améliorent pas. Par contre, Replace (et dans une moindre mesure Insert_tr) est très bénéfique, offrant une réduction du taux d'erreurs moyen de l'ordre de 40 % et une augmentation du score BLEU d'environ 47 %. Ces gains sont attribuables, du moins en partie, au rôle que joue le modèle de langue, qui permet par ailleurs d'améliorer les normalisations générées par *Dis*, comme nous l'avons vu. Malgré ces gains, nous obtenons des meilleurs résultats lorsque les normalisations de départ sont déjà de bonne qualité, intégrant un modèle de langue. Rappelons aussi que, lorsque les normalisation initiales sont bonnes, Replace n'a pas un effet favorable. Il nous semble que ces observations correspondent aux intuitions qu'on peut avoir par rapport à cette approche de la traduction (ou normalisation).

4.3 Évaluation sur le corpus de test

Les résultats de l'évaluation finale, effectuée sur le corpus de test, sont présentés dans la table 3. Nous évaluons le système qui fournit les meilleurs résultats sur le corpus de développement : la normalisation de départ est générée par Dis3, et la fonction Neighborhood utilise les opérations Swap, Insert_sp et Insert_tr pour générer des nouvelles hypothèses. Tout d'abord, on observe que les textos contiennent une proportion nettement plus élevée de formes non standard que ceux

Méthode	WER (%)	SER (%)	BLEU
Baseline	21,01	62,29	0,5683
Dis3	13,01	51,98	0,7230
Greedy_search	11,05	43,88	0,7674
TW	30,42	75,51	0,4051
TW+SW+IS	31,87	75,42	0,4202
TW+SW+IS+IT	29,37	74,63	0,4382
TW+SW+IS+IT+RE	17,78	51,81	0,5947

TABLE 2 – Impact de l'algorithme vorace de recherche

	WER	SER	BLEU
Baseline	28,90	68,60	0,4677
Greedy_search	19,32	57,70	0,6189

TABLE 3 – Évaluation sur le corpus de test

du corpus de développement, le WER étant 37, 6 % plus élevé. Ainsi, le WER des normalisations produites passe de 11, 05 % (sur le corpus de développement) à 19, 32 %. De plus, la diminution du WER observée en test, de 33 %, est inférieure à la diminution observée pendant la phase de développement (47 %). Or, si toute différence de WER de 30 % est considérée significative (Yvon, 2008), il mérite d'être souligné que nos résultats dépassent ce seuil. En ce qui concerne le score BLEU, le score des normalisations produites est beaucoup plus faible lorsqu'on évalue sur le corpus de test, mais l'augmentation du score BLEU (32 %) est cohérente avec celle que nous avons observée pendant le développement (35 %).

5 Conclusion

Dans ce travail, nous avons mis en application un algorithme de recherche vorace utilisé en traduction statistique dans le but de normaliser des textos. L'accent a été placé sur les fonctions qui génèrent la normalisation initiale et aux opérations permettant de générer des nouvelles hypothèses.

L'approche qui obtient les meilleurs résultats consiste à générer la normalisation initiale par décodage de type Viterbi à partir des modèles de traduction et de langue ; à utiliser les opérations d'alternance et d'insertion de mots afin de générer des nouvelles hypothèses ; et à maximiser la fonction de score. Cette méthode engendre une diminution du taux d'erreurs moyen par phrase de 33 % lors de l'évaluation finale, et une augmentation du score BLEU de plus de 30 %.

L'opération Replace, qui consiste à remplacer des mots dans la normalisation courante par d'autres équivalents tirés du modèle de traduction, n'a pas un effet bénéfique lorsque les normalisations initiales sont de bonne qualité. Or, lorsque celles-ci sont générées par une simple recherche locale du mot cible le plus probable pour chaque mot source, l'opération Replace permet d'améliorer la qualité des normalisations, notamment grâce à l'apport du modèle de langue.

Ces techniques simples fournissent des résultats qui nous semblent intéressants. Il nous paraît donc profitable de traiter la normalisation des textos comme un problème de traduction intralinguistique.

Remerciements

Nous désirons remercier Philippe Langlais, ainsi que les relecteurs, pour leurs commentaires et leurs suggestions sur ce travail. Nous remercions M. Langlais ainsi que Fabrizio Gotti pour les ressources mises à notre disposition. Nous remercions également le Fonds de recherche du Québec – Société et culture pour son soutien financier.

Références

Aw, A., Zhang, M., Xiao, J. et Su, J. (2006). A Phrase-Based Statistical Model for SMS Text Normalization. *In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 33–40, Sydney (Australie). Association for Computational Linguistics.

Beaufort, R., Roekhaut, S., Cougnon, L.-A. et Fairon, C. (2010). A Hybrid Rule/Model-Based Finite-State Framework for Normalizing SMS Messages. *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779, Uppsala (Suède). Association for Computational Linguistics.

Brown, P. F., Della Pietra, V. J., Della Pietra, S. A. et Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation : Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Kobus, C., Yvon, F. et Damnati, G. (2008). Normalizing SMS : are Two Metaphors Better than One ? *In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 441–448, Manchester (Angleterre). Coling 2008 Organizing Committee.

Langlais, P., Drouin, P., Paulus, A., Brodeur, E. R. et Cottin, F. (à paraître, 2012). Texto4science : a Quebec French Database of Annotated Short Text Messages. *In Proceedings of Language Resources and Evaluation Conference (LREC) 2012*, Istanbul (Turquie). ELRA.

Langlais, P., Patry, A. et Gotti, F. (2007). A Greedy Decoder for Phrase-Based Statistical Machine Translation. *In Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 104–113, Skövde (Suède).

Madnani, N. (2009). Querying and Serving N-gram Language Models with Python. *The Python Papers*, 4(2).

Papineni, K., Roukos, S., Ward, T. et Zhu, W.-J. (2001). Bleu : A Method for Automatic Evaluation of Machine Translation. Rapport technique RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center.

Stolcke, A. (2002). SRILM – An Extensible Language Modeling Toolkit. *In Proceedings of ICSLP*, Denver (États-Unis).

Yvon, F. (2008). Reorthography of SMS Messages. Rapport technique 2008-18, LIMSI-CNRS.