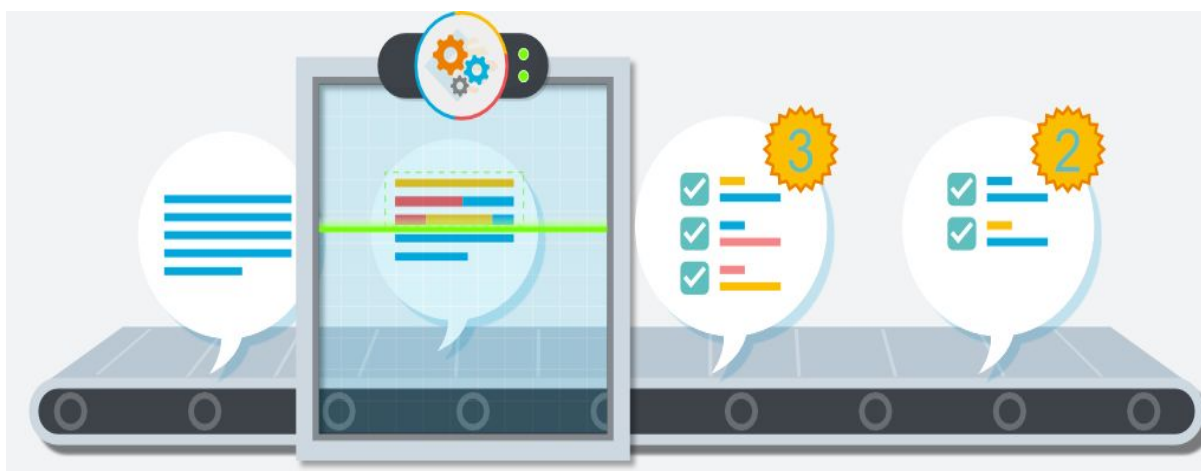


# Rapport du projet TAL

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](http://www.polytech.u-psud.fr)

## Introduction : Position du probl me

Dans ce rapport, nous analyserons des outils de traitements automatis s du langage sous la forme d'une  valuation entre plusieurs plateformes open source d'analyse linguistique.

Les outils ont  t  d couverts en TPs puis approfondis 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 d couper en token un texte. Ensuite les tokens vont  tre analys s morphologiquement pour savoir si le token appartient   la langue puis se voit attribu  des propri t s syntaxiques. Une analyse morpho-syntaxique est parfois n cessaire pour lever les ambigu t s gr ce   des r gles. Ensuite, une analyse syntaxique permet d'identifier le sens des mots et leur attribuer une  tiquette. Enfin, la reconnaissance d'entit s nomm es identifie certains tokens comme des lieux, heures, etc.

Dans ce projet, nous devons les utiliser pour voir ce dont ils sont capable et les comparer afin de percevoir les forces et les limites de chaque plateforme. Nous devons analyser les r sultats obtenus, puis trouver d' ventuelles pistes d'am lioration pour am liorer la pr cision des r sultats. Notre principale t che a  t  de rendre les r sultats exploitables (les outils  tant d j  complets et fournissant tous les scripts n cessaires   leur utilisation).

Ce projet a  t  r alis  par Th o Legras, Jordane Minet, et Alexis Proust,  tudiant en 5 me ann e   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 sa renomm   et son r  seau, le CEA dispose de nombreuses ressources et de nombreux experts dans de multiples domaines qui ont permis la cr  ation de LIMA. LIMA supporte l'ensemble des   tapes de l'analyse linguistique qu'un analyseur classique (morphologie, syntaxe, s  mantique, entit  s, cor  f  rences) selon une architecture modulaire et hautement configurable, avec des performances   lev  es aussi bien en vitesse qu'en qualit   d'analyse.

Une de ses particularit  s est le cross-linguisme, elle permet de travailler gratuitement sur des textes anglais, fran  ais ou m  me portugais. Outre le cross-linguisme d'une dizaine de langues, LIMA pr  sente de nombreux atouts. Il permet de traiter des documents multim  dias comme des images, des vid  os ou des paroles, et pas seulement du texte. Il offre aux utilisateurs le choix de la technologie, pour travailler sur diff  rentes tailles de donn  es et assurer le passage    l'  chelle (algorithmie et architecture, ou big analytics). Il est capable de prendre en compte les sp  cificit  s d'un domaine ce qu'il lui permet de d  passer les performances en pr  cision des outils g  n  riques du march  .

LIMA est un analyseur bas   sur des r  gles. L'avantage est que ces r  gles (  tant fix  es par des sp  cialistes) sont beaucoup plus pr  cises qu'un mod  le statistique obtenu via du machin learning. On est s  r de leur exactitude, on s'attend donc    obtenir des r  sultats sup  rieurs (plus pr  cis) par rapport aux mod  les utilisant du machine learning.

Les d  savantages sont eux aussi li  s au c  t   humain : on a besoin d'op  rateur humain pour fixer les r  gles, et si l'on veut que ce soit pr  cis, nous ferons appel    des sp  cialistes. Cette d  marche doit   tre effectu  e pour chaque langue pour laquelle nous voulons pouvoir utiliser l'outil. C'est un temps de d  veloppement extr  mement long pour des r  sultats qui, en fonction du contexte, peuvent se r  v  ler inutilement pr  cis. De plus si la langue   volue, il faudra modifier ces r  gles directement dans le code, ce qui peut s'av  rer 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'universit  de Stanford partage de nombreux outils gratuitement depuis plusieurs ann es, comme le Stanford CoreNLP. Il s'agit d'une librairie d velopp e en java.

Stanford CoreNLP met   disposition un ensemble d'outils d'analyse grammaticale. Il est capable de reconna tre les mots de base de plusieurs langues, que ce soit des noms d'entreprise, de personnes, etc. Il sait  galement 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  galement de nombreux d'atouts. Une API est disponible pour la majorit  des langages de programmation modernes, ce qui le rend facilement int grable   d'autres logiciels, et est plus accessible   une majorit  de personne. Stanford prend en charge un certain nombre de langues humaines principales, par d faut l'anglais, mais le moteur est compatible avec les mod les d'autres langues, comme l'arabe, le chinois, le fran ais, etc. Ils proposent m me des mod les optimis s pour travailler avec un anglais sans casse par exemple.

Stanford est un outils enti rement bas  sur du machine learning, au contraire de Lima permettant d'obtenir "rapidement" un mod le qui se montrera plus ou moins satisfaisant. Par rapport   un outils bas  sur des r gles, on a besoin ici de faire de l'apprentissage. On a donc besoin d'un corpus d'apprentissage annot . La qualit  de ce corpus annot  (sa taille, la pr cision des annotations, sa coh rence avec les cas qui seront rencontr  par l'outil dans le futur) influe directement sur la pr cision du mod le de ML qui sera obtenu. C'est un travail fastidieux mais n cessaire. N anmoins, comme on n'a pas besoin d'une pr cision aussi forte que pour un mod le   base de r gles, on peut confier ce travail   n'importe qui (puis faire une validation apr s). Si jamais on veut coller aux  volutions de la langue il suffit de modifier le corpus et de relancer l'apprentissage du mod le. Cela prendra moins de temps et sera moins compliqu  que de recoder les r gles   la main.

On gagne en rapidit  et facilit  d'adaptation de l'outils   de nouvelles langues, mais on perd en pr cision.

On peut trouver en ligne une version d mo : <https://corenlp.run/>

## C.NLTK

NLTK est une boîte à outil permettant la création de programmes pour l'analyse de texte. Elle a été créée en 2001 dans une université de Pennsylvanie. NLTK est puissant et documenté, ce qui en a fait 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 choses. 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 :

- Simplicit : il faut fournir un cadre intuitif, donner une connaissance pratique de la PNL de mani re simple et cacher la partie compliqu e.
- Coh rence: Il faut fournir un cadre uniforme avec les interfaces et les structures de donn es coh rentes avec des noms de m thodes intuitives.
- Extensibilit : Il faut fournir un cadre modulaire pour pouvoir ajouter des fonctionnalit s et des modules facilement.
- Modularit : Il faut fournir des composants qui peuvent  tre utilis s ind pendamment du reste de la bo te   outils.

L'approche hybride, selon comment elle est utilis e permet de combiner le meilleur des deux mondes (approche par r gles ou approche statistique), ou le pire. L'id e de l'approche hybride est d'utiliser des r gles pour les parties jug es critiques (ce qui permet de gagner en pr cision), et utiliser l'approche statistique aux endroits o  l'on souhaite pouvoir  tre plus flexible et qui n cessitent moins de pr cision (on en profite donc pour gagner du temps de d veloppement).

## 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 mots.

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 associés à des étiquettes. Afin de pouvoir comparer les résultats des différentes plateformes, il faut normaliser selon Universal pour les POSTags et CoNLL pour les entités nommées. 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 ayant été identifiés 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 :

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

Stanford :

```
STANFORD :
Warning: the reference and the candidate consists of different number of lines!
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.txt 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 :

```
tru@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 candidate consists of different number of lines!
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 r sultats

Nous avons eu beaucoup de probl mes avec les fichiers fournis. Apr s quelques manipulations, nous arrivons   obtenir de bons r sultats avec Lima pour la partie sur le postaging. De m me avec Stanford, mais seulement sur une partie des mots, un bug nous emp chant actuellement de tous les traiter. Cependant, avec NLTK, nous avons de mauvais r sultats parce que nous n'avons pas touch  au fichier apr s la g n ration. Et comme les tags ont  t  fait avec des outils diff rents, ils ne font pas les m mes d coupages. Par exemple pour la premi re phrase, le nom "Pierre Vinken" est d coup  en deux mots, alors que dans le fichier de r f rence fourni, il est identifi  comme une entit e, ce qui fait que lors de l' valuation, cela ne correspond pas. De plus, les d calages ne sont pas pris en compte dans le script d' valuation ce qui fait encore plus d'erreurs et provoque les mauvais r sultats. Pour LIMA et Stanford nous avons fait des scripts de normalisation pour parser les fichiers et les remettre sous le bon format. N anmoins m me le formatage que nous avons fait n'est pas parfait (on voit que le nombre de lignes n'est pas le m me quand on fait la comparaison avec la r f rence).

Malheureusement l'absence de normalisation utilisable pour tous les fichiers ne nous permet pas d' valuer la v ritable pr cision des mod les que nous utilisons. Il nous est impossible de savoir si l' cart entre les r sultats de deux mod les est due   la performance des mod les ou   une sortie trop diff rente de ce qui est attendu par le fichier de r f rence.



## Conclusion:

### A. R sum  du travail effectu 

Sur ce projet, nous avons Stanford presque enti rement fonctionnel sur le pos\_tagging et la reconnaissance d'entit  nomm e. Pour sa part, LIMA est enti rement fonctionnel. Pour NLTK, nous avons seulement termin  le pos\_tagging puisque la reconnaissance d'entit  nomm e ne fonctionne pas d  au format de sortie en arbre de NLTK.

### B. Limitation des plateformes

Les plateformes sont trop limit es par le format des donn es   fournir. De plus les analyses n'ont pas vraiment de valeurs puisque la comparaison est b te et m chante, s'il y a un d calage parce que certaines plateformes regroupent plusieurs entit s, on a une  valuation proche de 0%.

Par ailleurs on peut trouver des probl mes m me au niveau des sorties des outils. Par exemple LIMA sort la phrase suivante "Automobile Dealers' Association" l  o  dans le fichier de r f rence on a "Automobile Dealers Association". Cette l g re modification d truit la pr cision des mots et rend difficile le r alignement des tags.

### C. Piste d'am lioration

Il faudrait que la s paration des entit es dans le fichier de r f rence soit la m me que la s paration faite par les outils utilis s (ou au moins que des r gles puissent lier les deux). Nous aurions pu am liorer nos r sultats 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://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/>