POLYTECH PARIS-SUD

**Et5-Info**

Extraction d'Information – Traduction automatique

# Projet – Evaluation d'outils de TAL

## I. Le sujet : Evaluation de deux plateformes open source d'analyse linguistique

Une plateforme d'analyse linguistique standard se compose des modules suivants :

1. **Tokenisation :** Ce module consiste à découper les chaînes de caractères du texte en mots, en prenant en compte le contexte ainsi que les règles de découpage. Ce module utilise généralement des règles de segmentation ainsi que des automates d'états finis.
2. **Analyse morphologique :** Ce module a pour but de vérifier si le mot (token) appartient à la langue et d'associer à chaque mot des propriétés syntaxiques qui vont servir dans la suite des traitements. Ces propriétés syntaxiques sont décrites en classes appelées catégories grammaticales. La consultation de dictionnaires de formes ou de lemmes permet de récupérer les propriétés syntaxiques concernant les mots à reconnaître.
3. **Analyse morpho-syntaxqiue :** Après l'analyse morphologique, une partie des mots restent ambigus d'un point de vue grammatical. L'analyse morphosyntaxique réduit le nombre des ambiguïtés en utilisant soit des règles ou des matrices de désambiguïsation. Les règles sont généralement construites manuellement et les matrices de bi-grams et tri-grams sont obtenues à partir d'un corpus étiqueté et désambiguïsé manuellement.
4. **Analyse syntaxique :** Ce module consiste à identifier les principaux constituants de la phrase et les relations qu'ils entretiennent entre eux. Le résultat de l'analyse syntaxique peut être une ou plusieurs structures syntaxiques représentant la phrase en entrées. Ces structures dépendent du formalisme de représentation utilisé : un arbre syntagmatique, un arbre de dépendance ou une structure de traits. L'analyse en dépendance syntaxique consiste à créer un arbre de relations entre les mots de la phrase. Le module d'analyse syntaxique utilise des règles pour l'identification des relations de dépendance ou des corpus annotés en étiquettes morpho-syntaxiques et en relations de dépendance.
5. **Reconnaissance d'entités nommées :** Ce module consiste à identifier les dates, lieux, heures, expressions numériques, produits, évènements, organisations, présentes sur un ou plusieurs tokens, et à les remplacer par un seul token.

**Travail demandé**

Vous allez installer et évaluer deux plateformes d'analyse linguistique : CEA List LIMA et Stanford Core NLP.

**CEA List LIMA :** est une plateforme d'analyse linguistique multilingue utilisant des règles et des ressources validées par des experts linguistes.

**Stanford Core NLP :** est une boite à outils linguistiques utilisant l'apprentissage statistique à partir de corpus annotés.

**Consignes générales**

Le but de ce projet est de montrer que vous pouvez installer, évaluer et rédiger un rapport décrivant les principaux résultats, points forts et limitations d'une plateforme open source d'analyse linguistique.

Vous pouvez utiliser un dépôt ***git*** à partager entre les membres de votre groupe pour assurer la compatibilité entre les modifications que vous faites. Ce répertoire git devra être structuré comme suit :

1. Un fichier texte nommé ***README*** contenant les indications nécessaires pour exécuter les scripts Python que vous avez développés. Vous indiquerez notamment les options pour la ligne de commande ainsi que des exemples d'utilisation.
2. Un dossier ***src*** pour les codes source de vos scripts Python. La lisibilité du code sera l'un des critères de notation (commentaires précis, concis, clairs et bien orthographiés).
3. Un dossier **doc** pour votre rapport écrit en LATEX et compilé en un PDF de 5 `a 7 pages nommé ***nom1-nom2-nom3.pdf*** qui décrit votre travail. Vous y décrirez l'objectif du projet, les résultats d'évaluation des deux plateformes d'analyse linguistique, les points forts et les limitations de chaque plateforme. Si vous savez comment résoudre ces limitations, mentionnez-le et décrivez vos idées en les présentant comme des pistes pour un travail futur. Vous devez également inclure une petite section qui décrit la contribution respective de chaque membre du groupe. Le rapport peut être écrit en anglais ou en français.

Vous m'enverrez par mail (nasredine.semmar@cea.fr) un lien vers votre projet dès que vous aurez créé le répertoire git. Ensuite, lorsque le répertoire est prêt pour être noté, vous m'informerez par mail également au plus tard le vendredi 8 mars 2019.

La date limite pour la finalisation du projet est donc le **vendredi 8 mars 2019**.

## II. Installation de la plateforme d'analyse linguistique open source LIMA (1/2 heure)

Avant de démarrer l'installation de la plateforme LIMA sur un poste équipé d'une distribution Linux (de préférence Ubuntu 16.04 LTS), il faudrait préparer l'environnement d'exécution en installant les dépendances suivantes:

```
sudo apt-get install qt5-default libqt5xmlpatterns5 libqt5quick5 libqt5declarative5 libboost-all-dev libenchant1c2a libtre5
```

**Installation : https://github.com/aymara/lima/wiki/Install-Linux-packages**

1. Installation de Svmtool++
   a. Télécharger la version de Svmtool ++ correspondant à Ubuntu 16.04 LTS (svmtool-cpp-1.1.7-ubuntu16.deb)
   b. Installer Svmtool ++: `sudo dpkg -i svmtool-cpp-1.1.7-ubuntu16.deb`
2. Installation de Qhttpserver
   a. Télécharger la version de Qhttpserver correspondant à Ubuntu 16.04 LTS (qhttpserver-0.0.1-ubuntu16.04.deb)
   b. Installer Qhttpserver : `sudo dpkg -i qhttpserver-0.0.1-ubuntu16.04.deb`

3. Installation de LIMA
    a. Télécharger la version de LIMA correspondant à Ubuntu 16.04 LTS (lima-2.1.201901211052240800-d8065c6-Ubuntu16.04-x86_64.deb)
    b. Installer LIMA: `sudo dpkg -i lima-2.1.201901211052240800-d8065c6-Ubuntu16.04-x86_64.deb`

**Utilisation : https://github.com/aymara/lima/wiki/LIMA-User-Manual**

Les fichiers contenant les textes à analyser doivent être en UTF-8. Le résultat de l'analyse est au format CONLL-X (https://depparse.uvt.nl/DataFormat.html).

**Example:**

1. Créer le fichier « Sample_eng.txt" contenant la phrase "John ate dicious pizza with friends."
2. Lancer l'analyse linguistique : `analyzeText -l eng Sample_eng.txt > Sample_eng.txt.output`

LIMA output (Contenu du fichier Sample_eng.txt.output):

```
1       John    John    NP      PROPN   _       _       2       SUJ_V   _       _
2       ate     eat     V       VERB    _       _       _       _       _       _
3       dicious dicious NC      NOUN    _       _       4       ADJPRENSUB      _       _
4       pizza   pizza   NC      NOUN    _       _       2       COD_V   _       _
5       with    with    PREP    ADP     _       _       6       PREPSUB _       _
6       friends friend  NC      NOUN    _       _       4       COMPDUNOM       _       _
7       .       .       PONCTU  SENT    _       _       _       _       _       _
```

## Exercices

**Note :** Les exercices 2, 3 et 4 peuvent être faits sans avoir installé la plateforme LIMA (Exercice 1).

## 1. Utilisation de la plateforme LIMA pour l'analyse de textes

a. Editer le fichier « lima-lp-eng.xml » (/usr/share/config/lima) et identifier les principaux modules composant la plateforme d'analyse linguistique LIMA (voir la section /* Definition of pipelines */).
b. Lancer LIMA sur le fichier « wsj_0010_sample.txt » : `analyzeText -l eng wsj_0010_sample.txt`
c. A quoi ressemble le format du résultat de cette analyse ?
d. Activer dans le fichier « lima-lp-eng.xml » les loggers "specificEntitiesXmlLogger" et "disambiguatedGraphXmlLogger".
e. Lancer à nouveau LIMA sur le fichier « wsj_0010_sample.txt » et observer les sorties produites : `analyzeText -l eng wsj_0010_sample.txt`
f. Rediriger le résultat d'analyse vers le fichier « wsj_0010_sample.txt.conll » : `analyzeText -l eng wsj_0010_sample.txt > wsj_0010_sample.txt.conll`

## 2. Extraction d'entités nommées

A partir des sorties de l'analyseur LIMA « wsj_0010_sample.txt.se.xml » (ou « wsj_0010_sample.txt.conll »), écrire un programme Python permettant de représenter les entités nommées sous le format suivant :

Entité nommée Type Nombre d'occurrences Proportion dans le texte (%)

**Exemple :**

| **Entité nommée** | **Type** | **Nombre d'occurrences** | **Proportion dans le texte (%)** |
|---|---|---|---|
| Indianapolis | LOCATION | 1 | 14 (1/7) |

**Notes:**

- Mettre le résultat de la transformation du fichier « wsj_0010_sample.txt.disambiguated.xml » ou « wsj_0010_sample.txt.conll » dans le fichier « wsj_0010_sample.txt.ner.lima ».
- Appliquer cette transformation sur le fichier « formal-tst.NE.key.04oct95_sample.txt.conll » ou « formal-tst.NE.key.04oct95_sample.txt.se.xml ».

## 3. Analyse morpho-syntaxique

A partir de la sortie de l'analyseur LIMA « wsj_0010_sample.txt.disambiguated.xml » ou « wsj_0010_sample.txt.conll », écrire un programme Python permettant de représenter les étiquettes morpho-syntaxiques sous le format « Mot_Etiquette ».

**Exemple :**

Pour la phrase « When it's time for their biannual powwow, the nation's manufacturing titans typically jet off to the sunny confines of resort towns like Boca Raton and Hot Springs.", nous avons la representation suivante:

When_WRB it_PRP 's_VBZ time_NN for_IN their_PRP$ biannual_JJ powwow_NN ,_, the_DT nation_NN s_POS manufacturing_VBG titans_NNS typically_RB jet_VBP off_RP to_TO the_DT sunny_JJ confines_NNS of_IN resort_NN towns_NNS like_IN Boca_NNP Raton_NNP and_CC Hot_NNP Springs_NNP ._.

**Note:** Mettre le résultat de la transformation du fichier « wsj_0010_sample.txt.disambiguated.xml » ou « wsj_0010_sample.txt.conll » dans le fichier « wsj_0010_sample.txt.pos.lima ».

## 4. Evaluation de l'analyse morpho-syntaxique

1. **Evaluation à l'aide des étiquettes Penn TreeBank (PTB) :** Utiliser le programme Python « evaluate.py » pour évaluer l'analyseur morpho-syntaxique de la plateforme LIMA. L'évaluation se fait sur les fichier dans le nouveau format (wsj_0010_sample.txt.pos.lima ) (`python evaluate.py wsj_0010_sample.txt.pos.lima wsj_0010_sample.txt.pos.ref`)

   **Exemple :**
   ```
   python evaluate.py wsj_0010_sentence.pos.lima
   wsj_0010_sentence.pos.ref
   ```

   **Note:** Les deux fichiers « wsj_0010_sentence.pos.lima » et « wsj_0010_sentence.pos.ref » sont fournis pour illustrer leur format respectif et le résultat de l'évaluation.

2. **Evaluation à l'aide des étiquettes universelles :**
    a. Remplacer à l'aide d'un programme Python les étiquettes Penn TreeBank des fichiers « wsj_0010_sample.txt.pos.lima » et « wsj_0010_sample.txt.pos.ref » par les étiquettes universelles en utilisant la table de correspondance « POSTags_PTB_Universal.txt ».

       **Note :** Nommer les fichiers avec les étiquettes universelles comme suit : « wsj_0010_sample.txt.pos.univ.lima » et « wsj_0010_sample.txt.pos.univ.ref ».

    b. Utiliser le programme Python « evaluate.py » pour évaluer l'analyseur morpho-syntaxique de la plateforme LIMA selon les étiquettes universelles (`python evaluate.py wsj_0010_sample.txt.pos.univ.lima wsj_0010_sample.txt.pos.univ.ref`).
    c. Quelles conclusions peut-on avoir à partir de ces deux évaluations ?

**Note:** Certaines étiquettes de l'analyseur morpho-syntaxique de la plateforme LIMA ne font pas partie des étiquettes du Penn TreeBank. Il faut donc les remplacer dans le fichier « wsj_0010_sentence.pos.lima » avant de faire les deux évaluations.

- SCONJ => CC
- SENT => .
- COMMA => ,
- COLON => :

## III. Installation et évaluation de l'outil de désambiguïsation morpho-syntaxique de l'université de Stanford

**Informations sur l'outil : https://nlp.stanford.edu/software/tagger.shtml**

### 1. Installation

a. Installer Java sur Ubuntu 16.04 (Les outils TAL de Stanford nécessite Java 1.8+) :
    i. Mettre à jour les paquets courants : `apt-get update`
    ii. Installer la dernière version du Java Runtime Environment (JRE): `apt-get install default-jre`
    iii. Installer Java Development Kit (JDK): `apt-get install default-jdk`

    Note : Vérifier la version de Java à l'aide de : `java –version, javac –v`

b. Télécharger la basic English Stanford Tagger version 3.9.2 (stanford-postagger-2018-10-16.zip)
c. Décompresser le fichier téléchargé: `unzip stanford-postagger-2018-10-16.zip`
d. Analyser le texte du fichier « simple-input.txt) :

```
cd stanford-postagger-2018-10-16
```

```
./stanford-postagger.sh models/english-left3words-distsim.tagger sample-input.txt > sample-input.txt.pos
```

**Exemple :**

Le fichier "Sample_eng.txt" contient la phrase "John ate dicious pizza with friends."

```
./stanford-postagger.sh models/english-left3words-distsim.tagger
Sample_eng.txt > Sample_eng.txt.pos
```

Stanford POS tagger output (Fichier Sample_eng.txt.pos):

John_NNP ate_VBD dicious_JJ pizza_NN with_IN friends_NNS ._.

## 2. Evaluation

a. Lancer le POS tagger sur le fichier « wsj_0010_sample.txt » : `./stanford-postagger.sh models/english-left3words-distsim.tagger wsj_0010_sample.txt > wsj_0010_sample.txt.pos.stanford`
b. Calculer la précision de ce POS tagger en utilisant les étiquettes PTB : `python evaluate.py wsj_0010_sample.txt.pos.stanford wsj_0010_sentence.pos.ref`
c. Calculer la précision de ce POS tagger en utilisant les étiquettes universelles : `python evaluate.py wsj_0010_sample.txt.pos.univ.stanford wsj_0010_sample.txt.pos.univ.ref`
d. Quelles conclusions peut-on avoir à partir de ces deux évaluations ?

# IV. Installation et utilisation de l'outil de reconnaissance d'entités nommées de l'université de Stanford

**Informations sur l'outil : https://nlp.stanford.edu/software/CRF-NER.html**

## 1. Installation

e. Installer Java sur Ubuntu 16.04 (Les outils TAL de Stanford nécessite Java 1.8+) :
   i. Mettre à jour les paquets courants : `apt-get update`
   ii. Installer la dernière version du Java Runtime Environment (JRE): `apt-get install default-jre`
   iii. Installer Java Development Kit (JDK): `apt-get install default-jdk`

   Note : Vérifier la version de Java à l'aide de : `java –version, javac –v`
f. Télécharger la Stanford Named Entity Recognizer version 3.9.2 (stanford-ner-2018-10-16.zip)
g. Décompresser le fichier téléchargé: `unzip stanford-ner-2018-10-16.zip`
h. Analyser le texte du fichier « simple-input.txt) :

```
cd stanford-ner-2018-10-16
```

```
java -mx600m -cp stanford-ner.jar:lib/*
edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier
classifiers/english.all.3class.distsim.crf.ser.gz -textFile
Sample_NER_en.txt > Sample_NER_en.txt.output
```

**Exemple :**

Le fichier " Sample_NER_en.txt" contient la phrase "John ate dicious pizza with friends."

```
java -mx600m -cp stanford-ner.jar:lib/*
edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier
classifiers/english.all.3class.distsim.crf.ser.gz -textFile
Sample_NER_en.txt > Sample_NER_en.txt.output
```

Stanford NE recognizer output (Fichier Sample_NER_en.txt.output):

John/PERSON ate/O dicious/O pizza/O with/O friends/O ./O

## 2. Extraction d'entités nommées

A partir du résultat de l'outil de reconnaissance des entités nommées « wsj_0010_sample.txt.ner.stanford », écrire un programme Python permettant de représenter les entités nommées sous le format suivant :

Entité nommée Type Nombre d'occurrences Proportion dans le texte (%)

**Exemple :**

| **Entité nommée** | **Type** | **Nombre d'occurrences** | **Proportion dans le texte (%)** |
|---|---|---|---|
| John | PERSON | 1 | 1 (1/1) |

# V. Evaluation des outils de reconnaissance d'entités nommées du CEA List et l'université de Stanford

- Etablir une table de correspondances entre les étiquettes des entités nommées des outils du CEA List et l'université de Stanford.
- Ecrire un programme Python permettant de transformer le résultat de l'outil de reconnaissance d'entités nommées du CEA List en utilisant les étiquettes des entités nommées de l'outil l'université de Stanford.
- Ecrire un programme Python permettant de transformer les résultats des outils de reconnaissance d'entités nommées du CEA List et l'université de Stanford pour pouvoir utiliser le script d'évaluation « evaluate.py ».
- Evaluer les outils de reconnaissance d'entités nommées du CEA List et l'université de Stanford sur un corpus de référence.

## Extensions

### 1. Evaluation à grande échelle des outils du CEA List LIMA et Stanford CoreNLP pour les deux tâches POS tagging et NE recognition

- Ecrire un programme Python permettant de transformer une référence pour le POS tagging en étiquettes universelles.
- Ecrire un programme Python permettant de transformer une référence pour le NE recognition en étiquettes CoNLL-2003 (https://www.clips.uantwerpen.be/conll2003/ner/).
- Comparer les résultats des deux plateformes sur le texte ayant servi à produire les références.

**Note :** J'enverrai pour ceux qui souhaitent réaliser cette extension du projet les données nécessaires pour cette évaluation.

### 2. Evaluation du système de traduction statistique Moses sur un corpus de spécialité

**Informations sur Moses : http://www.statmt.org/moses/index.php?n=Main.HomePage**

**Guide d'installation : http://www.statmt.org/moses/?n=Moses.Baseline**

#### 1. Installation

a. Installer les packages logiciels nécessaires à la compilation et l'installation de Moses (git, Boost, etc.).
b. Installer Moses.
c. Installer GIZA++ et MGIZA.
d. Installer IRSTLM.

#### 2. Evaluation

a. Récupérer le corpus parallèle Europarl (http://opus.nlpl.eu/Europarl.php) pour le domaine général, Emea (http://opus.nlpl.eu/EMEA.php) pour le domaine médical et Ecb (http://opus.nlpl.eu/ECB.php) pour le domaine bancaire.
b. Evaluer Moses en utilisant le score BLEU.

**Note :** J'enverrai pour ceux qui souhaitent réaliser cette extension du projet les données nécessaires pour cette évaluation.