Konrad POWESKA

Mathilde SANTUCCI

ET5 INFO

Projet de Traitement Automatique des Langues

Evaluation d'outils de TAL

8 mars 2020

Polytech Paris-Sud

Maison de l'ingénieur – Bât. 620 – Centre scientifique d'Orsay – 91405 Orsay – France

Tél. : +33 (0)1 69 33 86 00 – Fax : +33 (0)1 69 41 99 58 – www.polytech.u-psud.fr

Sommaire

1 - Objectif du projet

Dans le cadre du cours de Traitement Automatique des Langues, nous avons pu découvrir formellement le concept d'analyse linguistique, qui permet d'analyser des corpus de textes pour en retirer le sens général. L'analyse linguistique peut être utilisée à diverses fins, notamment la traduction de textes. Son automatisation est un enjeu majeur actuel, car si la mondialisation nous a permis de nous connecter avec le reste du monde de manière très simplifiée, elle ne permet pas encore la compréhension du reste du monde, ce que promet l'automatisation de l'analyse syntaxique de manière très rapide.

Cependant, avant de traduire des textes, il faut en extraire le sens. Pour cela, il existe diverses plateformes, dont certaines open-source, qui permettent d'analyser des textes. Nous avons travaillé sur deux de ces plateformes :

- Stanford Core NLP : boîte à outils linguistiques utilisant l'apprentissage statistique à partir de corpus annotés.
- NLTK : boîte à outils linguistiques utilisant des approches hybrides combinant l'apprentissage automatique et des ressources linguistiques.

Nous avons également vu la plateforme CEA List LIMA, une plateforme d'analyse linguistique multilingue utilisant des règles et des ressources validées par des experts linguistes, mais elle n'était malheureusement pas compatible avec nos ressources et nous n'avons donc pas pu l'évaluer pour ce projet.

Avant de passer à la partie technique de ce projet, nous devons rappeler les différentes étapes d'analyse linguistique qui permettent d'extraire le sens d'un texte.

1 - Le découpage (tokenization): découpage des chaînes de caractères du texte en mots, en prenant en compte le contexte ainsi que les règles de découpage. On utilise pour cela des règles de segmentation ainsi que des automates d'états finis.

2. Analyse morphologique (morphological analysis): vérification de l'existence du mot (token) dans la langue cible. On associe à chaque mot des propriétés syntaxiques qui vont servir dans la suite des traitements. Ces propriétés syntaxiques sont décrites en classes appelées catégories grammaticales, et on peut les récupérer dans des dictionnaires

3. Analyse morpho-syntaxique (Part-Of-Speech tagging): Après l'analyse morphologique, une partie des mots restent ambigus d'un point de vue grammatical.

L’analyse morphosyntaxique réduit le nombre des ambiguités en utilisant des règles ou des matrices de désambiguisation, qui sont souvent obtenues manuellement.

4. Analyse syntaxique (Syntactic analysis ou Parsing): identification des principaux constituants de la phrase et des relations qu’ils entretiennent entre eux. L’analyse en dépendance syntaxique consiste à créer un arbre de relations entre les mots de la phrase. Le module d’analyse syntaxique utilise des règles pour l’identification des relations de dépendance ou des corpus annotés en étiquettes morpho-syntaxiques et en relations de dépendance.

5. Reconnaissance d'entités nommées (Named Entity recognition): Ce module consiste à identifier les dates, lieux, heures, expressions numériques, produits, évènements, organisations, présentes sur un ou plusieurs tokens, et à les remplacer par un seul token.

Nous allons ici nous concentrer sur deux étapes essentielles de ce processus : l’analyse morpho-syntaxique (étape 3) et la reconnaissance d’entités nommées (étape 5). Nous évaluerons donc la performance de ces deux étapes avec les deux plateformes mentionnées précédemment.

2 - Expérimentation

2. 1. Métriques d'évaluation

L'évaluation des performances est effectuée grâce au script evaluate.py. Ce dernier fournit la précision, le rappel et la "f-mesure".

La précision correspond au nombre d'éléments qui ont été placés dans la bonne classe comparé au total des éléments placés dans cette classe.

Le rappel correspond au nombre d'éléments qui ont été placés dans la bonne classe comparé au total des éléments qui appartiennent effectivement à cette classe.

Enfin, la "f-mesure" permet de synthétiser les deux métriques précédentes et de faire une sorte de moyenne, qui indique la qualité générale du résultat.

2. 2. Données de test

2. 2. 1. Analyse morpho-syntaxique

La donnée initiale était un fichier déjà analysé grâce à la plateforme LIMA. Pour avoir une donnée de test claire, il nous a donc fallu enlever les étiquettes LIMA, pour pouvoir travailler sur un fichier texte simple, prêt à être analysé par nos différentes plateformes. Par ailleurs, il nous fallait un fichier de référence : pour cela, nous avons dû transformer les étiquettes du fichier LIMA en étiquettes universelles.

Pour effectuer ces deux tâches, nous avons créé 2 scripts python, convertLimaToStanford.py et convertStanfordToUniv.py. En effet, n'ayant pas de tableau avec une correspondance directe entre les étiquettes de LIMA et les étiquettes universelles, nous avons dû passer par la création d'un fichier avec des étiquettes Stanford. Après l'exécution de ces deux scripts, nous avons obtenu un fichier avec des étiquettes universelles, que nous avons donc considéré comme notre fichier de référence. Grâce à un troisième script, extractText.py, nous avons finalement enlevé ces étiquettes pour obtenir un texte simple.

Ce texte simple a ensuite été traité par les plateformes de Stanford et NLTK, et est donc ressorti comme un texte annoté avec des étiquettes. Le script convertFormattedToUniv.py a été utilisé pour convertir les fichier avec étiquettes de Stanford et NLTK. Il était enfin possible de comparer les différents fichiers de sortie grâce au script evaluate.py (voir les résultats dans la partie 2.3).

2. 2. 2. Reconnaissance d'entités nommées

La donnée initiale était un fichier déjà analysé avec des étiquettes CoNLL-2003. Pour avoir une donnée de test claire, il nous a donc fallu enlever les étiquettes, pour pouvoir travailler sur un fichier texte simple, prêt à être analysé par nos différentes plateformes. Par ailleurs, il nous fallait un fichier de référence : pour cela, nous avons utilisé le fichier à étiquettes CoNLL-2003. Pour extraire le texte, nous avons à nouveau utilisé le script extractText.py.

Après avoir traité le texte via les plateformes Stanford et NLTK, nous avons obtenu deux fichiers avec des étiquettes différentes. Nous avons donc utilisé les scripts convertStanfordToConll.py et convertNltkToConll.py pour convertir les étiquettes en étiquettes universelles et pouvoir ensuite comparer les résultats.

Remarque : le script format.py permet de mettre en forme les textes, pour que l'évaluation ne rencontre pas de problème. Il est utilisé dans les deux étapes, pour l'analyse de la plateforme Stanford.

2. 3. Résultats

On remarque que les résultats sont très faibles ; nous pensons que c'est dû à une mauvaise coordination entre le formatage et le script evaluate.py. Nous n'avons malheureusement pas eu le temps de le régler, mais les calculs restant justes, ils permettent toujours de déterminer la plateforme la plus perfomante.

2. 3. 1. Analyse morpho-syntaxique

Stanford

```
Word precision: 0.0096287472702
Word recall: 0.00891134588884
Tag precision: 0.0096287472702
Tag recall: 0.00891134588884
Word F-measure: 0.00925616680185
Tag F-measure: 0.00925616680185
```

NLTK

```
Word precision: 0.00982727814175
Word recall: 0.00909508497933
Tag precision: 0.00982727814175
Tag recall: 0.00909508497933
Word F-measure: 0.00944701560189
Tag F-measure: 0.00944701560189
```

Selon celui ci, les deux plateformes obtiennent un score inférieur à 1% à l'analyse morpho-syntaxique. NLTK obtient un score légèrement supérieur, mais cette différence est négligeable.

2. 3. 2. Reconnaissance d'entités nommées

Stanford

```
Word precision: 0.0143027413588
Word recall: 0.0143027413588
Tag precision: 0.0143027413588
Tag recall: 0.0143027413588
Word F-measure: 0.0143027413588
Tag F-measure: 0.0143027413588
```

NLTK

```
Word precision: 0.046483909416
Word recall: 0.046483909416
Tag precision: 0.046483909416
Tag recall: 0.046483909416
Word F-measure: 0.046483909416
Tag F-measure: 0.046483909416
```

On remarque que pour la reconnaissance d'entités nommées, la plateforme NLTK est plus performante, et la différence entre les 2 plateformes est plus grande que pour l'analyse morpho-syntaxique ci-dessus.

Nos résultats semblent cependant peu réalistes. Selon notre hypothèse, un simple désalignement de lignes entre nos fichiers de résultat et la référence (une ligne en plus ou en moins en milieu de fichier) fausserait les calculs du script d'évaluation.

Malheureusement il n'est pas facile de corriger ce défaut, car Stanford ne respecte pas les nouvelles lignes dans son traitement. Malgré le fait que notre fichier `ne_test.txt` comporte bien une ligne vide entre le le premier paragraphe et la date en dessous, la sorte de Stanford n'en contient plus (les deux paragraphes sont sur la même ligne). Cela a été le cas également pour l'analyse morpho-syntaxique: le fichier de référence contenait "Pierre Vinken" sur une ligne, tandis que Stanford comme NLTK les ont séparé en deux lignes, ce qui qui peut fausser les calculs.

4 - Conclusion

4. 1. Résumé et répartition du travail

Nous avons travaillé sur les TP 1 et 3 uniquement, car le TP 2 nécessitait des ressources que nous n'avions pas. Ce problème s'est répercuté sur le projet, puisque nous n'avons pas non plus pu évaluer la plateforme LIMA. Nous avons toutefois pu évaluer les plateformes Stanford et NLTK et répondre aux questions du projet.

Pour effectuer ce travail, nous nous sommes répartis en les tâches en fonction de la plateforme : Mathilde a travaillé sur la plateforme Stanford (vue au TP 1) et Konrad sur la plateforme NLTK (vue au TP 3). Nous avons travaillé ensemble sur les TP.

4. 2. Discussion sur les résultats

Il est un peu difficile de conclure laquelle des deux plateformes est plus performante dans l'absolu, étant donné nos résultats très inférieurs à ce que nous attendions.

Néanmoins NLTK a eu un résultat très légèrement supérieur pour l'analyse morpho-syntaxique, et un résultat considérablement supérieur pour la reconnaissance d'entités nommées. On peut donc en conclure que cette plateforme a été supérieure pour notre utilisation.

Il nous paraît difficile de donner des pistes d'amélioration sur ces plateformes, car nous ne sommes pas des experts et que nous n'avons pas eu l'occasion d'en ressentir les limitations. Selon nous, ils faudrait surtout corriger le script d'évaluation, afin qu'il prenne en compte des lignes manquantes ou en trop.