Données Séquentielles et Symboliques : Translittération automatique

Alexandre Bérard, Mathias Millet, Charles Robin

13 janvier 2014

Résumé

1 Introduction

La translittération est le fait de traduire, entre deux langues dont les graphèmes diffèrent [1], un mot ayant la même (ou presque) prononciation dans les deux langues. C'est surtout le cas pour le vocabulaire technique, qui conserve la même prononciation entre plusieurs langues, mais dont la transcription va varier (en particulier lorsque les langues sont écrites avec des alphabets différents).

La translitération trouve un interêt lorsqu'elle est effectuée de manière automatique. En effet, avec l'aparition de nouveaux termes techniques, il est difficile de maintenir à jour des dictionnaires.

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.

1. Un graphème est l'unité de l'écrit correspondant au phonème, une unité de prononciation orale

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
```

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 [2]. Aussi, pour notre système de translittération nous utiliserons une distance de *Levenshtein* [3] 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 $\delta(y, y') = levenshtein(y, y')$

2. On compte le nombre de mots qui diffèrent.
3. Nombre minimal d'insertions, suppressions ou substitutions de lettres dans le mot original pour obtenir le mot cible.

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 sélectionnées 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.

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

Apprentissage 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 [4]. Un alignement se présente sous la forme suivante :

4. 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'

`[('#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*).

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. Pour l'exemple, les candidats sont `['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.

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).

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.

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.

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*.

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**

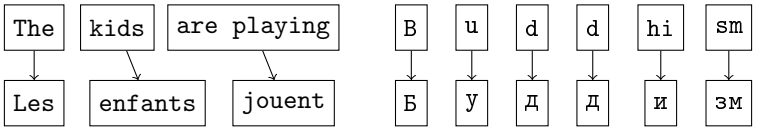
Support s	Longueur l	Confiance c	Précision	Distance	Règles
2	6	0.80	56.8%	0.82	104
2	5	0.80	**69.0%**	**0.58**	315
2	4	0.80	68.1%	0.59	584
3	5	0.80	67.3%	0.61	195
5	5	0.80	67.2%	0.61	**128**
1	5	0.80	64.5%	0.68	1490
2	5	0.75	68.5%	0.59	340
2	5	0.85	68.4%	0.59	287

Remarques Cette approche est relativement rudimentaire, et pourrait bénéficier de certaines améliorations.

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.

FIGURE 3 – Alignement de mots en traduction, et alignment de lettres en translittération



4 Conclusion et perspectives

Références

[1] Yaser Al-Onaizan and Kevin Knight. Machine transliteration of names in arabic text. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, SEMITIC '02, pages 1–13. Association for Computational Linguistics, 2002.

[2] Kevin Knight and Jonathan Graehl. Machine transliteration. *Computational Linguistics*, 24(4) :599–612, 1998.

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

[4] Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. A comparison of different machine transliteration models. *J. Artif. Int. Res.*, 27(1) :119–151, 2006.

[5] Anil Kumar Singh, Sethuramalingam Subramaniam, and Taraka Rama. Transliteration as alignment vs. transliteration as generation for crosslingual information retrieval. *TAL*, 51(2) :95–117, 2010.