République Tunisienne

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Sfax

Ecole Nationale d'Electronique et des
Télécommunications de Sfax

Génie
Informatique Industrielle

PFA-GII-22- 30

PROJET DE FIN D'ANNEE

Présenté à

L'Ecole Nationale d'Electronique et des Télécommunications de Sfax

Génie Informatique Industrielle

Par

Iheb Akermi

Analyse en temps réel et sentiment des tweets

Soutenu le 29 mai 2023, devant la commission

M.	**Ikram smaoui**	*Examinateur*
M.	**Mohamed koubaa**	*Encadrant*

Année Universitaire : 2024-2023

Analyse en temps réel et sentiment des tweets

Résumé

Ce rapport vise à analyser les données Twitter en temps réel en utilisant Tweepy et PySpark. Le rapport présente la méthodologie, les outils et les technologies utilisés pour la collecte, le stockage et l'analyse des données. Il explique comment se connecter à l'API Twitter, récupérer les tweets en temps réel, stocker et traiter les données à l'aide de PySpark, et visualiser les statistiques à l'aide de bibliothèques telles que matplotlib.pyplot. Le rapport présente les résultats et les discussions, mettant en évidence les tendances, les insights et l'analyse des sentiments des données collectées. Il conclut par un récapitulatif des principales conclusions et des suggestions d'amélioration et de développement futur.

Mot-cle : **API Twitter, Tweepy, Récupération de tweets en temps réel, Streaming, PySpark, Spark SQL , SparkSession, DataFrame Spark, Agrégation**

Abstract

The aim of this report is to analyze real-time Twitter data using Tweepy and PySpark. The report outlines the methodology, tools, and technologies used for data collection, storage, and analysis. It explains how to connect to the Twitter API, retrieve tweets in real-time, store and process the data using PySpark, and visualize the statistics using libraries such as matplotlib.pyplot. The report presents the results and discussions, highlighting the trends, insights, and sentiment analysis of the collected data. It concludes with a summary of the main findings and suggestions for future improvements and developments.

Keywords : **API Twitter, Tweepy, Récupération de tweets en temps réel, Streaming, PySpark, Spark SQL , SparkSession, DataFrame Spark, Agrégation**

Remerciement

Je tiens à exprimer ma profonde gratitude envers le professeur Mohamed Kobaa pour son soutien et son encadrement tout au long de ce projet. Sa connaissance approfondie, son expertise et sa disponibilité ont grandement contribué à la réussite de ce travail.

Je suis reconnaissant de l'opportunité qui m'a été donnée de bénéficier de ses précieux conseils et de sa guidance, qui ont joué un rôle déterminant dans la mise en œuvre et la réalisation de ce projet. Ses encouragements constants et sa confiance en mes capacités ont été une source de motivation essentielle.

Je tiens à le remercier infiniment pour son investissement et son implication, qui ont contribué à mon développement personnel et académique

Introduction Générale

L'utilisation des médias sociaux et des plateformes de partage d'informations a considérablement augmenté au cours des dernières années. Ces plateformes offrent une mine de données précieuses qui peuvent être exploitées pour comprendre les tendances, les opinions et les réactions des utilisateurs en temps réel. Dans ce rapport, nous nous concentrons sur l'analyse des données Twitter en temps réel en utilisant des techniques de collecte de données en continu et de traitement des données.

La collecte de données en temps réel est un processus essentiel pour obtenir des informations pertinentes à partir de Twitter. Nous utilisons l'API Twitter en conjonction avec la bibliothèque Tep, qui nous permet de nous connecter à Twitter et de récupérer les tweets en temps réel. L'API Twitter offre un accès aux flux de données publics, ce qui nous permet d'extraire des tweets en fonction de certains mots-clés, mots-dièse ou localisations.

Une fois les données collectées, nous utilisons Py Spark, une puissante plateforme de traitement de données distribuée, pour stocker et analyser les tweets en utilisant Spark SQL. Py Spark nous permet de manipuler de grands volumes de données de manière efficace en utilisant des fonctionnalités avancées de traitement parallèle. Nous utilisons également des fonctions Spark SQL pour agréger les données et extraire des statistiques significatives telles que le nombre de likes et de retweets.

La visualisation des statistiques est un aspect essentiel de notre analyse. Nous utilisons des bibliothèques telles que pyspark.sql.functions et matplotlib.pyplot pour représenter graphiquement les statistiques obtenues à partir des données Twitter. Les graphiques à barres sont utilisés pour visualiser les statistiques des likes et des retweets, fournissant ainsi une représentation claire et concise des tendances et des réactions des utilisateurs.

Les résultats de notre analyse des données Twitter en temps réel fournissent des informations précieuses sur les tendances et les insights liés aux sujets d'intérêt spécifiés. De plus, nous effectuons une analyse des sentiments globaux en utilisant des techniques de traitement du langage naturel pour comprendre les réactions des utilisateurs par rapport aux tweets collectés.

Chapitre : Introduction

1 Introduction

1.1 Présentation du projet et de ses objectifs

Le projet de streaming en temps réel et analyse des sentiments sur Twitter vise à collecter et analyser les tweets en direct afin de déterminer les sentiments exprimés par les utilisateurs sur la plateforme Twitter. L'objectif principal est de comprendre le climat émotionnel autour de sujets spécifiques, de produits, de marques ou d'événements, ce qui peut être précieux pour les entreprises, les chercheurs et les décideurs.

En utilisant les techniques de traitement du langage naturel (NLP) et l'analyse des sentiments, nous pouvons extraire des informations pertinentes à partir des tweets, tels que la polarité (positif, négatif ou neutre) des sentiments exprimés. Cela peut aider à évaluer la réputation d'une marque, à comprendre les opinions des utilisateurs sur un produit ou à détecter les tendances émotionnelles liées à un événement en temps réel.

1.2 Contexte et pertinence

Twitter est une plateforme de médias sociaux très populaire où les utilisateurs partagent leurs opinions, leurs réactions et leurs émotions de manière concise à travers des tweets. Jour des millions de tweets publiés chaque jour, il devient essentiel de pouvoir extraire des informations significatives à partir de ces données massives.

L'analyse des sentiments sur Twitter présente une grande pertinence dans de nombreux domaines. Markéting, les entreprises peuvent utiliser cette analyse pour surveiller la satisfaction des clients à l'égard de leurs produits ou services, identifier les problèmes émergents, ou évaluer l'impact d'une campagne marketing. Les chercheurs peuvent également utiliser cette analyse pour étudier les tendances émotionnelles de la société, suivre les réactions du public à des événements en direct ou mesurer l'opinion publique sur des questions spécifiques.

1.3 Technologies utilisées

Dans ce projet, nous utilisons plusieurs technologies pour collecter et analyser les données en temps réel sur Twitter. Les principales technologies utilisées sont les suivantes :

- Tweepy : Tweepy est une bibliothèque Python largement utilisée pour interagir avec l'API Twitter. Elle offre des fonctionnalités puissantes pour l'authentification, la collecte de données et la gestion des flux de tweets en temps réel.

- Pandas : Pandas est une bibliothèque Python populaire pour la manipulation et l'analyse des données. Nous utilisons Pandas pour prétraiter les données collectées, créer un DataFrame et effectuer des opérations de nettoyage et de transformation des données.

- Numpy: Numpy est une bibliothèque Python essentielle pour les calculs scientifiques et numériques. Nous pouvons l'utiliser pour effectuer des opérations mathématiques et statistiques sur les données.

- Matplotlib: Matplotlib est une bibliothèque de visualisation de données en Python. Elle nous permet de créer des graphiques et des visualisations pour mieux comprendre les résultats de notre analyse.

- Spark: Spark est un framework de traitement des données distribué et rapide. Nous utilisons Spark pour effectuer le traitement des données en temps réel à grande échelle. Il nous permet de gérer et d'analyser efficacement les données massives collectées à partir de Twitter.

1.4 Resume

Technologies, nous pouvons mettre en place un système robuste pour collecter et analyser les tweets en temps réel, extraire les sentiments exprimés par les utilisateurs et fournir des informations précieuses pour la prise de décision.

Chapitre : Collecte de données

2 Collecte de données

2.1 Utilisation de l'API Twitter

Pour collecter des tweets en temps réel, nous utilisons l'API Twitter, qui fournit un accès aux données publiques disponibles sur la plateforme. L'API Twitter nous permet de rechercher et de filtrer les tweets en fonction de certains critères tels que des mots-clés, des utilisateurs spécifiques ou des emplacements géographiques.

En utilisant la bibliothèque Tweepy en Python, nous pouvons facilement interagir avec l'API Twitter. Nous configurons l'API en fournissant des clés d'authentification (clé d'API, clé secrète d'API, jeton d'accès et jeton secret d'accès) qui nous sont fournies lors de la création d'une application Twitter.

Voici le code :

```
class TwitterAuthenticator():

    def authenticate_twitter_app(self):
        auth = OAuthHandler(config.API_Key, config.API_Key_Secret)
        auth.set_access_token(config.Access_Token, config.Access_Token_Secret)
        return auth
```

2.2 Présentation de la classe TwitterClient

La classe TwitterClient est une classe personnalisée que nous créons pour faciliter l'authentification et l'accès à l'API Twitter. Elle encapsule les fonctionnalités fournies par la bibliothèque Tweepy et fournit des méthodes pratiques pour interagir avec l'API.

Dans la classe TwitterClient, nous commençons par initialiser les clés d'authentification en utilisant les informations fournies par Twitter. Ensuite, nous créons une instance de l'API Twitter en utilisant ces clés.

La classe TwitterClient peut contenir des méthodes telles que:

- Méthode de recherche de tweets: Cette méthode utilise l'API pour effectuer une requête de recherche de tweets en utilisant des mots-clés spécifiques ou d'autres critères. Elle renvoie les tweets correspondants.

- Méthode de collecte de flux en temps réel: Cette méthode utilise l'API pour collecter un flux continu de tweets en temps réel. Elle peut être utilisée pour collecter des tweets en continu pendant une certaine période ou jusqu'à ce qu'une condition spécifique soit remplie.

Voici le code :

```
class TwitterClient():
    def __init__(self, twitter_user=None):
        self.auth = TwitterAuthenticator().authenticate_twitter_app()
        self.twitter_client = API(self.auth)

        self.twitter_user = twitter_user

    def get_twitter_client_api(self):
        return self.twitter_client

    def get_user_timeline_tweets(self, num_tweets):
        tweets = []
        for tweet in Cursor(self.twitter_client.user_timeline,
id=self.twitter_user).items(num_tweets):
            tweets.append(tweet)
        return tweets

    def get_friend_list(self, num_friends):
        friend_list = []
        for friend in Cursor(self.twitter_client.friends,
id=self.twitter_user).items(num_friends):
            friend_list.append(friend)
        return friend_list

    def get_home_timeline_tweets(self, num_tweets):
        home_timeline_tweets = []
        for tweet in Cursor(self.twitter_client.home_timeline,
id=self.twitter_user).items(num_tweets):
            home_timeline_tweets.append(tweet)
        return home_timeline_tweets
```

2.3 Utilisation de TwitterListener

TwitterListener est une classe personnalisée qui hérite de la classe StreamListener de Tweepy. Elle nous permet d'écouter et de capturer des tweets en temps réel à partir du flux Twitter.

Dans la classe TwitterListener, nous définissons des méthodes pour gérer différents événements tels que la réception d'un nouveau tweet, une erreur de connexion ou la fin du flux. Par exemple, la méthode on_status est appelée chaque fois qu'un nouveau tweet est reçu.

Dans la méthode on_status, nous pouvons définir des actions à effectuer, telles que l'enregistrement du tweet dans un fichier, l'extraction des informations pertinentes du tweet, ou l'analyse des sentiments.

Lors de l'utilisation de TwitterListener, nous créons une instance de cette classe et l'associons à l'API Twitter en utilisant la méthode stream de l'API. Cela nous permet de capturer les tweets en temps réel et de les traiter selon nos besoins.

En utilisant la classe TwitterClient et TwitterListener ensemble, nous pouvons authentifier notre application, accéder à l'API Twitter, et collecter les tweets en temps réel en utilisant les fonctionnalités de recherche ou de flux continu. Les tweets collectés peuvent être enregistrés dans

Voici le code :

```
class TwitterListener(Stream):

    """
    This is a basic listener that just prints received tweets to stdout.
    """
    def __init__(self, fetched_tweets_filename):
        self.fetched_tweets_filename = fetched_tweets_filename

    def on_data(self, data):
        try:
            print(data)
            with open(self.fetched_tweets_filename, 'a') as tf:
                tf.write(data)
            return True
        except BaseException as e:
            print("Error on_data %s" % str(e))
        return True

    def on_error(self, status):
        if status == 420:
            # Returning False on_data method in case rate limit occurs.
            return False
        print(status)
```

Chapitre : Prétraitement des données

3 Prétraitement des données

3.1 Processus de prétraitement

Le prétraitement des tweets collectés consiste à nettoyer et à préparer les données pour l'analyse ultérieure. Il comprend généralement les étapes suivantes :

- Suppression des caractères indésirables : Les tweets peuvent contenir des caractères spéciaux, des symboles ou des URL. Il est courant de supprimer ces éléments afin de ne conserver que le texte du tweet lui-même.

- Suppression des mentions et des mots-dièse : Dans de nombreux cas, les mentions d'utilisateurs ou les mots-dièse ne sont pas pertinents pour l'analyse des sentiments. Par conséquent, il est courant de les supprimer ou de les remplacer par un espace vide.

- Normalisation des mots : La normalisation des mots vise à réduire les variantes linguistiques et à ramener les mots à leur forme de base. Par exemple, convertir tous les mots en minuscules et supprimer la ponctuation.

- Suppression des mots vides : Les mots vides sont des mots couramment utilisés qui n'apportent pas beaucoup de sens à l'analyse, tels que "le", "et", "mais", etc. Ils peuvent être supprimés pour réduire le bruit dans les données.

3.2 Extraction des caractéristiques

Une fois les tweets prétraités, nous pouvons extraire les caractéristiques pertinentes pour notre analyse. Les caractéristiques couramment extraites des tweets peuvent inclure :

- Texte du tweet : Il s'agit du contenu principal du tweet qui contient les informations que nous souhaitons analyser.

- Date du tweet : La date et l'heure à laquelle le tweet a été publié. Cela peut être utile pour l'analyse temporelle ou pour détecter des tendances.

- Source du tweet : La source à partir de laquelle le tweet a été publié, telle que l'application Twitter ou le client utilisé. Cela peut fournir des informations sur la plateforme à partir de laquelle les utilisateurs publient des tweets.

- Likes et retweets : Le nombre de likes (j'aime) et de retweets que le tweet a reçus. Ces indicateurs peuvent être utilisés pour évaluer la popularité ou l'engagement d'un tweet.

3.3 Création d'un DataFrame

Une fois que nous avons extrait les caractéristiques pertinentes des tweets, nous pouvons les organiser dans une structure de données tabulaire appelée DataFrame. Pour cela, nous utilisons la bibliothèque Python appelée pandas.

Pandas offre une structure de données efficace et flexible pour manipuler et analyser des données tabulaires. Nous pouvons créer un DataFrame en utilisant les caractéristiques extraites des tweets, où chaque colonne du DataFrame représente une caractéristique spécifique et chaque ligne représente un tweet.

La création d'un DataFrame à partir des tweets collectés implique généralement l'utilisation de la fonction `DataFrame` de pandas, en passant les données sous forme de dictionnaire ou de liste de tuples. Chaque clé du dictionnaire ou élément du tuple représente une caractéristique, et les valeurs correspondantes représentent les valeurs de cette caractéristique pour chaque tweet

```
class TweetAnalyzer():
    """
    Functionality for analyzing and categorizing content from tweets.
    """


    def tweets_to_data_frame(self, tweets):
        df = pd.DataFrame([tweet.text for tweet in tweets])


        return df
```

Chapitre : Analyse de sentiment

4 Analyse de sentiment

4.1 Description de l'approche utilisée

L'approche utilisée pour l'analyse de sentiment est basée sur une méthode de mots-clés. Dans le code fourni, nous utilisons des listes de mots positifs et négatifs pour évaluer le sentiment exprimé dans les tweets. Les mots positifs sont associés à des émotions positives, tandis que les mots négatifs sont associés à des émotions négatives. L'objectif est d'attribuer un score de sentiment à chaque tweet, indiquant s'il est positif, négatif ou neutre.

4.2 Présentation des listes de mots positifs et négatifs

Dans le code, les listes de mots positifs et négatifs sont définies manuellement dans le code lui-même. Les mots positifs sont stockés dans la liste "positive_words", tandis que les mots négatifs sont stockés dans la liste "negative_words". Ces listes peuvent être modifiées et personnalisées en fonction des besoins de l'analyse de sentiment.

4.3 Explication de la fonction d'analyse

Dans le code fourni, la fonction d'analyse de sentiment s'appelle "analyze_sentiment". Cette fonction prend en entrée le texte d'un tweet et compare chaque mot du tweet avec les listes de mots positifs et négatifs pour déterminer le sentiment exprimé. La fonction attribue un score de sentiment en fonction de la présence de mots positifs ou négatifs dans le tweet. Si aucun mot positif ou négatif n'est trouvé, le score de sentiment est défini comme neutre (0).

Voici le code de la fonction "analyze_sentiment" :

```python
def analyze_sentiment(text):
    positive_words = ["Love", "Joy", "Happy", "Excitement", "Hope", "Success"]  # Liste des mots positifs
    negative_words = ["Hate", "Sadness", "Anger", "Failure", "Fear", "Stress"]  # Liste des mots négatifs

    sentiment_label = 0

    # Parcours des mots du texte du tweet
```

```
    for word in text.split():
        if word in positive_words:
            sentiment_label = 1  # Le tweet est positif
            break
        elif word in negative_words:
            sentiment_label = -1  # Le tweet est négatif
            break

    return sentiment_label
```

La fonction "analyze_sentiment" compare chaque mot du texte du tweet avec les mots des listes de mots positifs et négatifs. Si un mot positif est trouvé, le sentiment est considéré comme positif (1). Si un mot négatif est trouvé, le sentiment est considéré comme négatif (-1). Si aucun mot positif ou négatif n'est trouvé, le sentiment est considéré comme neutre (0).

Dans le code, cette fonction est ensuite appliquée à la colonne "text" du DataFrame pour attribuer un score de sentiment à chaque tweet. Le résultat est stocké dans une nouvelle colonne appelée "sentiment".

En résumé, l'analyse de sentiment des tweets repose sur une méthode de mots-cl

és où chaque mot du tweet est comparé avec des listes de mots positifs et négatifs. La fonction d'analyse de sentiment attribue un score de sentiment à chaque tweet, indiquant s'il est positif, négatif ou neutre.

Chapitre : Stockage et le traitement des données

5 Stockage et le traitement des données

5.1 Explication de l'utilisation de Spark Session

Pour travailler avec PySpark, nous utilisons SparkSession, qui est une classe principale de Spark permettant de créer une session Spark. Une session Spark représente une connexion unique à un cluster Spark et fournit un point d'entrée pour interagir avec les données. Elle offre diverses fonctionnalités, telles que le chargement de données, l'exécution de requêtes, le traitement parallèle et la gestion des ressources.

Dans notre code, nous avons déjà créé une session Spark à l'aide de la ligne suivante :

```python
spark = SparkSession.builder.appName("tweet").config("spark.drive.meory","20g").getOrCreate()
```

Cette ligne de code crée une session Spark avec le nom "tweet" et une configuration spécifique pour la mémoire (20 Go dans cet exemple). L'utilisation de `getOrCreate()` permet de récupérer une session Spark existante ou d'en créer une nouvelle si aucune session n'existe déjà

5.2 Chargement des données dans un DataFrame

Une fois que nous avons une session Spark, nous pouvons charger nos données à partir d'un fichier CSV dans un DataFrame Spark. Dans notre code, nous utilisons la fonction `read.csv()` de Spark pour charger les données à partir du fichier 'twitter_data.csv'. Voici la ligne de code correspondante :

```python
data = spark.read.csv('twitter_data.csv', header=True, inferSchema=True)
```

Cette ligne de code lit le fichier CSV en spécifiant les options `header=True` pour indiquer que le fichier contient un en-tête de colonne, et `inferSchema=True` pour permettre à Spark d'inférer les types de données des colonnes. Le DataFrame résultant, nommé 'data' dans notre cas, contient les données du fichier CSV, prêt à être utilisé pour l'analyse et le traitement ultérieurs.

5.3 Utilisation des fonctions Spark SQL et analyser les données

Une fois que nous avons chargé les données dans un DataFrame Spark, nous pouvons utiliser les fonctions de Spark SQL pour effectuer des opérations d'agrégation et d'analyse sur les données. Spark SQL fournit une interface SQL et des fonctions de traitement de données pour interroger et manipuler les DataFrames.

Dans notre code, nous avons déjà utilisé certaines des fonctionnalités de Spark SQL pour effectuer des opérations de base. Par exemple, nous avons utilisé les fonctions `printSchema()` pour afficher le schéma du DataFrame 'data', `show()` pour afficher les premières lignes du DataFrame, et `describe()` pour obtenir les statistiques descriptives du DataFrame.

Voici un exemple de l'utilisation de ces fonctions :

```python
data.printSchema()  # Affiche le schéma du DataFrame
data.show(5)        # Affiche les premières lignes du DataFrame
data.describe().show()  # Affiche les statistiques descriptives du DataFrame
```

En utilisant les fonctions Spark SQL, vous pouvez effectuer des opérations plus avancées telles que les jointures, les filtres, les regroupements, les agrégations et les calculs sur les colonnes. Ces opérations vous permettent d'extraire des informations significatives à partir des données et

de les utiliser pour votre analyse de sentiment ou d'autres objectifs.

En résumé, dans cette partie du rapport, nous avons expliqué l'utilisation de SparkSession pour créer une session Spark, chargé les données à partir d'un fichier CSV dans un DataFrame Spark, et utilisé les fonctions Spark SQL pour effectuer des opérations d'agrégation et d'analyse sur les données. Ces étapes vous permettent de préparer les données pour l'analyse ultérieure et d'effectuer des opérations sophistiquées sur les données Twitter collectées.

Chapitre : Visualisation des statistiques

6 Visualisation des statistiques

6.1 Importation des bibliothèques

Pour effectuer la visualisation des statistiques, nous devons importer deux bibliothèques : `pyspark.sql.functions` et `matplotlib.pyplot`. La bibliothèque `pyspark.sql.functions` est utilisée pour effectuer des opérations de manipulation de données sur les colonnes, tandis que `matplotlib.pyplot` est utilisée pour créer des graphiques.

Voici comment vous pouvez importer ces bibliothèques dans votre code Python :

```python
from pyspark.sql.functions import *
import matplotlib.pyplot as plt
```

6.2 Sélection des colonnes pertinentes

Dans cette étape, nous devons sélectionner les colonnes pertinentes dans notre DataFrame Spark pour calculer les statistiques. Dans notre cas, nous nous intéressons aux statistiques de "likes" (j'aime) et de "retweets" (retweets). Assurez-vous que votre DataFrame contient ces colonnes avant de procéder à cette étape.

Voici comment vous pouvez sélectionner les colonnes pertinentes dans votre DataFrame Spark :

```python
stats_df = tweets_df.select("likes", "retweets")
```

6.3 Agrégation des statistiques à l'aide de Spark SQL

Maintenant que nous avons sélectionné les colonnes pertinentes, nous pouvons utiliser Spark SQL pour agréger les statistiques. Spark SQL offre une syntaxe similaire à SQL pour effectuer des opérations d'agrégation sur les données.

Voici comment vous pouvez utiliser Spark SQL pour agréger les statistiques :

```python
stats_aggregated = stats_df.agg(sum("likes").alias("total_likes"), sum("retweets").alias("total_retweets"))
```

Dans cet exemple, nous utilisons la fonction `agg` pour agréger les valeurs de "likes" et de "retweets". La fonction `sum` est utilisée pour calculer la somme des valeurs de chaque colonne. Nous utilisons également `alias` pour renommer les colonnes agrégées.

6.4 Tracage de graphique à barres

Maintenant que nous avons agrégé les statistiques, nous pouvons les représenter graphiquement à l'aide d'un graphique à barres. La bibliothèque `matplotlib.pyplot` nous permet de créer facilement des graphiques dans Python.

Voici comment vous pouvez tracer un graphique à barres pour représenter les statistiques :

```python
likes = stats_aggregated.select("total_likes").first()[0]
retweets = stats_aggregated.select("total_retweets").first()[0]

# Création du graphique à barres
plt.bar(["Likes", "Retweets"], [likes, retweets])
plt.xlabel("Statistiques")
plt.ylabel("Nombre")
plt.title("Statistiques des Likes et des Retweets")

# Affichage du graphique
plt.show()
```

Dans cet exemple, nous utilisons la fonction `bar` de `matplotlib.pyplot` pour créer un graphique à barres. Nous spécifions les labels des axes x et y à l'aide des fonctions `xlabel` et `ylabel`, et donnons un titre au graphique avec la fonction `title`. Enfin, nous utilisons la fonction `show` pour afficher le graphique.

En suivant ces étapes, vous pourrez importer les bibliothèques nécessaires, sélectionner les colonnes pertinent

es, agréger les statistiques à l'aide de Spark SQL et tracer un graphique à barres représentant les statistiques des likes et des retweets de vos données Twitter.

Chapitre : Résultats et Discussions

7 Résultats et Discussions

7.1 Résultats et discussions

Dans cette section, nous présenterons les résultats obtenus à partir de l'analyse des données Twitter en temps réel. Nous pourrons inclure des tableaux, des graphiques ou d'autres formes de représentation des données pertinentes. Les résultats peuvent inclure des informations telles que le nombre total de tweets collectés, les tendances observées, les statistiques clés, etc.

Il est important de fournir une description détaillée des résultats et de les analyser de manière critique. Nous pouvons identifier des modèles ou des tendances intéressantes dans les données collectées, expliquer les variations observées, mettre en évidence les points saillants, etc. Il est également utile de comparer les résultats avec les objectifs initiaux de l'analyse ou avec d'autres études similaires.

7.2 Présentation des résultats

Dans cette sous-section, nous présenterons de manière claire et concise les résultats obtenus à partir de l'analyse des données Twitter en temps réel. Cela peut inclure des informations sur les mots-dièse les plus populaires, les utilisateurs les plus influents, les sujets tendance, les mentions fréquentes, etc. Les résultats peuvent être présentés sous forme de graphiques, de diagrammes ou de tableaux pour faciliter la compréhension.

Il est important de fournir des descriptions appropriées pour chaque résultat présenté afin d'expliquer leur signification et leur pertinence pour l'analyse. Par exemple, si un graphique montre l'évolution du nombre de tweets par heure, il est nécessaire de préciser les variations remarquables et d'en discuter les possibles raisons.

7.3 Discussion sur les tendances et les insights

Dans cette partie, nous discuterons des tendances et des insights observés dans les données Twitter en temps réel. Nous pourrons identifier des motifs intéressants, des comportements d'utilisateurs, des réactions à des

événements spécifiques, etc. Il est important de fournir des explications et des interprétations pour ces tendances et insights.

Nous pouvons également établir des liens avec des événements du monde réel ou des actualités qui pourraient expliquer certaines des tendances observées. Par exemple, si une augmentation significative du nombre de tweets est observée sur un sujet spécifique, nous pouvons discuter de la relation avec un événement majeur ou une campagne médiatique.

7.4 Analyse des sentiments et des réactions

Dans cette sous-section, nous analyserons les sentiments globaux exprimés dans les tweets collectés et les réactions des utilisateurs. Cela peut être réalisé en utilisant des techniques de traitement du langage naturel (NLP) pour détecter les sentiments positifs, négatifs ou neutres dans les tweets.

Nous pourrons présenter les résultats sous forme de graphiques ou de pourcentages pour illustrer la répartition des sentiments. De plus, nous pourrons discuter des principales tendances et des réactions des utilisateurs par rapport à un sujet spécifique ou à un événement.

Il est essentiel d'étayer nos discussions avec des exemples concrets de tweets ou de citations pour étayer nos observations

et nos conclusions.

En résumé, cette partie du rapport présentera les résultats obtenus à partir de l'analyse des données Twitter en temps réel, en fournissant des descriptions détaillées, des graphiques et des analyses critiques. Nous mettrons en évidence les tendances, les insights et les réactions des utilisateurs, en les reliant au contexte et aux objectifs de l'analyse.

Chapitre : Conclusions du Rapport

8 Conclusions du Rapport

8.1 Récapitulation des principales

Au cours de cette analyse des données Twitter en temps réel, nous avons collecté un grand nombre de tweets relatifs à un sujet spécifique et les avons analysés en utilisant des techniques de traitement du langage naturel et des outils tels que Tweepy et PySpark. À partir de cette analyse, nous avons pu tirer les conclusions suivantes :

- Tendance dominante : Nous avons identifié la tendance dominante sur Twitter concernant le sujet étudié. Cela nous a permis de comprendre les sujets les plus discutés et les préoccupations principales des utilisateurs.

- Insights clés : En analysant les tweets, nous avons obtenu des insights précieux sur les opinions, les sentiments et les réactions des utilisateurs par rapport au sujet. Nous avons pu identifier les points de vue majoritaires et minoritaires, les tendances émergentes, ainsi que les éléments qui suscitent le plus d'engagement et de réactions de la part des utilisateurs.

- Statistiques importantes : Nous avons agrégé et analysé les données pour extraire des statistiques clés telles que le nombre total de tweets, le nombre de likes et de retweets, la répartition géographique des utilisateurs, etc. Ces statistiques nous ont permis de quantifier l'ampleur de l'activité autour du sujet étudié.

- Analyse des sentiments : Nous avons utilisé des techniques d'analyse des sentiments pour évaluer le ton général des tweets. Cela nous a permis de déterminer si les utilisateurs avaient des sentiments positifs, négatifs ou neutres par rapport au sujet. Ces informations sont précieuses pour comprendre l'opinion générale et l'impact émotionnel du sujet sur les utilisateurs.

8.2 Perspectives d'amélioration et de développement futur

Bien que cette analyse des données Twitter en temps réel ait permis d'obtenir des insights intéressants, il existe des possibilités d'amélioration et de développement futur. Voici quelques perspectives à considérer :

- Amélioration de la collecte des données : Il est possible d'explorer des méthodes de collecte plus avancées, telles que l'utilisation de flux de données en continu pour obtenir une couverture plus large et en temps réel des tweets. Cela permettrait d'obtenir une image plus complète et en temps réel des conversations sur le sujet étudié.

- Utilisation de techniques d'analyse avancées : En plus de l'analyse des sentiments, il serait intéressant d'explorer d'autres techniques d'analyse avancées telles que l'analyse de réseau social, l'identification des influenceurs, l'analyse de la corrélation entre les variables, etc. Ces approches permettraient d'obtenir des informations plus approfondies et des relations plus complexes entre les données.

- Intégration de données supplémentaires : En complément des tweets, l'intégration de données supplémentaires telles que des données démographiques, des données contextuelles ou des données provenant d'autres sources de médias sociaux pourrait enrichir l'analyse et donner une perspective plus large sur le sujet.

- Développement d'une interface de visualisation interactive : La création d'une interface interactive permettant aux utilisateurs d'explorer les résultats de l'analyse de manière conviviale et intuitive serait une amélioration significative. Cela

Permettrait aux utilisateurs de découvrir eux-mêmes des insights et de poser des questions spécifiques sur les données.

En conclusion, cette analyse des données Twitter en temps réel nous a fourni des conclusions importantes sur le sujet étudié. Les insights obtenus nous ont permis de mieux comprendre les opinions, les sentiments et les réactions des utilisateurs. Cependant, il existe encore des possibilités d'amélioration et de développement futur pour affiner l'analyse et explorer davantage les données sociales.

8.3 Qr code