POLYTECH PARIS-SUD

Irteza SHEIKH MUHAMMAD

Maïssa KHAMIS

ET5 - INFO

Encadrant : M. Nasredine SEMMAR

Projet Traitement Automatique des Langues

08/03/2020

Github : https://github.com/Irteza-S/ProjetTAL

Introduction

Dans le cadre de notre UE Traitement automatique des langages, nous avons découvert les lois qui régissent une langue naturelle : elle est parlée et écrite par des humains et se plie à des règles de syntaxe, sémantique et grammaires grâce aux symboles qui la composent.

Ce sont des langues plus complexe à analyser de par leur possible ambigüité ou notion de paraphrase.

Le traitement automatique de ces langues est une science étroitement liée à la linguistique et à la recherche en sciences cognitives, en psychologie, en physiologie et en mathématiques. Aujourd'hui, il est pratiqué très couramment via des outils que l'on appelle plateforme d'analyse linguistique.

Ces outils possèdent les modules suivants : 1.Le découpage ou **Tokenisation**
2. L'analyse morphologique 3.L'analyse morpho-syntaxique 4 Analyse syntaxique ou **Parsing** 5. Reconnaissance d'entité nommées

Notre projet consiste en l'évaluation de trois de ces plateformes d'analyse linguistique : CEA List LIMA et Stanford Core NLP et NLTK.

Pour ce faire nous ne pouvons pas les comparer directement et nous allons donc pour chaque analyseur passer par les tags universel. Ce passage rendra notre étude plus objective.

Puis nous vous montrerons les principaux résultats obtenus avec ces trois outils, et enfin leurs points fort ainsi que leurs limitations.

Sommaire

I - Présentation des plateformes

1. CEA List LIMA

Créé par le List, institut de CEA Tech en France, c'est un outil d'analyse linguistique multilingues avec dix langues !

Il contient des modules et ressources pour l'analyse de textes développé par CEA List et utilise des règles formelles ainsi que des ressources mises à disposition par des professionnels du domaine linguistique. A partir d'enquêtes d'opinions, rapport techniques ou autre texte, en sachant que LIMA ajoute aux mots-clefs une relation au temps pour plus de pertinence.

2. Stanford Core NLP

Le NLP ou Natural Language Processing de l'université de Stanford est composé d'une série d'algorithmes pour comprendre les langues naturelles, qui se base contrairement à Lima sur l'apprentissage dit statistique à partir de corpus annotés.

Ils utilisent donc un mélange d'analyse de données avec du machine learning, deep learning et une approche probabiliste ce qui le rendrait plus robuste. Il peut gérer au total cinqs langues différentes et utilise Java.

3. NLTK

La Natural Language Toolkit est l'outil le plus ancien des trois, créé par Steven Bird et Edward Loper de l'Université de Pennsylvanie elle permet de traiter des données de langages humain avec Python .

Elle est à mi chemin entre l'approche de LIMA et de Stanford car elle utilise des approches hybrides combinant l'apprentissage automatique et des ressources linguistiques.

4. Choix pour l'étude

Ces trois plateformes sont open sources, et nous avons choisis pour notre étude de comparer LIMA, Stanford et NLTK pour l'analyse morpho-syntaxique et uniquement Lima et Stanford pour la reconnaissance d'entités nommées. Nous n'avons pas pu faire d'analyse de reconnaissance d'entités nommées avec NLTK car nous avons eu des problèmes de librairies nous empêchant d'exécuter nos scripts.

II - Evaluation de l'analyse morpho-syntaxique

L'analyse morpho-syntaxique sert à comparer la précision avec laquelle certains outils analysent un texte.

Une fois avoir exécuté les scripts nécessaires à obtenir les fichiers de résultats Lima, NLTK et Stanford dans le bon format (étiquettes universelles + 2 colonnes séparées par une tabulation + pas de lignes vides dans les fichiers comparés, cf ReadMe du GitHub), on peut passer à l'évaluation à l'aide du script evaluate.py.
On éxecute le script evaluate.py sur chacun des 3 fichiers de résultats.

On obtient les résultats suivants:

Outil	Word precision	Word recall	Tag precision	Word F-measure	Tag F-measure
Lima	0.00952	0.00891	0.00952	0.00921	0.00921
Stanford	0.01161	0.01074	0.01161	0.01116	0.01116
NLTK	0.01101	0.01019	0.01101	0.01059	0.01059

Tableau de comparaison résultats bruts Lima/Stanford/NLTK

L'outil Stanford semble être un peu plus performant que NLTK. L'outils Lima quand à lui reste moins performant que NLTK et Stanford. Cependant les valeurs diffèrent trop légèrement et sont trop faibles. Nous ne pouvons pas tirer plus de conclusions.

Pour aller plus loin dans la comparaison, nous allons modifier les fichiers de résultats Lima, Stanford et NLTK et aussi modifier le fichier de référence afin qu'il y ai les mêmes

mots sur la colonne de gauche (cf ReadMe du GitHub). Chaque outil aura donc son propre fichier de référence auquel on ne comparera uniquement les étiquettes. Cette étude complémentaire sert uniquement à évaluer les performances des différents outils à attribuer une étiquette à un mot.

Nos résultats sont les suivants :

Outil	Word precision	Word recall	Tag precision	Word F-measure	Tag F-measure
Lima	0.93545	0.93545	0.93545	0.93545	0.93545
Stanford	0.93657	0.93657	0.93657	0.93657	0.93657
NLTK	0.92769	0.92769	0.92769	0.92769	0.92769

Tableau de comparaison résultats formatés Lima/Stanford/NLTK

Ici les résultats de Stanford semblent être très légèrement meilleur que ceux de Lima et NLTK mais encore une fois nous ne remarquons pas de différence notable entre les différents outils. On peut expliquer cela part :

- le fait que le script evaluate.py compare ligne par ligne les deux fichiers passés en paramètres. Les outils évalués n'ont pas le même mécanisme de *tokenisation*, on se retrouve donc avec des fichiers de résultats qui n'ont pas le même nombres de lignes suivant l'outils. Certains phrases, voir mots (notamment les noms propres) ne sont pas découpées de la même manière
- même si nous avons essayé de contourner cette dernière contrainte, nous remarquons à l'oeil au vu des fichiers que les différences majeurs entre les outils apparaissent au niveau des noms propres, des dates, des chiffres… et malheureusement ces mêmes différences sont mal évaluées

Finalement même si nous ne pouvons pas tirer de réelle conclusion à l'heure actuelle, l'évaluation des différents outils à l'aide du script evaluate.py reste juste car on compare le résultat brut de chacun des outils à un résultat de référence. On évalue donc ici la capacité d'un outil à découper et à analyser le plus fidèlement possible le texte.

III - Evaluation de la reconnaissance d'entités nommées

Pour cette comparaison, nous avons utilisé le texte *pos-test.txt*, il est intéressant pour notre étude car il comporte de nombreux nom d'organisations, de localisations et de personnes. Nous évaluerons donc la précision des résultats de Lima et Stanford. Nous n'avons pas pu évaluer NLTK pour les entités nommées car nous avons rencontrés des problèmes dans les librairies utilisées par NLTK.

Avant de commencer l'évaluation, nous avons noté une petite erreur dans le fichier servant de référence : ne_reference.txt.conll ; en effet on peut retrouver dans ce fichier les étiquettes "*B-PER" et "I-PER"*, or les étiquettes CoNLL-2003 sont *"B-PERS" et "I-PERS"*. Pour ne pas modifier le fichier de référence, nous avons donc considéré les étiquettes "B-PER" et "I-PER" comme juste et de ce fait les fichiers de résultats des différents outils portent eux aussi les mêmes étiquettes *"B-PER" et "I-PER"*.

Voici les résultats de notre évaluation (post traitement ie ReadMe GitHub) :

Outil	Word precision	Word recall	Tag precision	Word F-measure	Tag F-measure
Lima	0.02068	0.02095	0.02068	0.02095	0.02081
Stanford	0.00923	0.00923	0.00923	0.00923	0.00923

Même si les valeurs obtenues restent faible, on remarque cette fois que les résultats de Lima de plus de deux fois plus précis que les résultats de Stanford.

Nous avons tout de même quelque remarques :

- le fichier de référence ne contient pas d'étiquettes "B-MISC" ou "I-MISC"
- Lima identifie les nombres, les dates (2008) et les caractères spéciaux ($) comme des *NUMBER*, *DATE* et *NUMEX*. Ces mêmes types ne sont pas représentés dans les étiquettes CoNLL-2003. Stanford lui n'identifie pas ces catégories.

IV- Points forts, limitations et difficultées rencontrées

En choisissant de comparer ces deux plateformes, il est intéressant de voir la différence entre un outils qui utilise des règles formelles et l'autre qui travail avec le machine learning et l'apprentissage en général .

Tout d'abord, la comparaison de deux outils aussi complexe n'est pas forcément optimale en ne traitant qu'un seul texte ou une seule langue comme nous l'avons fait. Le choix d'un registre de données différent peut influencer notre étude.

Les points fort de LIMA sont le nombre élevé de langues qu'elle arrive à traiter: anglais, français, allemand, espagnol, italien, russe, arabe, chinois et hongrois. Ce sont des langues dont la racine et l'alphabet ne se ressemblent parfois en rien.

LIMA se basant sur des règles faites manuellement par des êtres humains, cela permet une exactitudes des règle que l'on utilise, leurs modifications ainsi qu'un gain de temps par rapport à une méthode où l'on doit effectuer des périodes de tests training etc.

Cependant le fait que cette tâche ne soit pas automatisé la rend vulnérable, par exemple des experts en langues peuvent parfois se contredire, des règles peuvent devenir au fil du temps caduc ou même se superposer et se contredire.

Les points fort de Stanford NLP sont justement qu'en utilisant des techniques automatique d'apprentissage via des statistiques elle s'ajuste au fur et à mesure du contenu qu'on lui fait parvenir. Cette technique fait en sorte qu'elle reste forcément ancré dans la réalité de ce qui se fait et s'utilise à un instant T.

Pour NLTK, nous n'avons malheureusement pas pu en savoir davantage sur les limites et capacité de cet outil car nous avions eu des soucis de librairies nous empêchant d'exploiter NLTK.

Pour évaluer plus précisément ces plateformes nous pourrions tout d'abords modifer le script evaluate.py pour qu'il ne compare plus 2 à 2 les mots. La tokenisation semble poser problème dans notre évaluation mais nous ne savons pas comment y remédier.

V - Organisation

Irteza s'est occupé de l'intégralité des Tps (sauf 1 script), du Github du projet, ainsi que l'intégralité du code du projet. Maïssa s'est occupée du rapport avec Irteza.

VI - Annexes

```
irteza@ubuntu:~/Desktop/ProjetTAL/data/part1$ python2.7 ../../src/evaluate.py po
s_test.txt.pos.lima.univ.formated pos_reference.txt.univ.formated
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.00952942332253
Word recall: 0.00891134588884
Tag precision: 0.00952942332253
Tag recall: 0.00891134588884
Word F-measure: 0.00921002658564
Tag F-measure: 0.00921002658564
irteza@ubuntu:~/Desktop/ProjetTAL/data/part1$ python2.7 ../../src/evaluate.py po
s_test.txt.pos.stanford.univ.formated pos_reference.txt.univ.formated
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.0116140559857
Word recall: 0.0107487367938
Tag precision: 0.0116140559857
Tag recall: 0.0107487367938
Word F-measure: 0.0111646548022
Tag F-measure: 0.0111646548022
irteza@ubuntu:~/Desktop/ProjetTAL/data/part1$ python2.7 ../../src/evaluate.py po
s_test.txt.pos.nltk.univ.formated pos_reference.txt.univ.formated
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.0110184633711
Word recall: 0.0101975195223
Tag precision: 0.0110184633711
Tag recall: 0.0101975195223
Word F-measure: 0.0105921084021
Tag F-measure: 0.0105921084021
```

Figure 1 : résultats bruts analyse morphosyntaxique

```
irteza@ubuntu:~/Desktop/ProjetTAL/data/part1$ python2.7 ../../src/evaluate.py tm
pLima.txt tmpRefLima.txt
Word precision: 0.935457006038
Word recall: 0.935457006038
Tag precision: 0.935457006038
Tag recall: 0.935457006038
Word F-measure: 0.935457006038
Tag F-measure: 0.935457006038
irteza@ubuntu:~/Desktop/ProjetTAL/data/part1$ python2.7 ../../src/evaluate.py tm
pStanford.txt tmpRefStanford.txt
Word precision: 0.936574824419
Word recall: 0.936574824419
Tag precision: 0.936574824419
Tag recall: 0.936574824419
Word F-measure: 0.936574824419
Tag F-measure: 0.936574824419
irteza@ubuntu:~/Desktop/ProjetTAL/data/part1$ python2.7 ../../src/evaluate.py tm
pNltk.txt tmpRefNltk.txt
Word precision: 0.927691309987
Word recall: 0.927691309987
Tag precision: 0.927691309987
Tag recall: 0.927691309987
Word F-measure: 0.927691309987
Tag F-measure: 0.927691309987
```

Figure 2 : résultats sur fichiers modifiés analyse morphosyntaxique

```
irteza@ubuntu:~/Desktop/1$ python2.7 evaluate.py ne_test.txt.ne.lima.conll ne_re
ference.txt.conll.formated
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.0206835418103
Word recall: 0.0209518108351
Tag precision: 0.0206835418103
Tag recall: 0.0209518108351
Word F-measure: 0.0208168120539
Tag F-measure: 0.0208168120539
irteza@ubuntu:~/Desktop/1$ python2.7 evaluate.py ne_test.txt.ne.stanford.conll n
e_reference.txt.conll.formated
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.00923718712753
Word recall: 0.00923718712753
Tag precision: 0.00923718712753
Tag recall: 0.00923718712753
Word F-measure: 0.00923718712753
Tag F-measure: 0.00923718712753
```

Figure 3 : résultats bruts analyse entités nommées