ENSAI – Traitement Automatique du Langage

[TP2] Classification de documents et fouille d'opinion

L'objectif du TP est de réaliser et comparer plusieurs approches de classification de documents appliquées à la détection de polarité des opinions. On utilisera pour cela la base de données *Large Movie Review Dataset v1.0*[1] largement exploitée dans la littérature scientifique pour cette tâche. Nous avons déjà vu un extrait de cette base de données au TP précédent et mis en place un classifieur de type k plus proches voisins pour ces données.

On comparera deux approches au cours de cette séance : le classifieur des k plus proches voisins du TP précédent ; une approche exploitant le lexique de polarité SentiWordNet[2].

Préparation des données

La base de données comporte au total 50 000 commentaires de film extraits du site Internet Movie Database (IMDb), pour moitié positifs et pour moitié négatifs (pas de commentaires neutres). Elle est classiquement divisée en deux sous-ensembles équilibrés :

- un ensemble d'apprentissage de 25 000 commentaires (50 % positifs / 50 % négatifs)
- un ensemble de test de 25 000 commentaires (50 % positifs / 50 % négatifs)

Afin de limiter les temps de calcul, on utilisera pour le TP une version réduite de cette base de donnée en ne prenant que 5 000 exemples de chaque classe pour l'apprentissage et 2 000 de chaque pour le test. Ces données se trouvent respectivement dans les fichiers `imdb-trn-small.json` et `imdb-tst-small.json`[3]. Si vous avez du temps en fin de TP, vous pouvez lancer les traitements sur l'ensemble des données.

À faire :

1. Chargez les données d'apprentissage de `imdb-trn-small.json` et familiarisez vous avec ces données (*cf.* détails dans le texte du TP précédent).
2. Appliquez une chaîne de traitement `spaCy` de manière à *tokéniser* chaque entrée de la base, déterminer les étiquettes morphosyntaxiques des *tokens* et obtenir un plongement pour chacun des *tokens*. Lire les instructions ci-dessous avant de se lancer dans cette partie.

Important. Le traitement par `spaCy` de l'ensemble des données prend un peu de temps. Deux astuces permettent d'**accélérer ce traitement** par rapport à une application directe de `spaCy` sur chaque entrée : ne charger que les éléments pertinents de la chaîne de traitement ; utiliser la commande `pipe()` des pipelines `spaCy`. La commande ci-dessous donne un exemple de ces options pour traiter l'ensemble des textes de la liste `texts` (tableau où chaque entrée est un texte), en se limitant pour l'analyse du texte à la *tokénisation* et la détection des entités nommées. À vous de l'adapter pour vos besoins dans le cadre de ce TP pour ne conserver que les traitements nécessaires.

```
process = spacy.load("en_core_web_md")
for doc in process.pipe(texts, disable=["tagger", "parser"]):
    print([(ent.text, ent.label_) for ent in doc.ents])
```

Par ailleurs, il est important de **ne lancer l'analyse d'un texte qu'une et une seule fois**, et non à chaque fois que vous analysez le texte. On pourra par exemple compléter le tableau des données

[1] `http://ai.stanford.edu/~amaas/data/sentiment`
[2] `http://sentiwordnet.isti.cnr.it`
[3] Accessibles via `https://people.irisa.fr/Guillaume.Gravier/teaching/ENSAI/data`

d'apprentissage (colonne 1: classe, colonne 2: texte) en ajoutant une colonne contenant le résultat du traitement fait par `spaCy` de manière à pouvoir le réutiliser à tout moment.

Classifieur k plus proches voisins

On reprend tout d'abord les fonctions du TP précédent pour évaluer un classifier par k plus proches voisins (k=5) s'appuyant sur une représentation par *average word2vec* des textes. On limitera les textes aux seuls verbes et adjectifs (meilleure combinaison pour l'analyse de polarité) en considérant leur forme lémmatisée. Si vous n'avez pas eu l'occasion d'aborder la deuxième partie du TP1, c'est le moment d'écrire une fonction qui prend en entrée un document et qui renvoie le vecteur moyen des plongements des *tokens* du document – cf. TP1, dernière partie). On rappelle que si `X` (resp. `Y`) est une matrice regroupant l'ensemble des données d'apprentissage (resp. de test), chaque ligne étant un échantillon, la construction du classifieur et la classification se font à l'aide des instructions suivantes :

```
from sklearn.neighbors import NearestNeighbors
k = 5
neighbors = NearestNeighbors(n_neighbors=k).fit(X)
dist, idx = neighbors.kneighbors(Y)
```

La matrice `idx` contient pour chaque échantillon de `Y` les indices dans `X` des plus proches voisins, en ordre croissant de distance.

À faire :

3. En reprenant la fonction `avg_w2v` du TP précédent de manière à ne considérer que les lemmes correspondant aux adjectifs et aux verbes, construisez le classifieur au sens des 5 plus proches voisins (fonction `fit`).

4. Évaluez les performances du classifieur sur les données de test[4]

Utilisation d'un lexique sémantique

La deuxième méthode envisagée vise à utiliser SentiWordNet, un lexique sémantique de type WordNet dans lequel on associe à chaque *synset* des valeurs de polarités négative et positive. L'idée du système est donc de calculer pour chaque document un score de polarité positive et un score de polarité négative, en agrégeant (moyenne) les polarités associées à chacun des mots du document (éventuellement après filtrage par les catégories morphosyntaxiques) qui apparaissent dans SentiWordNet.

L'exemple suivant montre comment utiliser SentiWordNet avec `nltk`. Dans cet exemple, on récupère les `synsets` associés aux différents sens du mot *rat* en tant que verbe ou adjectif afin d'afficher leur polarité :

```
# load synsets for a word, possibly with tag specifications
# n=NOUN -- v=VERB -- a=ADJECTIVE

from nltk.corpus import sentiwordnet as swn

synsets = swn.senti_synsets('rat', 'av')
for item in synsets:
    print(item.synset.name(), item.pos_score(), item.neg_score(), item.obj_score())
```

[4] On devrait trouver dans les 77 % de classification correcte en prenant un vote majoritaire sur les *labels*.

À faire :

5. Écrivez une fonction `word_net_score` qui prend en entrée un document (tel que retourné par la chaîne de traitement `spaCy` mise en place en début de TP) et qui retourne les scores de polarités négative et positive. Le score de polarité positive (resp. négative) sera calculé comme le score positif (resp. négatif) moyen des adjectifs et verbes apparaissant dans un `synset`. Lorsqu'un mot possède plusieurs sens – *i.e.*, il apparaît dans plusieurs `synsets` – la polarité du mot sera donnée par la moyenne sur tous ses sens. On fera attention à se limiter aux sens correspondant à la catégorie morphosyntaxique du mot telle que déterminée par la chaîne de traitement (verbe ou adjectif) afin de limiter le nombre de sens.

6. Utilisez votre fonction `word_net_score` comme classifieur sur les données de test en utilisant une règle de décision simple, *i.e.*, text positif si le score moyen positif est supérieur au score moyen négatif.

Pour aller plus loin

Pour aller plus loin sur ce TP si vous le souhaitez, vous pouvez mettre en oeuvre d'autres approches de classification, .e.g., un classifieur bayésien naïf (qui marche assez bien sur ces données). Vous pouvez aussi remettre en cause les nombreux choix qui ont été faits : vous pouvez choisir les termes en fonction des scores de polarité dans SentiWordNet de manière à ne considérer que les termes les plus marqués du point de vue de la polarité ; combiner les différents classifieurs ; etc.

Dans les TP de TAL avancés pour les groupes qui suivent cette option, nous revisiterons ce corpus avec des approches neuronales.