mini-projet pour le cours d'enrichissement de corpus

Sommaire

Vous retrouverez tout mes fichiers sur mon GitHub :
github.com/deboraptor/annotation_automatique

Introduction

Je vous conseille de lire tout d'abord le README.md pour comprendre comment utiliser et exécuter les programmes pythons.

On va chercher à faire une annotation automatique sur deux corpus différents. Le premier aura été extrait de la page wikipédia de l'acteur britannique Henry Cavill : https://fr.wikipedia.org/wiki/Henry_Cavill

Et le deuxième est une sélection des derniers tweets trouvé sur la page https://twitter.com avec le filtre de recherche "langage : français" et les mots clés "Henry Cavill".

Dans un deuxième temps, nous ferons l'évaluation de ces modèles avec les mesure de précision, rappel et f-mesure. Pour ce faire, il a fallu que je fasse une annotation manuel des résultats. Nous discuterons de ces résultats plus en détails à la fin.

Notez que j'ai essayé plusieurs modules python et que ça s'est terminé en échec. Nous parlerons aussi de tout cela dans ce document.

Premiers pas

Au premier regard, quand l'annotation est faite automatiquement on remarque qu'il y a des ponctuations et des espaces qui correspondent aux retours chariot. Je me suis posée la question de savoir s'il fallait faire un pré-traitement et j'ai essayé de le faire avec des regex pour enlever la ponctuation mais ce n'était pas concluant.

J'ai voulu faire un affichage graphique plus intéressant qu'un simple retour dans un terminal. J'ai donc pris le parti de faire une page html avec BeautifulSoup pour le gérer.

J'ai choisi de faire le test pour deux corpus différents : un normé et un non-normé. Pour chaque corpus, je me suis contraintes à plus de 500 mots.

Le corpus normé est extrait de la page Wikipédia sur Henry Cavill. Pour le corpus non-normé, j'ai choisi de prendre des tweets sur le site twitter. Je les ai extrait à la main puis mis bout à bout dans un fichier texte exploitable. J'ai trié les tweets de manière à enlever ceux qui avaient du contenu explicite (évidemment, le premier tweet que j'ai eu en contenait..) et j'ai retiré les émojis encodé en unicode qui posaient un problème d'encodage.

SpaCy

Observation des résultats de SpaCy

Texte normé, wikipédia

Le texte ici est normé, c'est à dire qu'il répond aux exigences de la grammaire et de la conjugaison. Ce sont donc des mots qui apparaissent dans le dictionnaire et qui sont déjà connu de la machine. Les mots qui pourront poser problème sont donc les mots qui n'apparaissent pas dans celui-ci comme les noms propres et en effet, ce sont eux qui ont eu le plus de mal pour être correctement annotés.

les références, qui sont collées aux mots (romaine2, Cavill”3)

Texte non normé, tweets

Comme on pouvait s’y attendre, les résultats avec SpaCy sont moins concluants avec un texte non normé. En effet, nous pouvons observer :

- des fautes d’orthographe : geniale, synchroniser
- des hashtags : #TheMinistryofUngentlemanlyWarfare, #GuyRitchie
- des abréviations type SMS : pcq, tjs
- des onomatopées : krr, bah
- des expressions populaires chez les jeunes : fav, claqué, j’suis, crush
- des anglicismes : reboot, hyper, acting
- des émojis en ASCII : o_o

SpaCy a bien reconnu certains mots grâce à leur position dans la phrase. Par exemple, dans “c’est vraiment mon crush” le mot crush a bien été catégorisé par un nom car il est précédé par un déterminant. C’est aussi le cas pour “l’ont fancast” qui a bien été catégorisé en tant que verbe. Cependant, l’émoji “o_o” a été catégorisé en verbe, ce qui n’est pas du tout le cas.

Nous reviendrons plus en détail sur comment j’ai décidé de les annoter lors de mon annotation manuelle.

Annotation manuelle

Texte normé, wikipédia

Texte non normé, tweets

J’ai dû faire un choix pour les résultats et mon choix s’est scindé en deux catégorie que je vais donc expliciter ci-dessous.

- pour les hashtags, émojis et les onomatopées, j’ai décidé de les annoter en tant que X.
- les abréviations, les fautes d’orthographes, les anglicisme et les expressions populaires chez les jeunes seront annotées par la catégorie du mot qu’elles représentent.
    + bah annoté en tant qu’interjection (INTJ).

Stanza

J’ai voulu installer le module lorsque….

ERROR: Could not install packages due to an OSError: [Errno 28] No space left on device

Je n’ai plus de place sur ma partition, et comme c’est sur Linux je n’avais jamais géré ça. J’ai donc dû me renseigner pour ajouter de l’espace à ma partition.

NLTK

J'ai commencé par regarder le module NLTK. J'avais immédiatement remarqué qu'il fallait importer énormément de modules différents, ce qui peut porter à confusion.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.parse.corenlp import CoreNLPServer
from nltk.parse.corenlp import CoreNLPParser
```

Ensuite, il faut télécharger des ressources en mettant la commande sur Python. C'est utile de ne le faire qu'une seule fois, mais si je dois partager mon code il vaut mieux le laisser. Or, si je le laisse, il télécharge les ressources à chaque exécution du script…

Globalement, tout se passait bien jusqu'à que je décide d'afficher aussi les DEP. Pour obtenir les dépendances avec NLTK, il est nécessaire d'ouvrir un serveur, ce qui s'avère assez compliqué si je voulais partager mon code pour qu'il fonctionne également sur votre machine.

En bref, NLTK a l'air très complet mais assez complexe et j'ai décidé de le laisser de côté pour ce devoir, mais je me suis promise de retravailler dessus une prochaine fois.

Tree Tagger

J'ai eu énormément de problèmes avec le module TreeTaggerWrapper. Tout d'abord, c'est un module à installer à la main, ce qui n'est pas si grave. Sauf que dans le cas de Tree Tagger, il y a plusieurs fichiers à installer, et il faut les piocher sur un site qui a passé l'âge. Cela aurait été plus pratique d'avoir un fichier zip avec tout le nécessaire plutôt que d'avoir à aller chercher.

Mon module n'a tout simplement jamais fonctionné, j'ai passé beaucoup de temps à essayer de régler le problème. Voici le message d'erreur qui ne cessait de s'afficher peut importe la solution que j'essayais d'apporter :

```
treetaggerwrapper.py:739: FutureWarning: Possible nested set at position 8
punct2find_re = re.compile("([^ ])([[" + ALONEMARKS + "])",
/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py:
2043: FutureWarning: Possible nested set at position 152 DnsHostMatch_re =
re.compile("(" + DnsHost_expression + ")",
/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py:
2067: FutureWarning: Possible nested set at position 409 UrlMatch_re =
re.compile(UrlMatch_expression, re.VERBOSE | re.IGNORECASE)
/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py:
2079: FutureWarning: Possible nested set at position 192 EmailMatch_re =
```

```
re.compile(EmailMatch_expression, re.VERBOSE | re.IGNORECASE) Traceback
(most recent call last): File
"/home/debora/Desktop/Cours/S2/enrichissement_de_corpus/dm_annotation_autom
atique/annotation_automatique/projet/dm_corpus_spacy.py", line 147, in
<module> tagger = treetaggerwrapper.TreeTagger(TAGLANG='fr',
TAGDIR="../modele_treetagger") File
"/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py
", line 1006, in __init__ self._set_tagger(kargs) File
"/home/debora/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper.py
", line 1106, in _set_tagger raise TreeTaggerError("TreeTagger parameter
file invalid: " + \ treetaggerwrapper.TreeTaggerError: TreeTagger parameter
file invalid: french.par
```

J'ai tenté plusieurs solutions et en voici un aperçu :

- Donner le chemin du dossier du module

```
tagger = treetaggerwrapper.TreeTagger(TAGLANG='fr',
TAGDIR="/home/devoir_iris/lib/python3.10/site-packages/treetaggerwrapper-2.
3.egg-info)
```

- Renommer les fichiers french.par et french-chunker.par par french-utf8.par selon ce que j'avais pu trouver dans des forums
- Installer des versions ultérieures pour vérifier la compatibilité

Comme j'utilise un environnement virtuel, je me suis dis que c'était possiblement la cause, et étant donné que je vous conseille également de faire ça, j'ai pensé qu'il vallait mieux laisser tomber TreeTagger.

Les métriques

Comment on s'y prend ?

Je penser au début qu'il vallait mieux donner le résultat en une liste de listes car ce format sera plus simple à traiter pour faire le rappel, la mesure de précision et la m-mesure avec le module sklearn. J'avais ensuite aplati les listes pour qu'elles deviennent une seule liste de valeurs unidimensionnelles avec la méthode chain() du module itertools. Seulement, en débuggant mon code j'ai trouvé une solution plus simple.

J'ai choisi de faire la f-mesure sur les données "Mot", "Lemme" et "POS" car regarder manuellement les DEP prendrait trop de temps et serait en plus assez difficile à faire.

Mon premier résultat

```
Précision : 0.5
Rappel : 0.5
F-mesure : 0.5
```

Je me suis rendue compte que si le premier élément de la liste était le même, évidemment que ça ne marcherait pas. Il faut alors que je compare uniquement les POS pour que ce soit plus simple. A la base j'avais 532 mots dans le corpus mais comme je devais annoter à la main le reste j'ai décidé de diviser par deux le corpus.

Deuxième tentative

```
Précision : 0.0
Rappel : 0.0
F-mesure : 0.0
```

J'ai réussi à avoir d'autres résultats en faisant simplement une liste de strings avec [ligne_token, ligne_lemme, ligne_pos, ligne_dep].

n tentative

En suite, je me suis rendue compte qu'il fallait que je mette average="binary" pour que ce soit True ou False, et là j'ai obtenu un tout autre résultat :

```
Précision : 1.0
Rappel : 1.0
F-mesure : 1.0
```

Finalement, une réussite

```
Texte non normé :
Précision : 0.9559748427672956
Rappel : 1.0
F-mesure : 0.977491961414791

Texte normé :
Précision : 0.9656699889258029
Rappel : 1.0
F-mesure : 0.9825352112676057
```

Comme j'ai réussi à avoir tout les types de résultats possibles et inimaginables, je me disais que j'allais bien y arriver un jour. J'ai alors compris que je n'avais pas une liste de tuples, mais bien une seule liste de tuples, répétée plusieurs fois, soit des centaines de listes uniques avec un seul tuples à cause de ça : [ligne_token, ligne_lemme, ligne_pos, ligne_dep]

J'ai donc ensuite créé une boucle pour analyser les résultats. J'ai considéré que mon annotation manuelle était 100% juste et j'ai comparé chaque éléments de chaque tuples : si l'élément n'est pas le même, on met un 0 parce qu'on considère que la machine s'est trompée et si c'est le même on met un 1.

Analyse des résultats

Conclusion

J'ai passé beaucoup de temps à régler des problèmes de module et l'affichage en html qui était automatique. Malgré tout cela, j'ai beaucoup apprécié faire ce devoir et j'ai appris beaucoup de choses en python et sur l'analyse automatique. Ce sont ces genres de projets qui poussent à aller chercher de plus en loin pour avoir un résultats optimal.

Malheureusement, j'ai été prise de court par le temps et il a pas mal de choses que j'aurais aimé approfondir et que j'ai laissé de côté pour primer la qualité à la quantité, notamment pour l'affichage en html et le nombre de modules python utilisé. Même si je n'ai pas utilisé les modules NLTK et TreeTagger dans ce devoir, je me suis beaucoup renseigné sur eux et je compte les approfondir plus tard.

Je suis donc au final assez satisfaite de ce que j'ai pu faire car j'ai appris énormement de choses et parce que le projet était très intéressant et ça me motive pour en faire d'autres de mon côté dans le futur.