

Et5-Info

Module : Traitement Automatique des Langues

Année universitaire : 2021-2022

Projet

Consignes générales

- Vous avez le choix entre deux sujets. Le sujet 1 correspond à une analyse et une évaluation à grande échelle des deux plateformes d'analyse linguistique étudiées lors des TP1 et TP2. Le sujet 2 a pour objectif l'implémentation d'un désambiguateur morpho-syntaxique en utilisant un réseau de neurones récurrent de type LSTM (Long short-term memory). Ce sujet exige des compétences théoriques et pratiques sur les réseaux de neurones récurrents et le framework PyTorch.
- 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 ou en Word et compilé en un PDF de 5 à 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.
 4. Un dossier **data** pour les données manipulées. Veillez à utiliser les mêmes noms de fichiers utilisés en TPs et en projet.
 5. Un dossier **tp** pour les codes source de vos scripts Python des deux TPs. Il faut ajouter à ce dossier un **README** contenant les indications nécessaires pour exécuter ces scripts Python.

Vous m'enverrez par mail (nasredine.semmar@cea.fr) un lien vers votre projet **au plus tard le vendredi 25 février 2022.**

Sujet 1 : Evaluation de deux plateformes open source d'analyse linguistique

Travail demandé

Vous allez évaluer les deux plateformes d'analyse linguistique que vous avez utilisées en TP: Stanford Core NLP et NLTK. 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.

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

NLTK : est une boîte à outils linguistiques utilisant des approches hybrides combinant l'apprentissage automatique et des ressources linguistiques.

Remarques :

1. Pour pouvoir répondre à la 2ème question de la partie I (Evaluation de l'analyse morpho-syntaxique) vous aurez besoin de la table de correspondance entre les POS tags de la plateforme LIMA et les POS tags PTB (POSTags_LIMA_PTB_Linux.txt) ainsi que la table de correspondance entre les POS tags PTB et les POS tags Universels (POSTags_PTB_Universal_Linux.txt).
2. Le script « evaluate.py » ne fonctionne que si les deux fichiers à comparer ont les mêmes entrées (la 1ère colonne doit être la même), c'est-à-dire, le fichier résultat et le fichier de référence doivent avoir les mêmes mots sur la colonne de gauche.

I. Evaluation de l'analyse morpho-syntaxique

1. Utiliser le corpus annoté « pos_reference.txt.lima » pour extraire les phrases ayant servi pour produire ce corpus annoté et sauvegarder le résultat dans le fichier « pos_test.txt ». ***Dans ce corpus, une ligne vide indique la fin de la phrase courante.***
2. Convertir les tags du corpus annoté « pos_reference.txt.lima » en tags universels et sauvegarder le résultat dans le fichier « pos_reference.txt.univ ». Il faut utiliser les deux tables de correspondance « POSTags_LIMA_PTB_Linux.txt » et « POSTags_PTB_Universal_Linux.txt » pour réaliser cette transformation (LIMA→PTB→Universal).
3. Lancer les deux POS taggers sur le fichier « pos_test.txt ». Les résultats doivent avoir le format du corpus annoté « pos_reference.txt.lima » (2 colonnes séparées par une tabulation) et doivent être sauvegardés respectivement dans les fichiers suivants :

```
pos_test.txt.pos.stanford
pos_test.txt.pos.nltk
```
4. Convertir les résultats des deux POS taggers en utilisant les étiquettes universelles (Annexe 1). Il faut utiliser la table de correspondance « POSTags_PTB_Universal_Linux.txt » pour réaliser cette transformation (PTB→Universal). Les résultats de cette conversion doivent être sauvegardés respectivement dans les fichiers suivants :

```
pos_test.txt.pos.stanford.univ
pos_test.txt.pos.nltk.univ
```

5. Lancer l'évaluation des deux POS taggers. Pour réaliser cette évaluation, il faut supprimer la ligne vide (séparant les phrases) dans le fichier de référence « pos_reference.txt.univ » :

```
python evaluate.py pos_test.txt.pos.stanford.univ pos_reference.txt.univ
python evaluate.py pos_test.txt.pos.nltk.univ pos_reference.txt.univ
```
6. Quelles conclusions vous pouvez avoir à partir des résultats d'évaluation des deux POS taggers.

II. Evaluation de la reconnaissance d'entités nommées

1. Utiliser le corpus annoté « ne_reference.txt.conll » pour extraire les phrases ayant servi pour produire ce corpus annoté et sauvegarder le résultat dans le fichier « ne_test.txt ». ***Dans ce corpus, une ligne vide indique la fin de la phrase courante.***
2. Lancer les deux NE recognizers sur le fichier « ne_test.txt ». Les résultats doivent avoir le format du corpus annoté « ne_reference.txt.conll » (2 colonnes séparées par une tabulation) et doivent être sauvegardés respectivement dans les fichiers suivants :

```
ne_test.txt.ne.stanford
ne_test.txt.ne.nltk
```
3. Convertir les résultats des deux NE recognizers en utilisant les étiquettes CoNLL-2003 (<https://www.clips.uantwerpen.be/conll2003/ner/> -Annexe 2-). Les résultats de cette conversion doivent être sauvegardés respectivement dans les fichiers suivants :

```
ne_test.txt.ne.stanford.conll
ne_test.txt.ne.nltk.conll
```
4. Lancer l'évaluation des deux NE recognizers. Pour réaliser cette évaluation, il faut supprimer la ligne vide (séparant les phrases) dans le fichier de référence « ne_reference.txt.conll »:

```
python evaluate.py ne_test.txt.ne.stanford.conll ne_reference.txt.conll
python evaluate.py ne_test.txt.ne.nltk.conll ne_reference.txt.conll
```
5. Quelles conclusions vous pouvez avoir à partir des résultats d'évaluation des deux NE recognizers.

Annexe 1 : Correspondances entre les étiquettes du Penn TreeBank et les étiquettes universelles pour la désambiguïsation morpho-syntaxique.

Etiquette Penn TreeBank	Description	Etiquette Universelle
CC	conjunction, coordinating	CONJ
CD	cardinal number	NUM
DT	determiner	DET
EX	existential there	DT
FW	foreign word	X
IN	conjunction, subordinating or preposition	ADP
JJ	adjective	ADJ
JJR	adjective, comparative	ADJ
JJS	adjective, superlative	ADJ
LS	list item marker	X
MD	verb, modal auxiliary	VERB
NN	noun, singular or mass	NOUN
NNS	noun, plural	NOUN
NNP	noun, proper singular	NOUN
NNPS	noun, proper plural	NOUN
PDT	predeterminer	DET
POS	possessive ending	PRT
PRP	pronoun, personal	PRON
PRPDOL	pronoun, possessive	PRON
RB	adverb	ADV
RBR	adverb, comparative	ADV
RBS	adverb, superlative	ADV
RP	adverb, particle	PRT
SYM	symbol	X
TO	infinitival to	PRT
UH	interjection	X
VB	verb, base form	VERB
VBZ	verb, 3rd person singular present	VERB
VBP	verb, non-3rd person singular present	VERB
VBD	verb, past tense	VERB
VBN	verb, past participle	VERB
VBG	verb, gerund or present participle	VERB
WDT	wh-determiner	DET
WP	wh-pronoun, personal	PRON
WPDOL	wh-pronoun, possessive	PRON
WRB	wh-adverb	ADV
.	punctuation mark, sentence closer	.
,	punctuation mark, comma	,
:	punctuation mark, colon	:
(contextual separator, left paren	(
)	contextual separator, right paren)

Annexe 2 : Etiquettes CoNLL-2003 pour la reconnaissance es entités nommées

O: Words that are not named entities and referred to as 'Other'.

B-PERS: Beginning of Person Name.

I-PERS: Inside of Person Name.

B-ORG: Beginning of Organization Name.

I-ORG: Inside of Organization Name.

B-LOC: Beginning of Location Name.

I-LOC: Inside of Location Name.

B-MISC: Beginning of MiscellaneousWord.

I-MISC: Inside of MiscellaneousWord.

Sujet 2 : Un désambiguïseur morpho-syntaxique (Part-of-Speech tagger) basé sur des réseaux de neurones récurrents (LSTM)

Travail demandé

Vous allez utiliser une LSTM pour implémenter un POS tagger. Le fonctionnement du modèle neuronal est décrit comme suit :

Supposons que nous avons une phrase en entrée composée de M mots : w_1, \dots, w_M ou $w_i \in V$ (Vocabulaire). Supposons que T correspond à l'ensemble des étiquettes morpho-syntaxique et y_i est l'étiquette du mot w_i . Supposons que la prédiction de l'étiquette morpho-syntaxique du mot w_i est \hat{y}_i . La sortie du modèle est une séquence $\hat{y}_1, \dots, \hat{y}_M$ où $\hat{y}_i \in T$.

Pour réaliser la prédiction, il faut passer une LSTM sur la phrase en entrée. L'état caché au timestep i est représenté par h_i . Chaque étiquette morpho-syntaxique lui est assignée un seul index. La prédiction pour l'étiquette de \hat{y}_i est donnée par la formule suivante :

$$\hat{y}_i = \operatorname{argmax}_j (\log \operatorname{Softmax}(Ah_i + b))_j$$

L'étiquette morpho-syntaxique prédite correspond à l'étiquette ayant la valeur maximale dans le vecteur.

Documentation sur les LSTMs en Pytorch :

https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

Documentation sur les word embeddings :

https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html

Installation du framework PyTorch

1. Environnement Windows 10
 - a. Installer Anaconda : <https://mrmint.fr/installer-environnement-python-machine-learning-anaconda>
 - b. Lancer Anaconda en tant qu'administrateur :
 - i. Installation du Driver CUDA : `conda install cudatoolkit=11.0 cudnn`
 - ii. Installation de PyTorch : `conda install pytorch torchvision -c pytorch`
2. Environnement Linux (Ubuntu 18.04 LTS)
 - a. Installer Anaconda : <https://www.digitalocean.com/community/tutorials/how-to-install-anaconda-on-ubuntu-18-04-quickstart-fr>
 - b. Créer et activer un environnement Conda:
 - i. Création: `conda create -n env_pytorch python=3.7`
 - ii. Activation : `conda activate env_pytorch`
 - c. Installer PyTorch : `pip install torchvision`

I. Construction d'un POS tagger basé sur une LSTM à partir d'un petit corpus d'apprentissage

1. Préparation de données
 - a. Editer le programme «sample_lstm_pos_tagging_preparing_data.py» et identifier les principales étapes pour préparer les données d'apprentissage.
 - b. Lancer le programme «sample_lstm_pos_tagging_preparing_data.py» dans un environnement Pytporch et analyser les sorties.
2. Création du modèle neuronal
 - a. Editer le programme «sample_lstm_pos_tagging_training_model» et identifier les principales étapes pour créer le modèle neuronal.
 - b. Lancer le programme «sample_lstm_pos_tagging_training_model» dans un environnement Pytporch et analyser les sorties.

II. Amélioration du POS tagger basé sur une LSTM pour prendre en compte un gros corpus d'apprentissage

3. Construire à partir du corpus annoté « pos_reference.txt.lima » les données d'apprentissage (80% du corpus annoté) et les données de test (20% du corpus annoté).
4. Modifier le programme de préparation de données «sample_lstm_pos_tagging_preparing_data.py» pour prendre les nouvelles données d'apprentissage (80% du corpus annoté).
5. Modifier le programme de création et d'entraînement du modèle neuronal «sample_lstm_pos_tagging_preparing_data.py» pour prendre en compte les nouvelles données d'apprentissage (80% du corpus annoté) et les données de test (20% du corpus annoté).

III. Amélioration du POS tagger basé sur une LSTM en ajoutant une représentation basée sur des caractères

Dans l'exemple précédent, chaque mot avait un embedding qui a servi d'entrée au modèle neuronal. L'objectif de cet exercice est d'augmenter ces word embeddings avec une représentation basée sur des caractères.

Supposons que c_w est la représentation basée sur des caractères du mot w et que x_w est le word embedding, l'entrée du modèle neuronal sera la concaténation de x_w et c_w .

Pour obtenir la représentation au niveau du caractère, il faut passer une LSTM sur les caractères d'un mot et considérer w_c comme l'état caché final de cette LSTM.