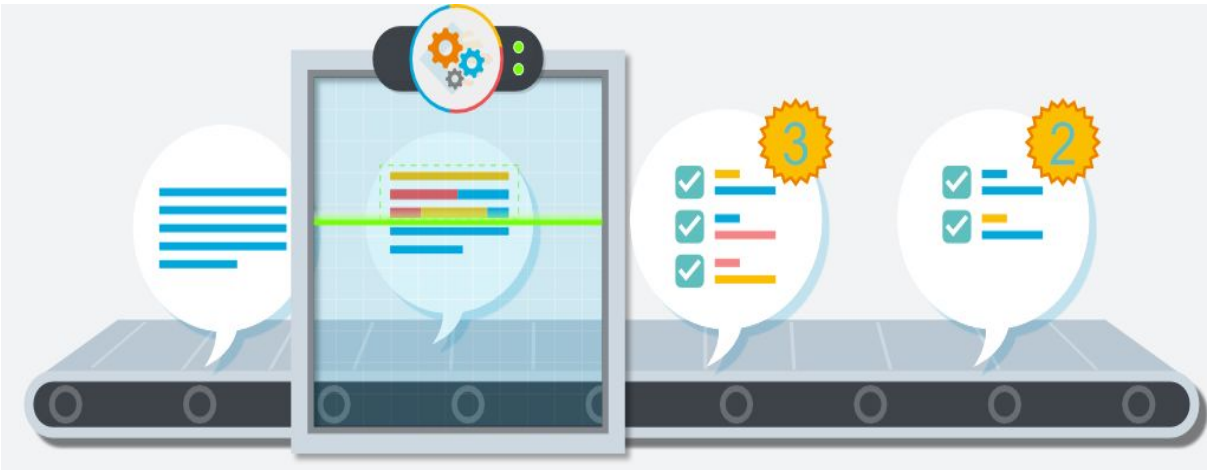
Rapport du projet TAL

Théo Legras

Jordane Minet

Alexis Proust

ET5 INFO 2019-2020



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

Introduction : Position du problème

Dans ce rapport, nous analyserons des outils de traitements automatisés du langage sous la forme d'une évaluation entre plusieurs plateformes open source d'analyse linguistique.
Les outils ont été découvert en TPs puis approfondi lors de ce projet. Les outils en question sont CEA List LIMA, Stanford Core NLP et NLTK.
Ces plateformes d'analyse classique nous permettent de découper en token un texte. Ensuite les tokens vont être analysé morphologiquement pour savoir si le token appartient à la langue puis se voit attribué des propriétés syntaxiques. Une analyse morpho-syntaxique est parfois nécessaire pour lever les ambiguïtés grâce à des règles. Ensuite, une analyse syntaxique permet d'identifier le sens des mots et leur attribuer une étiquette. Enfin, la reconnaissance d'entités nommés identifie certains tokens comme des lieux, heures, etc.

Dans ce projet, nous devions les utiliser pour voir ce dont ils sont capables et les comparer afin de percevoir les forces et les limites de chaque plateforme. Nous devons analyser les résultats obtenus, puis trouver d'éventuelles pistes d'amélioration pour améliorer la précision des résultats. Notre principales tâche à été de rendre les résultats exploitables (les outils étant déjà complet et fournissant tous les scripts nécessaires à leur utilisation).

Ce projet a été réalisé par Théo Legras, Jordane Minet, et Alexis Proust, étudiant en 5ème année à Polytech Paris-Saclay.

Description des plateformes:

A. Lima

LIMA est une boîte à outils de traitement du langage naturel. Elle sert d'analyseur linguistique de texte. Lima a été développé par le CEA LIST, laboratoire LASTI. LIMA est disponible gratuitement pour une grande partie des fonctionnalités, mais dispose également d'une version payante qui permet d'analyser d'autres langues comme l'arabe, le chinois, l'allemand, etc. Son fonctionnement se base sur l'utilisation de règles et des ressources validées par des experts linguistes. Grâce à son renommé et son réseau, le CEA dispose de nombreuses ressources et de nombreux experts dans de multiple domaine qui ont permis la création de LIMA.
LIMA supporte l'ensemble des étapes de l'analyse linguistique qu'un analyseur classique (morphologie, syntaxe, sémantique, entités, coréférences) selon une architecture modulaire et hautement configurable, avec des performances élevées aussi bien en vitesse qu'en qualité d'analyse.

Une de ses particularités est le cross-linguisme, elle permet de travailler gratuitement sur des textes anglais, français ou même portugais. Outre le cross-linguisme d'une dizaine de langues, LIMA présente de nombreux atouts. Il permet de traitement des documents multimédias comme des images, des vidéos ou des paroles, et pas seulement du texte. Il offre aux utilisateurs le choix de la technologie, pour travailler sur différentes tailles de données et assurer le passage à l'échelle (algorithmie et architecture, ou big analytics). Il est capable de prendre en compte les spécificités d'un domaine ce qu'il lui permet de dépasser les performances en précision des outils génériques du marché.

LIMA est un analyseur basé sur des règles. L'avantage est que ces règles (étant fixées par des spécialistes) sont beaucoup plus précises qu'un modèle statistique obtenu via du machin learning. On est sûr de leur exactitude, on s'attend donc à obtenir des résultats supérieurs (plus précis) par rapport aux modèles utilisant du machin learning.
Les désavantages sont eux aussi liés au côté humain : on a besoin d'opérateur humain pour fixer les règles, et si l'on veut que ce soit précis on fera appel à des spécialistes. Cette démarche doit être effectuée pour chaque langue pour laquelle on veut pouvoir utiliser l'outil. C'est un temps de développement extrêmement long pour des résultats qui, en fonction du context, peuvent se révéler inutilement précis. De plus si la langue évolue il faudra modifier ces règles directement dans le code, ce qui peut s'avérer fastidieux et long encore une fois.

Il existe un site pour tester en ligne LIMA : http://www.kalisteo.fr/demo/lima/

B. Stanford CoreNLP

Un groupe de recherche de l'université de Stanford partage de nombreux outils gratuitement depuis plusieurs années, comme le Stanford CoreNLP. Il s'agit d'une librairie développé en java.

Stanford CoreNLP met à disposition un ensemble d'outils d'analyse grammaticale. Il est capable de reconnaître les mots de base de plusieurs langues, que ce soit des noms d'entreprise, de personnes, etc. Il sait également normaliser les dates, les heures, etc. Ou bien extraire les relations syntaxiques entre les mots. En somme, tout ce qu'on attend d'une plateforme d'analyse linguistique.

Stanford dispose également de beaucoup d'atouts. Une api est disponible pour la majorité des langages de programmation modernes ce qui le rend facilement intégrable à d'autres logiciels, et est plus accessible à une majorité de personne. Stanford prend en charge un certain nombre de langues humaines principales, par défaut l'anglais, mais le moteur est compatible avec les modèles d'autres langues, comme l'arabe, le chinois, le français, etc. Ils proposent même des modèles optimisés pour travailler avec un anglais sans casse par exemple.

Stanford est un outils entièrement basé sur du machin learning, au contraire de Lima on va pouvoir obtenir "rapidement" un modèle qui se montrera plus ou moin satisfaisant. Par rapport à un outils basé sur des règles on a besoin ici de faire de l'apprentissage. On a donc besoin d'un corpus d'apprentissage annoté. La qualité de ce corpus annoté (sa taille, la précision des annotations, sa cohérence avec les cas qui seront rencontré par l'outil dans le futur) influe directement sur la précision du modèle de ML qui sera obtenu. C'est un travail fastidieux mais nécessaire. Néanmoins, comme on a pas besoin d'une précision aussi forte que pour un modèle à base de règles on peut confier ce travail à n'importe qui (puis faire une validation après). Si jamais on veut coller aux évolutions de la langue il suffit de modifier le corpus et de relancer l'apprentissage du modèle. Cela prendra moins de temps et sera moins compliqué que de recoder les règles à la main.
On gagne en rapidité et facilité d'adaptation de l'outils à de nouvelles langues, mais on perds en précision.

On peut trouver en ligne une version démo : https://corenlp.run/

C.NLTK

NLTK est une boite à outil permettant la création de programmes pour l'analyse de texte. Elle a été créé en 2001 dans une université de Pennsylvanie. NLTK est puissant et documenté ce qui a fait de lui une des plateformes les plus connues et les plus utilisées. Elle est développée en python et reste accessible à tous, développeur expérimenté ou débutant.

NLTK permet de faire beaucoup de chose. Il est capable de tout ce qu'un analyseur classique sait faire. Il fournit des interfaces intuitives avec un des corpus et des ressources lexicales, de nombreuses bibliothèques de traitement de texte, le balisage, la tokenisation, l'analyse,etc. Il se base sur de l'apprentissage automatique avec les arbres de décision, l'entropie maximale, Bayes naïfs, etc. Il peut également faire des interprétations sémantiques, des mesures d'évaluation, des probabilités et des estimations.

NLTK offre de nombreux avantages et a été conçu avec des objectifs clairs. Il y en a 4 :

- Simplicité: il faut fournir un cadre intuitif, donner une connaissance pratique de la PNL de manière simple et cacher la partie compliqué.
- Cohérence: Il faut fournir un cadre uniforme avec les interfaces et les structures de données cohérentes avec des noms de méthodes intuitives.
- Extensibilité: Il faut fournir un cadre modulaire pour pouvoir ajouter des fonctionnalités et des modules facilement.
- Modularité: Il faut fournir des composants qui peuvent être utilisés indépendamment du reste de la boîte à outils.

L'approche hybride, selon comment elle est utilisée permet de combiner le meilleur des deux mondes (approche par règles ou approche statistique), ou le pire. L'idée de l'approche hybride est d'utiliser des règles pour les parties jugées critiques (ce qui permet de gagner en précision), et utiliser l'approche statistique aux endroits où l'on veut pouvoir être plus flexible et qui nécessitent moins de précision (on en profite donc pour gagner du temps de développement).

Expérimentation :

A. Données de test

Les données utilisées pour les tests sont ceux fournis par le professeur, c'est-à-dire tout d'abord le sample composé de quelques phrases, puis le fichier test contenant un texte de plus de 10.000 lignes.

Une des premières choses que nous avons fait est de mettre en forme les données pour pouvoir les utiliser. Donc nous avons transformé le texte avec les outils à disposition. Les mots sont pris un par un, ou par groupe, puis associé à des étiquettes. Afin de pouvoir comparer les résultats des différentes plateformes, il faut normaliser selon CoNLL. Donc nous convertissons tous les LOCATION en B-LOC, suivi des I-LOC, pareil pour les autres tags. Une fois cela fait, nous les évaluons avec le script fourni par le professeur.

B. Métrique d'évaluation (précision, rappel, F mesure)

VP = vrai positif
FP = faux positif
VN = vrai négatif
VP = vrai positif

Pour évaluer nos plateformes, nous avons utilisé 3 facteurs.
Tout d'abord la précision. Elle permet de connaître la proportion d'identifications positives qui été effectivement correctes. elle se calcule avec la formule suivante:

$$\text{Précision} = \frac{VP}{VP + FP}$$

Le rappel est la proportion de résultats positifs réels a été identifiée correctement. On la calcule comme cela :

$$\text{Rappel} = \frac{VP}{VP + FN}$$

Enfin, la F mesure, représente une combinaison de la précision et du rappel. Elle se calcule comme cela :

$$F = 2 \cdot \frac{(\text{précision} \cdot \text{rappel})}{(\text{précision} + \text{rappel})}$$

C. Résultats

1. Postagging:

Lima :

```
ru@DESKTOP-IU5MC05:/mnt/e/Travail/ETS/TAL/projet/Polytech_S9_TAL_Projet/src$ python python/evaluate.py ../data/Exo1/
os_test.txt.pos.lima.univ.test ../data/Exo1/pos_reference.txt.univ
arning: the reference and the candidate consists of different number of lines!
ord precision: 0.905733186329
ord recall: 0.905733186329
ag precision: 0.905733186329
ag recall: 0.905733186329
ord F-measure: 0.905733186329
ag F-measure: 0.905733186329
```

Stanford :

```
STANFORD :
Warning: the reference and the
Word precision: 0.935152487961
Word recall: 0.935152487961
Tag precision: 0.935152487961
Tag recall: 0.935152487961
Word F-measure: 0.935152487961
Tag F-measure: 0.935152487961
```

NLTK :

```
semmar@semmar:~/Downloads/TP_1/pas utile$ python2.7 evaluate.py pos_reference.tx
t.univ pos_test.txt.pos.nltk.univ
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.00165365181442
Word recall: 0.00178677784395
Tag precision: 0.00165365181442
Tag recall: 0.00178677784395
Word F-measure: 0.00171763920034
Tag F-measure: 0.00171763920034
```

2. Named Entities:

Lima :

```
gru@DESKTOP-IU5MC05:/mnt/e/Travail/ETS/TAL/projet/Polytech_S9_TAL_Projet/src$ python python/evaluate.py ../data/Exo2/
ne_test.txt.ne.lima.conll ../data/Exo2/ne_reference.txt.conll
Warning: the reference and the candidate consists of different number of lines!
Word precision: 0.0128142423946
Word recall: 0.0130115216528
Tag precision: 0.0128142423946
Tag recall: 0.0130115216528
Word F-measure: 0.0129121285299
Tag F-measure: 0.0129121285299
```

Stanford :

```
Warning: the reference and the
Word precision: 0.0118196265395
Word recall: 0.0118196265395
Tag precision: 0.0118196265395
Tag recall: 0.0118196265395
Word F-measure: 0.0118196265395
Tag F-measure: 0.0118196265395
```

NLTK :

D. Analyse et discussion des résultats

Nous avons eu beaucoup de problèmes avec les fichiers fournis. Après quelques manipulations, nous arrivons à obtenir de bons résultats avec lima pour la partie sur le postaging. De même avec Stanford. Cependant, avec NLTK, nous avons de mauvais résultats parce que nous n'avons pas touché au fichier après la génération. Et comme les tags ont été fait avec des outils différents, ils ne font pas les mêmes découpages. Par exemple pour la première phrase, le nom Pierre Vinken est découpé en deux mots, alors que dans le fichier de référence fourni, il est identifié comme une entitée ce qui fait que lors de l'évaluation, cela ne correspond pas. De plus, les décalages ne sont pas pris en compte dans le script d'évaluation ce qui faire encore plus d'erreur et provoque les mauvais résultats. Pour LIMA et Stanford nous avons fait des scripts de normalisation pour parser les fichiers et les remettre sous le bon format. Néanmoins même le formatage que nous avons fait n'est pas parfait (on voit que le nombre de lignes n'est pas le même quand on fait la comparaison avec la référence.

Malheureusement l'absence de normalisation utilisable pour tous les fichiers ne nous permet pas d'évaluer la véritable précision des modèles que nous utilisons. Il nous est impossible de savoir si l'écart entre les résultats de deux modèles est due à la performance des modèles ou à une sortie trop différente de ce qui est attendu par le fichier de référence.

Conclusion:

A. Résumé du travail effectué

Sur ce projet, nous avons entièrement fait stanford ainsi que LIMA sur le pos_tagging et la reconnaissance d'entité nommée. Pour NLTK, nous avons seulement terminé le pos_tagging puisque la reconnaissance d'entité nommée ne fonctionne pas à cause du format de sortie en arbre de NLTK.

B. Limitation des plateformes

Les plateformes sont trop limités par le format des données à fournir. De plus les analyses n'ont pas vraiment de valeur puisque la comparaison est bête et méchante, s'il y a un décalage parce que certaines plateformes regroupent plusieurs entités, on a une évaluation proche de 0%.
Par ailleurs on peut trouver des problèmes même au niveau des sorties des outils. Par exemple LIMA sort la phrase suivante "Automobile Dealers' Association" là où dans le fichier de référence on a "Automobile Dealers Association". Cette légère modification détruit la précision des mots et rend difficile le réalignement des tags.

C. Piste d'amélioration

Il faudrait que la séparation des entitées dans le fichier de référence soit la même que la séparation faite par les outils utilisés (ou au moins que des règles puissent lier les deux). Nous aurions pu améliorer nos résultats et obtenir des chiffres plus exploitables en finalisant les fichiers de normalisation.

Sources:

LIMA :

http://listserv.linguistlist.org/pipermail/ln/2014-February/009451.html
https://www.researchgate.net/publication/269408618_Traitement_Automatique_des_Langues_Biens_Communs_Informationnels_et_Industries_de_la_Langue
https://github.com/aymara/lima/wiki
https://github.com/aymara/lima/wiki#the-lima-multilingual-nlp-tool
https://hal-cea.archives-ouvertes.fr/cea-01844458?gathStatIcon=true

Stanford Core NLP:

https://stanfordnlp.github.io/CoreNLP/#about
https://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf
http://www.erwanlenagard.com/general/tutoriel-implementer-stanford-corenlp-avec-talend-1354
https://www.supinfo.com/articles/single/4726-stanford-natural-language-processing-nlp

NLTK:

https://code.tutsplus.com/fr/tutorials/introducing-the-natural-language-toolkit-nltk--cms-28620
https://www.nltk.org/