

Extraction et Analyse des tweets sur le covid19

Plan

Introduction

Partie 1 : Spark Streaming & API Twitter (**covid19**)

Partie 2 : Analyse de sentiment des tweets sur le **vaccin du covid19**

Conclusion

Introduction

De nos jours, l'Internet se révèle plus que jamais un outil indispensable d'échange d'informations. Il nous offre une quantité considérable d'informations à une vitesse inédite et ses services s'adaptent de plus en plus aux besoins des internautes. Ceux-ci peuvent consulter l'Internet pour trouver de l'information, envoyer des e-mails, acheter des produits, lire des journaux en ligne, etc.

Pendant les dernières années, l'Internet a connu encore une plus vaste portée grâce au développement des médias sociaux. Basés sur des techniques de communication faciles et accessibles pour tous, ces médias favorisent les interactions sociales à travers l'Internet. Les médias sociaux se distinguent des médias traditionnels tels que les journaux, la télévision et la radio parce que leur utilisation est peu coûteuse et libre, de façon à permettre à tout le monde d'accéder à ou de publier de l'information. Ils subviennent aux besoins des individus d'échanger des opinions, de demander des conseils et de communiquer de façon rapide et facile.

Les médias sociaux qui ont récemment pu bénéficier d'un considérable essor sont les réseaux sociaux tels que Facebook, LinkedIn et Twitter. Ce sont des sites web qui rassemblent des identités sociales telles que des individus, des entreprises et des organisations qui peuvent échanger de l'information à travers des interactions sociales. Grâce à leur caractère maniable et leur accès libre, les réseaux sociaux bénéficient d'un succès croissant auprès du grand public.

Tous les récents développements dans le domaine d'échange d'informations et d'opinions ont donné un coup de volant aux applications informatiques conçues pour l'analyse et la détection de sentiments exprimés sur Internet. Présentée dans la littérature sous le nom d'opinion mining ou sentiment analysis, l'analyse des sentiments s'utilise entre autres pour la détection d'opinions sur des sites web et des réseaux sociaux, l'éclaircissement sur le comportement des consommateurs, la recommandation de produits et l'explication du résultat des élections. Elle consiste à rechercher des textes évaluatifs sur Internet tels que des critiques et des recommandations et à analyser de façon automatique ou manuelle les sentiments qui y sont exprimés afin de mieux comprendre l'opinion publique. L'analyse de sentiments des tweets de twitter est une des applications classiques du traitement du langage naturel, c'est le 'Hello World' du NLP.

Apache Spark, le moteur de traitement de données volumineuses à la mode qui offre des solutions plus rapides par rapport à Hadoop, peut être efficacement utilisé pour trouver des modèles de pertinence utiles à partir de ces sites. Il fournit des APIs de haut niveau en Java, Scala, Python et R, et un moteur optimisé qui supporte l'exécution des graphes. Il supporte également un ensemble d'outils de haut niveau tels que Spark SQL pour le support du traitement de données structurées, MLlib pour l'apprentissage des données, GraphX pour le traitement des graphes, et Spark Streaming pour le traitement des données en streaming.

Le projet comporte deux parties indépendantes :

- La première est une exploration en temps réel des données massives issues du réseau social Twitter, complétée d'une analyse statistique du contenu des tweets. Les données analysées sont des tweets portant sur le thème de covid19.
- Dans la deuxième partie de notre projet, l'idée sera de pouvoir faire de l'analyse de sentiments d'un tweet à partir des mots utilisés. Ce qui constituera une métrique de l'angoisse globale liée au Covid-19 qui règne sur Twitter, en fonction de ce qui y est publié.

Partie 1 : Spark Streaming & API Twitter

De nos jours, les données augmentent et s'accumulent plus rapidement que jamais. Actuellement, environ 90 % de toutes les données générées dans notre monde ne l'ont été qu'au cours des deux dernières années. En raison de ce taux de croissance fulgurant, les plateformes de big data ont dû adopter des solutions radicales afin de maintenir des volumes de données aussi importants. L'une des principales sources de données aujourd'hui sont les réseaux sociaux.

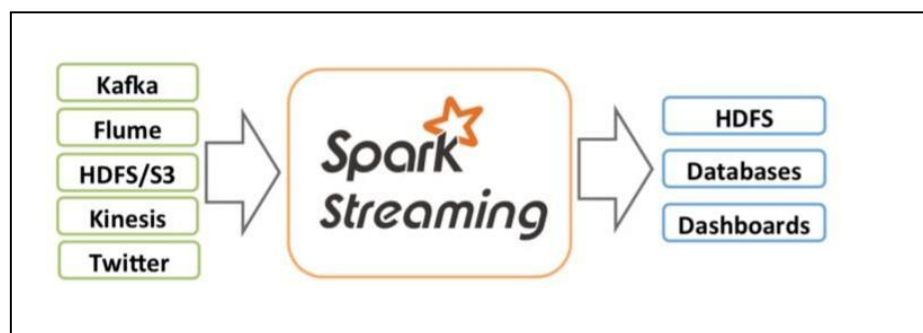
Apache Spark est un cadre efficace pour analyser de grandes quantités de données en temps réel et effectuer diverses analyses sur les données. Ce système de traitement offre la possibilité de calculs continus, car les données le traverse en permanence.

De nombreuses ressources discutent de Spark et de sa popularité dans le domaine du Big Data, mais il convient de souligner que ses principales fonctionnalités incluent le traitement en temps réel du Big Data à l'aide d'ensembles de données distribués résilients (RDD), le streaming et l'apprentissage automatique.

Les API sont la façon dont les programmes informatiques « communiquent » entre eux afin qu'ils puissent demander et fournir des informations. Cela se fait en permettant à une application logicielle d'appeler ce qu'on appelle un point de terminaison : une adresse qui correspond à un type spécifique d'informations que nous fournissons. Twitter autorise l'accès à certaines parties de notre service via des API pour permettre aux utilisateurs de créer un logiciel qui s'intègre à Twitter, comme une solution qui aide une entreprise à répondre aux commentaires des clients sur Twitter.

Spark streaming

Spark est connu pour supporter également le traitement des données en streaming. Les données peuvent être lues à partir de plusieurs sources tel que Kafka, Flume, Twitter ou des sockets TCP, et peuvent être traitées en utilisant des algorithmes complexes. Ensuite, les données traitées peuvent être stockées sur des systèmes de fichiers, des bases de données ou des dashboards. Il est même possible de réaliser des algorithmes de machine learning et de traitement de graphes sur les flux de données.



Spark streaming

En interne, il fonctionne comme suit : Spark Streaming reçoit des données en streaming et les divise en micro-batches, qui sont ensuite calculés par le moteur de spark pour générer le flux final de résultats.



Fonctionnement de spark streaming

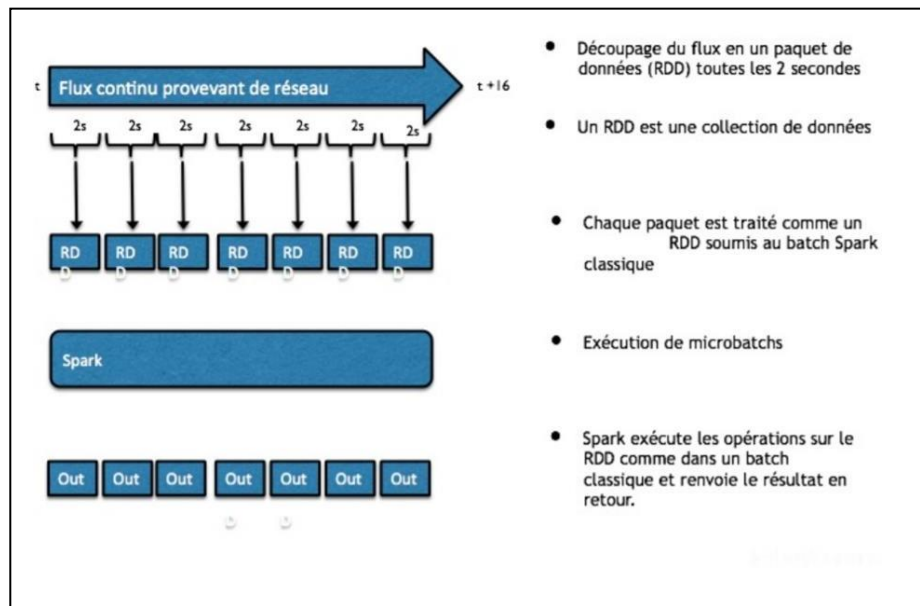
RDD ou Resilient Distributed Datasets, est une collection d'enregistrements avec calcul distribué, qui sont tolérants aux pannes, de nature immuable. Ils peuvent être utilisés en parallèle avec des API de bas niveau, tandis que leur fonctionnalité paresseuse permet à l'opération d'étincelle de fonctionner à une vitesse améliorée. Les RDD prennent en charge deux types d'opérations :

- **Transformations** : opérations paresseuses qui renvoient un autre RDD, ce RDD ne calcule pas à moins qu'une action ne soit effectuée dessus. Quelques exemples de transformations sont map(), flatmap(), filter().



- **Actions** : opérations qui déclenchent le calcul et renvoient des valeurs. Quelques exemples d'actions sont count, top(), savetofile()





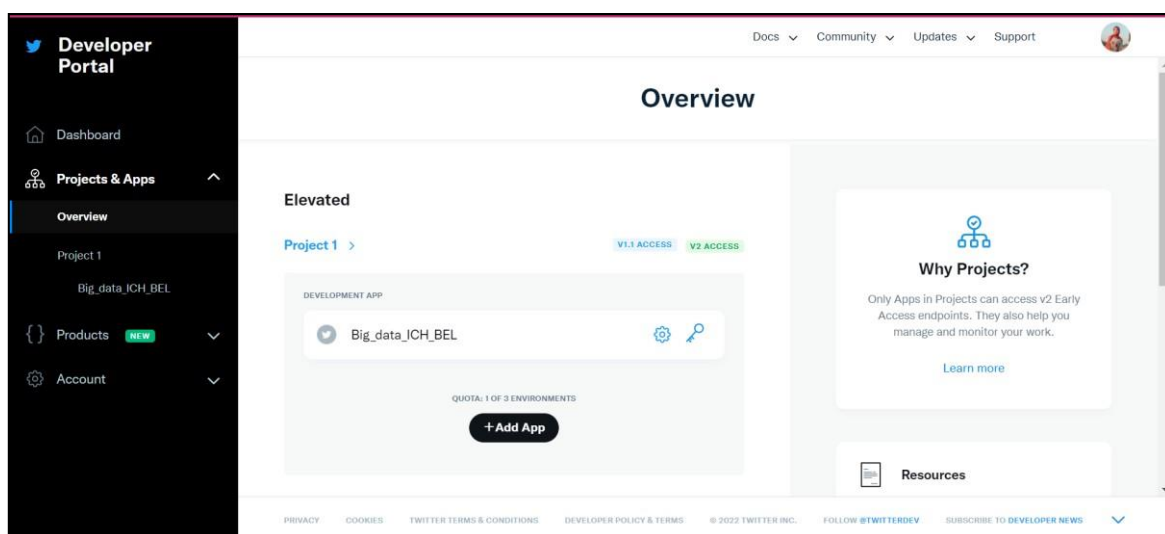
RDD Spark streaming

Mise en œuvre

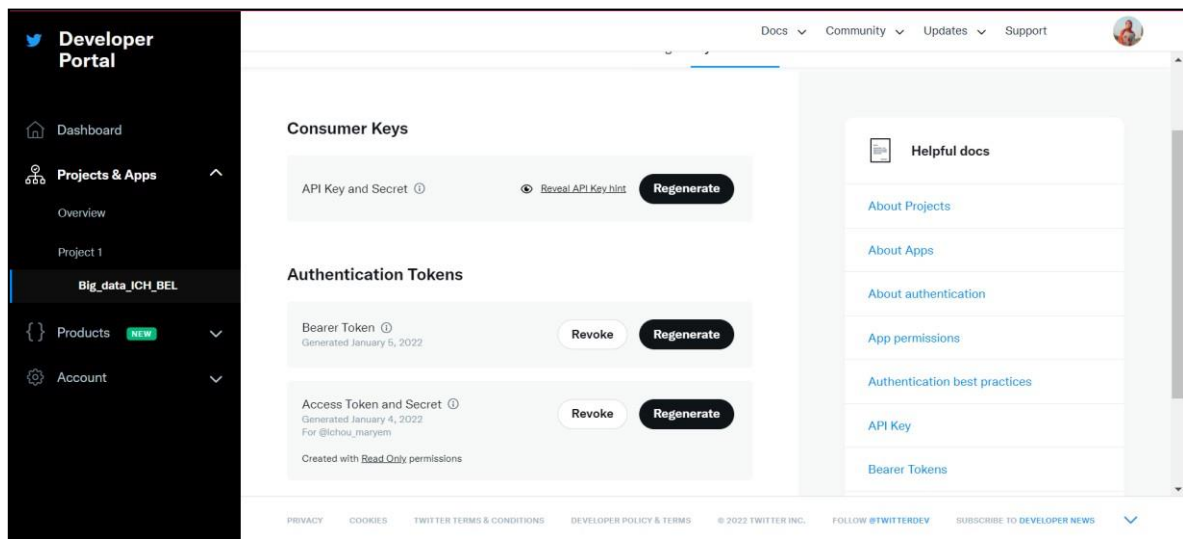
Nous avons créé une application simple qui lit les flux en ligne de Twitter à l'aide de Python, puis traite les tweets à l'aide d'Apache Spark Streaming pour identifier les hashtags et, enfin, renvoie les hashtags les plus populaires et visualise ces données.

Création de nos propres informations d'identification pour les API Twitter :

Afin d'obtenir des tweets de Twitter, nous avons créé notre application Twitter puis nous avons demandé un accès « **Elevator** » v2 access.



Après, nous avons obtenu nos clés d'authentification



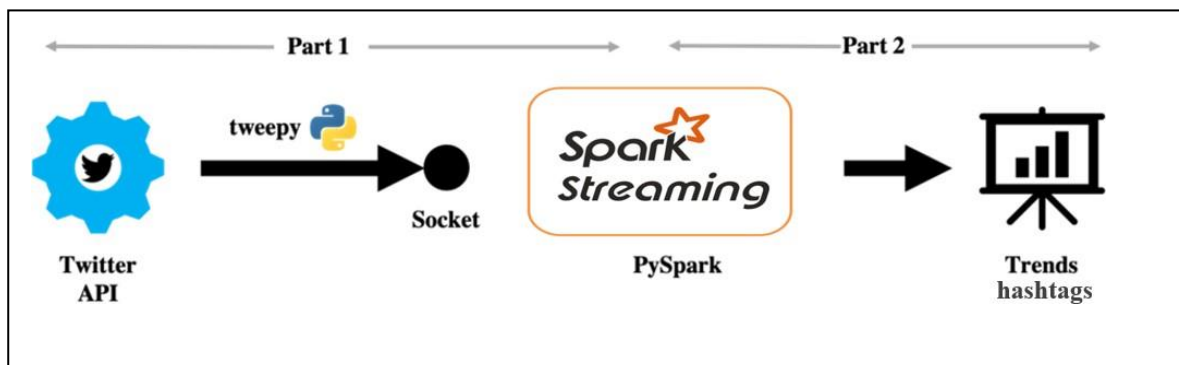
consumer_key= 'IsNmYLpt3SrvWN3EoiKheA1JJ'

consumer_secret= 'ZI6tUVTBnB3aVFiQWQXGnxzYndHcOmD2S0I7sfQEk6Ct7N7oNn'

access_token= '1476928546404806658-KcF0D70h8c8PAnsHygmY1mMidhqFNJ'

access_secret= 'qwEPD5kPxYC36wxPFkHh873pE7vPKiZA9fTHrWAE2YG7s'

Architecture :



Architecture

Nous avons utilisé Tweepy pour accéder à l'API de streaming de Twitter et au composant de streaming Spark avec socket TCP pour recevoir des tweets. Nous avons superposé les tweets sur RDD, puis récupérer les hashtags les plus populaires liés au thème covid19. Après cela, nous avons utilisé Spark SQL pour enregistrer les 10 meilleurs hashtags dans une base de données temporaire. Enfin, nous visualisons les résultats à l'aide des outils de visualisation de Python.

Il y a deux parties :

- 1/ Extraction des tweets par l'API Twitter.
- 2/ Prétraitement des tweets et recherche de #tags tendance.

I. Récupérer des tweets à partir de l'API Twitter

Nous avons utilisé nos informations d'identification de développeur pour nous authentifier et nous connecter à l'API Twitter. Nous avons créé également un socket TCP entre l'API de Twitter et Spark, qui attend l'appel du Spark Structured Streaming puis envoie les données Twitter. Ici, nous avons utilisé la bibliothèque Tweepy de Python pour nous connecter et obtenir les tweets de l'API Twitter.

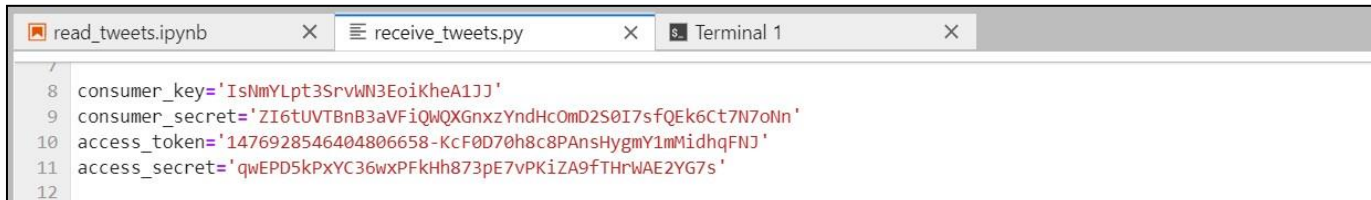
a) Importez-les packages nécessaires



```
1 import tweepy
2 from tweepy.auth import OAuthHandler
3 from tweepy import Stream
4 from tweepy.streaming import StreamListener
5 import socket # Le module socket est requis pour communiquer avec l'API Twitter sur notre machine locale.
6 import json # Pour travailler avec des objets json, nous aurons besoin du module json.
7
```

Nous utiliserons *tweepy*, un module Python qui se connecte à l'API Twitter, ainsi que les deux classes *Stream*, *StreamListener* pour créer le flux et *OAuthHandler* pour s'authentifier sur Twitter, tout ça pour créer notre pipeline de streaming de données dans *receive_tweets.py*. Le module *socket* est nécessaire pour communiquer avec l'API Twitter sur notre machine locale. Pour travailler avec des objets json, nous aurons besoin du module *json*.

b) Insérez nos identifiants



```
8 consumer_key='IsNmYLpt3SrvWN3EoiKheA1JJ'
9 consumer_secret='ZI6tUVTBnB3aVfiQWQXGnxzYndHcOmD2S0I7sfQEk6Ct7N7oNn'
10 access_token='1476928546404806658-KcF0D70h8c8PAnsHygmY1mMidhqFNJ'
11 access_secret='qWEpD5kPxYC36wxPFkHh873pE7vPKiZA9fThrWAE2YG7s'
12
```

c) Créer une instance StreamListener

```
read_tweets.ipynb x receive_tweets.py x Terminal 1 x
19 class TweetsListener(StreamListener):
20
21     def __init__(self, csocket):
22         self.client_socket = csocket
23         # we override the on_data() function in StreamListener
24     def on_data(self, data):
25         try:
26             message = json.loads( data )
27             print( message['text'].encode('utf-8') )
28             self.client_socket.send( message['text'].encode('utf-8') )
29             return True
30         except BaseException as e:
31             print("Error on_data: %s" % str(e))
32             return True
33
34     def if_error(self, status):
35         print(status)
36         return True
```

TweetListener représente une instance de StreamListener qui se connecte à l'API Twitter et renvoie un tweet à la fois. Dès que nous activons le Stream, il crée continuellement des instances.

La classe se compose de 3 méthodes ; le on_data, le on_errort et le __init__.

La méthode on_data lit le fichier JSON de tweet entrant, qui contient un tweet, et définit la partie à conserver. Certains exemples de parties peuvent être le message de tweet, les commentaires ou les hashtags. Dans notre cas, nous voulons extraire le texte réel du tweet. If_error s'assure que le flux fonctionne et __init__ configure le socket de l'API Twitter.

d) Envoyer des données depuis Twitter

```
39 def send_tweets(c_socket):
40     auth = OAuthHandler(consumer_key, consumer_secret)
41     auth.set_access_token(access_token, access_secret)
42     # commencer à envoyer des données depuis l'API Streaming
43     twitter_stream = Stream(auth, TweetsListener(c_socket))
44     twitter_stream.filter(track=['covid19']) #Nous avons choisi le thème du covid19
```

Afin de récupérer les données de l'API Twitter, nous devons d'abord authentifier notre connexion à l'aide des informations d'identification prédéfinies. Après authentification, nous diffusons les objets de données de tweet contenant notre mot-clé. TweetListener renvoie les tweets sous forme d'objets.

e) Lancez la diffusion en continu

```
49 if __name__ == "__main__":
50     new_skt = socket.socket()          # initiate a socket object
51     host = "192.168.1.102"            # local machine address
52     port = 5555                       # specific port for your service.
53     new_skt.bind((host, port))        # Binding host and port
54
55     print("Now listening on port: %s" % str(port))
56
57     new_skt.listen(5)                 # waiting for client connection.
58     c, addr = new_skt.accept()        # Establish connection with client. it returns first a socket object, c, and the address bound to the
socket
59
60     print("Received request from: " + str(addr))
61     # and after accepting the connection, we will send the tweets through the socket
62     send_tweets(c)
63
```

La diffusion en continu des données à partir de l'API Twitter nécessite la création d'un socket TCP d'écoute dans la machine locale (serveur) sur une adresse IP localhost (192.168.1.102) et un port prédéfini (Nous avons choisi le port : 5555). Un socket se compose d'un côté serveur, qui est notre machine locale, et d'un côté client, qui est l'API Twitter. Le socket ouvert du côté serveur écoute les connexions du client. Lorsque le côté client est opérationnel, le socket reçoit les données de l'API Twitter en fonction du sujet ou des mots-clés définis dans l'instance StreamListener, et transmet sa réponse avec la connexion socket à send_tweets pour envoyer les tweets à Spark.

II. Prétraitement des tweets et recherche de #tags tendance associées au covid19

a) Importez-les packages nécessaires



```
[1]: # Localiser Spark sur notre machine locale à l'aide de la bibliothèque findspark,
import findspark
findspark.init()
import pyspark

[2]: # import necessary packages
from pyspark import SparkContext # SparkContext est le point d'entrée de toute fonctionnalité de spark.
from pyspark.streaming import StreamingContext # pour traiter des flux de données qui arrivent en continu
from pyspark.sql import SQLContext
from pyspark.sql.functions import desc
```

Nous avons utilisé pyspark, qui est l'API Python pour Spark. Nous utilisons Spark Structured Streaming, qui est un moteur de traitement de flux construit sur le moteur Spark SQL, c'est pourquoi nous importons le pyspark.sql module. Nous importons également ses classes. Le findspark pour localiser spark sur notre machine locale.

b) Lancer SparkContext

```
[3]: # SparkContext est le point d'entrée pour toutes les fonctions Spark
sc = SparkContext()
# we initiate the StreamingContext with 10 second batch interval. #next we initiate our sqlcontext
ssc = StreamingContext(sc, 10)
sqlContext = SQLContext(sc)
```

Nous avons lancé le SparkContext. SparkContext est le point d'entrée pour toutes les fonctions spark. Lorsque nous exécutons une application Spark, un programme pilote s'exécute, qui sert de fonction principale, et le SparkContext est lancé ici. Les SparkContexts représentent une connexion à un cluster Spark et peuvent être utilisés pour créer des RDD, des accumulateurs et des variables de diffusion sur ce cluster. Ici, SparkContext génère un JavaSparkContext. À tout moment, un seul SparkContext peut être actif. Nous lançons ensuite StreamingContext() avec des intervalles de 10 secondes, ce qui signifie que les flux d'entrée seront divisés en lots toutes les 10 secondes pendant le processus de diffusion.

```
# initiate streaming text from a TCP (socket) source:
socket_stream = ssc.socketTextStream("192.168.1.102", 5555)
# lines of tweets with socket_stream window of size 60, or 60 #seconds windows of time
lines = socket_stream.window(60)
```

Ensuite, nous avons attribué la source d'entrée de diffusion en continu, puis configuré les données entrantes en lignes.

Nous utilisons le même numéro de port (5555) que nous avons spécifié dans la première partie pour envoyer les tweets et nous utilisons la même adresse IP puisque nous exécutons l'application localement. Nous utilisons également la fonction window() pour déterminer que nous analysons les tweets toutes les minutes (60 secondes) pour identifier les 10 premiers #tags.

Nettoyage des tweets qui commencent par # et enregistrement des 10 premiers tweets dans une table SQL temporaire « tweets » :

```
[5]: # nettoyé les tweets qui commencent par # et enregistrer les 10 premiers tweets dans une table SQL temporaire
from collections import namedtuple
fields = ("hashtag", "count")
Tweet = namedtuple('Tweet', fields)
# here we apply different operations on the tweets and save them to a temporary sql table
( lines.flatMap( lambda text: text.split(" ") ) ) #Splits to a list
# Checks for hashtag calls
.filter( lambda word: word.lower().startswith("#") )
.map( lambda word: ( word.lower(), 1 ) ) # Lower cases the word
.reduceByKey( lambda a, b: a + b ) # toutes les valeurs de la même clé seront additionnées
.map( lambda rec: Tweet( rec[0], rec[1] ) ) # stockage dans un tweet object
.foreachRDD( lambda rdd: rdd.toDF().sort( desc("count") ) ) # Trier le tweet Object
.limit(10).registerTempTable("tweets") ) # Enregistrer seulement 10 hashtags dans une table SQL temporaire
```

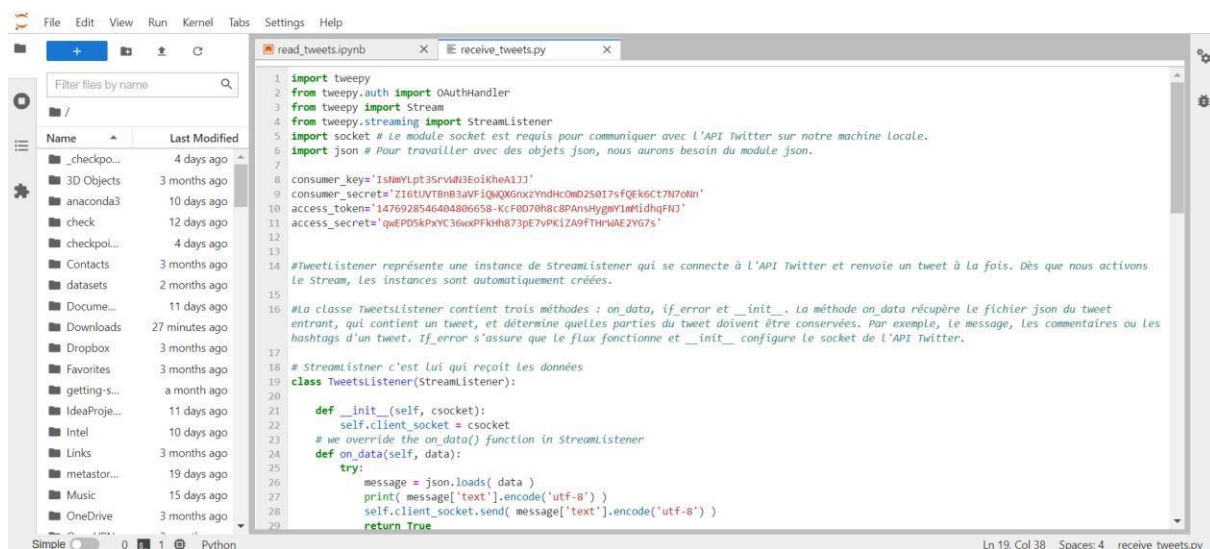
Pour enregistrer le nombre de balises, nous avons créé un objet namedtuple. Ensuite, nous avons utilisé flatmap() pour générer un tableau de tweets. Nous avons divisé tous les tweets en mots et les mettre en mots RDD. Ensuite, nous filtrerons uniquement les hashtags de tous les mots et les mapperons sur une paire de (hashtag, 1). Nous utilisons les fonctions lambda car elles nécessitent moins de mémoire et s'exécutent plus rapidement. Nous filtrons ensuite les tweets qui ne commencent pas par #. Ensuite, nous devons calculer combien de fois le hashtag a été mentionné. Nous pouvons le faire en utilisant la fonction reduceByKey. Cette fonction calculera combien de fois le hashtag a été mentionné pour chaque lot, c'est-à-dire qu'elle réinitialisera les comptes dans chaque lot.

La fonction foreachRDD() dans pySpark est une fonction importante qui permet un traitement plus rapide des RDD. Il est appliqué à chaque RDD pour le convertir en une trame de données et est ensuite stocké dans une table temporaire intitulée "tweets".

```
[6]: # start streaming and wait couple of minutes to get enough tweets
    ssc.start()
```

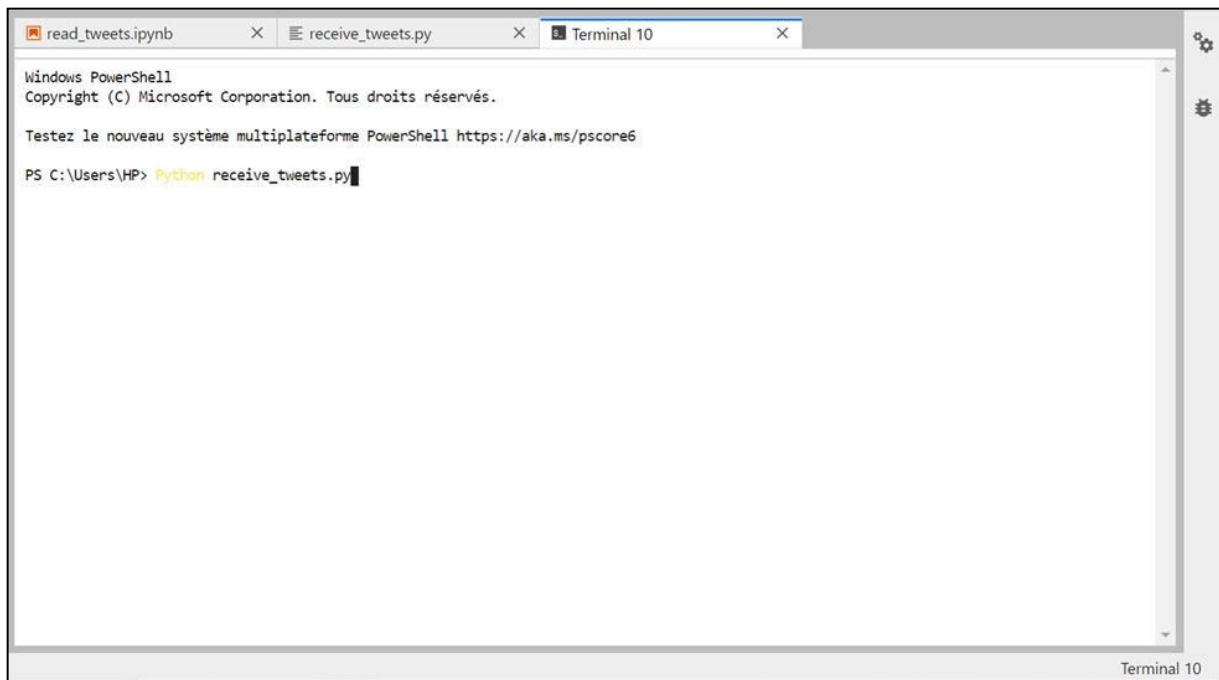
Exécution du code

JupyterLab est un environnement de développement interactif pour travailler avec des blocs-notes, du code et des données. Plus important encore, JupyterLab prend entièrement en charge les notebooks Jupyter. De plus, JupyterLab vous permet d'utiliser des éditeurs de texte, des terminaux, des visualiseurs de fichiers de données et d'autres composants personnalisés côte à côte avec des blocs-notes dans une zone de travail à onglets.



JupyterLab interface

Pour l'exécution, nous avons travaillé. Nous avons exécuté **receive_tweets.py** en premier dans le terminal de jupyterLab en tapant 'python receive_tweets.py' et après cela, nous pouvons commencer à diffuser en exécutant les cellules du notebook **read_tweets.ipynb**.



```
read_tweets.ipynb X receive_tweets.py X Terminal 10 X
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

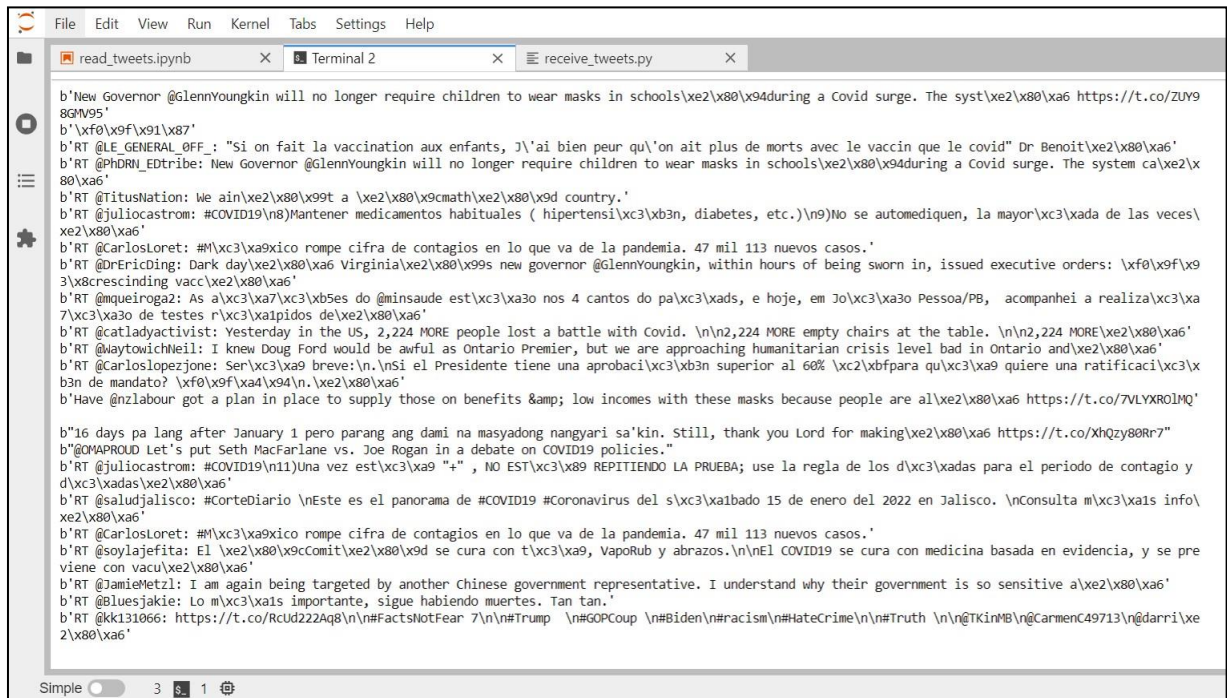
PS C:\Users\HP> python receive_tweets.py
```

Exécution du fichier python receive_tweets.py

```
PS C:\Users\HP> python receive_tweets.py
Now listening on port: 5555
Received request from: ('192.168.1.60', 56939)
b'2\n\nAre you part of the problem Tice?\n\nIs any politician, currently calling for the removal of 1 man, part of the pr\xe2\x80\xa6 htt
ps://t.co/dLqsYHn95S'
b'RT @MarkoCortes: Le deseamos pronta recuperaci\xf3n, presidente @lopezobrador_.'
```

Recevoir de tweets

Résultats



The screenshot shows a Jupyter Notebook interface with a terminal window open. The terminal displays a continuous stream of tweets, including mentions of the new Governor of Virginia, @GlennYoungkin, and various COVID-19 related news and discussions. The tweets are displayed in a monospaced font, with some lines wrapped. The terminal window has tabs for 'read_tweets.ipynb', 'Terminal 2', and 'receive_tweets.py'.

Tweets diffusés en continu

Visualisation de la table temporaire SQL « tweets » :

```
[7]: sqlContext.sql("SELECT * FROM tweets")

[7]: DataFrame[hashtag: string, count: bigint]

[13]: sqlContext.sql("SELECT * FROM tweets").show()

+-----+-----+
|      hashtag|count|
+-----+-----+
|      #covid19|   48|
|      #covid19|    7|
|      #omicron|    5|
|      #covid19|    4|
|      #corona|    3|
|      #covid|    3|
| #vaccinessavelives|    3|
|      #últimahora|    2|
|      #omicronvariant|    2|
|      #mumbai|    2|
+-----+-----+
```

Table temporaire SQL

Si nous réexécutons cette requête, nous allons remarquer que la table à changer puisque les tweets sont diffusés en temps réel, donc c'est tout à fait logique.

```
[7]: sqlContext.sql("SELECT * FROM tweets")

[7]: DataFrame[hashtag: string, count: bigint]

[14]: sqlContext.sql("SELECT * FROM tweets").show()

+-----+-----+
|      hashtag|count|
+-----+-----+
|      #covid19|   52|
|      #covid19,|    9|
|      #covid|    5|
|      #omicron|    5|
|      #covid-19|    4|
|      #covid19.|    3|
|      #últimahora|    3|
|      #corona|    3|
|      #vaccinessavelives|    3|
|      #pune|    2|
+-----+-----+
```

Réexécution de la table temporaire SQL

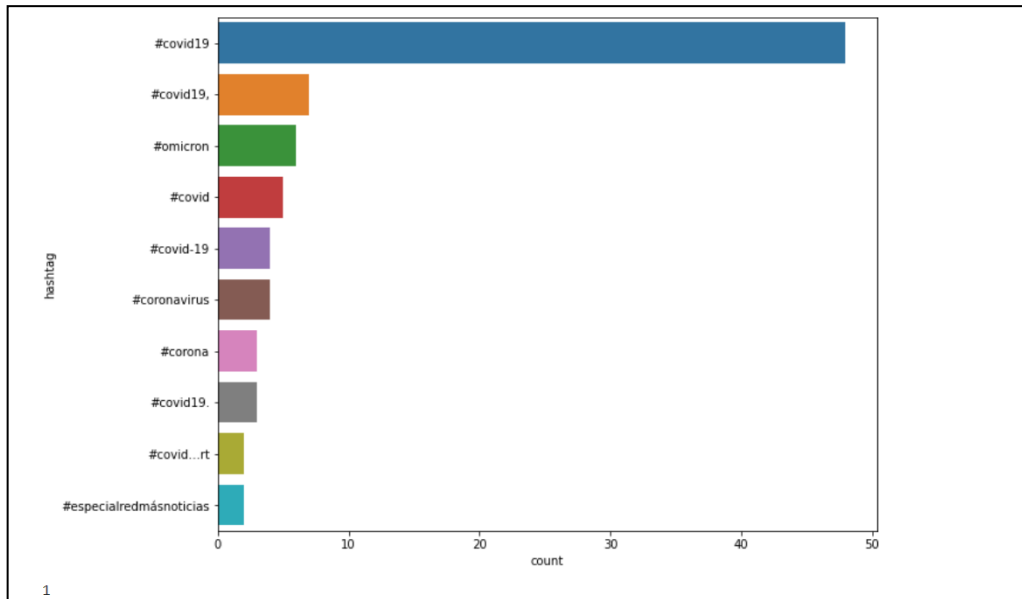
Analyse des hashtags :

Ensuite, nous importons nos bibliothèques de visualisation et dessinons le graphique des 10 meilleurs #tags pertinents pour le thème du « covid19 ». Nous exécutons notre streaming toutes les minutes pour vérifier les 10 meilleurs #tags seulement quelques (5) fois à des fins d'apprentissage.

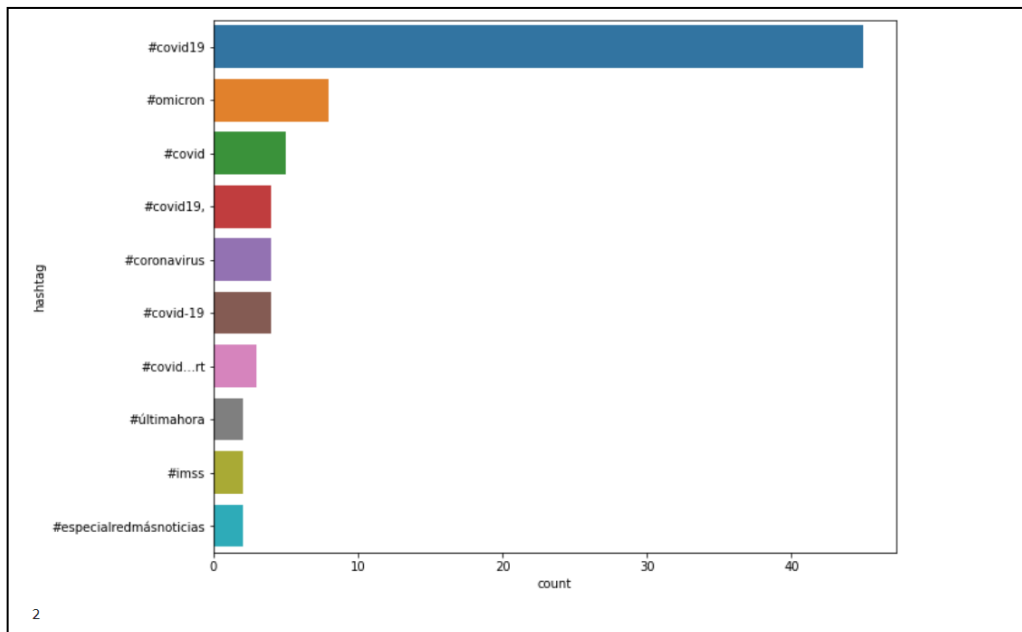
```
[15]: # import libraries to visualize the results
import time
from IPython import display
import matplotlib.pyplot as plt
import seaborn as sns
import pandas
get_ipython().run_line_magic('matplotlib', 'inline')
count = 0
while count < 5:

    time.sleep(5)
    top_10_tags = sqlContext.sql( 'Select hashtag, count from tweets' )
    top_10_df = top_10_tags.toPandas()
    display.clear_output(wait=True)
    plt.figure( figsize = ( 10, 8 ) )
    sns.barplot( x="count", y="hashtag", data=top_10_df)
    plt.show()
    count = count + 1
    print(count)
```

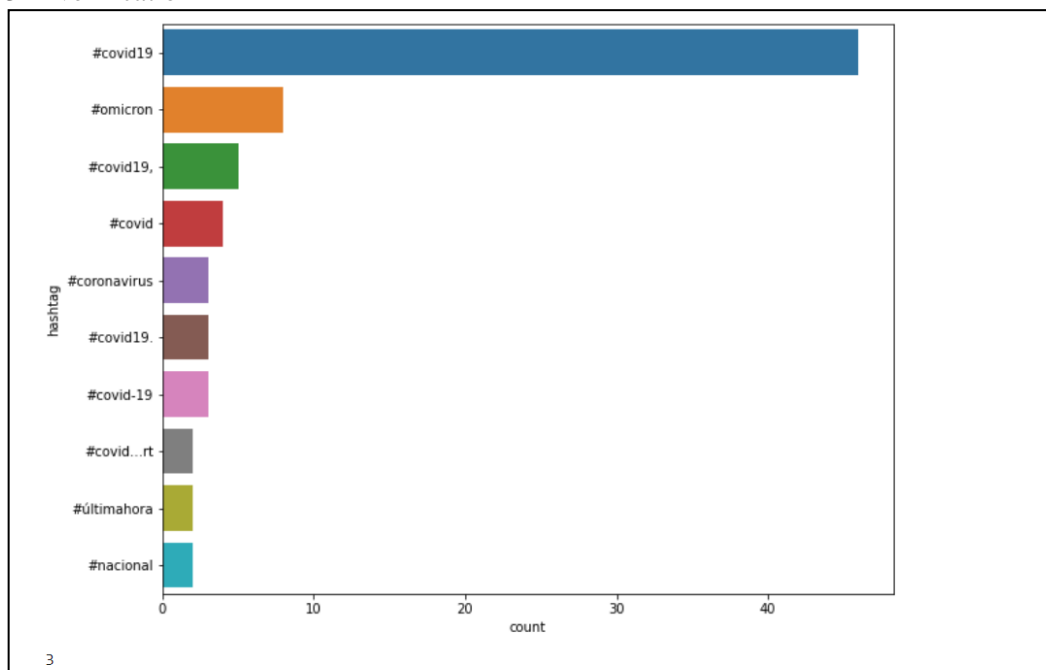

1^{ère} vérification



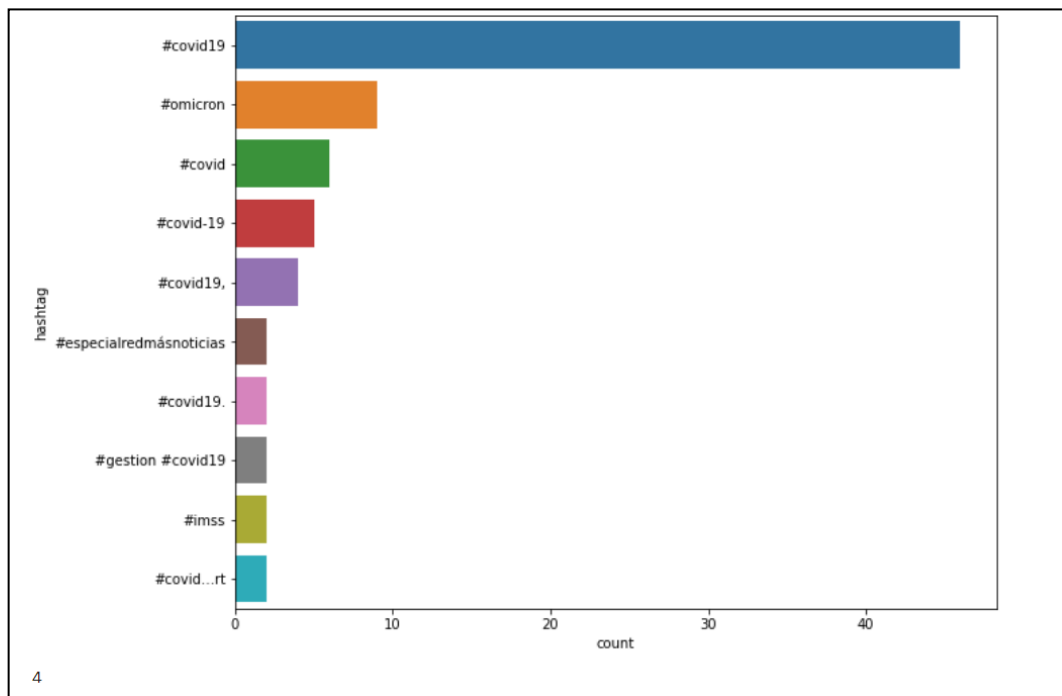
2^{ème} vérification



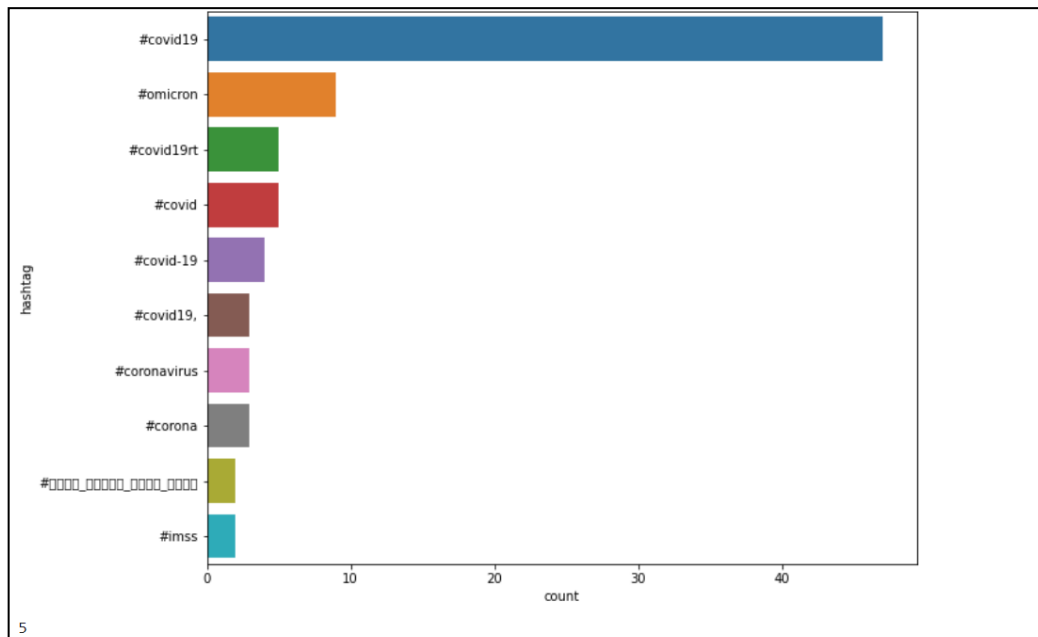
3^{ème} vérification



4^{ème} vérification



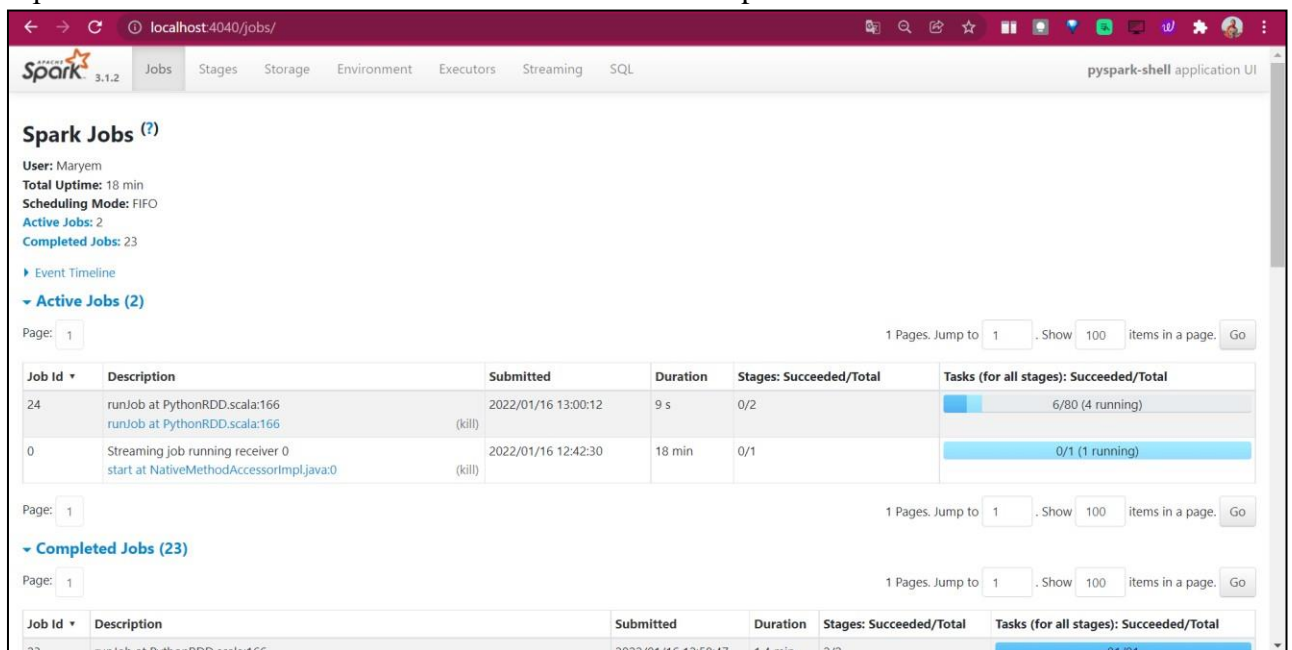
5^{ème} vérification



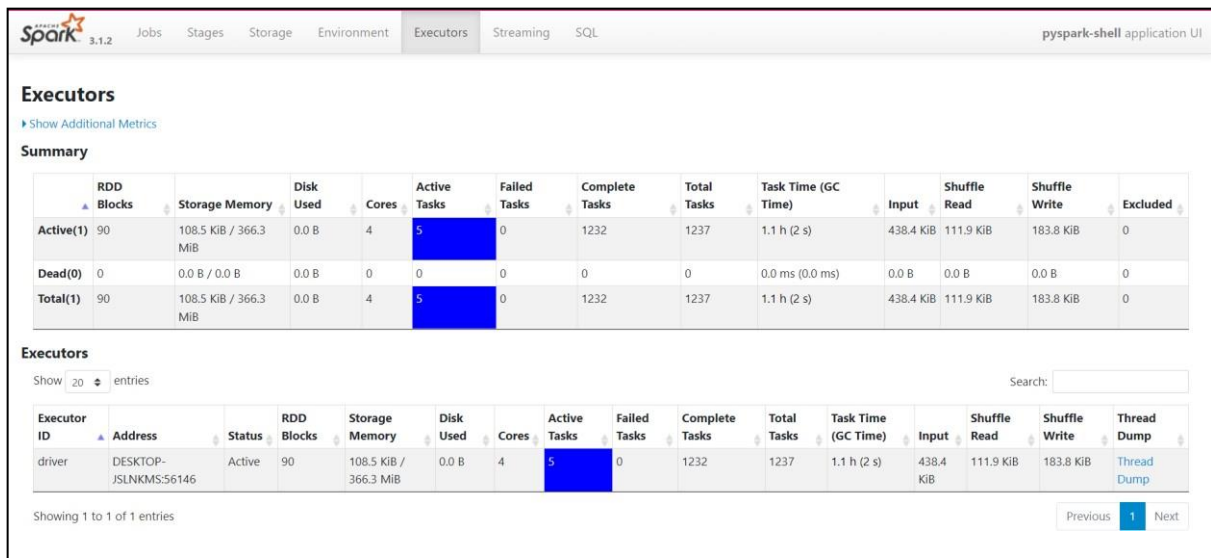
Nous remarquons très bien que le traitement se fait sur les données en temps réel.

Interface graphique spark

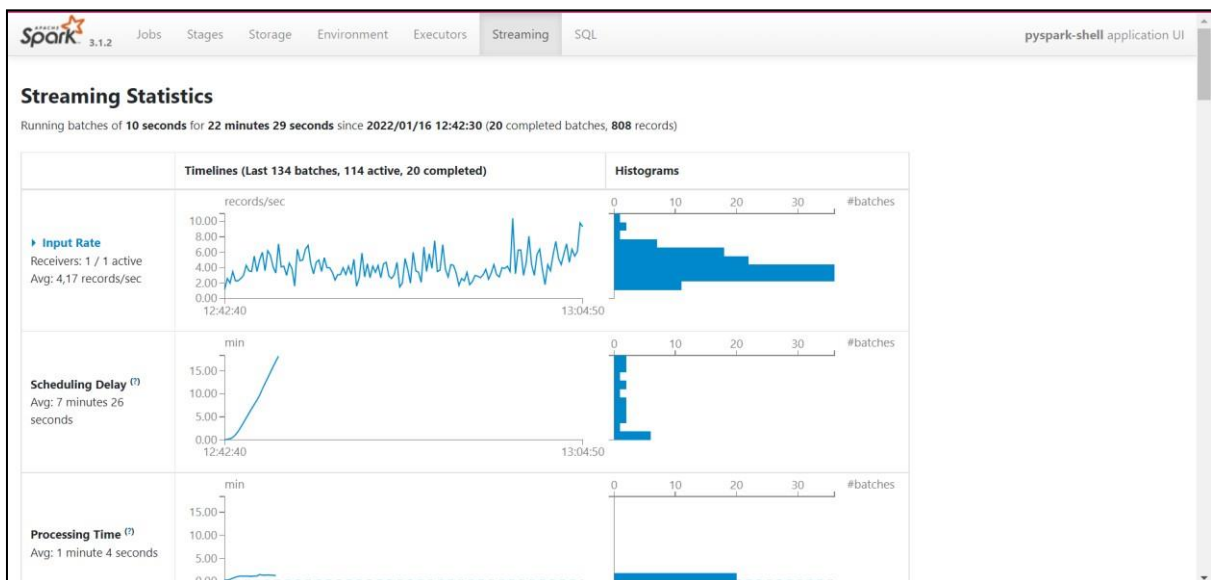
Spark possède aussi une interface graphique pour l'administration du cluster. Les captures d'écrans suivantes montrent la scalabilité dans Spark.



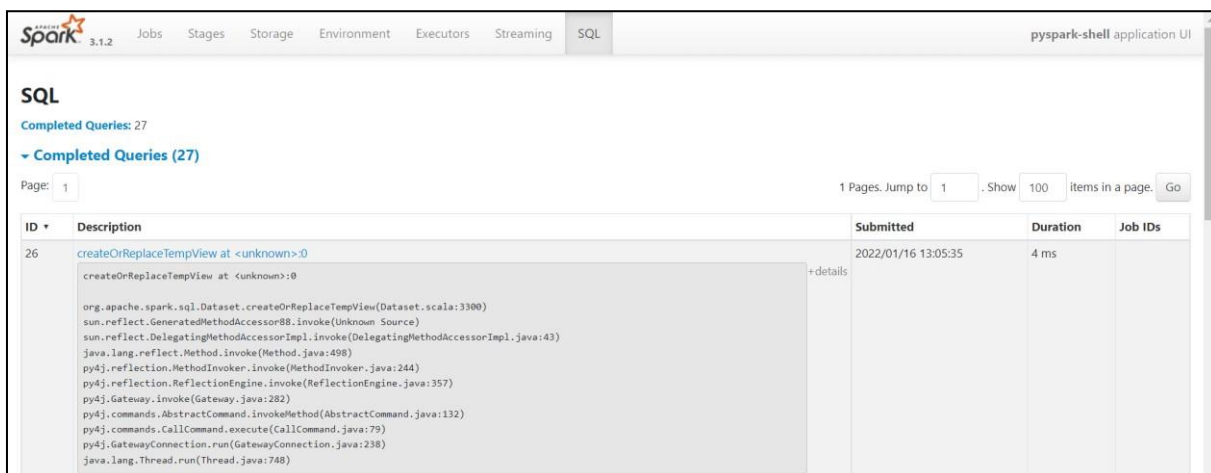
Spark jobs : active jobs



Executors



Streaming statistics



SQL

Partie 2 : Analyse de sentiment

Dans la littérature, l'analyse des sentiments (*sentiment analysis*) est également appelée *opinion Mining*, *opinion extraction*, *sentiment mining*, *subjectivity analysis*, *affect analysis*, *emotion analysis*, *review mining*, *appraisal extraction*, est un domaine de recherche qui consiste à analyser les sensations, les attitudes et les émotions des individus vis-à-vis des entités telles que les produits, les services et les organisations économique. L'analyse des sentiments est l'un des domaines de recherche les plus actifs en traitement automatique de langage naturel, Machine Learning, statistiques et linguistique depuis le début de l'année 2000. Les origines de l'analyse des sentiments se réfère aux des sciences de la psychologie, la sociologie et de l'anthropologie, qui se concentrent sur les émotions humaines.

L'analyse des sentiments consiste à construire des outils automatiques capables d'extraire des informations subjectives de textes en langage naturel, de manière à créer des connaissances structurées et exploitables pouvant être utilisées par un système d'aide à la décision ou un décideur.

Catégorisation des sentiments

Les phrases sont objectives ou subjectives. Lorsqu'une phrase est objective, aucune autre tâche fondamentale n'est requise. Lorsqu'une phrase est subjective, ses polarités (positive, négative ou neutre) doivent être estimées.

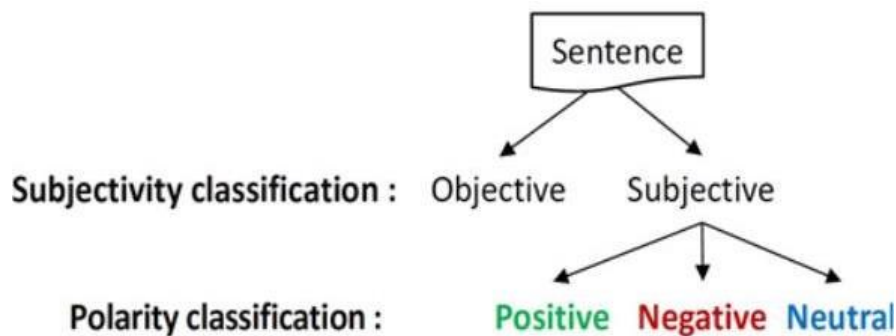


Figure 1 : Workflow de l'analyse de sentiments

La classification de subjectivité (Subjectivity classification) est la tâche qui distingue les phrases exprimant des informations objectives des phrases exprimant des vues et opinions subjectives.

L'iPhone est un smartphone – phrase objective

L'iPhone est génial – phrase subjective

La classification de polarité (Polarity classification) est la tâche qui distingue les phrases qui expriment des polarités positives, négatives ou neutres.

Niveaux d'analyse

Le premier choix lorsque l'on applique l'analyse des sentiments est de définir le texte qui va être analysé dans le cas d'une étude considérée. En général, il existe trois niveaux d'analyse : le niveau du document (Message level ou Document level), le niveau de la phrase (Sentence level) et le niveau des aspects (Entity and aspect level).

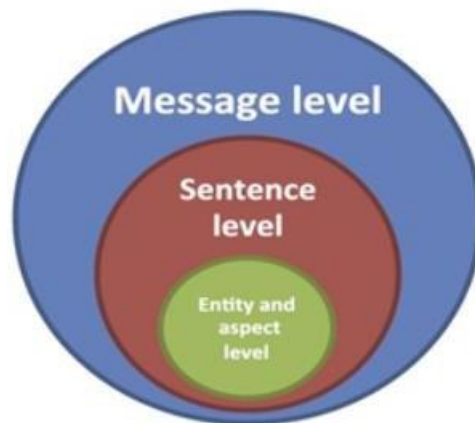


Figure 2 : Niveaux d'analyse

- **Niveau du document** : détermine la polarité d'un texte entier. L'hypothèse est que le texte n'exprime qu'une seule opinion sur une seule entité (par exemple, un seul produit).
- **Niveau de la phrase** : détermine la polarité de chaque phrase contenue dans un texte. L'hypothèse est que chaque phrase dans le texte exprime une opinion unique sur une entité unique.
- **Niveau des aspects** : Effectue une analyse plus fine que les autres niveaux. Il est basé sur l'idée qu'une opinion consiste d'un sentiment et une cible (d'opinion). Par exemple, la phrase « L'iPhone est très bon, mais il faut encore travailler sur la durée de vie de la batterie et les problèmes de sécurité » évalue trois aspects : iPhone (positif), la durée de vie de la batterie (négatif) et la sécurité (négative).

Types d'analyse de sentiments

Il existe de nombreux types d'analyses de sentiments allant des systèmes qui se concentrent sur la classification de la polarité (positif, négatif, neutre) aux systèmes qui détectent des émotions (en colère, heureux, triste, etc.) ou identifient des intentions (par exemple, intéressé, pas intéressé). Dans la section suivante, nous aborderons les types les plus importants.

- **Analyse fine des sentiments (fine-grained sentiment analysis)**

Au lieu de parler de phrases positives, négatives ou neutres, nous considérons les catégories suivantes :

- Très positive
- Positive
- Neutre
- Négative
- Très négative

Certains systèmes offrent également différentes classifications de polarité en identifiant si le sentiment positif ou négatif est associé à un sentiment particulier, tel que la colère, la tristesse ou des inquiétudes (sentiments négatifs) ou du bonheur, de l'amour ou de l'enthousiasme (sentiments positifs).

- **Détection d'émotion (Emotion detection)**

La détection des émotions vise à détecter des émotions telles que le bonheur, la frustration, la colère, la tristesse, etc. De nombreux systèmes de détection d'émotions sont basés sur l'utilisation de lexiques de sentiments (c'est-à-dire des listes des émotions) ou sur des algorithmes d'apprentissage automatique complexes.

- **Analyse de sentiments à base d'aspects (Aspect-Based Sentiment Analysis ABSA)**

Au lieu de classer le sentiment général d'un texte en positif ou en négatif, l'analyse de sentiments à base d'aspects permet d'analyser le texte afin d'identifier différents aspects et de déterminer le sentiment correspondant pour chacun. Les résultats sont plus détaillés, intéressants et précis car l'analyse à base d'aspects examine de manière précise les informations contenues dans un texte.

Algorithmes d'analyse des sentiments

Dans la littérature, il existe de nombreuses méthodes et algorithmes pour mettre en œuvre des systèmes d'analyse des sentiments, que l'on peut classer comme suit :

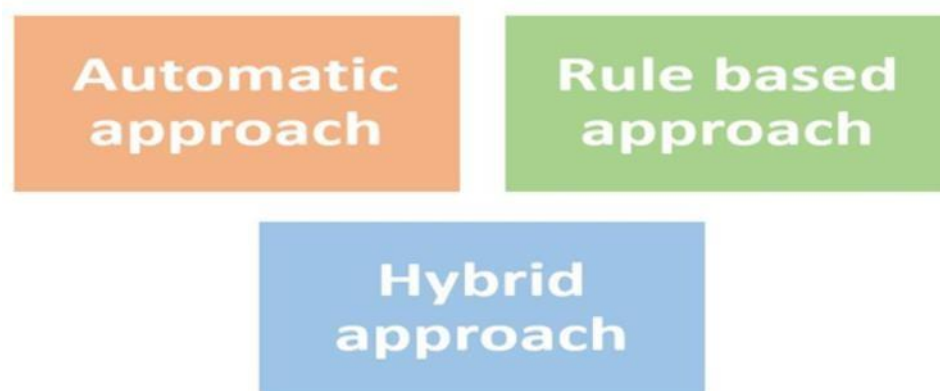
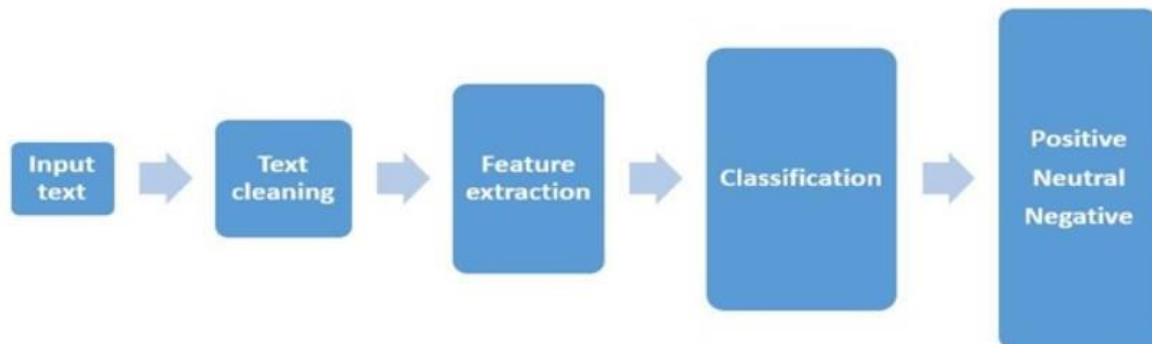


Figure 3 : Algorithmes d'analyse des sentiments

- ✓ **Approche automatique** : systèmes qui s'appuient sur des techniques d'apprentissage automatique à partir de données.

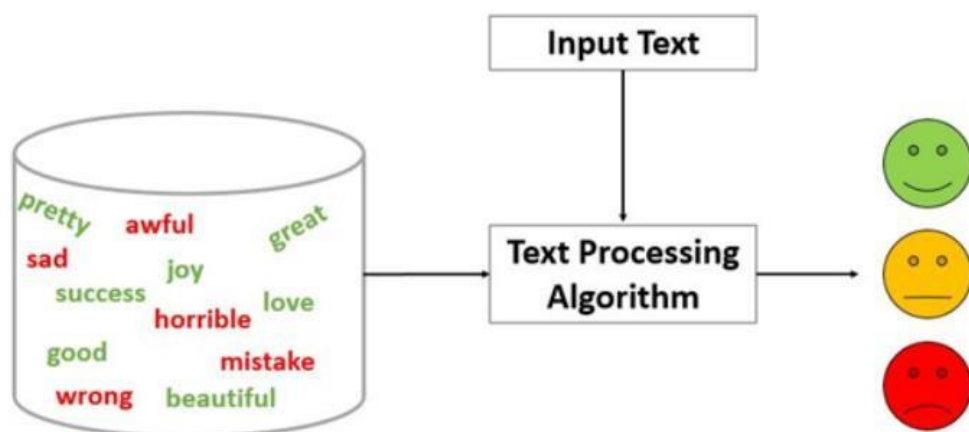
Les approches automatiques reposent sur des techniques d'apprentissage automatique (Machine learning). La tâche d'analyse des sentiments est généralement modélisée comme un problème de classification dans lequel un classificateur est alimenté avec un texte et renvoie la catégorie correspondante, par ex. positif, négatif ou neutre (en cas d'analyse de polarité).



- ✓ **Approche à base de règles (Rule-based) :** systèmes qui effectuent une analyse des sentiments basée sur un ensemble de règles.

L'approche à base de règles (ou l'approche lexicale) définit un ensemble de règles dans un type de langage de programmation (script) qui identifie la subjectivité, la polarité ou le sujet d'une opinion. Cette approche peut utiliser diverses entrées, telles que :

- Techniques classiques de NLP, telles que la racinisation, tokenisation, POS – tagging et Chunking.
- Autres opérations basées sur le lexique, ils utilisent le dictionnaire des sentiments avec des mots d'opinion et les faire correspondre avec les données pour déterminer la polarité.



- ✓ **Approche hybride** : systèmes combinant à la fois des approches basées sur des règles et des approches automatiques.

Le concept de méthodes hybrides est très intuitif : combinez simplement le meilleur des deux approches, celui basé sur des règles et celui automatique. Généralement, en combinant les deux approches, les méthodes peuvent améliorer la précision.

Extraction des tweets « Covid-19 vaccine »

Nous avons commencé par importer les packages nécessaires :

- **Tweepy** : Le module Tweepy nous permet d'interroger de façon très simple avec l'API Twitter afin de récupérer les Tweets.
- **OAuth** (Open Authentication) : est un protocole libre de gestion des autorisations sur une API. Elle permet d'utiliser de façon sécurisée une API. **OAuthHandler** pour s'authentifier sur **Twitter**.
- **NumPy** : est une bibliothèque pour le langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- **Pandas** : est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données.

```
import tweepy
from tweepy import API
from tweepy import Cursor
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
import numpy as np
import pandas as pd
```

L'API Twitter exige que toutes les requêtes utilisent **OAuth** pour s'authentifier. Nous devons donc créer les identifiants d'authentification requis pour pouvoir utiliser l'API. Ces informations d'identification sont les quatre chaînes de texte : **consumer_key**, **consumer_secret**, **access_token** et **access_secret_token**.

```
consumer_key="lCmUK0wkuQaPAsATiWaeFyx9I"
consumer_secret="RH9qCxUD0yzGaP9Sak0lX02t2vHgpszovslgwonNEuxr2C2CLaB"
access_token="1477255548294897665-tLSZnzoD5oRCSFxl2xHGbGQ7YNloF"
access_token_secret="z5mghK3BXUCY7gij39eeNaobD5tZ0j5w9AMuQrrtZRALA"
```

Les informations d'identification sont testées à l'aide de **verify_credentials()**. Si tout se passe bien, nous devrions voir un message indiquant **Authentication OK**.

Nous pouvons tester les identifiants à l'aide de l'extrait de code suivant :

```
# Authenticate to Twitter
auth = tweepy.OAuthHandler("lCmUK0wkuQaPAsATiWaeFyx9I", "RH9qCxUD0yzGaP9Sak0lX02t2vHgpszovslgwonNEuxr2C2CLaB")
auth.set_access_token("1477255548294897665-tLSZnzoD5oRCSFxl2xHGbGQ7YNloF", "z5mghK3BXUCY7gij39eeNaobD5tZ0j5w9AMuQrrtZRALA")

api = tweepy.API(auth)

try:
    api.verify_credentials()
    print("Authentication OK")
except:
    print("Error during authentication")

Authentication OK
```

Passons maintenant à lier notre compte à une application Twitter. C'est le cœur du process OAuth. Le concept de OAuth est de : **fournir un moyen de connexion sécurisé à la fois pour l'application et pour l'utilisateur de l'application.**

Tweepy s'occupe de tous les détails d'utilisation d'**OAuth** requis par l'API Twitter pour authentifier chaque demande. Il fournit une classe **OAuthHandler** que nous pouvons utiliser pour définir les informations d'identification à utiliser dans tous les appels d'API.

Cet extrait de code montre comment nous pouvons créer un objet **OAuthHandler** qui pourra ensuite être utilisé pour les appels d'API :

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

La classe API comporte de nombreuses méthodes qui permettent d'accéder aux points de terminaison de l'API Twitter. En utilisant ces méthodes, nous pouvons accéder aux fonctionnalités de l'API Twitter.

L'extrait de code ci-dessus, a créé un objet API que nous pouvons utiliser pour appeler les méthodes de l'API Twitter.

La sélection des tweets à analyser est réalisée au moyen de filtres basés sur le mot (**covid19 vaccine**). Ce mot permet d'identifier précisément la thématique supposée du tweet.

```
for tweet in api.search('covid19 vaccine'):
    print(tweet)
```

```
Status(_api=<tweepy.api.API object at 0x000001AC464E31F0>, _json={'created_at': 'Tue Jan 11 20:10:17 +0000 2022', 'id': 1480995500056846337, 'id_str': '1480995500056846337', 'text': 'RT @redstarneil: Update on pop up #COVID19 vaccination this week\n #VaccinesWork\nLocations include \nTue 11th @KensalRLibrary\nWed 12th #Kilbu...', 'truncated': False, 'entities': {'hashtags': [{'text': 'COVID19', 'indices': [34, 42]}, {'text': 'VaccinesWork', 'indices': [66, 79]}], 'symbols': [], 'user_mentions': [{'screen_name': 'redstarneil', 'name': 'Cllr Neil Nerva EUGB', 'id': 65481914, 'id_str': '65481914', 'indices': [3, 15]}, {'screen_name': 'KensalRLibrary', 'name': 'Friends of KRLibrary', 'id': 225438277, 'id_str': '225438277', 'indices': [108, 123]}], 'urls': []}, 'metadata': {'iso_language_code': 'en', 'result_type': 'recent'}, 'source': '<a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a>', 'in_reply_to_status_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': None, 'in_reply_to_user_id_str': None, 'in_reply_to_screen_name': None, 'user': {'id': 460962995, 'id_str': '460962995', 'name': 'QPARA', 'screen_name': 'Qparkres', 'location': 'Queen's Park, London NW6', 'description': 'We're your local residents' association working voluntarily to make the area an even better place to live and running Queen's Park Day & Open Gardens Studios', 'url': 'https://t.co/1sxKZh8Jy8', 'entities': {'url': {'urls': [{'url': 'https://t.co/1sxKZh8Jy8', 'expanded_url': 'http://www.qpark.org.uk/', 'display_url': 'qpark.org.uk', 'indices': [0, 23]}]}}, 'description': {'urls': []}}, 'protected': False, 'followers_count': 1474, 'friends_count': 201, 'listed_count': 23, 'created_at': 'Wed Jan 11 09:23:49 +0000 2012', 'favourites_count': 958, 'utc_offset': None, 'time_zone': None, 'geo_enabled': True, 'verified': False, 'statuses_count': 1576, 'lang': None, 'contributors_enabled': False, 'is_translator': False, 'is_translation_enabled': False, 'profile_background_color': 'C0DEED', 'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.png', 'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme1/bg.png', 'profile_background_tile': False, 'profile_image_url': 'http://pbs.twimg.com/profile_images/1
```

Chaque Tweet comporte de nombreux attributs qui décrivent le contexte dans lequel le tweet a été publié (ex : géolocalisation du tweet, profil de l'auteur du tweet, nombre de followers, etc..).

Pour ce projet, nous avons sélectionné les attributs suivants :

- **Tweet_id** : Identifiant du tweet
- **User_name** : le nom de l'utilisateur : Auteur du tweet
- **Date** : Date de création et l'heure.
- **Text** : Message du tweet (contenu textuel)
- **User_id** : Identifiant de l'utilisateur
- **User_statuses_count** : Le nombre de Tweets (y compris les retweets) émis par l'utilisateur.
- **User_followers** : Le nombre d'abonnés que le compte a actuellement.
- **User_location** : L'emplacement défini par l'utilisateur pour le profil de ce compte.
- **Like_count** : Le nombre de Tweets que cet utilisateur a émis pendant la durée de vie du compte.
- **Retweet_count** : Indicateur que ce tweet est un tweet Retweeté, et le nombre de retweet.
- **User_verified** : Si True, indique que l'utilisateur a un compte vérifié.

La commande **DataFrame** permet d'organiser toutes nos données sur les tweets dans un tableau pandas.

```
df = pd.DataFrame(columns = ['tweet_id', 'user_name', 'date', 'text', 'user_id', 'user_statuses_count',  
                             'user_followers', 'user_location', 'like_count', 'retweet_count', 'user_verified'])
```

Le **streaming** nous permet de surveiller activement les tweets qui correspondent à certains critères en **temps réel**.

Les curseurs sont implémentés en tant que classe Tweepy nommée **Cursor**. Pour utiliser un curseur, nous sélectionnons la méthode API à utiliser pour récupérer les éléments et le nombre d'éléments souhaités. L'objet **Cursor** se charge de récupérer les différentes pages de résultats de manière transparente.

L'objet **Cursor** a une méthode **items()** qui renvoie un itérable que nous pouvons utiliser pour parcourir les résultats. Nous pouvons transmettre à **items()** le nombre d'éléments de résultat que nous souhaitons obtenir.


```
def stream(data, file_name):
    i = 0
    for tweet in tweepy.Cursor(api.search, q=data, count=100, lang='en').items():
        print(i,end='\r')
        df.loc[i, 'tweet_id'] = tweet.id
        df.loc[i, 'user_name'] = tweet.user.name
        df.loc[i, 'date'] = tweet.created_at
        df.loc[i, 'text'] = tweet.text
        df.loc[i, 'user_id'] = tweet.user.id
        df.loc[i, 'user_statuses_count'] = tweet.user.statuses_count
        df.loc[i, 'user_followers'] = tweet.user.followers_count
        df.loc[i, 'user_location'] = tweet.user.location
        df.loc[i, 'like_count'] = tweet.favorite_count
        df.loc[i, 'retweet_count'] = tweet.retweet_count
        df.loc[i, 'user_verified'] = tweet.user.verified
        df.to_csv('{}\{}.csv'.format(file_name))
        i+=1
    if i == 10000:
        break
    else:
        pass
```

On exporte ensuite le dataframe au format **csv** sous le nom :

« **tweet_extraction_covid19_vaccine** ».

```
stream(data = ['covid19 vaccine'], file_name = 'tweet_extraction_covid19_vaccine')
9999
```

Tous les tweets ont été créés le **11/01/2022**, ce qui représente un total de **9999** tweets à analyser.

df.head()									
	tweet_id	user_name	date	text	user_id	user_statuses_count	user_followers	user_location	like_count
0	1480995707943211008	Colorado Department of Public Health & Environ...	2022-01-11 20:11:07	📍Chapel Hills Mall\n1710 Briargate Blvd, Color...	331244103	9176	50810	Colorado	0
1	1480995703216164864	Colorado Department of Public Health & Environ...	2022-01-11 20:11:05	📍Ball Arena\n1000 Chopper Cir, Denver, CO 8020...	331244103	9176	50810	Colorado	0
2	1480995693279862784	Colorado Department of Public Health & Environ...	2022-01-11 20:11:03	📍Arapahoe Community College (Lot B)\n5900 S Sa...	331244103	9176	50810	Colorado	0
3	1480995600946458624	Savannah Deschner	2022-01-11 20:10:41	RT @SKGov: #COVID19SK Summary for January 11, ...	886614449423036417	15562	247	Regina SK Canada	0
4	1480995572937015299	David Mossey	2022-01-11 20:10:34	RT @CityNewsTO: A new policy mandating #COVID1...	2392851672	30042	439		0

Analyse de sentiments

Nous effectuerons une analyse des sentiments pour le vaccin du COVID-19 en utilisant les données des Tweets **covid19_vaccine**, collectées à l'aide du package Python **tweepy** pour accéder à l'API Twitter.

Commençons par importer les packages :

- **NLTK** : est une bibliothèque Python dédiée au traitement naturel du langage ou Natural Language Processing.

Le **NLTK**, ou **Natural Language Toolkit**, est une suite de bibliothèques logicielles et de programmes. Elle est conçue pour le traitement naturel symbolique et statistique du langage anglais en langage Python. C'est l'une des bibliothèques de traitement naturel du langage les plus puissantes.

- **Re** : Les expressions régulières ou expressions rationnelles ne font pas partie du langage Python en soi mais constituent un langage à part. Python nous permet d'exploiter leur puissance et fournit un support pour les expressions régulières via son module standard **re**.

Les expressions régulières sont des schémas ou des motifs utilisés pour effectuer des recherches et des remplacements dans des chaînes de caractères.

- **Seaborn** : est une bibliothèque permettant de créer des graphiques statistiques en Python. Elle est basée sur Matplotlib, et s'intègre avec les structures Pandas.

Nous allons maintenant importer l'ensemble de données à partir des Tweets, que nous avons mis dans le fichier CSV : **tweet_extraction_covid19_vaccine**.

```
df=pd.read_csv('tweet_extraction_covid19_vaccine.csv')
```

```
df.head()
```

	Unnamed: 0	tweet_id	user_name	date	text	user_id	user_statuses_count	user_followers	user_location
0	0	1480995707943211008	Colorado Department of Public Health & Environ...	2022-01-11 20:11:07	📍Chapel Hills Mall\n1710 Briargate Blvd, Color...	331244103	9176	50810	Colorado
1	1	1480995703216164864	Colorado Department of Public Health & Environ...	2022-01-11 20:11:05	📍Ball Arena\n1000 Chopper Cir, Denver, CO 8020...	331244103	9176	50810	Colorado
2	2	1480995693279862784	Colorado Department of Public Health & Environ...	2022-01-11 20:11:03	📍Arapahoe Community College (Lot B)\n5900 S Sa...	331244103	9176	50810	Colorado
3	3	1480995600946458624	Savannah Deschner	2022-01-11 20:10:41	RT @SKGov: #COVID19SK Summary for January 11, ...	886614449423036417	15562	247	Regina SK Canada
4	4	1480995572937015299	David Mossey	2022-01-11 20:10:34	RT @CityNewsTO: A new policy mandating #COVID1...	2392851672	30042	439	NaN

Une rapide inspection de la base de données, nous permet de voir que la compréhension de certains tweets est difficile. Le nettoyage sera important.

Le nettoyage des tweets comprendra plusieurs choses :

- Enlever les emojis
- Enlever les website URL
- Changer les abréviations
- Retirer la ponctuation
- Enlever les stopwords

Nous pouvons voir que notre ensemble de données a 10 000 lignes et 12 colonnes. Mais comme nous n'aurions pas besoin de toutes les colonnes, nous allons maintenant sélectionner les plus importantes pour notre analyse et créer une nouvelle base de données.

```
needed_columns=['user_name', 'date', 'text']
df=df[needed_columns]
df.head()
```

	user_name	date	text
0	Colorado Department of Public Health & Environ...	2022-01-11 20:11:07	📍Chapel Hills Mall\n1710 Briargate Blvd, Color...
1	Colorado Department of Public Health & Environ...	2022-01-11 20:11:05	📍Ball Arena\n1000 Chopper Cir, Denver, CO 8020...
2	Colorado Department of Public Health & Environ...	2022-01-11 20:11:03	📍Arapahoe Community College (Lot B)\n5900 S Sa...
3	Savannah Deschner	2022-01-11 20:10:41	RT @SKGov: #COVID19SK Summary for January 11, ...
4	David Mossey	2022-01-11 20:10:34	RT @CityNewsTO: A new policy mandating #COVID1...

Maintenant que nous avons un nouvel ensemble de données avec les données importantes pour notre analyse, nous devons vérifier les types de données de la base de données.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   user_name   10000 non-null  object 
 1   date        10000 non-null  object 
 2   text        10000 non-null  object 
dtypes: object(3)
memory usage: 234.5+ KB
```

Nous pouvons voir que nos trois colonnes ont le même type de données. Mais pour la colonne date, on peut voir qu'elle est spécifique à la seconde du tweet. Comme nous n'aurions pas besoin de données aussi précises pour notre analyse, nous ne prendrons que le jour, le mois et l'année du tweet.

```
df.user_name=df.user_name.astype('category')
df.user_name=df.user_name.cat.codes # assign a unique numerical code to each category
df.date=pd.to_datetime(df.date).dt.date
```

```
df.head(5)
```

	user_name	date	text
0	1456	2022-01-11	📍 Chapel Hills Mall\n1710 Briargate Blvd, Color...
1	1456	2022-01-11	📍 Ball Arena\n1000 Chopper Cir, Denver, CO 8020...
2	1456	2022-01-11	📍 Arapahoe Community College (Lot B)\n5900 S Sa...
3	6132	2022-01-11	RT @SKGov: #COVID19SK Summary for January 11, ...
4	1725	2022-01-11	RT @CityNewsTO: A new policy mandating #COVID1...

Maintenant que nous avons fini d'importer notre ensemble de données, nous pouvons continuer à traiter nos données pour l'analyse.

Pour traiter nos données, nous devons sélectionner la colonne de texte de notre ensemble de données :

```
texts=df.text
texts
```

```
0      📍 Chapel Hills Mall\n1710 Briargate Blvd, Color...
1      📍 Ball Arena\n1000 Chopper Cir, Denver, CO 8020...
2      📍 Arapahoe Community College (Lot B)\n5900 S Sa...
3      RT @SKGov: #COVID19SK Summary for January 11, ...
4      RT @CityNewsTO: A new policy mandating #COVID1...
...
9995   The court hearing over Trump's liability for #...
9996   RT @CDCgov: New @CDCMMWR shows 2 doses of Pfiz...
9997   RT @KahlonRav: Check out these links for up to...
9998   RT @MollyThomasTV: To think, last Jan, we were...
9999   RT @501Awani: The Nikkei COVID-19 Recovery Ind...
Name: text, Length: 10000, dtype: object
```

La première étape de notre traitement consisterait à supprimer l'URL de tous les tweets, car nous n'en avons pas besoin. Après cela, nous convertirons tout le texte en minuscules pour une analyse plus facile. Enfin, nous supprimerons également toutes les ponctuations des textes.

Supprimer les URLs des tweets :

```
remove_url=lambda x:re.sub(r'http\S+', '',str(x))
texts_lr=texts.apply(remove_url)
texts_lr
```

0	📍Chapel Hills Mall\n1710 Briargate Blvd, Color...
1	📍Ball Arena\n1000 Chopper Cir, Denver, CO 8020...
2	📍Arapahoe Community College (Lot B)\n5900 S Sa...
3	RT @SKGov: #COVID19SK Summary for January 11, ...
4	RT @CityNewsTO: A new policy mandating #COVID1...
	...
9995	The court hearing over Trump's liability for #...
9996	RT @CDCgov: New @CDCMMWR shows 2 doses of Pfiz...
9997	RT @KahlonRav: Check out these links for up to...
9998	RT @MollyThomasTV: To think, last Jan, we were...
9999	RT @501Awani: The Nikkei COVID-19 Recovery Ind...

Name: text, Length: 10000, dtype: object

Convertir les tweets en minuscules :

```
to_lower=lambda x: x.lower()
texts_lr_lc=texts_lr.apply(to_lower)
texts_lr_lc
```

0	📍chapel hills mall\n1710 briargate blvd, color...
1	📍ball arena\n1000 chopper cir, denver, co 8020...
2	📍arapahoe community college (lot b)\n5900 s sa...
3	rt @skgov: #covid19sk summary for january 11, ...
4	rt @citynewsto: a new policy mandating #covid1...
	...
9995	the court hearing over trump's liability for #...
9996	rt @cdcgov: new @cdcmmwr shows 2 doses of pfiz...
9997	rt @kahlonrav: check out these links for up to...
9998	rt @mollythomastv: to think, last jan, we were...
9999	rt @501awani: the nikkei covid-19 recovery ind...

Name: text, Length: 10000, dtype: object

Supprimer les ponctuations :

```
remove_puncs= lambda x:x.translate(str.maketrans('', '',string.punctuation))
texts_lr_lc_np=texts_lr_lc.apply(remove_puncs)
texts_lr_lc_np
```

0	📍chapel hills mall\n1710 briargate blvd colora...
1	📍ball arena\n1000 chopper cir denver co 80204\...
2	📍arapahoe community college lot b\n5900 s sant...
3	rt skgov covid19sk summary for january 11 2022...
4	rt citynewsto a new policy mandating covid19 v...
	...
9995	the court hearing over trump's liability for j...
9996	rt cdcgov new cdcmmwr shows 2 doses of pfizerb...
9997	rt kahlonrav check out these links for up to d...
9998	rt mollythomastv to think last jan we were all...
9999	rt 501awani the nikkei covid19 recovery index ...

Name: text, Length: 10000, dtype: object

Supprimer les stopwords

Maintenant que nous avons supprimé tous les caractères inutiles de notre texte, nous allons maintenant supprimer les stopwords du texte. Un stopwords est un mot couramment utilisé (comme « le », « un », « une », « en ») 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 en tant que résultat d'une requête de recherche. Cela réduira le bruit dans notre analyse.

```
more_words=[]
stop_words=set(stopwords.words('english')) #nltk package
stop_words.update(more_words)

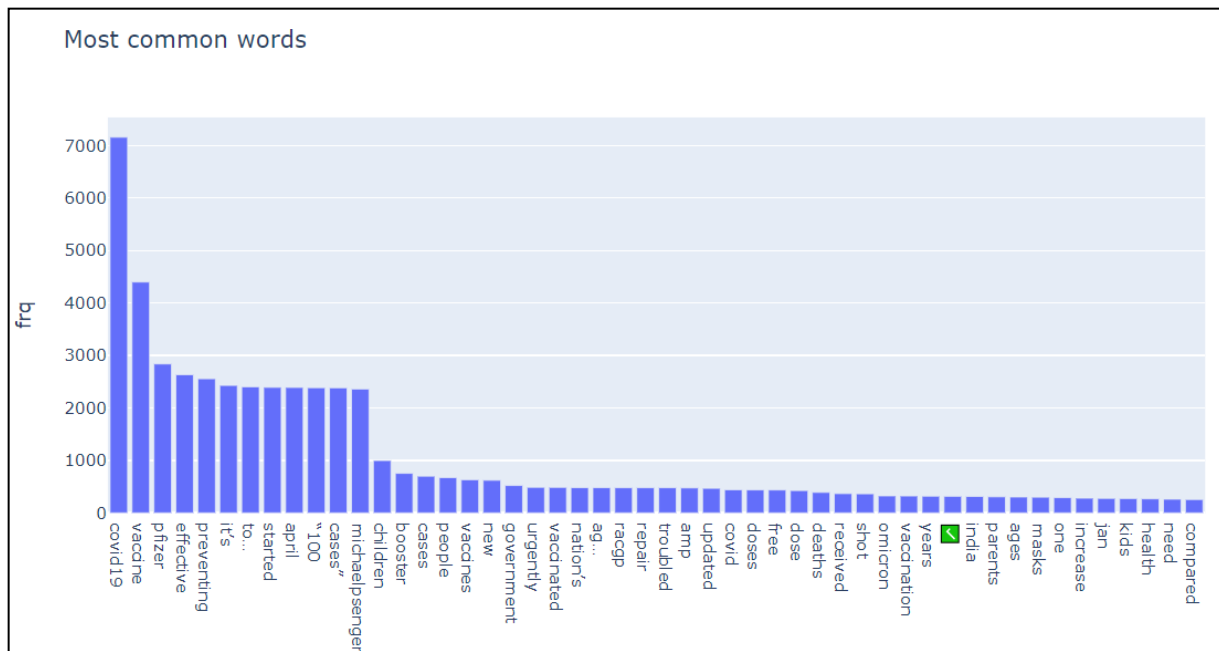
remove_words=lambda x: ' '.join([word for word in x.split() if word not in stop_words])
texts_lr_lc_np_ns=r=texts_lr_lc_np.apply(remove_words)
texts_lr_lc_np_ns

0      📍 chapel hills mall 1710 briargate blvd colorad...
1      📍 ball arena 1000 chopper cir denver co 80204 s...
2      📍 arapahoe community college lot b 5900 santa f...
3      skgov covid19sk summary january 11 2022 daily ...
4      citynewsto new policy mandating covid19 vaccin...
...
9995    court hearing trump's liability jan6 judge ami...
9996    new cdmmwr shows 2 doses pfizerbiontech covid...
9997    kahlonrav check links date covid19 info 📍 • f...
9998    mollythomastv think last jan eagerly awaiting ...
9999    501awani nikkei covid19 recovery index ranks 1...
Name: text, Length: 10000, dtype: object
```

Avant d'analyser les sentiments des tweets, nous allons faire une analyse sur le texte lui-même. Tout d'abord, nous allons lister tous les mots sur chacun des tweets, et également les visualiser. Le but est de voir les mots les plus courants de tous les tweets.

```
word_counts=Counter(words_list).most_common(50)
word_df=pd.DataFrame(word_counts)
word_df.columns=['word','frq']
display(word_df.head(5))
# px=import plotly.express
px.bar(word_df,x='word',y='frq',title='Most common words')
```

	word	frq
0	covid19	7159
1	vaccine	4399
2	pfizer	2838
3	effective	2633
4	preventing	2556



Supprimer les emojis :

```
df['text']=df['text'].apply(lambda x: remove_emoji(x))
display(df)
```

	user_name	date	text
0	1456	2022-01-11	Chapel Hills Briargate Blvd Colorado Springs ...
1	1456	2022-01-11	Ball Chopper Cir Denver CO an appointment
2	1456	2022-01-11	Arapahoe Community College Lot S Santa Fe Dri...
3	6132	2022-01-11	RT SKGov Summary for January Daily statist...
4	1725	2022-01-11	RT CityNewsTO A new policy mandating vaccinat...
...
9995	943	2022-01-11	The court hearing over Trump's liability for ...
9996	6106	2022-01-11	RT CDCgov New CDCMMWR shows doses of PfizerBi...
9997	6551	2022-01-11	RT KahlonRav Check out these links for up to d...
9998	4702	2022-01-11	RT MollyThomasTV To think last Jan we were all...
9999	8178	2022-01-11	RT The Nikkei Recovery Index ranks about co...

10000 rows × 3 columns

Puisque nous avons terminé le traitement des données de texte, nous pouvons maintenant placer le texte nettoyé dans notre Dataframe principal.


```
display(df.head(5))
df.text=texts_lr_lc_np_ns
display(df.head(5))
```

	user_name	date	text
0	1456	2022-01-11	Chapel Hills Briargate Blvd Colorado Springs ...
1	1456	2022-01-11	Ball Chopper Cir Denver CO an appointment
2	1456	2022-01-11	Arapahoe Community College Lot S Santa Fe Dri...
3	6132	2022-01-11	RT SKGov Summary for January Daily statist...
4	1725	2022-01-11	RT CityNewsTO A new policy mandating vaccinat...

Analyse de sentiments

Maintenant que nous avons fini de préparer les données pour notre analyse, nous pouvons continuer notre analyse des sentiments. L'analyse des sentiments peut être organisée en sentiments neutres, positifs ou négatifs. Pour le savoir, nous utiliserons

SentimentIntensityAnalyzer, qui évaluera si le tweet contient un sentiment positif, négatif ou neutre.

```
sid=SentimentIntensityAnalyzer()
ps=lambda x:sid.polarity_scores(x)
sentiment_scores=df.text.apply(ps)
sentiment_scores
```

0	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
1	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
2	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
3	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
4	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
...	
9995	{'neg': 0.141, 'neu': 0.859, 'pos': 0.0, 'comp...
9996	{'neg': 0.084, 'neu': 0.714, 'pos': 0.201, 'co...
9997	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...
9998	{'neg': 0.0, 'neu': 0.822, 'pos': 0.178, 'comp...
9999	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...

Name: text, Length: 10000, dtype: object

Nous pouvons voir qu'il y a 'neg' pour le sentiment négatif, 'neu' pour le sentiment neutre, 'pos' pour le sentiment positif et **compound** comme taux moyen du sentiment. Nous allons nous concentrer sur ce taux.

Pour le sentiment négatif, le taux sera plus proche de -1, et l'inverse vaut pour le sentiment positif, qui sera plus proche de 1. Le sentiment neutre sera un 0.

```
sentiment_df=pd.DataFrame(data=list(sentiment_scores))
display(sentiment_df)
```

	neg	neu	pos	compound
0	0.000	1.000	0.000	0.0000
1	0.000	1.000	0.000	0.0000
2	0.000	1.000	0.000	0.0000
3	0.000	1.000	0.000	0.0000
4	0.000	1.000	0.000	0.0000
...
9995	0.141	0.859	0.000	-0.2023
9996	0.084	0.714	0.201	0.4215
9997	0.000	1.000	0.000	0.0000
9998	0.000	0.822	0.178	0.3818
9999	0.000	1.000	0.000	0.0000

10000 rows × 4 columns

Pour notre analyse, nous allons créer une autre colonne appelée label, où nous étiquèterons les scores en fonction de la valeur de polarité du compound.

Ici, au lieu d'avoir 3 classes, nous avons choisis que 2 classes : **positive**, **negative**

```
labelize=lambda x:('positive' if x>0 else 'negative')
sentiment_df['Label']=sentiment_df.compound.apply(labelize)
display(sentiment_df.head(10))
```

	neg	neu	pos	compound	Label
0	0.000	1.000	0.000	0.0000	negative
1	0.000	1.000	0.000	0.0000	negative
2	0.000	1.000	0.000	0.0000	negative
3	0.000	1.000	0.000	0.0000	negative
4	0.000	1.000	0.000	0.0000	negative
5	0.000	1.000	0.000	0.0000	negative
6	0.000	0.820	0.180	0.5106	positive
7	0.000	0.794	0.206	0.3818	positive
8	0.000	1.000	0.000	0.0000	negative
9	0.182	0.649	0.169	-0.0516	negative

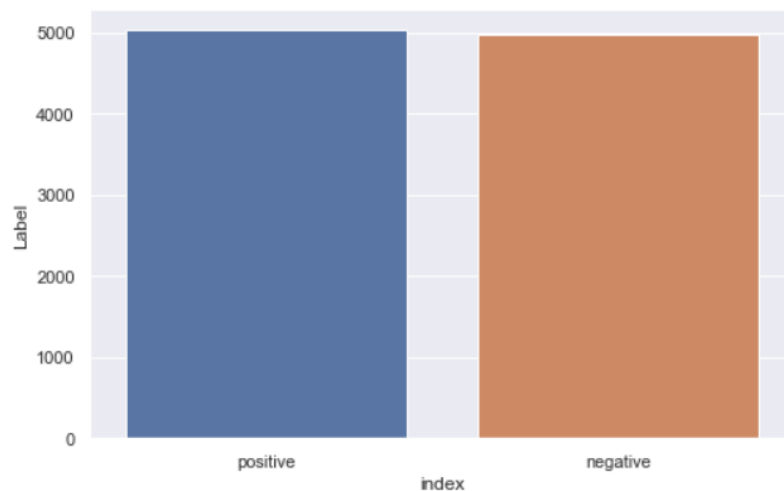
Maintenant que nous avons l'étiquette pour chaque tweet, nous allons joindre la colonne Label dans notre base de données principale, ensuite nous compterons le nombre de tweets positives et négatives de notre dataframe et le visualiserons :

```
counts_df=data.Label.value_counts().reset_index()
display(counts_df)
```

	index	Label
0	positive	5022
1	negative	4978

```
plt.figure(figsize=(8,5))
sns.barplot(x='index',y='Label',data=counts_df)
```

<AxesSubplot:xlabel='index', ylabel='Label'>



Word Cloud est une technique de visualisation de données utilisée pour représenter des données textuelles dans lesquelles la taille de chaque mot indique sa fréquence ou son importance.

```
from wordcloud import WordCloud
from wordcloud import STOPWORDS
```

Voici un nuage de mots dans lequel on peut apercevoir quels sont les mots les plus utilisés dans les tweets de ses utilisateurs (plus la taille de la police d'un mot est grande, plus le nombre d'occurrences de celui-ci est important. Ici nous avons fait le nuage de mots pour la classe positive et la classe négative.

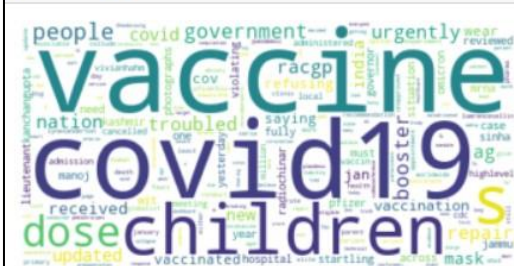
- **Classe positive :**

```
pos_tweets=data[data["Label"]=="positive"]
txt=" ".join(tweet for tweet in pos_tweets["text"])
stop_words = ["amp"] + list(STOPWORDS)
wordcloud = WordCloud(collocations = False,background_color = 'white', stopwords = stop_words).generate(txt)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



- **Classe négative :**

```
neg_tweets=data[data["Label"]=="negative"]
txt=" ".join(tweet.lower() for tweet in neg_tweets["text"])
wordcloud = WordCloud(collocations = False,background_color = 'white',stopwords=stop_words).generate(txt)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



Ensuite, nous allons commencer notre travail de classification de sentiments des tweets à l'aide d'une régression logistique, mais cette fois en utilisant **PySpark**.

```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover
```

```
appName = "Sentiment Analysis in Spark"
spark = SparkSession \
    .builder \
    .appName(appName) \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

```
spark_tweet = spark.createDataFrame(data)
```

```
spark_tweet = spark.createDataFrame(data)
```

```
spark_tweet.show(n=5, truncate=True)
```

```
+-----+-----+-----+-----+
|user_name|      date|      text|  Label|
+-----+-----+-----+-----+
|    1456|2022-01-11|chapel hills mall...|negative|
|    1456|2022-01-11|ball arena chopp...|negative|
|    1456|2022-01-11|arapahoe communit...|negative|
|    6132|2022-01-11|skgov summary ja...|negative|
|    1725|2022-01-11|citynewsto new po...|negative|
+-----+-----+-----+-----+
only showing top 5 rows
```

Nous allons diviser notre jeu de données en un échantillon d'entraînement (80%) dans lequel nous allons apprendre les paramètres du modèle et un échantillon test (10%) dans lequel nous allons les tester, et (10%) pour la validation.

```
(train_set, val_set, test_set) = spark_tweet.randomSplit([0.80, 0.10, 0.10], seed = 2000)
```

Passons à tourner notre modèle de régression logistique afin de classifier les sentiments positifs ou négatifs des tweets.

```

from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline

tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashtf = HashingTF(numFeatures=2**16, inputCol="words", outputCol='tf')
idf = IDF(inputCol='tf', outputCol="features", minDocFreq=5) #minDocFreq: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "Label", outputCol = "label")
pipeline = Pipeline(stages=[tokenizer, hashtf, idf, label_stringIdx])

pipelineFit = pipeline.fit(train_set)
train_df = pipelineFit.transform(train_set)
val_df = pipelineFit.transform(val_set)
train_df.show(5)

```

	user_name	date	text	label	words	tf	features
1	2022-01-11	sick bullshit app...	0.0	[sick, bullshit, ...]	(65536,[2731,5548...]	(65536,[2731,5548...]	
1	2022-01-11	time covaxin cova...	0.0	[time, covaxin, c...	(65536,[26460,398...]	(65536,[26460,398...]	
1	2022-01-11	usfda drwoodcockf...	1.0	[usfda, drwoodcoc...	(65536,[150,5548,...]	(65536,[150,5548,...]	
1	2022-01-11	usfda drwoodcockf...	0.0	[usfda, drwoodcoc...	(65536,[150,5548,...]	(65536,[150,5548,...]	
3	2022-01-11	warns repeated b...	0.0	[warns, repeated,...]	(65536,[11233,118...]	(65536,[11233,118...]	

only showing top 5 rows

- **Tokenizer** : La **tokenisation** consiste essentiellement à diviser une phrase, un paragraphe ou un document texte entier en unités plus petites, telles que des mots ou des termes individuels. Chacune de ces petites unités s'appelle des **tokens (words)**. Dans notre cas, tokenizer divise notre tweet (contenu textuel du tweet) en mots afin que le sens du texte pourrait facilement être interprété en analysant les mots présents dans le texte.
- **Hashing** : Avec le hachage, le contexte du texte sera transformé en valeur de hachage comme une valeur de taille de bit où chaque contexte sera calculé et attribué une valeur unique.
- **TF-IDF** : Cette technique calcule l'importance du mot dans le document. TF, Term Frequency, mesure la fréquence d'un terme dans chaque document. IDF, Inverse Document Frequency, mesure l'importance du terme dans tous les documents. Ensuite, deux valeurs sont multipliées, $TF * IDF$, et nous découvrons l'importance de chaque mot dans le document.

Nous avons obtenu comme performance du modèle : 97,23%

```

from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=100)
lrModel = lr.fit(train_df)
predictions = lrModel.transform(val_df)
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")
evaluator.evaluate(predictions)

0.9722866533187262

```

Conclusion

A travers ce projet, nous avons pu voir la puissance du NLP qui a permis de classifier de manière fiable le sentiment des tweets. Cette analyse de sentiments Twitter pourrait avoir plusieurs domaines d'application dont voici quelques exemples spécifiques :

- **La surveillance des médias sociaux :** Détecter les tweets à sentiments négatifs pourrait permettre de réduire le harcèlement et le déferlement de violence se produisant sur Twitter.
- **Campagnes politiques :** L'analyse des sentiments sur Twitter pourrait permettre d'analyser la popularité d'un candidat politique et de prédire ainsi le vainqueur d'une élection présidentielle par exemple.
- **Fidéliser la clientèle d'une entreprise :** L'analyse des sentiments sur Twitter permet de suivre ce qui se dit à propos d'un produit ou service vendu par une entreprise et peut ainsi aider à détecter les clients en colère.

Malgré tous ces outils dont on dispose et malgré tous les efforts de milliers de chercheurs, les défis liés au NLP sont encore difficile à résoudre. La construction de sémantiques fiables et flexibles n'est pas encore parfaitement maîtrisée. Et nous sommes encore très loin d'une vraie compréhension du langage par les IA.