Twitter Sentiment Analysis

Rapport du mini-Projet moteurs de recherches distribués

Réalisé par :

- NBADOU Ossama

Encadré par :

- Pr. EL GHALI Btihal

Table des matières

I. INTRODUCTION :

Les réseaux sociaux font l'objet d'une attention accrue de nos jours. Des opinions publiques et privées sur une grande variété de sujets sont exprimées et diffusées en permanence via de nombreux médias sociaux. Twitter est l'un des médias sociaux qui gagne en popularité. Twitter offre aux organisations un moyen rapide et efficace d'analyser les opinions des clients sur ce qui est essentiel à leur réussite sur le marché. Le développement d'un programme d'analyse des sentiments est une approche à utiliser pour mesurer de manière computationnelle les perceptions des clients. Ce rapport présente la conception d'une analyse des sentiments, en extrayant une quantité de tweets. Les résultats classent les opinions des auteurs via les tweets en positif, négatif et neutre ce qui est représenté dans un graphique circulaire et plusieurs autres visualisations que nous allons les voir par la suite.

II. Etude littérature :

1. Elasticsearch :

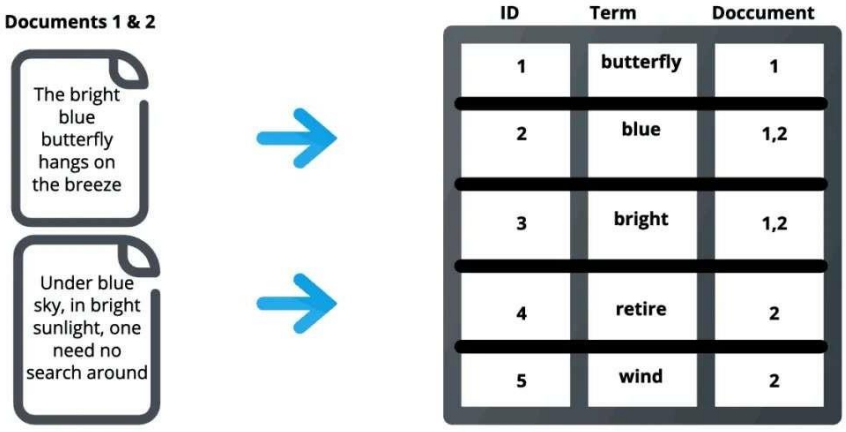
Elasticsearch est un moteur de recherche et d'analyse distribué et open-source basé sur Apache Lucene et développé en Java. Au départ, il s'agissait d'une version évolutive du cadre de recherche open-source Lucene, puis la possibilité d'étendre horizontalement les indices Lucene a été ajoutée. Elasticsearch vous permet de stocker, de rechercher et d'analyser d'énormes volumes de données rapidement et en temps quasi réel, et de fournir des réponses en quelques millisecondes. Il est capable d'obtenir des réponses rapides parce qu'au lieu de rechercher le texte directement, il recherche dans un index. Il utilise une structure basée sur des documents plutôt que sur des tables et des schémas, et est fourni avec des API REST étendues pour le stockage et la recherche de données. Au fond, Elasticsearch est un serveur capable de traiter des requêtes JSON et de vous renvoyer des données JSON.

Les documents sont l'unité de base de l'information qui peut être indexée dans Elasticsearch exprimée en JSON, qui est le format mondial d'échange de données sur Internet. Vous pouvez considérer un document comme une ligne dans une base de données relationnelle, représentant une entité donnée - la chose que vous recherchez. Dans Elasticsearch, un document peut être plus qu'un simple texte, il peut s'agir de n'importe quelle donnée structurée codée en JSON. Ces données peuvent être des nombres, des chaînes de caractères et des dates. Chaque document possède un identifiant unique et un type de données donné, qui décrit le type d'entité que représente le document. Par exemple, un document peut représenter un article d'encyclopédie ou des entrées de journal d'un serveur Web.

Un index est une collection de documents qui ont des caractéristiques similaires. Un index est l'entité de plus haut niveau que vous pouvez interroger dans Elasticsearch. Vous pouvez considérer l'index comme

similaire à une base de données dans un schéma de base de données relationnelle. Tous les documents d'un index sont généralement liés logiquement. Dans le contexte d'un site de commerce électronique, par exemple, vous pouvez avoir un index pour les clients, un pour les produits, un pour les commandes, et ainsi de suite. Un index est identifié par un nom qui est utilisé pour faire référence à l'index lors des opérations d'indexation, de recherche, de mise à jour et de suppression des documents qu'il contient.

Un index dans Elasticsearch est en fait ce que l'on appelle un index inversé, qui est le mécanisme par lequel tous les moteurs de recherche fonctionnent. Il s'agit d'une structure de données qui stocke une correspondance entre le contenu, tel que des mots ou des chiffres, et son emplacement dans un document ou un ensemble de documents. En gros, il s'agit d'une structure de données de type hashmap qui vous dirige d'un mot vers un document. Un index inversé ne stocke pas directement les chaînes de caractères, mais divise chaque document en termes de recherche individuels (c'est-à-dire chaque mot), puis fait correspondre chaque terme de recherche aux documents dans lesquels ces termes de recherche apparaissent. Par exemple, dans l'image ci-dessous, le terme "meilleur" apparaît dans le document 2, il est donc associé à ce document. Cela permet de savoir rapidement où trouver les termes recherchés dans un document donné. En utilisant des indices inversés distribués, Elasticsearch trouve rapidement les meilleures correspondances pour les recherches en texte intégral, même dans des ensembles de données très volumineux.

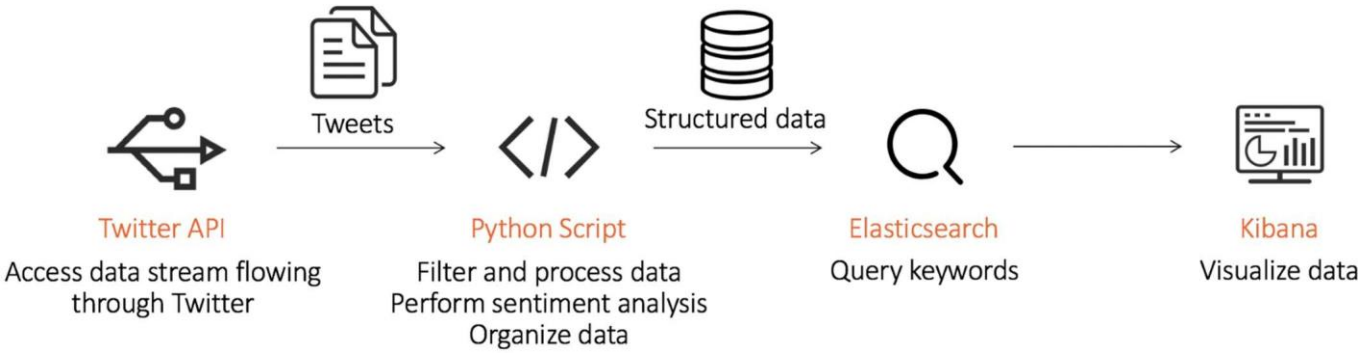


2. Kibana :

Kibana est un outil de visualisation et de gestion des données pour Elasticsearch qui fournit des histogrammes, des graphiques linéaires, des camemberts et des cartes en temps réel. Il vous permet de visualiser vos données Elasticsearch et de naviguer dans la pile Elastic. Vous pouvez choisir la façon dont vous donnez forme à vos données en commençant par une question pour savoir où la visualisation interactive vous mènera. Par exemple, comme Kibana est souvent utilisé pour l'analyse des journaux, il vous permet de répondre à des questions sur l'origine de vos visites Web, vos URL de distribution, etc. Si vous ne construisez pas votre propre application au-dessus d'Elasticsearch, Kibana est un excellent moyen de rechercher et de visualiser votre index avec une interface utilisateur puissante et flexible. Cependant, un inconvénient majeur est que chaque visualisation ne peut fonctionner que par rapport à un seul index/modèle d'index. Ainsi, si vous avez des index avec des données strictement différentes, vous devrez créer des visualisations distinctes pour chacun. Pour des cas d'utilisation plus avancés, Knowi est une bonne option. Il vous permet de joindre vos données Elasticsearch à travers plusieurs index et de les mélanger avec d'autres sources de données SQL/NoSQL/REST-API, puis de créer des visualisations à partir de ces données dans une interface utilisateur conviviale.

III. Réalisation :

1. Le workflow :



2. Code :

- Récupération des API depuis Twitter :

```python
# the config file that has your twitter access information.

consumer_key = "EfDxTE4CnldCd8nKiHM4D9nq2"
consumer_secret = "yjJQpI43Uwud4Ffi1LEnfODb9NN8VxDrxqroBYRsWn7fY2AbqB"
access_token = "1356761794983972916-yezL1QQ4biMbT9NXNCw66GnZswAxXW"
access_token_secret = "qqt6XAt0K7lktJRzJMRZzsaavdhd8zYOUxLCBbN8TaHvb"

```

- Création de DataFrame :

Dans cette étape, nous allons de l'avant et créons un cadre de données avec des données extraites de Twitter Utilisez TwitterSearchScraper.

```python
import snscrape.modules.twitter as sntwitter
import pandas as pd

# Creating list to append tweet data to
tweets_list1 = []

# Using TwitterSearchScraper to scrape data and append tweets to list
for i,tweet in enumerate(sntwitter.TwitterSearchScraper("NFT").get_items()):
    if i>100:
        break
    tweets_list1.append([tweet.date, tweet.id, tweet.content, tweet.user.username])

# Creating a dataframe from the tweets list above
tweets_df1 = pd.DataFrame(tweets_list1, columns=['Datetime', 'Tweet Id', 'Text', 'Username'])
print(tweets_df1.head())
```

- Exécution python :

Les bibliothèques utilisées sont :

```python
from tweepy import OAuthHandler
from tweepy import Stream
import json
import snscrape.modules.twitter as sntwitter
import pandas as pd

import sys

# Import twitter keys and tokens
from config import *

# Import listener
from textblob import TextBlob
from elasticsearch import Elasticsearch

```

La fonction Get-sentiment nous permet d'analyser les tweets à l'aide de TextBlob ; elle nous permet de déterminer également la polarité de la sensation, si polarité < 0 alors la sensation est négative ;

De même, si la polarité > 0, alors le ressenti est Positif, et enfin si la polarité == 0, alors le ressenti est neutre.

```python
def _get_sentiment(self, decoded):
    # Pass textual data to TextBlob to process
    tweet = TextBlob(decoded)
    author=tweet.split(' ')[0]

    # [0, 1] where 1 means very subjective
    subjectivity = tweet.sentiment.subjectivity
    # [-1, 1]
    polarity = tweet.sentiment.polarity

    # Determine if sentiment is positive, negative, or neutral
    if polarity < 0:
        sentiment = "negative"
    elif polarity == 0:
        sentiment = "neutral"
    else:
        sentiment = "positive"

    return polarity, subjectivity, sentiment, author

```

On met les données dans ElasticSearch,

```
45          # Inject data into Elasticsearch
46          doc = {"author":author,
47                 "polarity": polarity,
48                 "subjectivity": subjectivity,
49                 "sentiment": sentiment,
50                }
51
52          es = Elasticsearch("http://elastic:changeme@localhost:920
53          es.index(index=self.index,
54                   doc_type=self.doc_type,
55                   body=doc)
```

On utilise le mot-clés « Congress »,et on crée notre index.

```
88      # Obtain the input topics of your interests
89      topics = []
90      if len(sys.argv) == 1:
91          # Default topics
92          topics = ['Congress']
93      else:
94          for topic in sys.argv[1:]:
95              topics.append(topic)
96
97      # Change this if you're not happy with the index and type nam
98      index = "tweet-sentiment"
99      doc_type = "new-tweet"
100
```

V. Visualisation des données avec Kibana:

Nous avons utilisé Kibana pour crée un Dashboard qui visualise les données collectées.

Total number of tweets

303

Count of records

Count of sentiments

Top values of sentiment.keywor	Count of records
neutral	154
positive	81
negative	68

Top 5 authors

Top values of autho	Top values of sentir	Count of records
Congress	negative	10
Congress	neutral	2
@INCIndia	neutral	9
I	positive	4
I	neutral	2
@donwinslow	negative	2
@donwinslow	neutral	2
@dsrajpurohit291	neutral	4
Other	neutral	80
Other	positive	53
Other	negative	34

[No Title]