ROYAUME DU MAROC
UNIVERSITE MOHAMMED V
FACULTE DES SCIENCES RABAT

Master Ingénierie de Données et Développement Logiciel

NLP

Documentation mini-projet
Sentiment analysis

Réalisé par:

BENZYANE Zakaria
KOURMOU Omar

Sous la supervision de :

Pr. MAHMOUDI Abdelhak

Plan

- Introduction.
- Analyse des sentiments
- Pourquoi les données Twitter
- Prétraitement des données.
- Words Embedding && Sentiment Classification.
- Topic Modelling à l’aide de LDA.
- Conclusion.

Introduction

Avec l’avènement du web et l’explosion des sources des données tels que les sites d’avis, les blogs et les micro-blogs est apparu la nécessite d’analyser des millions des postes, des tweets ou d’avis afin de savoir ce que pensent les internautes.

Analyse des sentiments

• L'objectif de l'analyse des sentiments est d'analyser une grande quantité de données afin d'en déduire les différents sentiments qui y sont exprimés. Les sentiments extraits peuvent ensuite faire l'objet de statistiques sur le ressenti général d'une communauté.

• Aussi appelé extraction d'opinion, notre objectif est de déterminer si les données (tweets) sont positives, négatives ou neutres.

Pourquoi les données Twitter ?

- Site de microblogging populaire.
- Messages texte courts de 140 caractères.
- Plus de 240 millions d'utilisateurs actifs.
- 500 millions de tweets sont générés chaque jour.
- L'audience de Twitter varie d'un homme ordinaire à une célébrité.
- Les utilisateurs discutent souvent des sujets courant (comme covid 19, le sujet de notre dataset).
- Les tweets sont de petite longueur et donc sans ambiguïté.

Prétraitement des données

Une technique utilisée pour convertir les données brutes en un ensemble de données propre.
En d'autres termes, chaque fois que les données sont collectées à partir de différentes sources (Twitter dans notre cas), elles sont collectées au format brut, ce qui n'est pas réalisable pour l'analyse.

* La fonction lower ()

Python, **lower ()** est une méthode intégrée utilisée pour la gestion des chaînes. La méthode **lower ()** renvoie la chaîne en minuscules de la chaîne donnée. Il convertit tous les caractères majuscules en minuscules. S'il n'existe aucun caractère majuscule, il renvoie la chaîne d'origine.

* La fonction clean ()

Une fonction de la bibliothèque **Preprocessor** qui se charge de prétraitement pour les données de tweet écrites en Python.

La fonction **clean ()** prend actuellement en charge le nettoyage :

- ✓ URLs
- ✓ Hashtags
- ✓ Emojis

Stop words

Dans le NLP, les mots inutiles (données) sont appelés **stop words**.

Stop words: Un **stop word** est un mot couramment utilisé (comme «le», «a», «un», «dans») qu'un moteur de recherche a été programmé pour ignorer, à la fois lors de l'indexation des entrées pour la recherche et lors de leur récupération. à la suite d'une requête de recherche.

NLTK (Natural Language Toolkit) en python a une liste de **stop words** stockés dans 16 langues différentes ,Nous pouvons même modifier la liste en ajoutant les mots de notre choix dans le fichier english.txt dans le répertoire des stop words.

Suppression des stop words avec NLTK

Le programme suivant supprime les stop words :

```
Entrée [20]: STOPWORDS = set(stopwords.words('english'))
             def remove_stopwords(text):
                 """custom function to remove the stopwords"""
                 return " ".join([word for word in str(text).split() if word not in STOPWORDS])
```

* Les ponctuations

Suppression des ponctuations

Le programme suivant supprime les ponctuations :

Entrée [21]:
```
# Removal of punctions

PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

df_eng['clean_text'] = df_eng['clean_text'].apply(lambda text: remove_punctuation(text))
```

* Stemming

L'objectif principal de **Stemming** est de réduire les mots à leur forme racine.
Cette technique de NLP fonctionne sur le principe que certains mots ayant une orthographe légèrement différente mais presque la même signification doivent être dans le même jeton.
Ainsi, nous coupons les suffixes des mots pour un traitement efficace.

Il existe plusieurs types d'algorithmes de **stemming** disponibles et l'un des plus célèbres est le **PorterStemmer**, qui est largement utilisé.

Exemple :

Entrée [22]:
```
from nltk.stem.porter import PorterStemmer
```

* Lemmatization

La **lemmatization**, contrairement au Stemming, réduit correctement les mots fléchis en s'assurant que le mot racine appartient à la langue. Dans la **lemmatization**, le mot racine est appelé **Lemma** . Un lemma (lemmas pluriel ou lemmata) est la forme canonique, la forme du dictionnaire ou la forme de citation d'un ensemble de mots.

Python NLTK fournit un lemmatiseur **WordNet** qui utilise la base de données WordNet pour rechercher des **lemmas** de mots.

Exemple

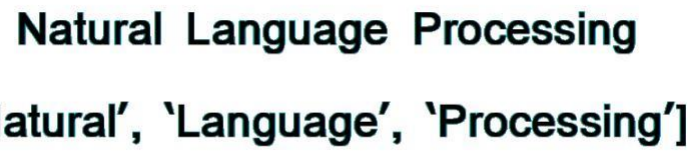
Entrée [24]:

```
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
wordnet_map = {"N":wordnet.NOUN, "V":wordnet.VERB, "J":wordnet.ADJ, "R":wordnet.ADV}
def lemmatize_words(text):
    pos_tagged_text = nltk.pos_tag(text.split())
    return " ".join([lemmatizer.lemmatize(word, wordnet_map.get(pos[0], wordnet.NOUN)) for word, pos in pos_tagged_text])

df_eng['clean_text'] = df_eng['clean_text'].apply(lambda text: lemmatize_words(text))
```

* Tokenization

La **tokenization** est l'une des tâches les plus courantes lorsqu'il s'agit de travailler avec des données texte, elle consiste essentiellement à diviser une phrase, une expression, un paragraphe ou un document texte entier en unités plus petites, telles que des mots ou des termes individuels. Chacune de ces petites unités est appelée **tokens**.

La figure ci-dessous visualise cette définition :



Exemple :

```
Entrée [25]: from nltk.tokenize import word_tokenize

Entrée [26]: df_eng['clean_text'] = df_eng['clean_text'].apply(lambda text: word_tokenize(text))
```

Word Embeddings && Sentiment Classification

Avant de pouvoir commencer avec la classification, nous devons trouver un moyen de représenter les mots.
Maintenant, il existe deux façons populaires de le faire: **Word Vectors** et **Word Embeddings.**

* Les ***Word Vectors*** sont des vecteurs épars de haute dimension (principalement des 0) où chaque vecteur représente un mot qui est simplement codé à chaud.
* Les ***Word Embeddings***, contrairement aux word Vectors, représentent des mots dans des vecteurs denses. Les mots sont mappés dans un espace significatif où la distance entre les mots est liée à leur similitude sémantique.

Topic Modelling à l’aide de LDA

* **Topic Modelling** est une approche non supervisée utilisée pour trouver un ensemble de mots appelés «***topics***» dans un document texte.

* Ces ***topics*** se composent de mots qui se produisent fréquemment ensemble et partagent généralement un thème commun. Et par conséquent, ces ***topics*** avec l'ensemble de mots prédéfini peuvent être utilisés comme facteurs pour décrire au mieux l'ensemble du document.

* **Topic Modelling** nous fournit des méthodes pour organiser, comprendre et résumer de grandes collections de données textuelles.

Conclusion

L'analyse des sentiments dispose d'une grande variété d'applications qui pourraient bénéficier de ses résultats, telles que l'analyse de l'actualité, le marketing, la réponse aux questions, les bases de connaissances, etc. le défi de ce domaine est de développer la capacité de la machine à comprendre les textes comme le font les lecteurs humains.