Université de Paris

Extraction non supervisée de lexique bilingue

Guillaume Wisniewski
guillaume.wisniewski@limsi.fr

mars 2020

1 Extraction de lexique

L'objectif de ce TP est d'extraire automatiquement un lexique bilingue d'un ensemble de textes. Un lexique bilingue est un dictionnaire [1] qui associe à un mot d'une langue source la liste de ses traductions possibles dans une langue cible. La Table 1 donne un extrait d'un lexique français–anglais.

cuisine	cooks, cook, cooking, kitchen, food...
comprendre	understand, understood, understanding, realize, ...
être	human, being, be, ...
économie	free-market, Economy, economics, market, economy, ...

TABLE 1 – Extrait d'un lexique français–anglais

L'extraction sera *non supervisée* : la méthode utilisée prendra en entrée uniquement un ensemble de paire de phrases (une phrase en français et sa traduction en anglais). On appelle « corpus parallèle » un tel corpus et on qualifie de « parallèles » des phrases qui sont traductions l'une de l'autre. La Table 2 donne un exemple de corpus parallèle.

L'approche proposée est fondée sur une théorie linguistique, l'*hypothèse distributionnelle* : le sens d'un mot peut se déduire directement du contexte dans lequel celui-ci apparait. La généralisation au cas bilingue de cette hypothèse peut se formuler de la manière suivante : un mot français et un mot anglais qui *co-occurent* (apparaissent dans une paire de phrases parallèles) fréquemment ont une grande chance d'être traduction l'un de l'autre. Ainsi sur l'exemple du corpus de la Table 2, il est naturel de supposer que *cuisine* se traduit soit par *cooking*, soit par *kitchen* puisque ce sont les seuls mots qui apparaissent dans toutes les traductions.

1. aussi bien au sens informatique qu'au sens « général »

Living on my own, I really miss my Mom's **cooking**.
Vivant seul, la **cuisine** de ma mère me manque.
She left the **kitchen** with the kettle boiling.
Elle quitta la **cuisine** avec la bouilloire.
Is there any coffee in the **kitchen**?
Y a-t-il encore du café dans la **cuisine** ?
Cooking runs in my family.
La **cuisine** c'est de famille.
Both boys and girls should take **cooking** class in school.
Garçons et filles devraient suivre des cours de **cuisine** à l'école.

TABLE 2 – Exemple d'un corpus parallèle français–anglais

Πάω στο σπίτι μας.
Je vais chez nous (*lit.* dans notre maison).
Το σπίτι μου είναι μεγάλο.
Ma maison est grande
Το σπίτι της Μελίνας είναι κοντά στην παραλία.
La maison de Mélina est à côté de la plage.
Με λένε Μελίνα.
Je m'appelle Mélina.
Ένα σπίτι του χωριού κάηκε
Une maison du village a brûlé.
Αγαπώ την Μελίνα.
J'aime Mélina.

TABLE 3 – Exemple d'un corpus parallèle grec–français

1. En considérant le corpus français–grec, décrit Table 3 pouvez-vous donner (en les justifiants) les traductions en grec des mots `Elli`, `est` et `maison`.

2 Approche naïve

Vous trouverez sur le site du cours deux fichiers contenant le roman *Voyage au centre de la Terre* en anglais et en français. Les documents ont été pré-traités pour :

— être alignés au niveau des phrases : la i^{e} ligne du fichier en français est la traduction de la i^{e} ligne du fichier anglais.

— segmentés en mots : cette segmentation consiste à séparer certains signes (par exemple « `à_l'école.` » est ré-écrit en « `à_l'_école_.` ») et à en regrouper d'autres (« 100 000 » est récrit en « 100000 »).

L'extraction d'un lexique à partir de ce corpus parallèle repose sur la construction d'un table de co-occurences. Cette table peut-être modélisée par une variable de type `HashMap<String, HashMap<String, Integer>` > qui associe à un mot français (la 1re clef), une `HashMap` dont les clefs sont l'ensemble des mots anglais co-occurant avec le mot français et les valeurs le nombre de paires de phrases dans lesquelles le mot français et le mot anglais apparaissent tous les deux. Par exemple, si le corpus est constitué de deux phrases :

doc n° 1	doc n° 2
la vache et le veau	the cow and the calf
le chien et le chat	the dog and the cat

la table extraite sera :

```
{'chat': {'and': 1, 'the': 1, 'dog': 1, 'cat': 1},
 'chien': {'and': 1, 'the': 1, 'dog': 1, 'cat': 1},
 'et': {'and': 2, 'calf': 1, 'cat': 1, 'cow': 1, 'dog': 1, 'the': 2},
 'la': {'and': 1, 'the': 1, 'cow': 1, 'calf': 1},
 'le': {'and': 2, 'calf': 1, 'cat': 1, 'cow': 1, 'dog': 1, 'the': 2},
 'vache': {'and': 1, 'calf': 1, 'cow': 1, 'the': 1},
 'veau': {'and': 1, 'the': 1, 'cow': 1, 'calf': 1}}
```

2. Quel est l'intérêt de la segmentation en mot ?
3. Écrivez une méthode `countSentencesWithWord` dont l'entrée est un nom de fichier et qui renvoie une `HashMap` associant à chaque mot du fichier le nombre de phrases dans lequel il apparait. **Attention** : on cherche à déterminer le nombre de phrases dans lequel un mot apparait, et même si un mot apparait plusieurs fois dans une même phrase, il ne faut le compter qu'une fois. Il est possible de supprimer les doublons d'une liste en utilisant l'instruction :

```
HashSet<String> uniqWord = new HashSet<String>(Arrays.asList(tab));
```

qui à partir d'un tableau de `String` (par exemple `["a", "b", "a"]`) construit une *liste* sans doublons (si on reprend l'exemple précédent `["b", "a"]`).

4. Écrivez une méthode `buildCoocTable` qui prend en argument deux noms de fichiers décrivant un corpus parallèle et retourne la table de co-occurence de ce corpus. À nouveau, il faut penser à supprimer les doublons à l'intérieur d'une phrase avant de réaliser les comptages.
5. À l'aide de la méthode `printSortedCoocTable`, afficher la table de co-occurences par ordre de fréquence décroissant.
6. Interprétez le résultat obtenu : est-ce que cette méthode permet de construire un « bon » lexique ?

Remarque : pensez à regarder le squelette qui vous est fourni sur le site du cours.

3 Tests statistiques

L'approche de la section précédente n'est utilisable que si l'on dispose d'un moyen de distinguer les associations « porteuses de sens » (celles que l'on observe dans un corpus parce que les mots sont traduction l'un de l'autre) des associations dues au hasard (par exemple, celles que l'on observe que parce que notre corpus est trop petit ou à cause des mots fréquents).

Les *tests de significativité* sont des outils statistiques qui répondent à cet objectif. Dans la suite du TP nous proposons d'utiliser un de ces tests, le *likelihood ratio*, pour filtrer la table des co-occurences construite dans la partie précédente et ne garder que les associations les *significatives*. Ce test évalue l'intérêt d'une association entre le mot a et le mot b à partir de sa table de contingence. Cette table décrit :

— $n(a, b)$ le nombre de paires de phrases dans lesquelles a et b co-occurent ;
— $n(\neg a, b)$ le nombre de paires de phrases dans lesquelles b apparait, mais pas a ;
— $n(a, \neg b)$ le nombre de paires de phrases dans lesquelles a apparait, mais pas b ;
— $n(\neg a, \neg b)$ le nombre de paires de phrases dans lesquelles ni a, ni b n'apparaissent.

La table 4 résume ces notations.

	nbre phrase avec a	nbre phrase sans a	
nbre phrase avec b	$n(a, b)$	$n(\neg a, b)$	$n(b)$
nbre phrase sans b	$n(a, \neg b)$	$n(\neg a, \neg b)$	$n(\neg b)$
	$n(a)$	$n(\neg a)$	N

TABLE 4 – Table de contingence

Le *likelihood ratio* est défini par :

$$G^2 = 2 \cdot N \left[\sum_{a? \in \{a, \neg a\}} \sum_{b? \in \{b, \neg b\}} p(a? \text{ et } b?) \cdot \log \frac{p(a? \text{ et } b?)}{p(a?) \cdot p(b?)} \right]$$

où $p(x?)$ est la fréquence de l'événement $x?$ (le rapport entre le nombre de fois où $x?$ est vrai et le nombre de fois où $x?$ est vrai *ou* faux). Par exemple, $p(a) = \frac{n(a)}{N}$ et $p(a, \neg b) = \frac{n(a,\neg b)}{N}$.

7. On note $n(a)$ (resp. $n(b)$) le nombre de phrases dans lesquelles a (resp. b) apparaît et N le nombre total de phrases. Exprimez $n(\neg a, \neg b)$, $n(\neg a, b)$ et $n(a, \neg b)$ en fonction de $n(a)$, $n(b)$, $n(a, b)$ et N.
8. Écrivez une méthode qui calcule le *likelihood ratio* d'une paire mot français / mot anglais. Cette méthode prendra en paramètre :
   — le nombre de phrases du corpus ;
   — le nombre de phrases dans lesquelles le mot apparait ;
   — le nombre de phrases dans lesquelles le mot anglais apparait ;
   — le nombre de phrases parallèles dans lesquelles la paire de mot apparait.
9. Écrivez une méthode qui calcule le likelihood ratio de l'ensemble des paires de mots du corpus. Cette méthode renverra une instance de `HashMap<String, Double>` qui associe à une paire de mots (obtenu, par exemple, en concaténant le mot français et le mot anglais) son score. Seules les paires dont le likelihood ratio n'est pas infini devront être ajoutées au résultat. Pour savoir si un `Double` est infini, on peut utiliser l'instruction

   ```
   lr.equals(Double.NaN)
   ```
10. Afficher les associations par ordre décroissant de *likelihood ratio* croissant. Que peut-on en conclure ?