Université de Montréal

IFT6285 (TALN) — Devoir7
Étiquetage morphosyntaxique avec NLTK

Contact :
Philippe Langlais
RALI/DIRO
Université de Montréal
+1 514 343 61 11 ext: 47494
felipe@iro.umontreal.ca
http://www.iro.umontreal.ca/~felipe/

dernière compilation : 3 novembre 2020 (22:26)

Contexte

NLTK est une plateforme bien connue du traitement des langues qui offre de nombreuses implémentations d'algorithmes faciles à utiliser ainsi qu'un accès à une cinquantaine de corpus populaires. Dans ce devoir, vous allez explorer les facilités d'étiquetage morpho-syntaxique (POS) offertes par cette plateforme et les comparerez avec celles de Spacy.

Données

Vous allez travailler sur un extrait du corpus Penn Tree Bank disponible via NLTK. Le Penn Tree Bank est une ressource payante distribuée par LDC (Linguistic Data Consortium) offrant des phrases manuellement arborées syntaxiquement et NLTK vous offre un accès gratuit à 3914 de ces phrases. Une fois NLTK installé (librairie et corpora), vous avez accès au corpus sous la forme d'une liste de phrases, chaque phrase étant représentée par une liste de tuples (mot, tag) :

```
from nltk.corpus import treebank
print(f'len {len(treebank.tagged_sents())}')
print(treebank.tagged_sents()[0])
```

dont l'exécution devrait produire :

```
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'),
 ('years', 'NNS'), ('old', 'JJ'), (',', ','), ('will', 'MD'), ... ]
```

Vous prendrez les 3000 premières phrases pour entraîner vos modèles, et les suivantes pour réaliser des tests.

À faire

1. installez NLTK et les données (au moins le Penn Treebank, 44è item de cette liste).
2. Prenez connaissance du package TaggerI de NLTK.
3. Utiliser la classe `CRFTagger` afin :
   — d'entraîner un tagger sur les données sur Penn Tree Bank,
   — de sauver ce modèle sur disque,
   — d'évaluer ce modèle (à l'aide de la fonction `evaluate`).

   Par défaut, la fonction utilisée pour extraire les features (voir ici) ne prend en considération que le mot en cours de prédiction. Proposez votre fonction de façon à prendre en considération le contexte (par exemple le mot avant). Quel est l'impact de votre fonction sur les performances de votre modèle ?
4. Utiliser la classe `BrillTagger` afin :
   — d'entraîner un modèle transformationnel en prenant comme étiqueteur par défaut un étiqueteur `RegexpTagger` (vous pouvez prendre celui proposé à même la documentation),
   — d'entraîner un modèle transformationnel en prenant comme étiqueteur par défaut un modèle crf que vous avez développé,
   — d'évaluer ces modèles. Vous étudierez l'impact de métaparamètres comme le nombre de règles à retenir ou les patrons de règles.

   Vous analyserez le type de règles apprises en fonction du modèle de base (`RegexpTagger` ou `CRFTagger`) utilisé. Vous étudierez si le tagger transformationnel appris peut servir à son tour de base à un autre tagger transformationnel (*tbl(tbl(base))*). En combien d'itérations ce processus (*tbl(tbl(...tbl(base))...)*) converge ?
5. Encapsuler le tagger de Spacy dans une classe héritée de `TaggerI` de façon à a) l'évaluer sur la partie de test du Penn Tree Bank et b) l'utiliser comme tagger de base dans l'approche transformationnelle. La propriété `tag_` d'un token Spacy contient une étiquette POS compatible avec le jeu d'étiquettes utilisé par défaut dans NLTK.

   Attention : la création par la fonction `nlp` d'un objet de type Doc dans Spacy entraîne le découpage de la phrase en mots. Ce découpage ne sera à priori pas compatible avec celui du Penn Tree Bank, ce qui

posera des problèmes lors de l'évaluation avec la fonction `evaluate` de TaggerI. Vous devez donc fonctionner à *tokenisation* fixe, c'est-à-dire celle imposée par le treebank de NLTK. Pour cela, vous pouvez vous inspirer de ce code :

```
import spacy

model = "en_core_web_sm"      # try also  the _lg one
nlp = spacy.load(model,
                 disable=["parser", "ner"]) # to go faster
# we want to do this:
# doc = nlp('hello world !')
#
# but the tokenization would change from the one in treebank
# which would cause problems with the function evaluate
# so instead  do this more convoluted thing:
tokens_of_my_sentence = ['hello', 'world', '!']
doc = spacy.tokens.doc.Doc(nlp.vocab, words=tokens_of_my_sentence)
for _, proc in nlp.pipeline:
   doc = proc(doc)

# now doc is ready:
for t in doc:
   print(f'{t.text:20s} {t.tag_}')
```

Observez (à l'aide de la fonction `evaluate`) la performance du tagger de Spacy sur le corpus de test du Penn Tree Bank. Tentez d'expliquer pourquoi les performances ne sont pas aussi élevées que celles indiquées ici.

Vous devez produire un **rapport** d'au plus 3 pages (format pdf, en français ou en anglais) qui contient les analyses des approches qui vous sont demandées.

Remise

La remise est à faire sur Studium sous le libellé `devoir7`. Vous devez remettre votre code, votre rapport (format pdf, texte en anglais ou en français) dans une archive (gzip, tar, tar.gz) dont le nom est préfixé de `devoir7-name1` ou `devoir7-name1-name2` selon que vous remettez seul ou a deux, où `name1` et `name2` sont à remplacer par l'identité des personnes faisant la remise (`prénom_nom`). Assurez vous que le nom des personnes impliquées dans le devoir soit indiqué sur tous les documents remis (code et rapport). Le devoir est à remettre en groupe d'au plus deux personnes au plus tard vendredi 13 novembre à 23h59.

Note : Aucun modèle n'est demandé.