PFA :

L’analyse des sentiments via les réseaux sociaux

Réalisé par :

EL Outmadi Abderrahim

Encadré Par :

M.Laanya Hicham

Remerciements

Tout d'abord, on voudrait remercier le corps professoral et administratif de l'Institut National des Postes et Télécommunications, pour la qualité de l'enseignement offert et le soutien de l'équipe administrative.

On souhaite ensuite adresser nos remerciements notre professeur Monsieur Laanya Hicham , il a su nous faire confiance lors de cette aventure dans le monde de recherche et a partagé ses connaissances de manière très pédagogique. On le remercie aussi pour sa disponibilité et la qualité de son encadrement dans ces conditions.

Sommaire :

Introduction :

Nick Martin, expert mondial de l'engagement sur les médias sociaux chez Hootsuite, définit le sentiment sur les médias sociaux :

« Le sentiment sur les médias sociaux correspond à l'impression positive ou négative qui se dégage d'une publication ou d'une interaction. »

Contrairement aux idées reçues, toute publicité n'est donc pas bonne à prendre.

Les 10 000 publications dans lesquelles vous avez été identifié sur Twitter la semaine dernière vous ont peut-être donné le sourire, mais votre joie risque d'être de courte durée si ces mentions s'avèrent négatives.

La mesure ou plutôt l'analyse des sentiments sur les médias sociaux est un élément clé de tout programme de veille des médias sociaux. En effet, il vous permet de comprendre ce que ressent l'auteur d'une publication et peut vous fournir le contexte nécessaire pour agir et répondre.

Dans ce projet vous aurez besoin d'un compte développeur Twitter pour pouvoir utiliser le Twitter API.

Puisque nous avons maintenant nos informations de connexion pour les développeurs Twitter (c'est-à-dire les clés API et le jeton d'accès), nous pouvons procéder à l'authentification de notre programme. Tout d'abord, nous devons importer la bibliothèque Twitter, puis créer un objet Twitter API avec les informations d'identification dont nous avons parlé, comme suit:

```
twitter_api = twitter.Api(consumer_key='z6W3ZLobKsJmPePLOyzLT1eCf',
                          consumer_secret='Tp5o5NpZVkGoXWjIa9qQnbuYdjHymL5GMb73O100Yla4wGjkpx',
                          access_token_key='875363267048349697-cVuMnLdzBHGbzuWJclc1BVe3pFspKfk',
                          access_token_secret='0wJIesEYxucO1xMFzxFGX9kAvK8Cmkf0mqYRnSfkfPkaN')

print(twitter_api.VerifyCredentials())
```

La dernière instance n'est la que pour vérifier que le twitter API fonctionne . La valeur que compte twitter_API est quelque chose comme la réponse JSON suivante:

{"created_at": "Thu Jun 15 14:44:01 +0000 2017", "default_profile": true, "favourites_count": 14, "followers_count": 3, "friends_count": 29, "id": 875363267048349697, "id_str": "875363267048349697", "name": "ABDERRAHIM EL OUTMADI", "profile_background_color": "F5F8FA",.....}

Section A: Création de la fonction pour construire l'ensemble de Testing :

Nous pouvons maintenant commencer à créer une fonction qui import le TestSet. Fondamentalement, cela va être une fonction qui prend un mot-clé de recherche comme entrée, recherche les tweets qui incluent ce mot-clé et les renvoie sous forme d'objets twitter.

La mise en garde ici, cependant, est que Twitter limite le nombre de requêtes que vous pouvez faire via l'API à des fins de sécurité. Cette limite est de 180 demandes par fenêtre de 15 minutes.

Par souci de simplicité, nous limiterons la recherche à 100 tweets pour l'instant, sans dépasser le nombre de requêtes autorisé. Notre fonction de recherche des tweets (c'est-à-dire l'ensemble de test) sera:

```
def Construction_Du_TestSet(mot_clé):
    try:
        Tweet_récupéré = twitter_api.GetSearch(mot_clé, count=100)
        print(str(len(_Tweet_récupéré)) +" Tweets ont été récupéré contenant le mot :  " + mot_clé)
        return [{"text": status_Du_Tweet.text, "label": None} for status_Du_Tweet in _Tweet_récupéré]
    except:
        print("Erreur !!")
        return None

mot_clé = input("Saisir le mot clé : ")
Data_To_Use_For_Testing = Construction_Du_TestSet(mot_clé)
```

Comme vous vous en doutez, cette fonction renverra une liste de tweets contenant notre mot-clé de recherche.

Section B: Préparation de l'ensemble de Training :

Dans cette section, nous utiliserons également notre instance de Twitter API de la dernière section. Nous utiliserons un ensemble de training téléchargeable. Les tweets étaient tous étiquetés comme positifs ou négatifs, selon le contenu. C'est exactement à cela que sert un ensemble de training.

Pour cette tâche, nous utiliserons le Corpus de Niek Sanders de plus de 5000 tweets classés à la main, ce qui le rend assez fiable. Il faut noter aussi que Twitter n'autorise pas le stockage de tweets sur un appareil personnel, même si toutes ces données sont accessibles au public. Par conséquent, le corpus comprend un mot-clé (sujet du tweet), une étiquette (pos / neg) et un numéro d'identification de tweet pour chaque tweet (c'est-à-dire une ligne dans notre corpus CSV).

```python
def Construire_Le_Training_Set(FichierCorpus, FichierTweets):
    import csv
    import time
    L = []
    with open(FichierCorpus, 'r') as csvfile:
        lecture_ligne = csv.reader(csvfile, delimiter=',')
        for row in lecture_ligne:
            try:
                status_tweet = twitter_api.GetStatus(row[2]) # on importe le tweet correspond au tweet_id (row[2] =tweet_id)
                print("Le tweet récupéré est : " + status_tweet.text)    # on extrait la partie texte du tweet qu'on vient d'importer .
                L.append({"tweet_id": row[2],"text":status_tweet.text, "label": row[1], "topic": row[0]}) # Enfin de l'execution on formera une liste contenat le training set.
                time.sleep(900 / 180)
            except:
                continue

    with open(FichierTweets, 'w') as csvfile:  #maintenant nous les écrivons dans le fichier CSV vide
        Ecriture_Ligne = csv.writer(csvfile, delimiter=',')
        for tweet in L:
            try:
                Ecriture_Ligne.writerow([tweet["tweet_id"], tweet["text"], tweet["label"], tweet["topic"]])
            except Exception as e:
                print(e)
    return L

FichierCorpus = "C:/Users/hp/Desktop/Fichiercorpus.csv"
FichierTweets = "C:/Users/hp/Desktop/FichierTweets.csv"
Data_To_Use_For_Training  = Construire_Le_Training_Set(FichierCorpus, FichierTweets)
```

Tout d'abord, nous définissons la fonction pour prendre deux entrées, qui sont toutes deux des chemins de fichier:

FichierCorpus est le chemin du fichier de corpus CSV de Niek Sanders que nous avons téléchargé. Ce fichier comprend le text de chaque tweet ,ainsi que le sujet, l'étiquette et l'ID.

FichierTweets est le chemin d'accès du fichier dans lequel nous aimerions enregistrer les tweets complets.

Le code traite de l'obtention du texte des tweets en fonction des ID. Nous parcourons les tweets dans **FichierCorpus**, appelant l'API sur chaque tweet pour obtenir texte qui lui est associé en utilisant le **Status_tweet.text** . Ensuite, nous le poussons dans la liste L puis dormons pendant cinq minutes (900/180 secondes) afin de respecter la limite de requêtes que nous pouvons faire dont nous avons parlé .

Prenons maintenant le temps de laisser notre script télécharger les tweets (ce qui prendra des heures). Le résultat est un TrainingSet qui sera stocké dans le variable **Data_To_Use_For_Training.**

Section C: prétraitement des tweets dans les ensembles de données

Avant de passer à la section de classification proprement dite, il y a du nettoyage à faire. Parlons de ce qui compte et de ce qui n'a pas d'importance dans l'analyse des sentiments. Les mots sont la partie la plus importante . Cependant, quand il s'agit de choses comme la ponctuation, vous ne pouvez pas obtenir le sentiment à partir de la ponctuation. Par conséquent, la ponctuation n'a pas d'importance pour l'analyse des sentiments. De plus, les composants du tweet comme les images, les vidéos, les URL, les noms d'utilisateur, les emojis, etc. ne contribuent pas à la polarité (qu'elle soit positive ou négative) du tweet.

Nous savons donc ce que nous devons garder dans les tweets et ce que nous devons retirer. Cela s'applique aux ensembles de TrainingSet et de TestingSet. Nous rédigeons le code suivant :

```
stopwords = set(stopwords.words('english') + list(punctuation) + ['AT_USER', 'URL'])

def prétraitement_Du_Tweets(Data_to_process):
    Tweets_Prétraités = []
    for tweet in Data_to_process:
        tweet["text"] = tweet["text"].lower()
        tweet["text"] = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'URL', tweet["text"])
        tweet["text"] = re.sub('@[^\s]+', 'AT_USER', tweet["text"])
        tweet["text"] = re.sub(r'#([^\s]+)', r'\1', tweet["text"])
        tweet["text"] = word_tokenize(tweet["text"])
        a=[mot for mot in tweet["text"] if mot not in stopwords]
        Tweets_Prétraités.append((a,tweet["label"]))
    return Tweets_Prétraités

TrainingSet_Prétraité = prétraitement_Du_Tweets(Data_To_Use_For_Training)
TestSet_Prétraité = prétraitement_Du_Tweets(Data_To_Use_For_Testing)
```

La fonction **prétraiement_Du_Tweets** boucle simplement tous les tweets qui y sont entrés. Ce dernier effectue le prétraitement proprement dit en transformant d'abord tout le texte en lettres minuscules. C'est simplement

parce que, dans presque tous les langages de programmation, «home» n'est pas interprété de la même manière que «hOMe». Par conséquent, il est préférable de normaliser tous les caractères en minuscules dans toutes nos données.

Deuxièmement, les URL et les noms d'utilisateur sont supprimés du tweet. Ensuite, le signe numérique (c'est-à-dire #) est supprimé de chaque hashtag, afin d'éviter que les hashtags soient traités différemment. Enfin et surtout, les caractères en double sont supprimés afin de garantir qu'aucun mot important ne soit traité, même s'il est énoncé de manière inhabituelle (par exemple, «hooome» devient «home»).

Enfin, le texte du tweet est décomposé en mots afin de faciliter son traitement dans les étapes à venir.

Notez que notre code a supprimé les caractères en double dans les mots comme nous l'avons mentionné plus tôt. Cependant, il n'a pas supprimé les mots en double du texte, mais les a plutôt conservés. En effet, le mot en double joue un rôle dans la détermination de la polarité du texte .

Section D: Classificateur Naive Bayes

Naive Bayes Classifier est un algorithme de classification qui s'appuie sur le théorème de Bayes. Ce théorème fournit un moyen de calculer un type ou une probabilité appelé probabilité postérieure, dans lequel la probabilité qu'un événement A se produise dépend d'un contexte probabiliste connu (par exemple, la preuve de l'événement B).

Tout ce que nous aurons besoin pour notre tâche est qu'un classificateur Naive Bayes dépend du théorème de Bayes toujours. Avant de poursuivre, donnons un bref aperçu des étapes à suivre:

1- Construire un vocabulaire (liste de mots avec leur fréquences) de tous les mots résidant dans notre ensemble de Training .

2- Associez le contenu des tweets à notre vocabulaire - mot par mot.

3- Construisez notre vecteur d'entité.

4- Branchez notre vecteur d'entités au Naive Bayes Classifier.

Étape D.1: Construire le vocabulaire

Dans ce cas, cela inclut tous les mots résidant dans l'ensemble de training que nous avons, car le modèle peut les utiliser tous de manière relativement égale. Le code ressemblera à ceci:

```
def Liste_Des_Mots(TrainingSet_Prétraité):
    Ensemble_Des_Mots = []
    for (mots, sentiment) in TrainingSet_Prétraité:
        Ensemble_Des_Mots.extend(mots)
    DesMots_avec_leur_fréquence= nltk.FreqDist(Ensemble_Des_Mots).keys()
    return DesMots_avec_leur_fréquence
```

Il s'agit simplement de créer une liste de l'ensemble des mots que nous avons dans l'ensemble de training, en la décomposant en mots avec fréquence. Ces **Desmots_avec_leur_frequence** sont essentiellement une liste de mots distincts, dont chacun a sa fréquence (nombre d'occurrences dans l'ensemble de training) comme un clé.

Étape D.2: Faire correspondre les tweets avec notre vocabulaire:

Cette étape est cruciale, car nous passerons en revue tous les mots de notre ensemble de Training (c'est-à-dire notre liste **DesMots_avec_leur_fréquence**), en

comparant chaque mot au tweet que nous avons , en associant un nombre au mot suivant:

étiquette 1 (vrai): si le mot du vocabulaire réside dans le tweet

étiquette 0 (faux): si le mot du vocabulaire ne réside pas dans le tweet

C'est assez simple à coder:

```
def Tirer_des_caract(tweet):
    appartenance_des_mots = {}
    for mot in Caract_des_mots :
        appartenance_des_mots['Est ce qu\'il contient le mot ?(%s)' % mot] = (mot in set(tweet))
    return appartenance_des_mots
```

Étape D3: Création de notre vecteur d'entités:

Appelons maintenant les deux dernières fonctions que nous avons écrites. Cela va construire notre vecteur, avec lequel nous pouvons passer à la phase du Training.

```
Caract_des_mots = Liste_Des_Mots(TrainingSet_Prétraité)
Caract_de_TrainingSet = nltk.classify.apply_features(Tirer_des_caract, TrainingSet_Prétraité)
```

Notre Vecteur Final est **Caract_de_TrainingSet .**

Étape D.4: Le training du classificateur :

Grâce à NLTK, il ne nous faudra qu'un appel de fonction pour former le modèle en tant que classificateur Naive Bayes, puisque ce dernier est déjà intégré à la bibliothèque:

```
NBayesClassifier = nltk.NaiveBayesClassifier.train(Caract_de_TrainingSet)
```

Section E: Test du modèle

Maintenant testons notre classificateur à notre 100 tweets que nous avons importé depuis Twitter , en obtenant le vote majoritaire des étiquettes retournées par le classificateur, puis en affichant le pourcentage total positif ou négatif des tweets.

```
NBResultLabels = [NBayesClassifier.classify(Tirer_Des_Caract(tweet[0])) for tweet in TestSet_Prétraité ]

# Maintenant vous aurez la nature du sentiment est qu'il est positive ou pas
if NBResultLabels.count('positive') > NBResultLabels.count('negative'):
    print("C'est un sentiment positive avec le pourcentage  " + str(100*NBResultLabels.count('positive')/len(NBResultLabels)) + "%")
else:
    print("C'est un sentiment négative avec le pourcentage" + str(100*NBResultLabels.count('negative')/len(NBResultLabels)) + "%")
```

Bibiliographie :

Stackoverflow.com

Github.com