Données Séquentielles et Symboliques : Translittération automatique

Alexandre Bérard, Mathias Millet, Charles Robin

14 janvier 2014

Résumé

1 Introduction

La translittération est le fait de remplacer chaque graphème d'un système d'écriture par graphème ou un groupe de graphèmes d'un autre système, indépendamment de la prononciation. La translittération peut être par exemple utilisée pour écrire un nom propre d'un alphabet dans un autre.

La translittération automatique est un domaine particulier de la translittération, qui va trouver des application par exemple pour traduire des mots qui ne sont pas encore dans les dictionnaires, ou traduire automatiquement des requêtes effectuées dans des systèmes d'écriture peu utilisés sur internet [Singh et al., 2010].

1.1 Jeux de données

Les données nous ont été fournies par Vincent Claveau. Nous disposons de deux jeux de données, concernant respectivement la translittération de l'*espagnol* au *portugais* et de l'*anglais* vers le *russe*. Chacun de ces jeux de données est divisé en deux fichiers, un fichier pour l'apprentissage et un fichier pour l'évaluation. Le fichier d'apprentissage contient plusieurs milliers d'entrées (3057 pour le portugais, et 7262 pour le russe), chaque entrée correspondant à un mot dans la langue originale et sa transcription dans le langage cible. Les entrées dans le fichier d'évaluation, elles, peuvent cependant contenir plusieurs transcriptions pour un même mot.

FIGURE 1 – Entrées dans le fichier d'apprentissage Espagnol-Portugais

```
2996: #fotocopiado# #fotocópia#
2997: #hexanoles# #hexanóis#
2998: #catalasa# #catalase#
```

FIGURE 2 – Entrées dans le fichier d'évaluation Espagnol-Portugais

```
182: #centrifugación# #centrifugação# other unknown
183: #centriolo# #centríolo# OR #centríolos# other unknown
184: #ceramida# #ceramida# OR #ceramidas# other unknown
```

TABLE 1 – Longueur moyenne des mots dans chaque langue

Langue	Longueur moyenne
Espagnol	11.74
Portugais	11.52
Anglais	12.27
Russe	11.64

1.2 Processus d'évaluation

Ensembles de test et d'apprentissage Comme nous disposons de données de test et d'apprentissage, la question de la méthode d'échantillonage à utiliser ne se pose pas.

Métriques d'évaluation Nous devons choisir une métrique afin de mesurer les performances de nos classifieurs, et fournir une moyen de comparaison (avec d'autres classifieurs, une baseline, ou un opérateur humain). Nous mesurons la précision de notre système, c'est à dire la proportion de mots correctement traduits. Cependant, ce n'est pas suffisant car si la transcription choisie n'est pas correcte, il se peut qu'elle ne soit pas loin de la vérité.

Le même problème se pose dans la traduction automatique, de manière encore plus accentuée car le vocabulaire est beaucoup plus grand et les candidats de traduction sont souvent bien plus nombreux. Même dans les systèmes de traduction modernes, il est très fréquent que la traduction comporte au moins une erreur sans toutefois empêcher la compréhension de la phrase traduite. Une métrique couramment utilisée dans ce type de système, est le *Word Error Rate*, qui est une distance d'édition (Levenshtein) adaptée aux phrases [1]. Aussi, pour

1. On compte le nombre de mots qui diffèrent.

notre système de translittération nous utiliserons une distance de *Levenshtein*[2] pour évaluer la distance entre la proposition de notre système et la/les transcription(s) attendu(es).

Dans les données de test, les mots peuvent posséder plusieurs transcriptions acceptables. Dans ce cas, on mesure la distance au mot le plus similaire à la proposition du système.

$d(x, y) = \min_{y' \in traductions(x)}\{\delta(y, y')\}$

où $\delta(y, y') = levenshtein(y, y')$

Pour la précision, nous jugeons que la transcription est correcte si elle est égale à une des transcriptions possible.

Ces deux métriques donnent un résumé acceptable des performances du système.

2 Règles de substitution

Nous avons observé que dans le cas de l'espagnol et le portugais, les mots sont très similaires dans les deux langages. Dans les données d'apprentissage, la distance d'édition moyenne entre un mot espagnol et sa transcription en portugais est de *2.0*. Lorsque nous utilisons, comme *baseline* pour notre système de translittération, un système qui se contente de renvoyer le mot original, nous obtenons donc une précision de *51.0%* et une distance d'édition moyenne de *1.06* (au mot le plus similaire dans les candidats).

Nous avons ensuite constaté qu'un système mettant en jeu les trois règles de substitution suivantes (que nous avons sélectionné manuellement), obtenait une précision de *58.6%*, et une distance d'édition moyenne de *0.76*.

```
is#->e#
ción#->ção#
ido#->ídeo#
```

La progression est flagrante, avec seulement trois règles. Cela nous a amené à penser qu'un système sélectionnant automatiquement des règles de substitution basiques, pourrait obtenir de très bons résultats.

2.1 Application des règles

Dans le fichier de règles, les règles sont listées par ordre de priorité. Lorsqu'un mot doit être traduit, on tente d'abord d'appliquer la première règle, puis la deuxième, etc. L'application d'une règle est simplement la substitution dans le mot de toutes les occurences de la partie gauche de la règle par la partie droite de la règle. Afin d'éviter les incohérences, nous prenons soin de ne pas appliquer une règle sur une portion du mot qui a déjà été modifiée. Si la règle chevauche

2. Nombre minimal d'insertions, suppressions ou substitutions de lettres dans le mot original pour obtenir le mot cible.

une partie modifiée par une autre règle déjà appliquée, elle est tout simplement ignorée.

2.2 Apprentissage

Alignement Pour l'apprentissage de règles, nous commençons par aligner toutes les données d'apprentissage. Comme les mots sont très similaires, nous utilisons une simple distance de Levenshtein : l'alignement est construit à partir de la séquence d'opérations qui minimise la distance d'édition [3]. Un alignement se présente sous la forme suivante :

```
[('#catars', '#catars'), ('is', 'e'), ('#', '#')]
```

De cet alignement, on pourrait déduire la règle simple : `is -> e`. Seulement, cette règle est beaucoup trop générale, et possède probablement un taux de confiance faible. Il est possible cependant que les règles `sis -> se`, ou `is# -> e#` soient plus fiables.

D'un autre côté, si la règle `catarsis -> catarse` est très fiable, son utilité est très discutable. En effet, elle revient a faire de l'apprentissage par cœur, et n'a aucun pouvoir de généralisation.

L'objectif est donc de produire des règles ayant un bon compromis entre *fiabilité* (qui sera évaluée par un *taux de confiance*), et généralisation (évaluée par la longueur de la règle, et son *support*).

Génération des candidats La production des règles se fait donc en énumérant, pour chaque alignement dans le corpus d'apprentissage, la liste des candidats de partie gauche de règle. Dans le cas de l'exemple, la liste des candidats est : `['is', 'sis', 'is#', 'sis#', ...]`. Seuls les candidats de longueur supérieure au seuil l sont conservés. Les candidats sont générés à partir des parties du mot dont l'alignement diffère, en étendant itérativement vers la droite et vers la gauche.

Calcul du support On calcule ensuite le support de chacun de ces candidats sur l'ensemble des données (`len([w for w in words if 'is' in w])`). Les candidats avec un support supérieur au seuil s sont conservés. Cette étape permet de s'assurer que les règles sont suffisamment générales, et aussi de réduire le temps d'exécution (car l'étape suivante est beaucoup plus coûteuse).

Taux de confiance L'étape suivante est la génération de partie droite de règle. En parcourant l'ensemble de données, on calcule pour chaque candidat la partie droite de règle offrant le meilleur taux de confiance. Si ce taux de confiance est au-dessus du seuil c, alors la règle est conservée.

3. Il y a quatre types d'opérations : *égal*, *insertion*, *suppression*, *substitution*. Les lettres successives concernées par des opérations *égal* sont regroupées. Et les trois autres types d'opérations sont regroupés. Dans l'exemple, la lettre 'i' subit une *suppression*, et la lettre 's' une *substitution* par 'e'. Dans le résultat, 'is' est aligné avec 'e'

Par exemple, la règle `is -> e` est jugée incorrecte pour l'alignement suivant : `[('#hist', '#hist'), ('o', 'ó'), ('ria#', 'ria#')]`. En effet, le motif `'is'` y est traduit par `'is'`, et non `'e'`. De la même manière, pour l'alignement suivant : `[('#imagina', '#imagina'), ('ción', 'ção'), ('#', '#')]`, la règle `nac -> naç` est incorrecte, car incomplète. Le taux de confiance d'une règle est calculé en divisant le nombre de fois où la règle est jugée correcte, par le nombre total d'occurences de la partie gauche de la règle.

Tri des règles La dernière étape est de trier les règles selon leur ordre de priorité. Les règles les plus spécifiques doivent être appliquées avant les règles générales (sinon elles ne pourront jamais être appliquées). Nous trions donc les règles par ordre de taille (décroissante), une règle ne pouvant pas être plus générale qu'une règle plus courte. En fonction de la valeur du paramètre l, cette approche peut donner *+1% à +2%* en précision par rapport à un tri inversé. Une piste d'amélioration pourrait être de considérer leur taux de confiance dans l'ordre de tri des règles.

2.3 Résultats

Le score maximal que nous parvenons à atteindre avec ce système est de *69.0%*. Ces résultats sont sur le jeu de données *Espagnol-Portugais*.

TABLE 2 – Types de règles et résultats

Règles	Précision	Distance d'édition
Aucune	51.0%	1.06
3 (manuel)	58.6%	0.76
584 ($s = 2, l = 5, c = 0.8$)	**69.0%**	**0.58**

TABLE 3 – Valeur des différents paramètres et résultats

Support s	Longueur l	Confiance c	Précision	Distance	Règles
2	6	0.80	56.8%	0.82	104
2	5	0.80	**69.0%**	**0.58**	315
2	4	0.80	68.1%	0.59	584
3	5	0.80	67.3%	0.61	195
5	5	0.80	67.2%	0.61	**128**
1	5	0.80	64.5%	0.68	1490
2	5	0.75	68.5%	0.59	340
2	5	0.85	68.4%	0.59	287

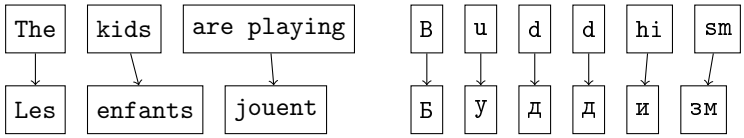
Nous n'avons pas considéré l'utilisation de cette méthode sur les données *Anglais-Russe*, car l'alphabet est complétement différent. L'hypothèse de simi-

larité des mots dans les deux langages n'est donc plus vérifiée. D'autres méthodes sont plus adaptées, notamment les méthodes *statistiques* (traduction statistique, CRF).

3 Translittération statistique

Le problème de la translittération est relativement proche de la traduction automatique. En traduction automatique, on traduit des séquences de mots d'un langage à un autre. Cela se caractérise par l'association à chaque mot (ou groupe de mots) de la phrase original, d'un mot (ou groupe de mots) dans le langage cible, avec un réordonnancement possible. En translittération, on associe des lettres ou groupes de lettres.

FIGURE 3 – Traduction et translittération



La différence essentielle, par rapport à la traduction automatique est l'absence de réarrangements dans le résultat. En effet, dans certaines paires de langages, les mots doivent être réordonnés (e.g. les adjectifs entre le français et l'anglais). Ce n'est pas le cas dans la translittération.

Traduction statistique L'approche de traduction automatique avec le plus de succès est la *traduction statistique*. La traduction statistique se base sur la *loi de Bayes* :

$$e = argmax_{e \in e*} p(e|f) = argmax_{e \in e*} p(f|e)p(e)$$

Où f est la phrase à traduire, $e*$ est l'ensemble des traductions possible, et e la traduction choisie. La probabilité $p(e)$ est le *modèle de langue*. Il s'agit de la probabilité d'observer la phrase e dans la langue cible. $p(f|e)$ est le modèle de traduction et dépend de l'approche utilisée. Ces probabilités sont souvent estimées avec un *maximum de vraissemblance* sur de larges corpus de textes alignés dans les deux langues.

La traduction syntagmatique (ou *phrase-based traduction*), considère des *syntagmes* comme unité de langage ([Koehn et al., 2003]). Un syntagme (ou *phrase*) est un groupe de mots constituant une unité syntaxique (par ex., "are playing" est un syntagme). Les anciennes approches basées sur les mots se heurtaient à certaines difficultés, notamment pour traduire les expressions (par ex. "kick the bucket"/"casser sa pipe").

Dans le cas de la translittération, l'approche syntagmatique est également très intéressante. Dans la *Figure 3*, "sm" et son équivalent russe sont des syntagmes.

Dans notre travail, nous utiliserons donc le framework de traduction statistique (syntagmatique) *Moses*, [Koehn et al., 2007].

3.1 Moses

Nous devons dans un premier temps modifier les données, pour que Moses les considère comme des phrases. Pour cela il suffit d'insérer des espaces entre les lettres (chaque lettre est un mot). Pour empêcher tout réarrangement de mots (traduction monotone), nous mettons le paramètre *distortion-limit* à la valeur 0. Les résultats sont similaires, mais le temps d'exécution largement réduit.

Nous nous sommes inspirés du travail de [Matthews, 2007], où la translittération de noms propres (du corpus *Gigaword*) est effectuée, en utilisant Moses.

La création d'un système de traduction avec Moses se fait en plusieurs étapes (manuel d'utilisation, [Koehn, 2014]) :

- Utilisation d'*IRSTLM* (ou autre) pour générer le modèle de langue à partir des données de la langue cible (les mots du russe ou du portugais dans notre cas) ;
- Alignement des données avec *Giza++* ;
- Création du modèle de traduction.

Paramètres Les paramètres d'apprentissage sont les suivants :

- Ordre des N-grammes du modèle de langue (3 est la valeur retenue) ;
- Réordonnancement (monotone dans notre cas) ;
- Longueur maximale des phrases (par défaut 20, 5 donne d'aussi bon résultats) ;

Dans Moses, la probabilité finale d'une traduction est le produit de quatre probabilités : la probabilité du modèle de traduction syntagmatique, la probabilité du modèle de langue, le modèle de réordonnancement (qui n'est pas considéré dans notre cas), et finalement la pénalité de longueur. Les paramètres du modèle sont les poids attribués à ces différentes probabilités.

3.2 Résultats

TABLE 4 – Valeurs optimales des poids

Données	Pénalité de longueur	Modèle de langue	Modèle de traduction
ENG-RUS	-0.3	0.65	`[1,0,1,0.2]`
SPA-POR	-0.7	0.35	`[1,0,1,0.2]`
Par défaut	-1	0.5	`[0.2,0.2,0.2,0.2]`

TABLE 5 – Résultats de la baseline (paramètres par défaut)

Données	Précision	Distance moyenne	Variance
ENG-RUS	59.1%	2.10	15.41
SPA-POR	56.9%	1.21	4.30

TABLE 6 – Résultats avec paramètres optimaux

Données	Précision	Distance moyenne	Variance
ENG-RUS	**67.8%**	**1.75**	14.47
SPA-POR	**71.5%**	**0.73**	3.46

FIGURE 4 – Histogramme des distances pour les données *ENG-RUS*

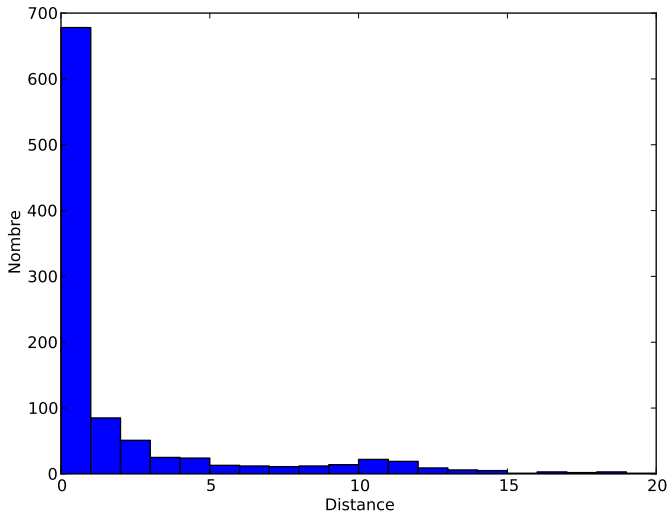
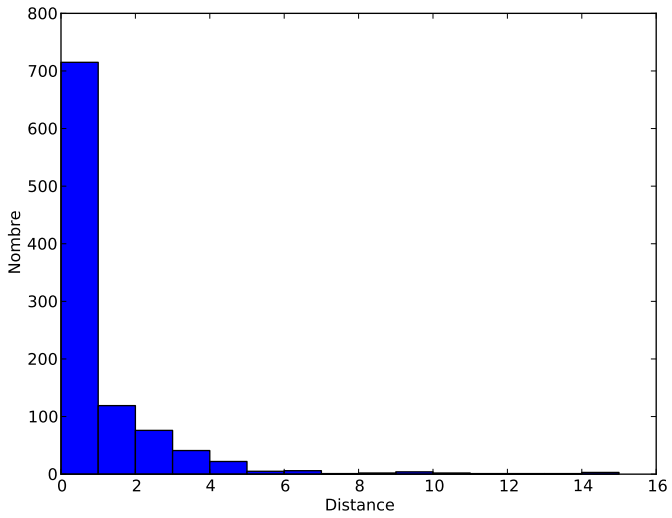


FIGURE 5 – Histogramme des distances pour les données *SPA-POR*



4 CRF

Les CRF *Conditionnal Random Fields* sont des modèles statisitques permettant de prendre plus de contexte qu'un classifieur "ordinaire". Ils permettent l'étiquettage de séquences. Ils ont été introduit par [Lafferty et al., 2001]. L'équation principale des CRF est donnée à l'équation 1.

$$p(\mathbf{s}|\mathbf{o}) = \frac{1}{Z_0} exp(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(...)) \quad (1)$$

Les f_k sont des fonctions *feature* données par l'utilisateur nécessitant certains arguments, et les λ_k sont des poids qui sont appris grâce aux données d'apprentissage, $\mathbf{s}$ et $\mathbf{o}$ étant respectivement une séquence d'état et la séquence des observations (ici les lettres). Les fonctions features sont des fonctions qui ont pour valeur 0 sur presque tout leur domaine et 1 sinon.

Nous avons utilisé dans le cadre de ce projet, l'implémentation CRF++ [4].

4.1 Apprentissage

Dans le cas de l'apprentissage des CRF, nous avons dû en premier lieu aligner lettre à lettre les mots et leurs traductions dans l'ensemble d'apprentissage. Pour celà nous avons utilisé l'algorithme *dpalign* dont une implémentation en perl est disponible sur le site du cours. Nous avons ensuite donné ces données à CRF++ en plus d'un fichier *template* décrivant les fonctions features que nous souhaitions utiliser. Nous avons choisi comme fonctions features des unigrammes. Le contenu du fichier template est le suivant :

```
# Unigram
U0:%x[0,0]
U1:%x[-1,0]
U2:%x[1,0]
U3:%x[-2,0]
U4:%x[2,0]
U5:%x[-3,0]
U6:%x[3,0]

# Bigram
B
```

4.2 Évaluation

Nous testons ensuite le modèle CRF obtenu grace aux données de test, en suivant le protocole décrit à la section 1.2. Il est important de noter qu'aucun alignement n'a été réalisé sur ces données, ce qui a pu poser quelques problèmes. En effet, il est possible d'étiquetter une et une seule étiquette par observation, il est donc impossible de traduire un mot par un mot plus long.

4. https ://code.google.com/p/crfpp/

4.3 Résultats

Un récapitualatif de nos résultats est disponible ci-dessous sur le jeu de données *Espagnol-Portugais*. Nous trouvons des résultats un peu moins bons que grace aux règles de substitutions, cependant ils sont meilleurs que notre baseline ainsi qu'avec les trois règles de substitutions trouvées manuellement.

Règles	Précision	Distance d'édition
Baseline	51.0%	1.06
3 subst.	58.6%	0.76
CRF	**63.9%**	**0.69**

5 Conclusion et perspectives

Références

[Koehn, 2014] Koehn, P. (2014). *Moses, Statistical Machine Translation System, User Manual and Code Guide*. University of Edinburgh.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., and Moran, C. (2007). Moses : Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, ICML '01.

[Matthews, 2007] Matthews, D. (2007). Machine transliteration of proper names. Master's thesis, University of Edinburgh.

[Singh et al., 2010] Singh, A. K., Subramaniam, S., and Rama, T. (2010). Transliteration as alignment vs. transliteration as generation for crosslingual information retrieval. *TAL*, 51(2) :95–117.