POLYTECH PARIS-SUD

Et5-Info

Module : Traitement Automatique des Langues - Traduction Automatique

TP 2 - Analyse linguistique avec la plateforme CEA LIST Lima

Contexte :

La traduction automatique à base de corpus des textes des langues morphologiquement riches nécessite un pré-traitement impliquant au minimum une analyse morpho-syntaxqiue des données d'apprentissage. L'objectif de ce TP est l'installation et l'expérimentation de deux plateformes d'analyse linguistique nécessaires à la préparation des corpus d'apprentissage des modèles de traduction statistique Moses et neuronal OpenNMT.

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.

<u>Travail demandé</u>

Vous allez installer et expérimenter la plateforme d'analyse linguistique du CEA LIST Lima (une boite à outils linguistiques à base de règles).

I. Installation de la plateforme d'analyse linguistique open source LIMA

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-2.1.202001241207530100-9e12819-Ubuntu16.04-x86_64.deb)
    b. Installer LIMA: `sudo dpkg -i lima-2.1.202001241207530100-9e12819-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-U (https://universaldependencies.org/format.html).

Exemple:

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

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

```
# global.columns = = ID      FORM    LEMMA   UPOS    XPOS    FEATS   HEAD    DEPREL  DEPS    MISC
# sent_id = 1
# text = John ate dicious pizza with friends.
1       John    John    PROPN   _       _       2       SUJ_V   _       Pos=1|Len=4
2       ate     eat     VERB    _       _       _       _       _       Pos=6|Len=3
3       dicious dicious NOUN    _       _       4       ADJPRENSUB      _       Pos=10|Len=7
4       pizza   pizza   NOUN    _       _       2       COD_V   _       Pos=18|Len=5
5       with    with    ADP     _       _       6       PREPSUB _       Pos=24|Len=4
6       friends friend  NOUN    _       _       4       COMPDUNOM       _
        Pos=29|Len=7|SpaceAfter=No
7       .       .       SENT    _       _       _       _       _       Pos=36|Len=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 -p main 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" et lancer à nouveau LIMA sur le fichier « wsj_0010_sample.txt » et observer les sorties produites : `analyzeText -l eng -p main wsj_0010_sample.txt`
e. Rediriger le résultat d'analyse vers le fichier « wsj_0010_sample.txt.conllu » : `analyzeText -l eng -p main wsj_0010_sample.txt > wsj_0010_sample.txt.conllu`

2. Extraction d'entités nommées

A partir des sorties de l'analyseur LIMA « wsj_0010_sample.txt.se.xml » (ou « wsj_0010_sample.txt.conllu »), 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)

3. Analyse morpho-syntaxique

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

Mettre le résultat de la transformation (application de ce programme Python) de la sortie de l'analyseur LIMA « wsj_0010_sample.txt.disambiguated.xml » ou « wsj_0010_sample.txt.conllu » dans le fichier « wsj_0010_sample.txt.pos.lima ».

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 représentation 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 ._.

Après l'application du programme Python sur cet exemple, nous aurons la nouvelle représentation ci-dessous :

When WRB

it PRP

's VBZ

time NN

for IN

…

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.pos.ref`)

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

   Notes:
   - 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.
   - Pour calculer la précision de l'analyseur morpho-syntaxique de la plateforme LIMA à l'aide du script Python evaluate.py, il faut que les tokens (1ère colonne) des fichiers wsj_0010_sample.txt.pos.lima et wsj_0010_sample.pos.ref soient identiques. Pour ce faire, il faut écrire un programme Python qui permet de ne garder que les tokens identiques dans les deux fichiers wsj_0010_sample.txt.pos.lima et wsj_0010_sample.pos.ref.
   - Vous pouvez essayer ces programmes Python sur des textes de petite taille comme ceux des fichiers wsj_0010_sentence.pos.lima et wsj_0010_sentence.pos.ref.

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 => :