Rapport de Projet - TAL

1. Introduction

Le but de ce projet est l'écriture d'un rapport sur deux outils de traitement automatique des langues: **NLTK** et Stanford **CoreNLP**. Nous présenterons brièvement leur fonctionnement et nous comparerons leurs performances sur les tâches de POS tagging et de reconnaissance d'entité nommés.

2. Présentation des plateformes d'analyse linguistique

2.1. Stanford CoreNLP :

- La plateforme Stanford ***CoreNLP*** a une framework d'annotation codée en java qui contient les principales composantes nécessaires à toutes les étapes du traitement automatique des langues confer **Figure_1**. C'est l'un des outils de traitement automatique de la langue les plus utilisés. Dans ce paragraphe, nous essayerons de mettre en lumière son fonctionnement et ses points forts.

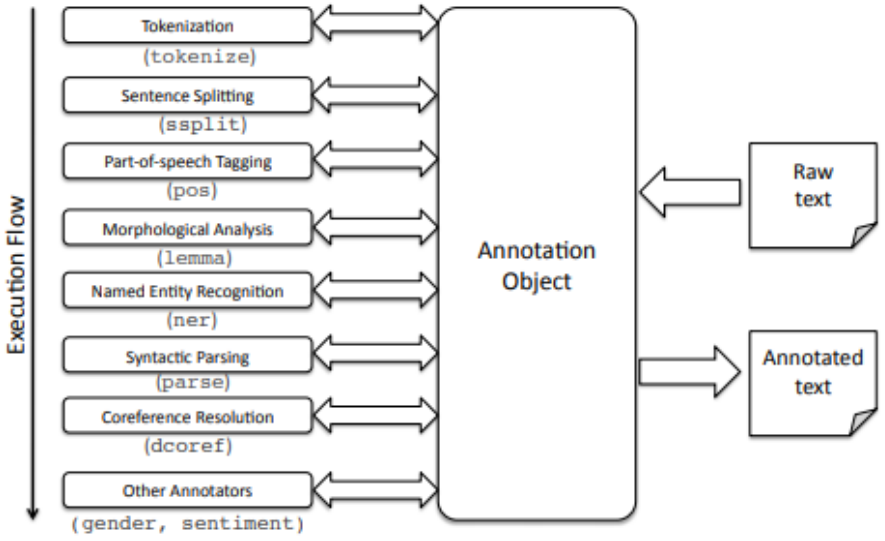


Figure_1 : Pipeline de traitement de langue de l'outils Stanford CoreNLP

- En effet, lorsqu'un texte brut est soumis à l'outil Stanford, il est mis dans un objet de type Annotation. Ensuite cet objet est soumis à une séquence de divers traitements réalisés par les annotateurs qui rajouterons les résultats de leur traitement à l'objet Annotation contenant le texte brut. Ainsi, toutes les étapes stockent des résultats d'analyses qui pourraient très bien servir à la réalisation d'autres traitements.

- Le but pour Stanford était de développer un processus de traitement de langue uniformisé qui permet d'accéder de manière efficace à diverses informations qu'on peut obtenir à travers l'analyse d'un texte. Ainsi, on a une interface qui regroupe un ensemble de traitements, permettant d'avoir en un bloc un ensemble diverse d'informations concernant le texte traité, afin de pouvoir exploiter ces informations simultanément pour affiner et faciliter le traitement automatique du texte. Tout cela réuni dans un framework portable, basé sur java, un langage objet permettant d'avoir de la finesse et de la simplicité au niveau du code.

- C'est un outil qui permet donc une prise en charge facile pour l'utilisateur, qui n'a besoin que d'avoir son texte brut et de faire appel à l'annotateur qu'il souhaite appliquer pour tout de suite avoir le résultat dans un fichier. Ainsi, il peut lui-même sélectionner les étapes du pipeline, et les annotateurs auxquels il souhaite faire appel pour faire son propre pipeline de traitement automatique de la langue. De plus, s'il le souhaite, il a la possibilité d'adapter un peu plus son pipeline à ses besoins en ajoutant un ou plusieurs annotateurs a l'outil Stanford CoreNLP.

- Cette dernière capacité s'avère particulièrement utile car Stanford CoreNLP n'est pas en mesure de traiter toutes les langues, mais parce qu'il est plutôt ouvert, n'importe quelle langue pourrait faire ses annotateurs en fonction de ses besoins, tout en bénéficiant des annotateurs déjà disponibles, confer **Figure_2**.

Annotator	Ara-bic	Chi-nese	Eng-lish	Fre-nch	Ger-man
Tokenize	✓	✓	✓	✓	✓
Sent. split	✓	✓	✓	✓	✓
Truecase			✓		
POS	✓	✓	✓	✓	✓
Lemma			✓		
Gender			✓		
NER		✓	✓		✓
RegexNER	✓	✓	✓	✓	✓
Parse	✓	✓	✓	✓	✓
Dep. Parse		✓	✓		
Sentiment			✓		
Coref.			✓		

Figure_2 : les différents langages humains supportés par les différents annotateurs en 2014

Les approches utilisées pour réaliser les différents annotateurs des divers composant analytiques de l'outils Stanford CoreNLP :

- Des corpus annotés associés permettant de mettre des techniques de Machine Learning supervisé

- Des dictionnaires de règles, nécessitant aussi de grosses ressources du langage humain considéré.

2.2. NLTK :

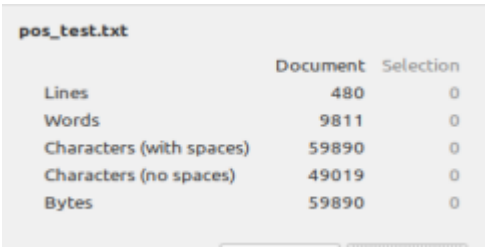
- NLTK permet de créer des scripts pour l'analyse de textes. C'est une des plateformes les plus populaires dans ce domaine car elle est très efficace et bien documentée, ce qui la rend plus facile à utiliser que d'autres. Il met à disposition des interfaces intuitives, des corpus, et des modules de tokenization, de tagging, de reconnaissance d'entités nommées, et de bibliothèques de traitement de texte. Il utilise l'apprentissage automatique avec une approche utilisant notamment la classification naïve bayésienne, les arbres de décision, l'entropie maximale, et autres…

- Il peut également faire des interprétations sémantiques, des mesures d'évaluation, des probabilités et des estimations. NLTK a pour but d'être simple d'utilisation, en proposant des fonctions modulaires et intuitives, permettant à l'utilisateur de faire évoluer son raisonnement en développant ses scripts de traitement et d'analyse au fur et à mesure de l'avancement de son projet.

3. Evaluation de l'analyse morpho-syntaxique

- Décrire le corpus d'évaluation (nombre de phrases, nombre de mots)

Nous avons utilisé le fichier pos_test.txt:



3.1. Stanford Core NLP : résultats

```
yawo@yawo-VirtualBox:~/Documents/TAL/[TAL] Projet$ python evaluate.py pos_test.txt.pos.stanford.univ pos_ref
erence.txt.univlinear
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.190454504053
Word recall: 0.208995502249
Tag precision: 0.170598415156
Tag recall: 0.187206396802
Word F-measure: 0.199294700724
Tag F-measure: 0.178516965307
```

Ces résultats ont été trouvés à partir du format d'affichage de stanford; les étiquettes et les mots sont séparés “_” et on ne retourne à la ligne qu’à la fin d’une phrase. Nous nous sommes rendus que le programme evaluate.py marche mieux ainsi.

3.2. NLTK : résultats

```
Word precision: 0.009728012705975779
Word recall: 0.0090032154340836
Tag precision: 0.009728012705975779
Tag recall: 0.0090032154340836
Word F-measure: 0.009351591201870317
Tag F-measure: 0.009351591201870317
```

Pour les résultats ci-dessous, on observe une baisse significative des résultats, qui est liée à un problème de formatage des données au cours des conversions entre les étiquettes par exemple.

4. Evaluation de la reconnaissance d’entités nommées

Pour la reconnaissance d’entités nommées, le corpus utilisé, présent dans le fichier ne_test.txt est composé de 8800 mots.

En terme de métriques :

La précision est la proportion des items pertinents parmi l'ensemble des items proposés

Le rappel est la proportion des items pertinents proposés parmi l'ensemble des items pertinents. Il est inférieur à la précision et permet de connaître l’exhaustivité des résultats.

La F-measure combine la précision et le rappel comme suit :

F = 2*((précision*rappel) / (précision+rappel))

5. Points forts, limitations et difficultés rencontrées

Des difficultés à mettre en forme les documents de manière homogène ont entraîné une baisse significative dans la partie nltk.

6. Organisation

Qui a fait quoi ?

- TP1: Yawo Woassili Adufu
- TP2: Martin Boujon

- Github du projet: Yawo Woassili Adufu
- Code du projet: Yawo Woassili Adufu, Martin Boujon, Fares Ghodhbani
- Rédaction du rapport: Yawo Woassili Adufu, Martin Boujon, Fares Ghodhbani