RAPPORT PROJET TAL

Evaluation de deux plateformes open source d'analyse linguistique

KÉVIN LACOSTE
YAO SHI
MAX M'BOURRA

10 MARS 2019

Table des matières

Introduction

Le traitement automatique des langues est un domaine vaste et qui prend de plus en plus d'importance avec le temps. Car nous sommes aujourd'hui dans une société d'intercommunication entre les personnes du monde entier. De ce fait, nous avons un besoin grandissant d'application permettant de traiter des morceaux d'informations électronique sous la forme de document électronique, mail, page web et d'autre. De plus, dans le domaine de l'IA, la modélisation du mécanisme du langage est un domaine passionnant.

Pour la traduction automatique des langues, nous avons plusieurs approches possibles. Certaines de ces approches sont plus efficaces que d'autres. La toute première approche a été la traduction mot à mot avec réarrangement des mots et utilisation de dictionnaire. Cette approche nous donne des résultats très peu convaincant. Cela s'explique par le fait qu'elle ne prend pas en compte l'aspect contextuel des mots et de la phrase. Cette approche date des années 60.

Depuis, l'ensemble des recherches dans le domaine s'est orienté vers le sens, avec la Syntatic Structure de Noam Chomsky en 1957, qui se base sur l'analyse syntaxique des langues et la définition formelle des grammaires. Dans la poursuit des recherches orientées vers le sens, nous avons la sémantique procédurale de Woods. C'est une approche procédurale du sens, qui se base sur le traitement logique des sens. Ces approches ont conduit à des avancées et à quelques réussites dans le milieu. Par exemple avec TAUM meteo, en 1970 qui effectue la traduction automatique de bulletins météos. Ces approches sont efficaces dans l'ensemble. Cependant, les systèmes issus de ces approches se limitent à chaque fois à un domaine précis.

Ensuite nous avons dans les années 80, l'approche symbolique. Cette approche est une synthèse de l'ensemble des autres approches. Elle mélange la linguistique et la logique. L'approche symbolique a conduit à créer les premières grosses grammaires informatiques. Cette approche a deux limites, la première est le manque de robustesse du système et la second est le manque d'applications, qui implémentent le système.

Les approches les plus récentes sont l'approche par corpus et règle, l'approche par apprentissage automatique et l'approche statistique. L'approche par corpus demande d'avoir une taille conséquente du corpus. Depuis les années 90 cela est possible, grâce à la baisse des prix de l'espace mémoire et de la grande quantité de document numérisé et électronique. La grande quantité de donnée permet d'avoir des corpus de taille conséquent mais aussi d'implémenter des systèmes permettant d'avoir des résultats plus importants. L'approche par apprentissage automatique utilise les réseaux de neurones. Cette approche utilise les réseaux de neurone de type RNN (Recurrent Neural Network). Cela est possible aujourd'hui grâce à l'évolution de la puissance de calcul et la puissance des cartes graphiques. L'approche par apprentissage automatique permet d'avoir de bon résultat.

Pour résumer l'état de l'art, dans le domaine du TAL, plusieurs approches se sont succédé. Chaque approche a apporté une évolution dans le domaine. Aujourd'hui, c'est l'approche par corpus statistique et apprentissage automatique, qui est le plus utilisé.

Dans le cadre du projet de TAL, nous avons expérimenté deux plateformes d'analyses linguistiques. Ces plateformes linguistiques sont Lima et Stanford. Les deux plateformes sont open source, c'est l'une des raisons pour lesquelles nous les avons utilisés.

Plateforme d'analyse linguistique

1. Plateforme CEA List Lima

La plateforme CEA Lima est une plateforme linguistique open source. Cette plateforme supporte différentes langues. Elle en supporte 5 langues : le français, l'anglais, le chinois et l'arabe. La plateforme est basée sur le NLP, Natural Language Processing. La plateforme Lima prend en entrer un texte dans une langue et ensuite nous retourne un document xml avec les éléments du texte découpé en entité nommées. A chaque entité nommés le système lui associe des étiquettes propres à lima. L'avantage de lima, c'est ça flexibilité. Il permet de faire une configuration dynamique de l'application. L'application permet aussi d'ajouter des modules de langue supplémentaire. C'est pour cela que le système peut gérer l'arabe. Lima possède de très bonne performance. De plus l'application garde de bonne performance avec différente configuration des processeurs.

2. Plateforme Stanford

La plateforme Stanford CoreNLP, est une plateforme développée en java. La plateforme se base aussi sur le Natural Language Processing. Les pipelines de la plateforme combinent plusieurs composant d'analyse de langage naturel. Comme pour la plateforme lima, la plateforme stanford prend en entrer un texte et nous retourne un fichier XML. Ce fichier est annoté avec des étiquettes d'entités nommés spécifique au système stanfordCore NLP.

Partie II : Utilisation de LIMA

Exercice 1 : analyse de tests

Après le lancement de l'analyse via la commande `analyeText -l eng wsj_0010_sample.txt` , le résultat de l'analyse linguistique apparaît dans la sortie standard . Chaque token analysé dans le texte est affiché sur une ligne (sauf pour le saut de ligne qui est juste affiché), chaque champ étant séparé par une tabulation, de la façon suivante (source : https://depparse.uvt.nl/DataFormat.html) :

- **ID** : identifiant du token analysé dans la phrase
- **FORM** : Mot ou ensemble de mots encapsulés en tant que token
- **LEMMA** : Identique au champ FORM , hormis dans le cas des mots à forme variable (verbes par exemple) où il abrite sa forme neutre (pour un verbe, sa base verbale)
- **CPOSTAG** : PostTag décrivant la nature grammaticale du token de manière grossière dans un premier temps.
- **POSTAG** : PostTag décrivant la nature grammaticale du token de manière plus précise. C'est ce qui sera utilisé pour la suite.
- **FEATS** : champ contenant le type d'entité représenté par le token si celui-ci peut être classé de la sorte (Numex.NUMBER, Person.PERSON, etc.)

D'autres champs (**HEAD**, **DEPREL**, **PHEAD** et **PDEPREL**) existent mais ne seront pas utilisés pour la suite.

En plus de cette sortie standard générée par LIMA, en activant les loggers *specificEntitiesXmlLogger* et *disambiguatedGrapfXmlLogger*, LIMA se met à génerer deux nouveaux fichiers : si le nom du fichier en entré est représenté par <input>, les fichiers *<input>.se.xml* et *<input>.disambiguated.xml* sont générés, le premier listant les différents tokens générés précedemment avec leur catégorie grammaticale, le second listant toutes les entités nommées détectées par LIMA.

En addition à ces fichiers, il est possible de récupérer la sortie générée par LIMA dans un fichier facilement parsable en redirigeant la sortie standard vers un fichier .conll . Cela est intéressant car toutes les informations qui seront utiles par la suite se trouveront dans ce fichier.

Exercice 2 : extraction d'entités nommées

Le script *limaEntitiesExtractor.py* permet de récupérer dans la sortie standard la liste des entités nommées, leur type, leur nombre d'occurences dans le texte ainsi que la proportion qu'ils représente dans le texte.

Il prend en entrée le fichier.conll généré par LIMA en redirigeant la sortie standard, car toutes les informations nécessaires y sont présentes : Nom, Type d'entité et indirectement le nombre de fois que le mot apparaît (un dictionnaire permet de vérifier cela) ainsi que le nombre de mots dans le texte pour calculer le dernier paramètre affiché.

Voici les résultats obtenus pour le fichier *wsj_0010_sample.txt* comme exemple:

```
125     Numex.NUMBER    1       0.91
Boca Raton      Location.LOCATION       1       0.91
Hoosier Person.PERSON   1       0.91
Indianapolis    Location.LOCATION       1       0.91
National Association    Organization.ORGANIZATION       1       0.91
Rust Belt       Location.LOCATION       1       0.91
```

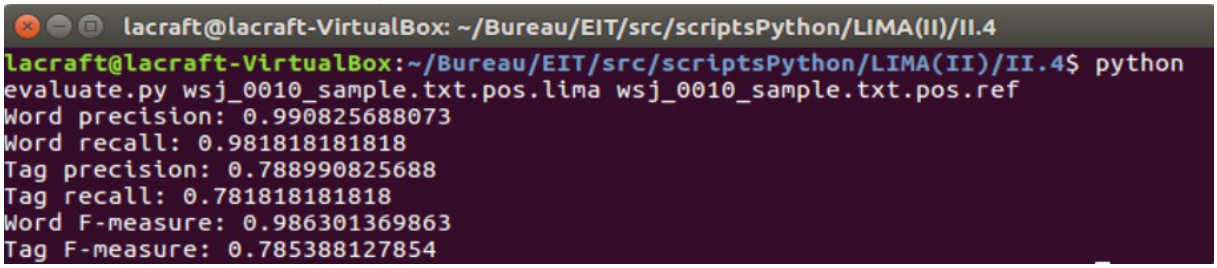
Exercice 3 : Analyse morpho-syntaxique

Le script *syntaxicAnalyser.py* permet de traiter un fichier .conll venant de LIMA et d'en créer un nouveau contenant le texte représenté par le .conll avec comme suffixe TAG, TAG étant le PostTag grammatical représenté par **POSTAG**. Le résultat obtenu utilise les tags de la Penn TreeBank (PTB) en suffixe des mots. Il est aussi possible d'utiliser ce script sur le fichier équivalent au format .disambiguated.xml afin d'obtenir un résultat équivalent, à ceci près que les mots entrés sont les lemmas (donc base verbale par exemple et non pas sa version conjuguée) et qu'aucun saut de ligne n'est disponible.

Exercice 4 : Evaluation de l'analyse morpho-syntaxique

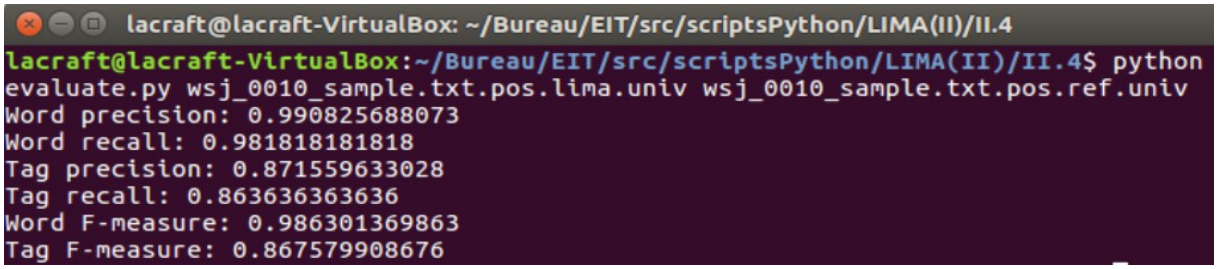
En utilisant le script *evaluate.py* fourni, il est possible avec un fichier de référence d'évaluer les performances d'analyse syntaxique de LIMA. Malheureusement, le fichier de référence pour *wsj_0010_sample.txt*, *wsj_0010_sample.txt.pos.ref*, n'est pas dans le bon format pour être comparé au fichier obtenu précedemment, *wsj_0010_sample.txt.pos.lima* . En effet, contrairement au fichier généré précedemment, cette référence possède pour chaque ligne un mot du texte original avec son PostTag. Un script du nom de *convertRefToLine.py* a donc été réalisé afin de pallier au problème. De plus, un nettoyage du fichier a dû s'imposer car certains caractères non voulus étaient présents (comme le caractère]) et les sauts de ligne n'ont pas pu être récupérés.

Une fois ce petit souci résolu, nous pouvons évaluer les performances de LIMA sur cette analyse syntaxique :



Les résultats obtenus sont plutôt bons, avec une précision des tags de presque 79 %. Cependant, avec de grands corpus, les 21 % d'erreurs peuvent être très pénalisantes pour la traduction et donc il est aussi possible de dire qu'ici l'analyse morpho-syntaxique de LIMA n'est pas assez bonne.

Cependant, qu 'en est-il quand les tags utilisés sont les tags universelles, standardisée ? Grâce au script *translateToUniv.py*, et avec la table de traduction *POSTags_PTB_Universal.txt*, il est possible d'effectuer à la volée la conversion des PostTags des fichier de test et fichier de référence. Voici donc le nouveau résultat obtenu pour les deux nouveaux fichiers avec tags universels :



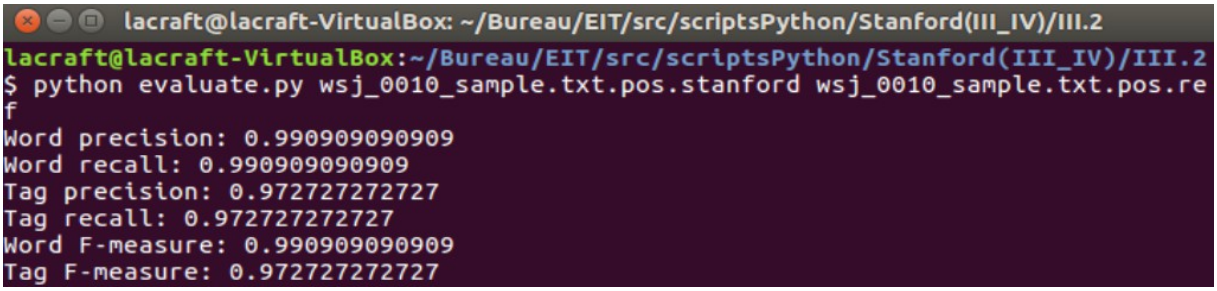
Il est possible de s'apercevoir que les résultats obtenus sont meilleurs ici, et à cela il y a plusieurs raisons :

- Le nombre de tags PTB est plus élevé que celui des tags universels et sont plus précis. Cela a pour conséquence « d 'améliorer » les performances de l'analyseur morpho-syntaxique en apparence
- Certains tags générés par LIMA n'étaient pas présents dans le fichier de référence, notamment les tags COMMA (représenté par « , ») , COLON (représenté par « : »), etc. Ces éléments qui étaient considérés comme incorrects ne le sont donc plus.

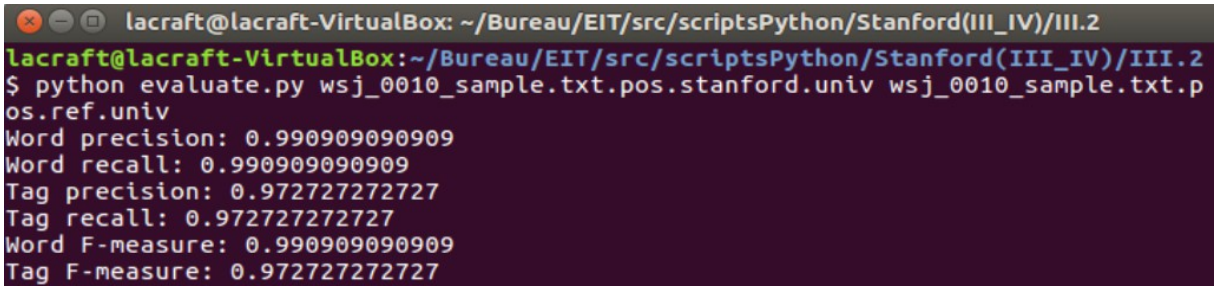
Partie III : Utilisation de l'outil de désambiguïsation morpho-syntaxique de l'université de Stanford

L'outil de désambiguïsation morpho-syntaxique de Stanford est un outil permettant d'effectuer la même opération que celle faite précedemment avec LIMA. Comme LIMA, cet outil utilise de base les étiquettes PTB et les affiche en suffixe aussi de la même façon (_TAG). Cet outil affiche le résultat de son analyse dans la sortie standard il faut donc, comme pour LIMA et la production des fichiers .conll, rediriger la sortie standard dans un fichier allant contenir le résultat de notre analyse.

Afin d'évaluer les performances de l'outil de Stanford, nous pouvons ré-utiliser le script *evaluate.py*, et après comparaison de ce fichier avec *wsj_0010_sample.txt.pos* précédent (référence) et ré-agencement du fichier Stanford de manière à avoir les mêmes sauts de ligne et structure de texte, nous obtenons les résultats suivants :



Il est possible de se rendre compte que les résultats de l'analyse sont meilleurs que ceux obtenus précédemment, avec une précision des tags de 97 % à comparer avec les 87 % précédents. Cependant, passer à des tags universels via le script utilisé précédemment n'améliore pas les performances de l'analyseur :



Nous pouvons en déduire que l'analyseur de Stanford utilise des tags plus précis et fonctionnels, en tout cas plus fonctionnels que ceux de LIMA, même si les tags universels, moins précis, font que LIMA arrive malgré tout à s'en sortir malgré le fait que l'outil de Stanford apparaît meilleur.

Aussi, il est important de mentionner qu'une réelle comparaison des deux logiciels demanderait leur utilisation sur une multitude de corpus afin de pouvoir faire une moyenne des résultats obtenus.

Partie IV : Utilisation de l'outil de reconnaissance d'entités nommées de l'université de Stanford

Parmi les outils de traduction de Stanford se trouve aussi un outil de reconnaissance d'entités nommées. Cependant, contrairement à celui de LIMA, cet outil génère un fichier de sortie ayant une structure assez similaire à celle des fichiers d'analyse morpho-syntaxique précédents, à ceci près que :

- Les _ sont remplacés par des /
- Les tags disponibles sont les tags **PERSON**, **ORGANIZATION** et **LOCATION**, tags équivalents aux tags **Person.PERSON**, **Organization.ORGANIZATION** et **Location.LOCATION** de LIMA
- Tous les mots dont aucun tag n'a été trouvé ont le suffixe « /O »

Le problème avec ce fichier est que le surplus de tags O inutiles rend le fichier peu lisible, nous avons donc réalisé un script permettant de pouvoir afficher les entités nommés détectées de la même façon que dans la partie II : *stanfordEntitiesExtractor.py*. Voici le résultat de ce script :

```
Association     ORGANIZATION    1       0.87
Boca    LOCATION        1       0.87
Hot     LOCATION        1       0.87
Indianapolis    LOCATION        1       0.87
Manufacturers   ORGANIZATION    1       0.87
National        ORGANIZATION    1       0.87
of      ORGANIZATION    1       0.87
Raton   LOCATION        1       0.87
Springs LOCATION        1       0.87
```

En comparant ce résultat à celui équivalent pour LIMA, il apparaît que l'outil de Stanford met plus l'emphase sur les Organisations que LIMA (Par exemple « National » qui en réalité est un morceau de « National Association of Manufacturers », qui n'est pas présent chez LIMA). Au contraire, LIMA avait trouvé plus de lieux que Stanford (Par exemple « Rust Belt » qui n'est malheureusement pas réellement un lieu dans le contexte). Quelle peut donc être la meilleure des deux plateformes en ce qui concerne la détection des entités nommées ?

Partie V : Evaluation des outils de reconnaissance d'entités nommées du CEA List (LIMA) et de l'université de Stanford

Afin de pouvoir évaluer la précision de ces outils de reconnaissance, il faut pouvoir trouver une notation commune aux sorties des reconnaissances d'entités de LIMA, Stanford et ENAMEX (fichier référence). De plus, il faut aussi modifier les 3 fichiers à comparer de façon à ce que ceux-ci aient la même structure (sauts de lignes, espaces, etc.)

Création de la table de traduction pour LIMA

Afin de pouvoir traduire les tags d'entités nommés de LIMA vers la notation de Stanford, il a fallu créer une petite table de traduction permettant de passer d'une notation à une autre. Il a fallu prendre en compte tous les tags disponibles avec cet outil de LIMA et les traduire en 4 types d'entités : O (aucune entité), PERSON, LOCATION et ORGANIZATION.

Avec cette table, le fichier .conll crée par LIMA, le script *translateLIMAEntToStanfordEnt.py* et un reformattage à la main pour que la référence et ce fichier de test aient la même structure, le premier fichier est prêt pour la comparaison au format Stanford.

Traduction de la référence ENAMEX au format Stanford

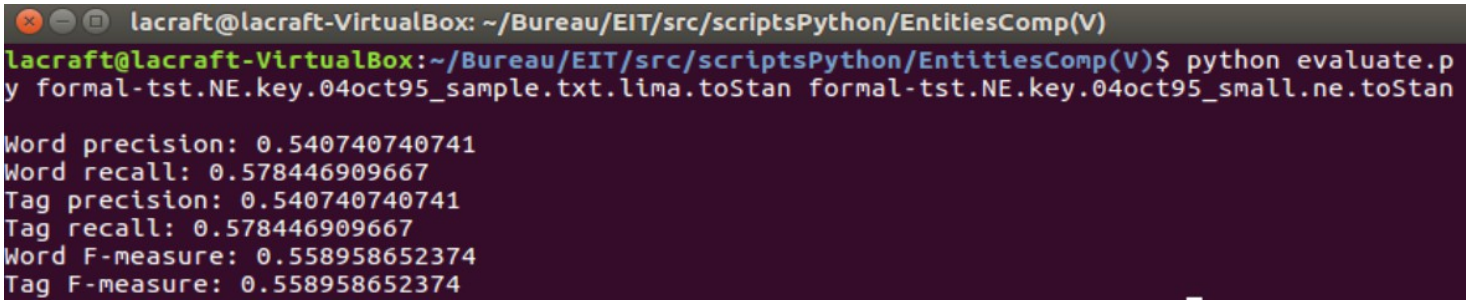
Malheureusement, la référence nécessite aussi une traduction afin de pouvoir être comparée avec les autres fichiers de test. Pour ce faire, le script *convertENAMEXToStanford.py* s'occupe d'effectuer la conversion en utilisant des expressions régulières de manière à détecter les tags intéressants dans notre cas (<ENAMEX TAG="PERSON"> par exemple), à les traduire sous le format Stanford (dans cet exemple /PERSON) et à mettre sur les autres mots peu intéressants le tag /O (non-reconnu).

Creation et formattage du fichier Stanford

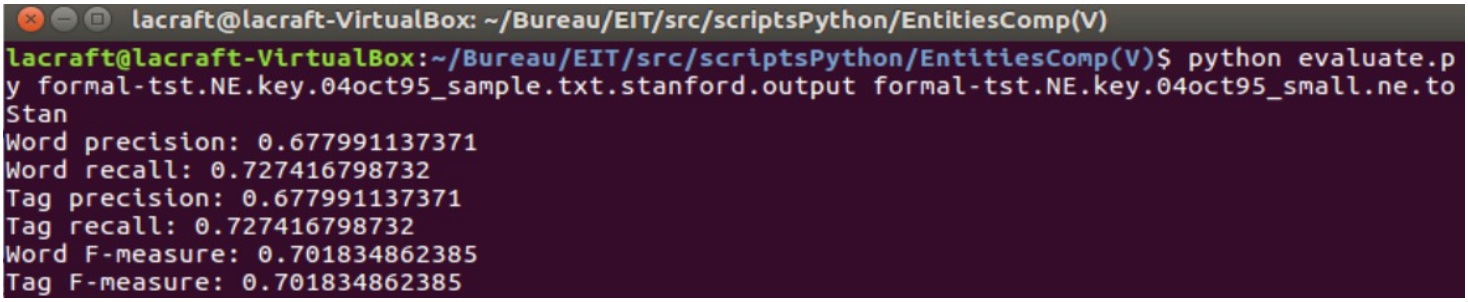
En ce qui concerne le fichier d'entités nommées de Stanford, Il ne suffit plus qu'à mettre en forme le fichier obtenu en traitant notre texte de référence (*formal-tst.NE.key.04oct95_small.txt*) avec l'outil de Stanford de la partie IV et à le mettre en forme de façon à pouvoir le comparer avec les deux fichiers précédents. Ainsi, nous pouvons commencer à comparer nos trois fichiers.

Comparaisons

- LIMA



- Stanford



Il s'avère, après comparaison avec la référence ENAMEX, que l'extracteur d'entités nommées de Stanford est plus précis et efficace que celui de LIMA, avec 10-15% de précision supplémentaire. Cette différence n'est pas énorme, mais montre malgré tout que l'outil de Stanford est en général plus efficace que LIMA. Cependant, une autre conclusion se dégage : il paraît plus difficile de détecter des entités nommées avec précision que pour les unités grammaticales.

Répartition du travail

Kévin LACOSTE : rédaction du rapport, travail Partie IV et V, screens

Max M'BOURRA : rédaction du rapport, travail Partie II

Yao SHI : rédaction du rapport, travail Partie III