Projet Master 1 informatique

Traitement Automatique du Texte en Intelligence Artificielle

Sujet

Analyse des sentiments - Implémentez un algorithme de classification des sentiments des tweets / critiques / messages, etc. en négatif / positif (jeu de données de SEMEVAL).

Université Nice Sophia Antipolis
Membre de UNIVERSITÉ CÔTE D'AZUR

Année 2018

Chloé Maccarinelli

Table des matières

A- Api twitter et python

1) Connexion à l'Api twitter

La première étape consiste à faire le lien entre notre programme et l'API twitter. Nous n'allons pas ici détailler le fonctionnement de l'API. L'accès à twitter est possible ici grâce à la librairie python *tweepy*. Ci-dessous, on retrouve le code pour l'accès à l'API et un petit test pour vérifier que tout fonctionne.

```
# coding: utf8
import tweepy


# information API twitter
consumer_key = '[redacted]'
consumer_secret = '[redacted]'
access_token = '[redacted]'
access_secret = '[redacted]'

# connexion à l'API twitter
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)

#affichage d'un tweet avec un id donnée pour tester l'accès à twitter
tweet = api.get_status('637859860017647616')
print(tweet.text)
```

Figure 1 : Accès à l'API twitter

Nous récupérons bien le contenu d'un tweet avec son identifiant ; l'accès est donc fonctionnel.

```
E:\PROGRAMMES\Python\python.exe E:/COURS/MASTER1/TATIA/Analyse_sentiments/download-tweets.py
#tv Ind vs SL 3rd Test Day 3: Cricket live score and Sony Six live streaming info: Watch the live teleca... http://t.co/mUlHw4cN00 #Sony
```

Figure 2 : tweet correspondant à l'id renseigné

2) Parcours de fichiers

Nous voulons à présent parcourir notre fichier *id_tweets.input.txt*, qui contient tous les identifiants de tweets et les afficher. Ce fichier provient de données SEMEVAL sur lesquelles nous travaillerons tout le projet.

```
#affichage des tweets avec les ids données dans notre fichier txt id_tweets.input.txt

with open("../Analyse_sentiments/tweeterData/input/devtest/id_tweets.input.txt", "r") as f:
    for line in f:
        if line != '\n':
            try:
                tweet = api.get_status(line)
                print(tweet.text)
            except Exception as e:
                pass
```

Figure 3: parcours des ids de tweets

On précise que si un identifiant de tweet ne correspond à aucun tweet (supprimé ou modifié) alors on passe à la ligne suivante. On obtient ainsi les tweets comme ci-dessous.

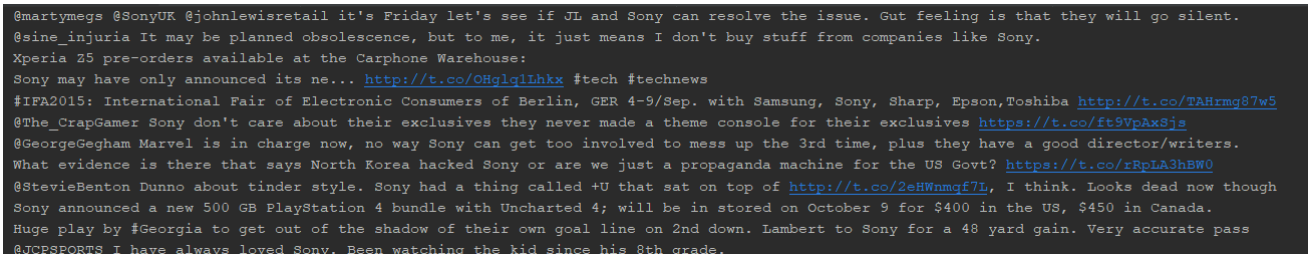

Figure 4 : affichage des tweets

B- Nettoyage des tweets

1) Le prétraitement de base

Avant toute analyse il faut traiter les tweets afin de ne garder que les parties significatives. Pour cela, on va pour chacun des tweets récupérer effectuer les modifications suivantes :

- Suppression des lien hypertextes
- Suppression des adresses mails
- Suppression des hashtags

Pour la suppression des liens hypertextes, on procède comme suit :

```
tweet = re.sub(r'https?:\/\/.*[\r\n]*', '', tweets.text)
```

Pour chaque tweet s il contient un « mot » commençant par http ou https alors on le supprime

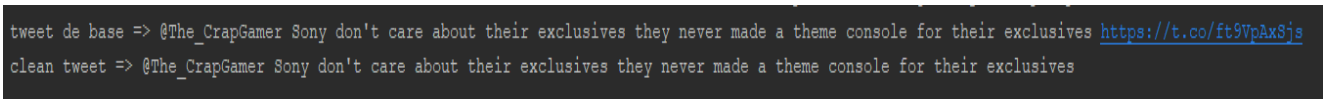

Figure 5: suppression du lien hypertexte

Pour la suppression des noms d'utilisateurs, tags ou adresses mails on effectue une recherche de tous les mots de la forme « @mot »

```
tweetat =re.sub('@[\w\.\-]+','',tweet_http)
```

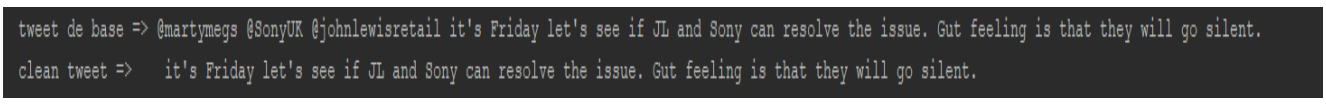

Figure 6 : supression des noms utilisateurs

Pour les hashtags c'est la même chose :

```
tweet = re.sub('#[\w\.\-]+', '', tweetat)
```

```
tweet de base => Sony's pulling a #TASM2 by beginning pre-production on a Goosebumps sequel even though the 1st one hasn't come out yet. #FuckingStupid
clean tweet => Sony's pulling a  by beginning pre-production on a Goosebumps sequel even though the 1st one hasn't come out yet.
```

Figure 7 : suppression des #

2) Le prétraitement lexical

Une fois que nous avons récupéré les tweets sans les mots « insignifiants » nous pouvons nous attaquer à la lemmatisation. C'est-à-dire que nous allons regrouper les mots d'une même famille et ainsi les réduire en lemme. Cette méthode va nous permettre de ne pas compter plusieurs mots de la même famille en plusieurs différents. Plus concrètement, pour les mots *aime* et *aimera*, on aura deux occurrences de *aime* et pas une occurrence de *aime* et une de *aimera*.

➔ Avant de s'attaquer à la lemmatisation, nous devons découper notre tweet en 'mots' ; c'est la tokenisation.

```
tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, reduce_len=True)
tweet_tokens = tokenizer.tokenize(tweet)
```

On obtient alors la liste suivante :

```
['a', 'good', 'read', 'about', "what's", 'happening', 'to', 'minecraft', 'in', 'schools', '...']
tweet de base => A good read about what's happening to Minecraft in schools...
```

Figure 8 : tokens

On lemmatise maintenant le groupe de mots obtenu par tokenisation

On remarque bien que *happening* est devenu *happen* et *schools*, *school.*

```
tweet de base => A good read about what's happening to Minecraft in schools
http://t.co/qjjB5HfnAw
clean tweet => a
clean tweet => good
clean tweet => read
clean tweet => about
clean tweet => what'
clean tweet => happen
clean tweet => to
clean tweet => minecraft
clean tweet => in
clean tweet => school
```

Figure 9 : lemmatisation d'un tweet

Après succession de toutes les modifications sur les tweets, on obtient une liste de mots comme ci-dessous :

```
tweet de base => #IFA2015: International Fair of Electronic Consumers of Berlin, GER 4-9/Sep. with Samsung, Sony, Sharp, Epson,Toshiba http://t.co/TAHrmg87w5
['intern', 'fair', 'of', 'electron', 'consum', 'of', 'berlin', 'ger', '4-9', 'sep', 'with', 'samsung', 'soni', 'sharp', 'epson', 'toshiba']
```

Figure 10 : liste de mots des tweets modifiés

Dans cette liste on remarque que certains mots sont inutiles pour une analyse de sentiments tels que les déterminants. Pour ne pas en tenir compte, la librairie NLTK propose une liste de « stopwords » qui regroupes tous ces mots « inutiles ».

C- Calcul des polarités

```
tweet de base => #IFA2015: International Fair of Electronic Consumers of Berlin, GER 4-9/Sep. with Samsung, Sony, Sharp, Epson,Toshiba http://t.co/TAHrmq87w5
['intern', 'fair', 'electron', 'consum', 'berlin', 'ger', '4-9', 'sep', 'samsung', 'soni', 'sharp', 'epson', 'toshiba']
```

Figure 11: liste de mots des tweets modifiés sans 'stopwords'

Une fois les traitements effectués sur les tweets il est temps de déterminer leur polarité.

Pour cela on peut utiliser deux méthodes :

- Classification des mots
- Apprentissage automatique

1) La classification des mots

1ère étape : récupérer les listes de mots positifs et négatifs

Ces deux listes proviennent de https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

2ème étape : comptabiliser le nombre de mots positifs et négatifs

Pour chaque token (voir étape de pré-traitement) on regarde s'il apparaît dans la liste des mots positifs ou négatifs.

Si le nombre de mots positifs est égal à celui de mots négatifs alors on considère le tweet comme neutre, s'il est supérieur le tweet sera positif sinon négatif.

On obtient alors une classification comme ci-dessous :
ici on a un tweet neutre puisque le nombre de mots positifs est égal à celui de mots négatifs.

```
tweet de base => #IFA2015: International Fair of Electronic Consumers of Berlin, GER 4-9/Sep. with Samsung, Sony, Sharp, Epson,Toshiba http://t.co/TAHrmq87w5
['intern', 'fair', 'electron', 'consum', 'berlin', 'ger', '4-9', 'sep', 'samsung', 'soni', 'sharp', 'epson', 'toshiba']
0
0
1
#IFA2015: International Fair of Electronic Consumers of Berlin, GER 4-9/Sep. with Samsung, Sony, Sharp, Epson,Toshiba http://t.co/TAHrmq87w5 neutral
```

Figure 12 : polarité des mots

Ces résultats sont stockés dans un fichier .txt afin de pouvoir comparer nos résultats avec les résultats déjà obtenus par SEMEVAL.

2) L'apprentissage automatique

Globalement le traitement des données est le même que pour la classification par mot mis à part qu'au lieu d'utiliser une liste de mots négatifs et positifs on utilise ici une liste de tweets positifs et négatifs (listes disponibles avec nltk.corpus). On crée donc nos données d'entrainement et de test de la manière suivante :

On import les listes de tweets positifs et négatifs :

```
pos_tweets = twitter_samples.strings('positive_tweets.json')
neg_tweets = twitter_samples.strings('negative_tweets.json')
```

Après traitement des tweets (tokenisation, lemmatisation, …) on crée nos listes de tweets positifs et négatifs et on mélange ces deux listes pour en faire des données d'entrainement et de test.

```
pos_tweets_set = []
for tweet in pos_tweets:
    pos_tweets_set.append((bag_of_words(tweet), 'positive'))

neg_tweets_set = []
for tweet in neg_tweets:
    neg_tweets_set.append((bag_of_words(tweet), 'negative'))

from random import shuffle

shuffle(pos_tweets_set)
shuffle(neg_tweets_set)


test_set = pos_tweets_set[:3000] + neg_tweets_set[:3000]
train_set = pos_tweets_set[3000:] + neg_tweets_set[3000:]
```

D- Analyse des résultats

1) La classification des mots

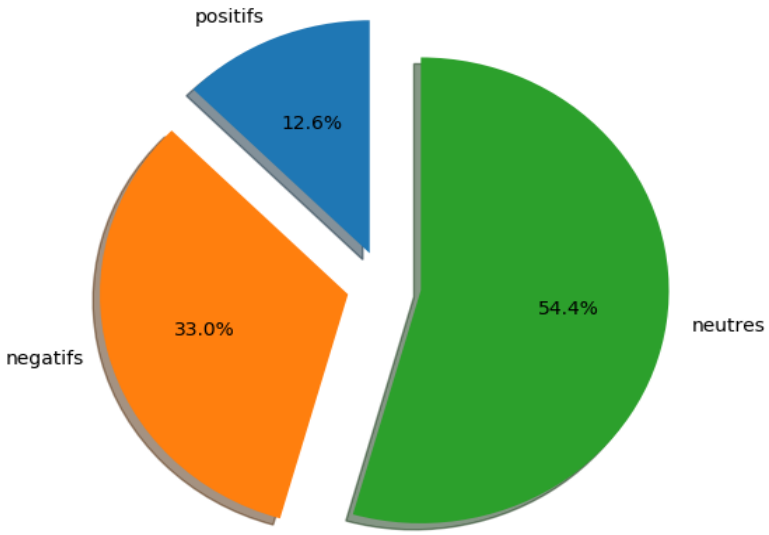

Figure 13 : classification des tweets

Le diagramme ci-dessus représente la classification des tweets effectuée avec notre algorithme et le diagramme ci-dessous celle des résultats que nous devrions obtenir.

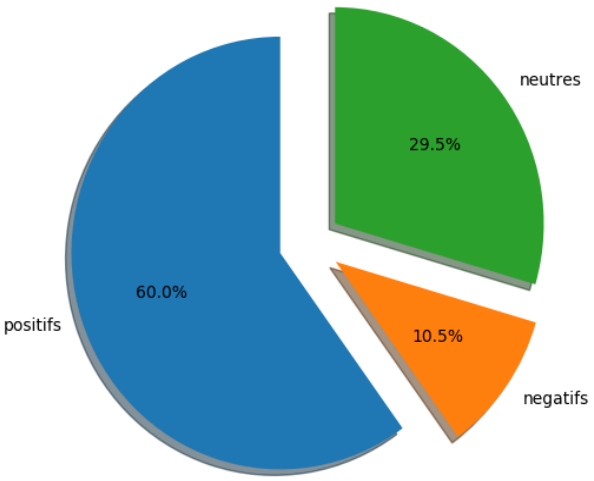

Figure 14 : classification des tweets

On compare donc les résultats obtenus et ceux attendus.

On récupère uniquement les données avec les identifiants identiques pour les résultats obtenus et attendus. On a au total 668 tweets à comparer.

Les tweets positifs : grâce à notre algorithme on trouve 64 tweets positifs identiques aux 401 attendus.

Les tweets négatifs : grâce à notre algorithme on trouve 159 tweets négatifs identiques aux 70 attendus.

Les tweets neutres : grâce à notre algorithme on trouve 117 tweets neutres identiques aux 197 attendus.

Pourquoi ?

On remarque que dans les résultats obtenus 209 tweets sont neutres alors qu'ils sont positifs dans les résultats attendus. En tenant compte de ce cas particulier (on passe les tweets positifs au nombre de 273) on obtiendrait alors les résultats suivants :

- Positifs : 68,08 %
- Négatifs : 44,02%
- Neutres : 59,39%

Globalement avec la méthode de classification des mots, on a uniquement une chance sur 2 d'avoir une polarité cohérente avec le tweet.

2) L'apprentissage automatique

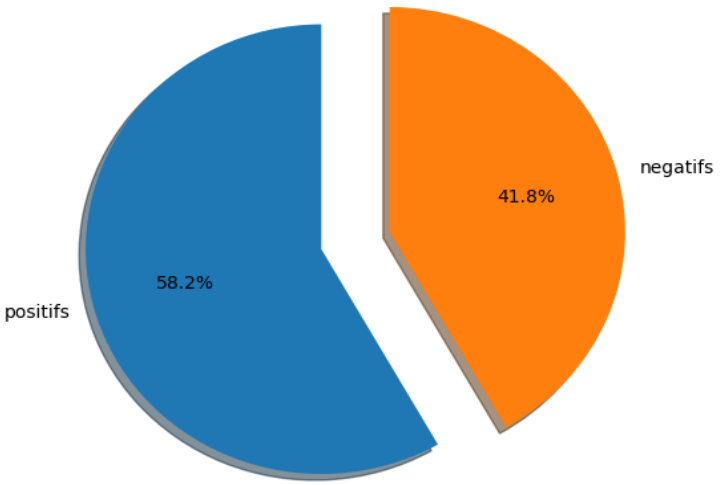

Figure 15 : classification des tweets

Le diagramme ci-dessus représente la classification des tweets effectuée avec notre algorithme et le diagramme ci-dessous celle des résultats que nous devrions obtenir.

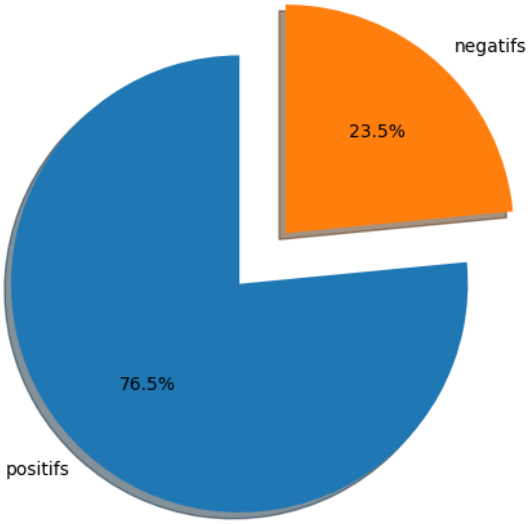

Figure 16 : classification des tweets

On compare donc les résultats obtenus et ceux attendus.

On récupère uniquement les données avec les identifiants identiques pour les résultats obtenus et attendus. On a au total 823 tweets à comparer.

Les tweets positifs : grâce à notre algorithme on trouve 388 tweets positifs identiques aux 629 attendus.

Les tweets négatifs : grâce à notre algorithme on trouve 102 tweets négatifs identiques aux 193 attendus.

On obtient alors les résultats suivants :

Positifs : 61,69 %

Négatifs : 52,85 %

Globalement avec la méthode d'apprentissage automatique on a environ une chance sur 2 d'avoir une polarité cohérente avec le tweet.

E- Classification avec des données de journaux

Jusqu'ici nous avons traité des données que l'on a récupérée directement via l'Api twitter. Nous allons donc tester nos algorithmes sur des textes issus de journaux.
Testons avec cet article :

« A man displayed a gun during an argument with a group of mostly Somali-American teenagers at a McDonald's in Minnesota after he wrongly suggested they were using welfare assistance to pay for their food, members of the group said.

The confrontation with the man, which was partly captured on a video that has been widely circulated on Twitter, occurred on Monday at a McDonald's restaurant in Eden Prairie, Minn., a suburb southwest of Minneapolis, and ended without any shots fired or injuries sustained.

On Wednesday, the Eden Prairie police arrested Lloyd Edward Johnson, 55, under probable cause for second-degree assault, the city said in a statement. Mr. Johnson was being held at the Hennepin County Adult Detention Center. Joyce Lorenz, a spokeswoman for the city, said on Thursday that the Hennepin County attorney would decide whether to file any charges. »

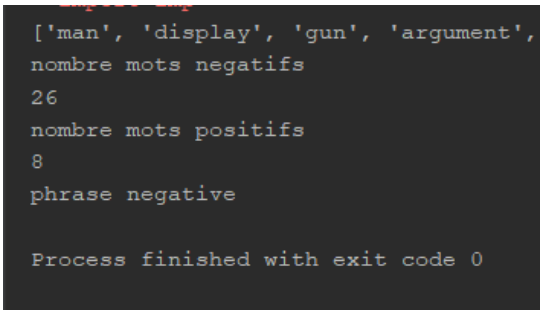

Figure 17 : résultat de la classification par mots

L'article précédent parle de l'arrestation d'un homme après avoir pointé une arme sur un adolescent. On comprend que le sentiment principal de l'article est négatif. Dans cet exemple nous voulons montrer principalement que dans un texte structuré et sans fautes d'orthographes, n'importe quel algorithme peut obtenir des classifications de sentiments assez correct.

F- Conclusion

L'analyse de données issues de twitter m'a posé de nombreux problèmes pour ce qui est de l'analyse et du traitement des textes avant la classification. Ces complications sont principalement dues au langage et à l'orthographe assez familière. Pour la classification avec apprentissage automatique nous pourrions avoir de meilleurs résultats en proposant encore plus de données d'entrainements. C'est d'ailleurs pour cette raison que nous utilisons les données d'entrainement proposées par la librairie nltk.corpus avec ses twitter.samples .