

**MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE  
SCIENTIFIQUE**

**UNIVERSITÉ DE SOUSSE**

**المعهد العالي للإعلامية وتقنيات الاتصال بحمام سوسة**



**Institut supérieur de l'informatique et des technologies de la  
communication - HAMMAM SOUSSE**

## **Rapport de projet Data Mining**

Spécialité : Téléinformatique

## **Classification des Tweets**

**Réaliser par :**

Kassdallah Sabrine

**Encadrée par :**

Superviseur académique : Mr. Lotfi Ben Romdhane Professeur en informatique

Superviseur académique : Mr. Khemais Abdallah Enseignant en data science

# Sommaire

Introduction.....	5
Définition.....	5
Les étapes du data Mining .....	5
Problématique.....	6
État de l'art.....	6
Préparation des tweets.....	6
Création d'application.....	7
Api key .....	7
Access token .....	8
Préparation datasets .....	8
Tweepy .....	8
Définition.....	8
Installation.....	9
GetOldTweets3 .....	9
Définition.....	9
Installation.....	9
Identifiants et autorisation .....	10
Récupérer les tweets .....	10
Concaténation de datasets .....	11
Import libraires python.....	12
Classification du texte.....	12
Définition de l'ensemble des mots.....	13
Vectorisation et standardisation.....	13
TF.....	13
IDF.....	13
Distance de Jaccard.....	14
KMeans Clustering .....	15
Clustered Datasets : Question 4.....	16
Les tweets représentatifs de datasets total .....	17
Les tweets représentatifs de catégorie sport .....	17
Les tweets représentatifs de catégorie politics .....	18
Les tweets représentatifs de catégorie economy .....	18
Les tweets représentatifs de catégorie social .....	19
Les tweets représentatifs de catégorie culture .....	19

<b>Les tweets représentatifs de catégorie health .....</b>	<b>20</b>
<b>Conclusion et perspectives .....</b>	<b>20</b>

# Liste des figures

Figure 1:les étapes de Data Mining .....	5
Figure 2: Création d'application .....	7
Figure 3: Api key.....	7
Figure 4: Access token .....	8
Figure 5 : Installation de Tweepy .....	9
Figure 6 : Installation de GetOldTweets3 .....	9
Figure 7 : credentials and authorization.....	10
Figure 8 : Récupérer les tweets par text_query .....	10
Figure 9 : enregistrer le fichier en csv .....	11
Figure 10 : Concaténation de datasets .....	11
Figure 11 : Importation des libraires python .....	12
Figure 12 : nettoyage des tweets .....	12
Figure 13 : Vectorisation des tweets .....	14
Figure 14: calculer la Distance de Jaccard entre les tweets .....	14
Figure 15 : Appliquer l'algorithme de clustering.....	15
Figure 16: Représentation du volumes des tweets .....	16
Figure 17 : Les tweets représentatifs de datasets total .....	17
Figure 18 : Les tweets représentatifs de catégorie sport .....	17
Figure 19 : Les tweets représentatifs de catégorie politics .....	18
Figure 20 : Les tweets représentatifs de catégorie economy.....	18
Figure 21 : Les tweets représentatifs de catégorie social.....	19
Figure 22 : Les tweets représentatifs de catégorie culture .....	19
Figure 23 : Les tweets représentatifs de catégorie health .....	20

# Introduction

## Définition

Data Mining est un l'ensemble des algorithmes, méthodes et technologies inspirés de plusieurs autres disciplines, propres ou non au dm pouvant servir à remplacer ou à aider l'expert humain ou le décideur dans un domaine spécifique dans le cadre de prise de décision, et ce en fouillant dans des bases de données décisionnelles des corrélations, des associations, des comportements homogènes, des formules de lien entre indicateurs, des spécifications par rapport à une thématique bien déterminée, etc. Cet ensemble de techniques peut être une étape fondamentale dans tout processus ECD (KDD) ou bien l'intervention de l'intelligence artificielle, la reconnaissance de forme, et la statistique décisionnelle dans tout processus de Business Intelligence afin de transformer tout système d'aide à la décision en un système DECISIONNEL au vrai sens du mot.

## Les étapes du data Mining

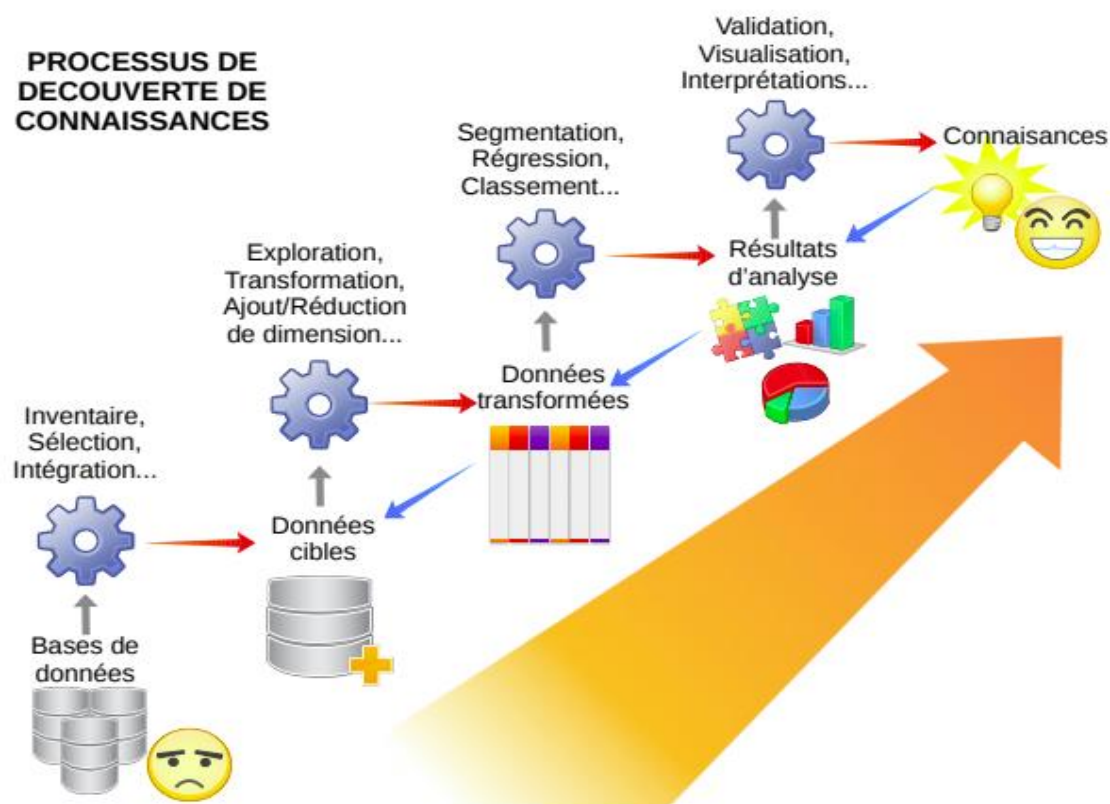


Figure 1: les étapes de Data Mining

## Problématique

Les postes publiés sur Twitter ou Instagram reflètent l'interaction d'utilisateurs avec les événements réels qui se déroulent dans le monde, comme les élections, les événements sportifs et culturels, les catastrophes naturelles, etc. Ces événements réels ont un impact direct sur la quantité de tweets mises en ligne.

Le suivi de ces événements sur les réseaux sociaux généralement et sur Twitter plus précisément est un défi audacieux pour les chercheurs, tout d'abord parce qu'un sujet sur Twitter est caractérisé par plusieurs termes (ces termes peuvent être des hashtags) qui peuvent changer dynamiquement où certains peuvent devenir moins utilisés et d'autres peuvent apparaître. Donc il est incontournable de trouver un moyen pour couvrir tous ces termes utilisés pendant le processus d'analyse. Cela représente l'un de nos objectifs dans ce travail. Mais avant de chercher les nouveaux termes, il faut pouvoir identifier les ensembles de tweets qui parlent du même sujet et qui représentent un fil de discussion, ce qui définit l'objectif principal de notre travail.

## État de l'art

L'analyse des postes dans les réseaux sociaux est une nouvelle discipline qui émerge en informatique. Des contributions comme la détection des événements, la détection des maladies et l'extraction des connaissances depuis Twitter, sont les travaux les plus proches de notre problématique.

Dans cette partie nous allons citer quelques travaux connexes que nous avons synthétisés, afin d'inspirer les différentes étapes de notre processus d'analyse en général. Ensuite, nous parlerons des différentes stratégies de représentation du texte et nous concluons par une étude comparative entre les algorithmes de clustering dont nous allons nous servir.

## Préparation des tweets

Pour résoudre notre problème ,on collecte un ensemble des tweets à travers API twitter.

Il y a des étapes nécessaires que nous avons fait pour bénéficier d'API twitter

## Création d'application

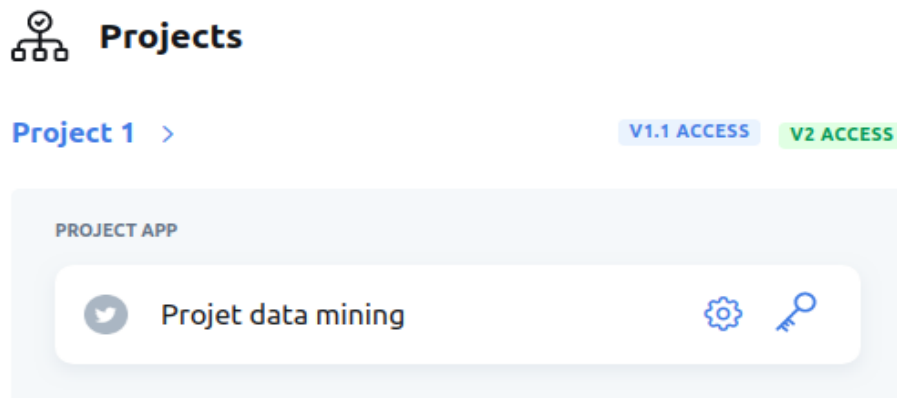


Figure 2: Création d'application

## Api key

Here are your API key and secret. Have you saved them?



For security, we will be hiding these starting 01/12/2021. If something happens, you can always regenerate them.

**API key:** l8mPBrUN2jGSXCd85Rdrvz0xq



**API key secret:** z53SiNFIPhKPT8JT66qtzNsAT9gtXMcvpM6nAg3eYHI9G  
BhmY4





Figure 3: Api key

## Access token

**Here are your new access token & secret. Have you saved them?** ✕

---

For security, this will be the last time we'll display these. If something happens, you can always regenerate them.

<b>Access token:</b>	1331376961801904137-gAKXY38FBf9TDw4DQxYYTKkf3b8t3B	
<b>Access token secret:</b>	zCTuCNZaqDRE0luT4dP2OUR3HkwJJQ9TM7y5ye30Mvaln	

Yes, I saved them

Figure 4: Access token

## Préparation datasets

Pour collecter les tweets de chaque catégorie nous avons utilisé les deux bibliothèques suivantes :

✓ Tweepy

### Définition

Tweepy est un package Python open source qui vous offre un moyen très pratique d'accéder à l'API Twitter avec Python. Tweepy comprend un ensemble de classes et de méthodes qui représentent les modèles de Twitter et les points de terminaison d'API, et il gère de manière transparente divers détails d'implémentation, tels que :

- ✓ Data encoding and decoding
- ✓ HTTP requests
- ✓ Results pagination
- ✓ OAuth authentication
- ✓ Rate limits
- ✓ Streams



## Installation

```
In [1]: !pip install tweepy

Collecting tweepy
  Downloading tweepy-3.9.0-py2.py3-none-any.whl (30 kB)
Requirement already satisfied: six>=1.10.0 in /opt/conda/lib/python3.7/site-packages (from tweepy) (1.14.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /opt/conda/lib/python3.7/site-packages (from tweepy) (1.2.0)
Requirement already satisfied: requests[socks]>=2.11.1 in /opt/conda/lib/python3.7/site-packages (from tweepy) (2.23.0)
Requirement already satisfied: requests>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from requests-oauthlib>=0.7.0->tweepy) (2.23.0)
```

Figure 5 : Installation de Tweepy

## ✓ GetOldTweets3

### Définition

Pour la plupart des projets d'exploration de texte ou de classification, l'extraction de tweets est l'une des étapes initiales les plus importantes. La méthode bien connue consiste à extraire les tweets avec tweepy et à créer un compte développeur sur Twitter. Pour certaines raisons de sécurité, Twitter prend près de 15 jours pour vérifier le processus de création d'un compte développeur. Ainsi, l'utilisation de cette bibliothèque python facilite le processus. Un autre avantage de l'utilisation de cette bibliothèque est que les tweets sont des tweets assez récents. On peut recevoir des tweets des mois précédents ou même des semaines.

## Installation

```
In [2]: !pip install GetOldTweets3

Collecting GetOldTweets3
  Downloading GetOldTweets3-0.0.11-py3-none-any.whl (13 kB)
Requirement already satisfied: lxml>=3.5.0 in /opt/conda/lib/python3.7/site-packages (from GetOldTweets3) (4.5.0)
Collecting pyquery>=1.2.10
  Downloading pyquery-1.4.3-py3-none-any.whl (22 kB)
Requirement already satisfied: lxml>=3.5.0 in /opt/conda/lib/python3.7/site-packages (from GetOldTweets3) (4.5.0)
Collecting cssselect>0.7.9
  Downloading cssselect-1.1.0-py2.py3-none-any.whl (16 kB)
Installing collected packages: cssselect, pyquery, GetOldTweets3
Successfully installed GetOldTweets3-0.0.11 cssselect-1.1.0 pyquery-1.4.3
```

Figure 6 : Installation de GetOldTweets3

## Identifiants et autorisation

Avant de commencer, Tweepy vous permet à disposer des informations d'identification pour utiliser son API. L'extrait de code suivant montre comment se fait l'autorisation.

### Credentials and Authorization

```
import tweepy
consumer_key = "I8mPBrUN2jGSXCd85Rdrvz0xq"
consumer_secret = "z53SiNF1PHkPT8JT66qtzNsAT9gtXMcvpM6nAg3eYHI9GBhmY4"
access_token = "1331376961801904137-aBQRLq75ju0J9B80e5g0TtAvVZ7fq5"
access_token_secret = "Bh03ncc6XxzXYgVUcBXaxe5LqBiYx9zIK6AZRH9yW3Vkh"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth,wait_on_rate_limit=True)
```

Figure 7 : credentials and authorization

## Récupérer les tweets

Les paramètres de recherche sur lesquels je me suis concentré sont **q** et **count**. **q** est censé être la requête de recherche de texte avec laquelle vous souhaitez effectuer la recherche, et **count** est à nouveau le nombre maximal de tweets le plus récents que vous souhaitez extraire de cette requête de recherche spécifique. Dans cet exemple, je gratte les 3000 tweets les plus récents qui étaient pertinents pour le thème sport.

```
# Input search query to scrape tweets and name csv file
# Max recent tweets pulls x amount of most recent tweets from that user
text_query = ' Sport '
count = 3000
#screen_name = screen_name
# Calling function to query X amount of relevant tweets and create a CSV file
text_query_to_csv(text_query, count)
```

Figure 8 : Récupérer les tweets par text\_query

## Save file csv

```
Sport= pd.read_csv('./ Sport -tweets.csv' )  
Sport.head(3000)
```

	user	Text
0	dante3346	@dannyy0y6 @trucifer_x I don't know how old yo...
1	67dmy19	@NewsNationNow So not a sport.
2	Lautaro_Arribas	RT @MarianoReyOK: Matias Ledesma en "La 94 Spo...
3	AileenAdrienne	RT @BBCSport: Paul Pogba "has to change teams"...
4	mali_hussein	RT @SunChelsea: Meet Marina Granovskaia, the m...
...	...	...

Figure 9 : enregistrer le fichier en csv

**Remarque :** pour chaque catégorie que nous avons utilisée on refait la même modification pour le **text\_query** et **count**

## Concaténation de datasets

Après avoir récupérer les tweets de chaque catégorie on fait la concaténation

```
import os  
import glob  
import pandas as pd  
datasets = pd.concat([economy, social, culture, health, politics, sport],ignore_index=True)  
datasets.head(20000)
```

	user	Text
0	ReaveleyBarbara	@TruthOuter @lara_lazar "Those who do not lear...
1	_letsbebadguys	RT @Sillyshib: So, basically the government ar...
2	aplemkseriously	RT @brooklynmarie: Today I got the idea to put...
3	FixSheltersNow	@B_J_Davis @NeilAxelrod @LisaMarieBoothe You m...
4	GrantThorntonAU	COVID has pressed the pause button on the econ...
...	...	...

Figure 10 : Concaténation de datasets

## Import libraires python

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import tweepy
import csv
import os
import pandas as pd

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import spacy
from sklearn.model_selection import train_test_split
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.tokenize import RegexpTokenizer, WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import string
from string import punctuation
import collections
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import en_core_web_sm
```

Figure 11 : Importation des libraires python

## Classification du texte

Dans cette partie on va Nettoyer les tweets et on applique les différentes techniques de NLTK tel que : lemmatisation, tokenisation, suppression des arrêts, ponctuations, hashtags et mentions

```
# remove the hashtags, mentions and unwanted characters from the tweet texts
def clean_text(df, text_field):
    df[text_field] = df[text_field].str.lower()
    df[text_field] = df[text_field].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|(^0-9A-Za-z \t)|(\w+:\w+\S+)|^rt", "", elem))
    return df

clean_tweets = clean_text(datasets, 'Text')
clean_tweets.head()
```

	user	Text
0	ReaveleyBarbara	lazar those who do not learn history are doom...
1	_letsbebadguys	so basically the government are playing chic...
2	aplemkseriously	today i got the idea to put together a threa...
3	FixSheltersNow	jdavis you must be kidding seriously begging...
4	GrantThorntonAU	covid has pressed the pause button on the econ...

Figure 12 : nettoyage des tweets

## Définition de l'ensemble des mots

Dans cette partie on va définir l'ensemble des mots reliés à chaque thème, pour effectuer ce processus on a utilisé **Related Words**

Related Words fonctionne sur plusieurs algorithmes différents qui sont en concurrence pour obtenir le plus haut résultat dans la liste. Un de ces algorithmes utilise l'incorporation de mots pour convertir les mots en de nombreux vecteurs dimensionnels qui représentent leur signification

L'incorporation de mots fait partie d'un ensemble de techniques de modélisation du langage et d'apprentissage de fonctionnalités dans le traitement du langage naturel (NLP) où des mots ou des phrases du vocabulaire sont mappés à des vecteurs de nombres réels. Conceptuellement, il s'agit d'une intégration mathématique d'un espace avec de nombreuses dimensions par mot à un espace vectoriel continu avec une dimension beaucoup plus faible.

Lien pour obtenir des mots connexes : <https://relatedwords.org/>

## Vectorisation et standardisation

Vectoriser les ensembles de mots, puis les standardiser. **TFIDF** sera utilisé afin de s'occuper des mots les moins fréquents. La standardisation est parce que TFIDF favorise les phrases longues et il y aura des incohérences entre la longueur des tweets et la longueur de l'ensemble de mots.

### TF

TF : Term Frequency est une notation de la fréquence du mot dans le document actuel.

### IDF

IDF : Inverse Document Frequency est une notation de la rareté du mot dans les documents.

```
def get_vectors(*strs):
    text = [t for t in strs]
    vectorizer = TfidfVectorizer(text)
    vectorizer.fit(text)
    return vectorizer.transform(text).toarray()

socialvector = get_vectors(social)
economic_vector = get_vectors(economy)
culture_vector = get_vectors(culture)
health_vector = get_vectors(health)
politics_vector = get_vectors(politics)
sport_vector = get_vectors(sport)
```

Figure 13 : Vectorisation des tweets

## Distance de Jaccard

La similitude Jaccard est bonne pour les cas où la duplication n'a pas d'importance, la similitude cosinus est bonne pour les cas où la duplication est importante lors de l'analyse de la similitude du texte. Pour deux descriptions de produits, il sera préférable d'utiliser la similitude Jaccard car la répétition d'un mot ne réduit pas leur similitude.

```
def jaccard_similarity(query, document):
    intersection = set(query).intersection(set(document))
    union = set(query).union(set(document))
    return len(intersection)/len(union)
# jaccard_score(socialvector, economic_vector)

#for similarity of 1 and 2 of column1
# jaccard_similarity('dog lion a dog', 'dog is cat')

def get_scores(group, tweets):
    scores = []
    for tweet in tweets:
        s = jaccard_similarity(group, tweet)
        scores.append(s)
    return scores

#sport scores
sp_scores = get_scores(sport, datasets.tweets.to_list())
sp_scores[:10]

[0.6785714285714286,
 0.8214285714285714,
 0.6785714285714286,
 0.7142857142857143,
```

Figure 14: calculer la Distance de Jaccard entre les tweets

Pour chaque catégorie on refait la même chose et on calcule la distance entre deux tweets

## KMeans Clustering

Pour traiter les données d'apprentissage, l'algorithme K-means dans l'exploration de données commence par un premier groupe de centres de gravité (centroids) sélectionnés au hasard, qui sont utilisés comme points de départ pour chaque cluster, puis effectue des calculs itératifs (répétitifs) pour optimiser les positions des centres de gravité (centroids).

Il arrête la création et l'optimisation des clusters lorsque :

Les centres de gravité (centroids) se sont stabilisés- il n'y a pas de changement dans leurs valeurs car le regroupement a réussi.

Le nombre d'itérations défini a été atteint.

```
# fitting kmeans to dataset
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300, random_state=0)
Y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[Y_kmeans==0, 0], X[Y_kmeans==0, 1], s=100, c='violet', label= 'Cluster 1')
plt.scatter(X[Y_kmeans==1, 0], X[Y_kmeans==1, 1], s=100, c='cyan', label= 'Cluster 2')
plt.scatter(X[Y_kmeans==2, 0], X[Y_kmeans==2, 1], s=100, c='green', label= 'Cluster 3')
# plt.scatter(X[Y_kmeans==3, 0], X[Y_kmeans==3, 1], s=100, c='blue', label= 'Cluster 4')
# plt.scatter(X[Y_kmeans==4, 0], X[Y_kmeans==4, 1], s=100, c='magenta', label= 'Cluster 5')
plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1], s=100, c='black', label='Centroids' )
plt.title('Clusters of tweets in economic and social groups')
plt.xlabel('economic tweets')
plt.ylabel('social tweets')
plt.legend()
plt.show()
```

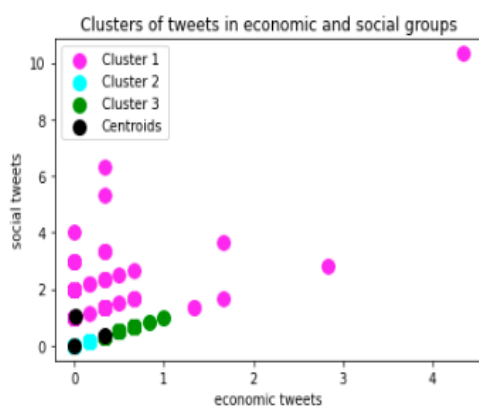


Figure 15 : Appliquer l'algorithme de clustering

Par la suite on refait la même méthode pour le reste des catégories deux à deux.

## Clustered Datasets : Question 4

Dans cette partie on va étudier ou bien obtenir le tweet le plus représentatif de chaque catégorie.

Voilà un graphique à secteurs pour afficher le nombre total de tweets dans chaque catégorie.

```
fig = plt.figure(figsize =(10, 20))
a = pivot_clusters.drop(['total'], axis = 1)
plt.pie(a.loc['Total'], labels = a.columns)
plt.title('A pie chart showing the volumes of tweets under different categories.')
plt.show()
```

A pie chart showing the volumes of tweets under different categories.

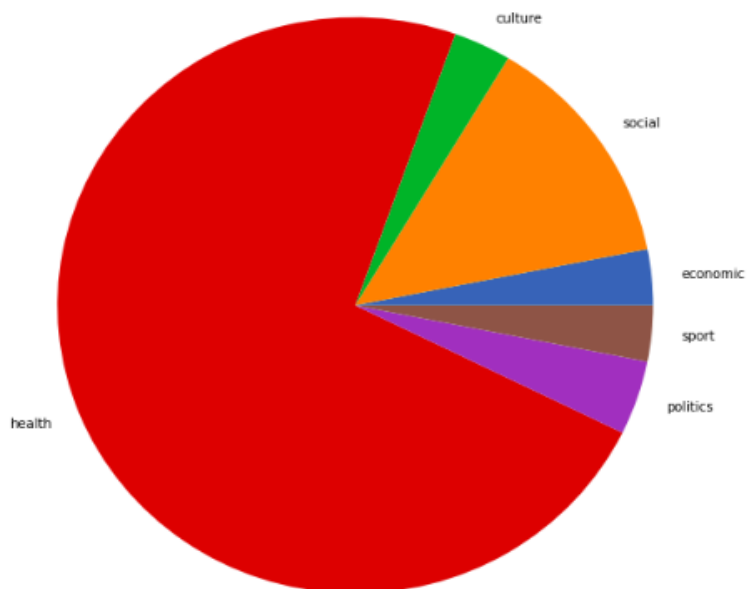


Figure 16: Représentation des volumes des tweets

L'énorme pourcentage en santé pourrait résulter de la pandémie actuelle, Covid19, tout le monde en parle donc un énorme volume de tweets.

Les tweets sociaux suivent, cela pourrait être lié à l'ensemble de mots définis comme des mots liés au social. La plupart de ces termes sont généraux, donc si un tweet qui était peut-être plus lié à l'économie pouvait avoir plus de mots sociaux que de mots d'économie et donc classé comme social, il s'agit donc principalement d'un biais dans la méthode de classification.



## Les tweets représentatifs de datasets total

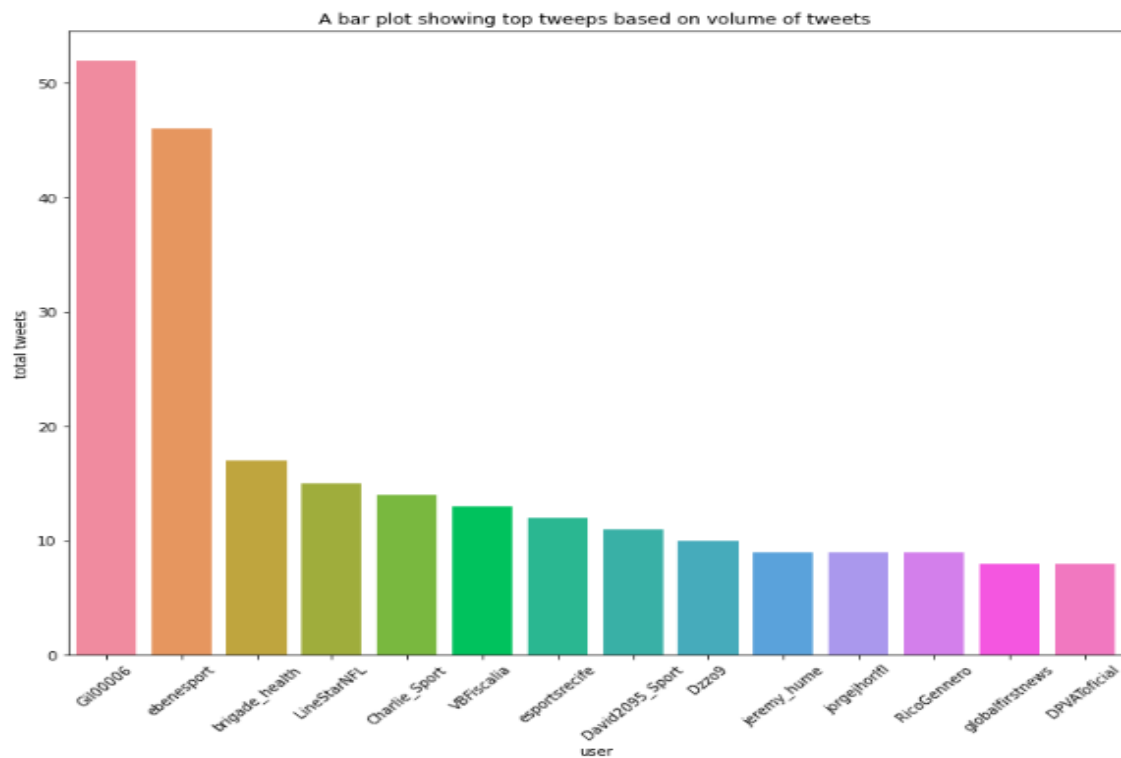


Figure 17 : Les tweets représentatifs de datasets total

## Les tweets représentatifs de catégorie sport

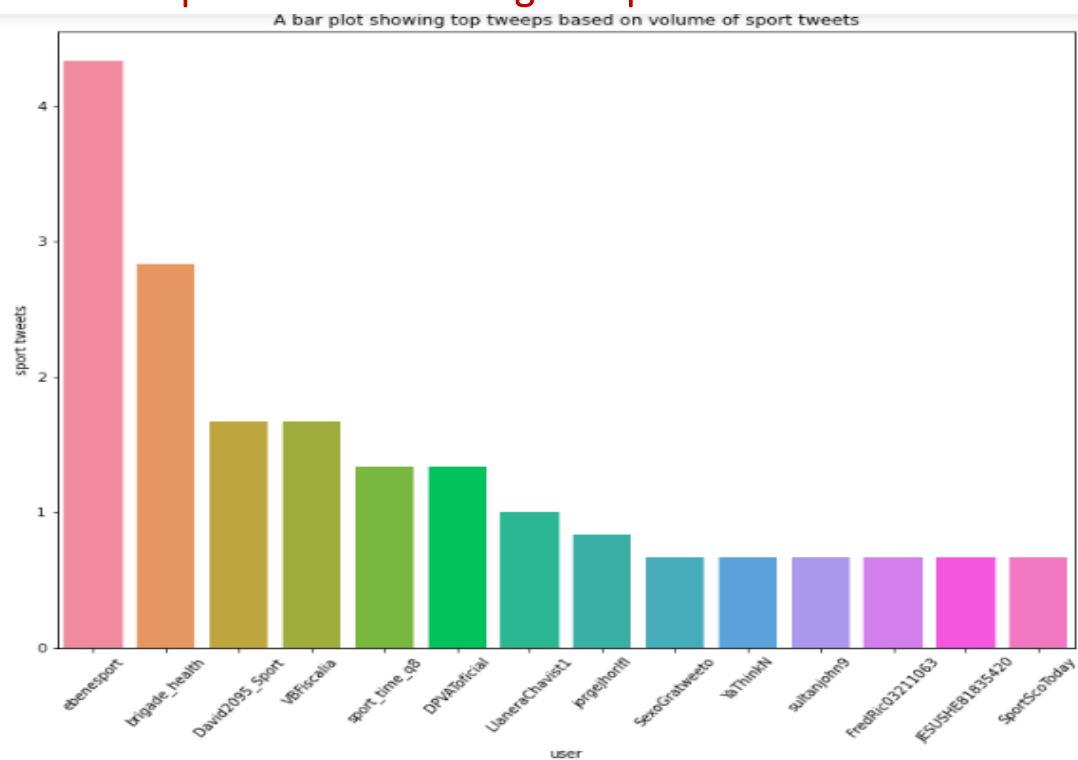


Figure 18 : Les tweets représentatifs de catégorie sport

## Les tweets représentatifs de catégorie politics

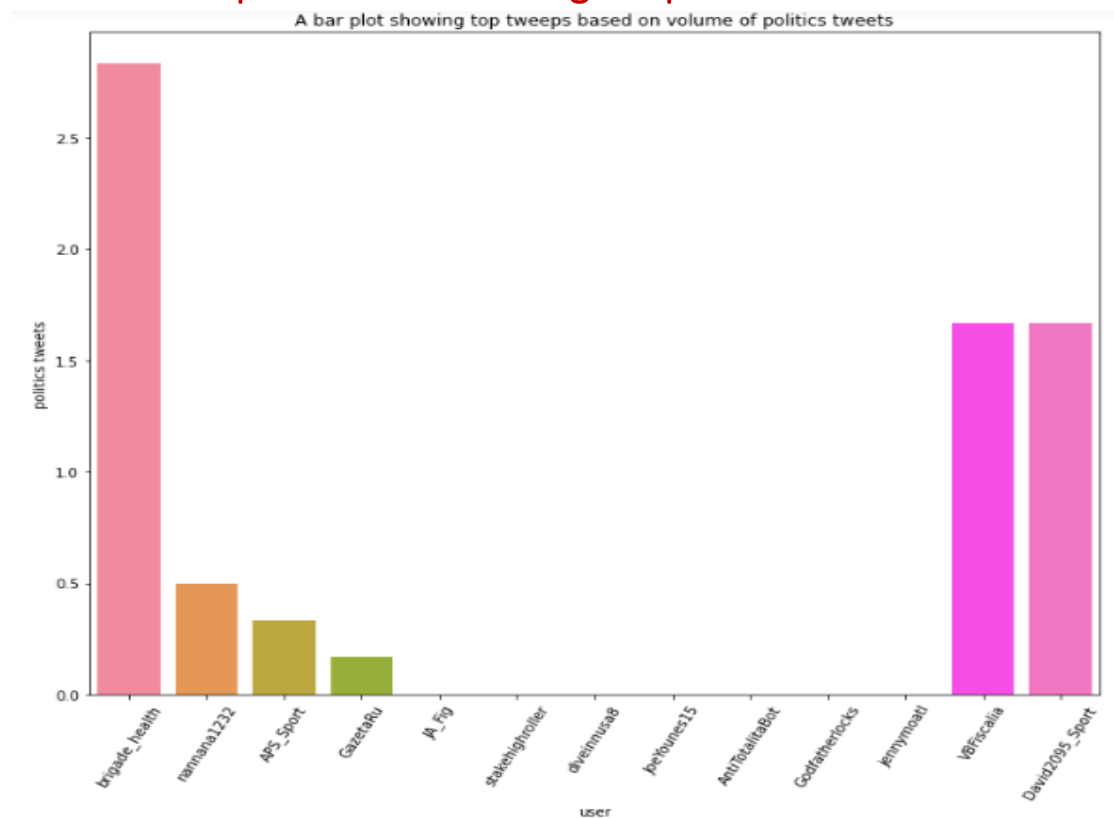


Figure 19 : Les tweets représentatifs de catégorie politics

## Les tweets représentatifs de catégorie economy

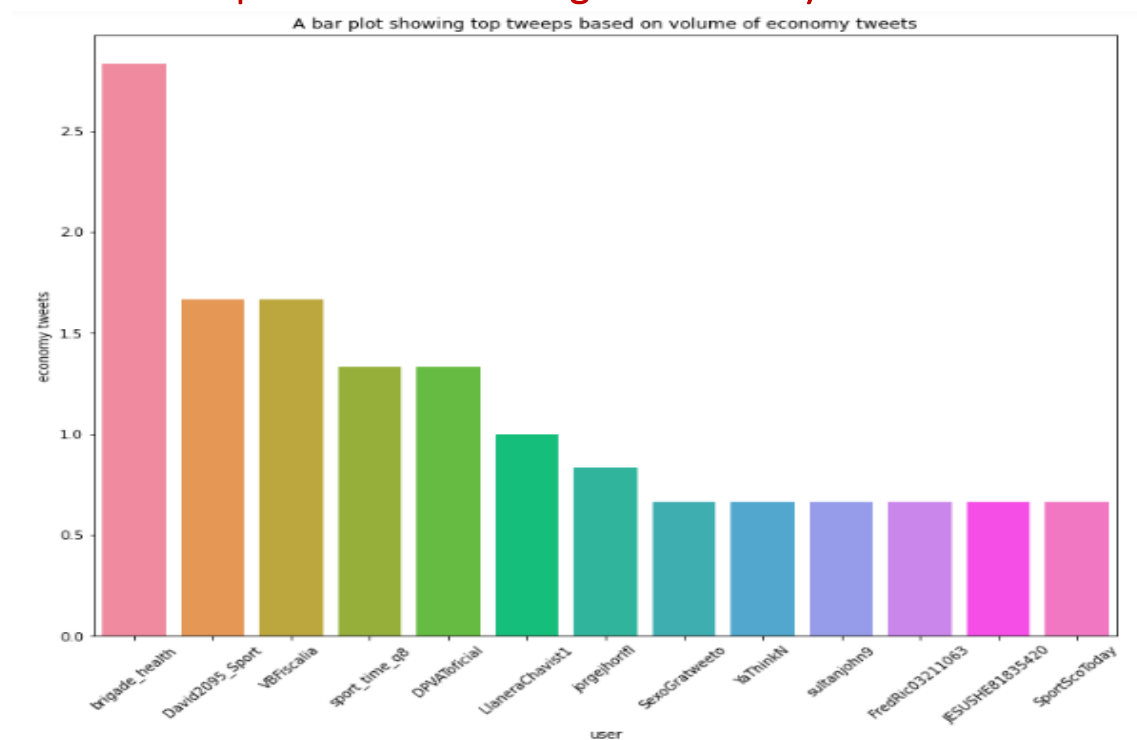


Figure 20 : Les tweets représentatifs de catégorie economy

## Les tweets représentatifs de catégorie social

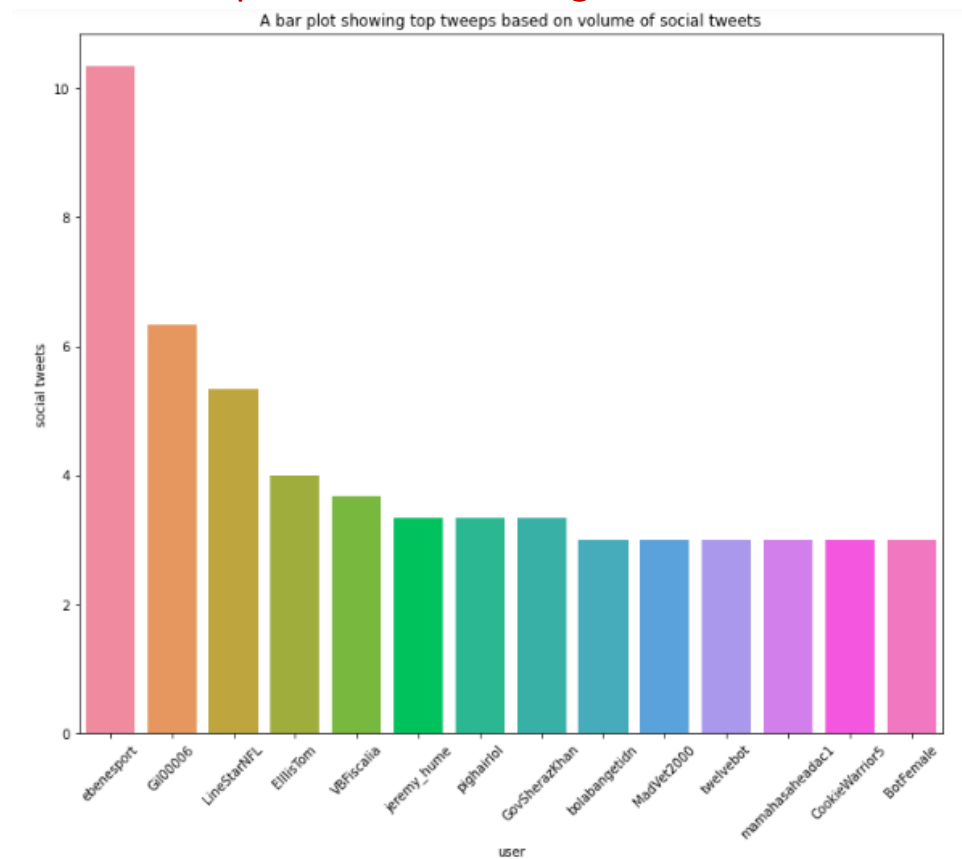


Figure 21 : Les tweets représentatifs de catégorie social

## Les tweets représentatifs de catégorie culture

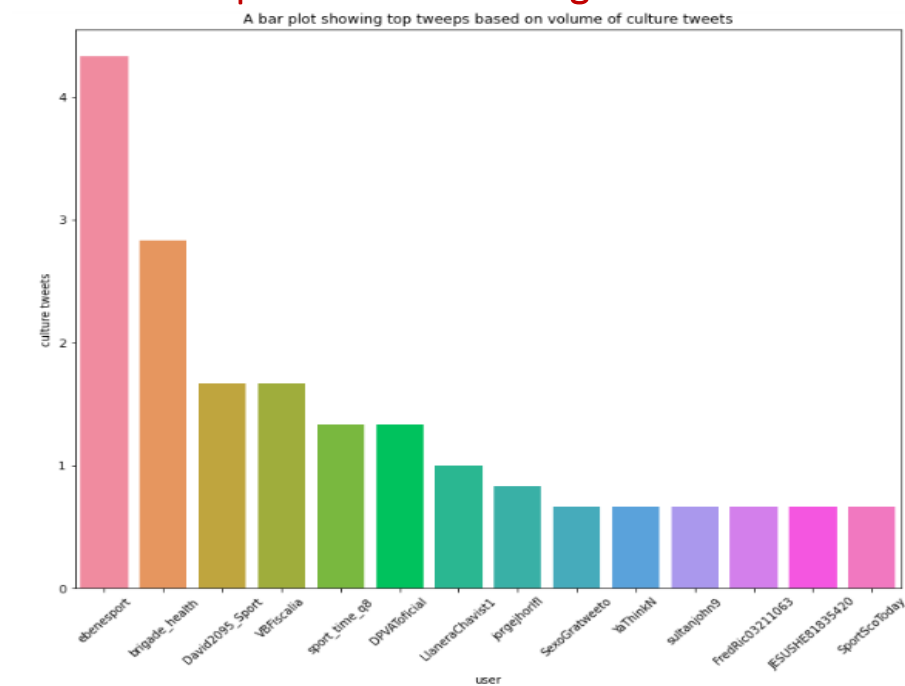


Figure 22 : Les tweets représentatifs de catégorie culture

## Les tweets représentatifs de catégorie health

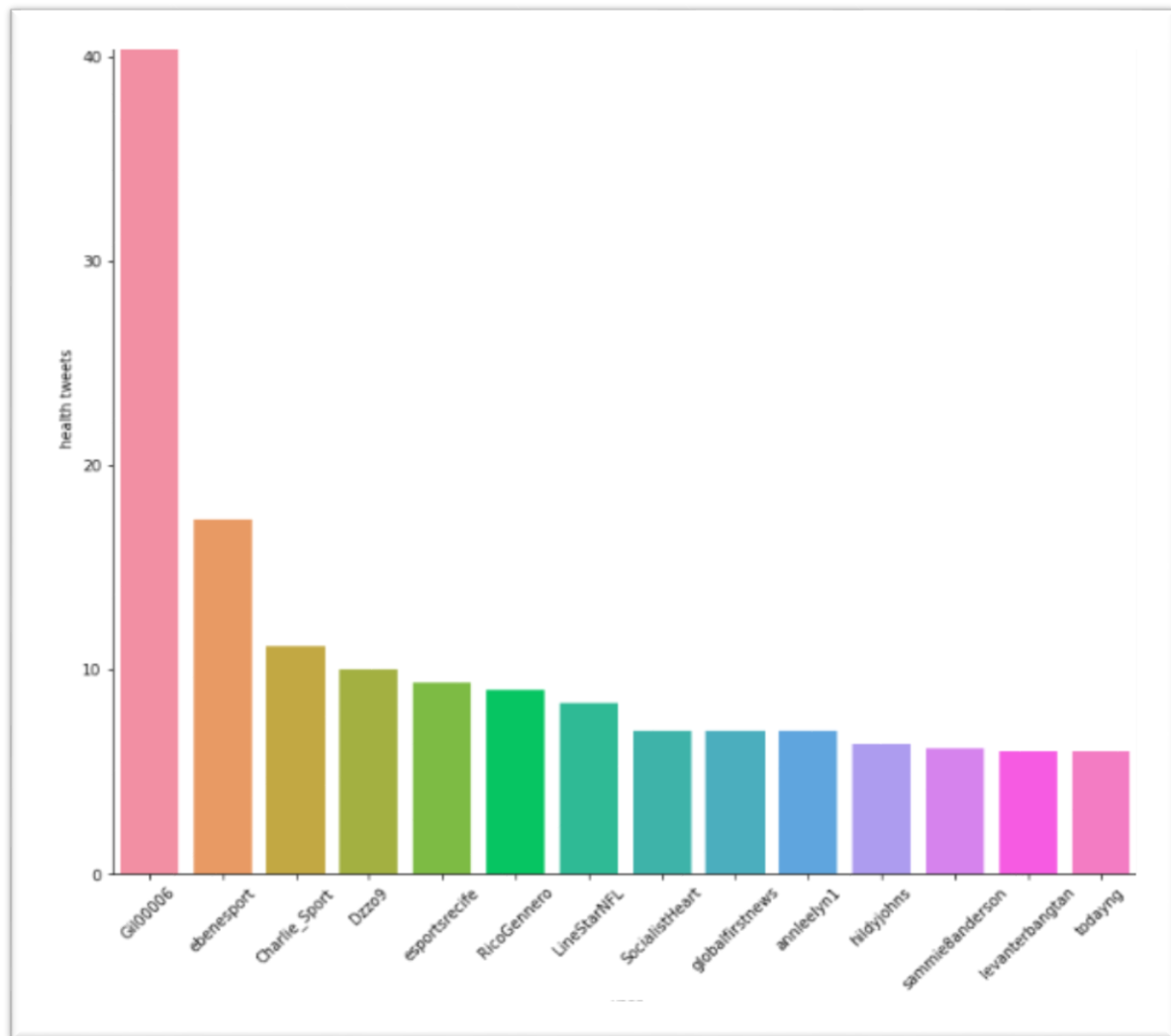


Figure 23 : Les tweets représentatifs de catégorie health

## Conclusion et perspectives

A partir d'une requête de collecte qui contient les mots : "sport, politics, Heath, Economy, social, culture " nous avons pu identifier des fils de discussions qui parlent d'événement mondial "covid19 sur Twitter, ce qui représente notre objectif principal. Aussi à la fin de notre processus nous arriverons à suggérer

automatiquement d'autres termes liés à cet événement qui peuvent enrichir la requête de collecte, pour le deuxième objectif de ce travail.

Notre contribution consiste à effectuer un regroupement (Clustering) sur le flux de Twitter en passant par une phase de nettoyage et une phase de conversion du texte vers des vecteurs numériques, en fonction de la présence des termes statistiquement.

Comme perspective, nous pouvons évoquer :

- Créer un jeu de test et de validation de grande taille (>1T tweets)
- Le passage vers le Clustering en ligne (en temps réel par des fenêtres temporelles).
- L'utilisation d'une méthode de transformation du texte à base sémantique comme LDA (Latent Dirichlet Allocation) ou LSA (Latent Semantic Analysis) est une autre perspective scientifique pour pouvoir la comparer avec l'approche statistique déjà appliquée.