31/03/2024

Rapport d’analyse

Projet NoSQL

Franklin essono & Nadir Zourdani
ESIEA

Contents

1. Introduction

Avec l'avancement actuel du big data, les bases de données NoSQL sont devenues des outils inestimables pour gérer des ensembles de données vastes et diversifiés. Ce projet explore deux systèmes de gestion de bases de données NoSQL significatifs : MongoDB et Neo4j. MongoDB, une base de données orientée document, offre une grande flexibilité dans le stockage et la récupération de

données non structurées, tandis que Neo4j, une base de données orientée graphe, excelle dans la modélisation et l'analyse des relations complexes entre les données.

L'objectif principal de ce projet était de développer une application en Python capable d'interagir efficacement avec ces bases de données NoSQL hébergées dans le cloud. Cette application vise à exécuter une série de tâches spécifiques : se connecter aux bases de données, effectuer diverses interrogations et manipulations de données, et enfin, analyser et visualiser les résultats obtenus pour en tirer des insights pertinents.

Ce rapport présente le processus de développement de l'application, la logique employée pour répondre aux questions posées par les bases de données, et une analyse des résultats obtenus. Enfin, il conclut sur les leçons apprises tout au long de ce projet et les perspectives. En abordant ces points, ce document vise à fournir une vue d'ensemble complète des défis et des accomplissements associés à la manipulation et à l'analyse de données NoSQL dans un environnement cloud.

2. Logique de résolution des questions

• MongoDB Manager (mongodb_manager.py)

Dans le cadre de notre projet, le fichier mongodb_manager.py joue un rôle crucial dans l'interrogation et la manipulation de données au sein de la base de données MongoDB. Voici comment ce gestionnaire aborde les questions posées :

- **Gestion des entités :** Le gestionnaire établit une connexion avec la base de données MongoDB et utilise différentes méthodes pour comptabiliser les entités telles que les utilisateurs, les tweets et les hashtags. Ces comptages sont fondamentaux pour répondre aux questions concernant les statistiques de base de l'application.
- **Interrogation par hashtags :** Le gestionnaire effectue des requêtes spécifiques pour récupérer le nombre de tweets contenant des hashtags particuliers et le nombre d'utilisateurs uniques ayant utilisé un hashtag donné. Cette fonctionnalité répond aux questions relatives à l'engagement et à la popularité des sujets sur la plateforme.
- **Analyse des discussions :** À travers une série de méthodes, le gestionnaire identifie les tweets qui initient des discussions et évalue les conversations en calculant la plus longue discussion basée sur le nombre de réponses. Cela permet d'aborder des questions sur l'interaction et la portée des discussions entre utilisateurs.
- **Évaluation de popularité :** En triant et en limitant les requêtes, le gestionnaire détermine les tweets et les hashtags les plus populaires. Cette analyse est cruciale pour comprendre les tendances et les intérêts des utilisateurs.
- **Relations entre utilisateurs :** Le gestionnaire explore la dynamique sociale en identifiant les utilisateurs avec un nombre significatif de followers ou ceux qui suivent de nombreux autres utilisateurs, aidant à répondre aux questions sur l'influence et les réseaux sociaux.

• Neo4j Manager (neo4j_manager.py)

Le fichier neo4j_manager.py s'attaque aux aspects de la gestion des données orientées graphe avec Neo4j :

- **Connexion et requêtes Cypher :** Le gestionnaire établit une connexion sécurisée avec Neo4j et exécute des requêtes Cypher pour analyser les relations entre les utilisateurs. Cela comprend

la récupération du nombre de followers et de suivis pour des utilisateurs spécifiques, comme 'Spinomade'.

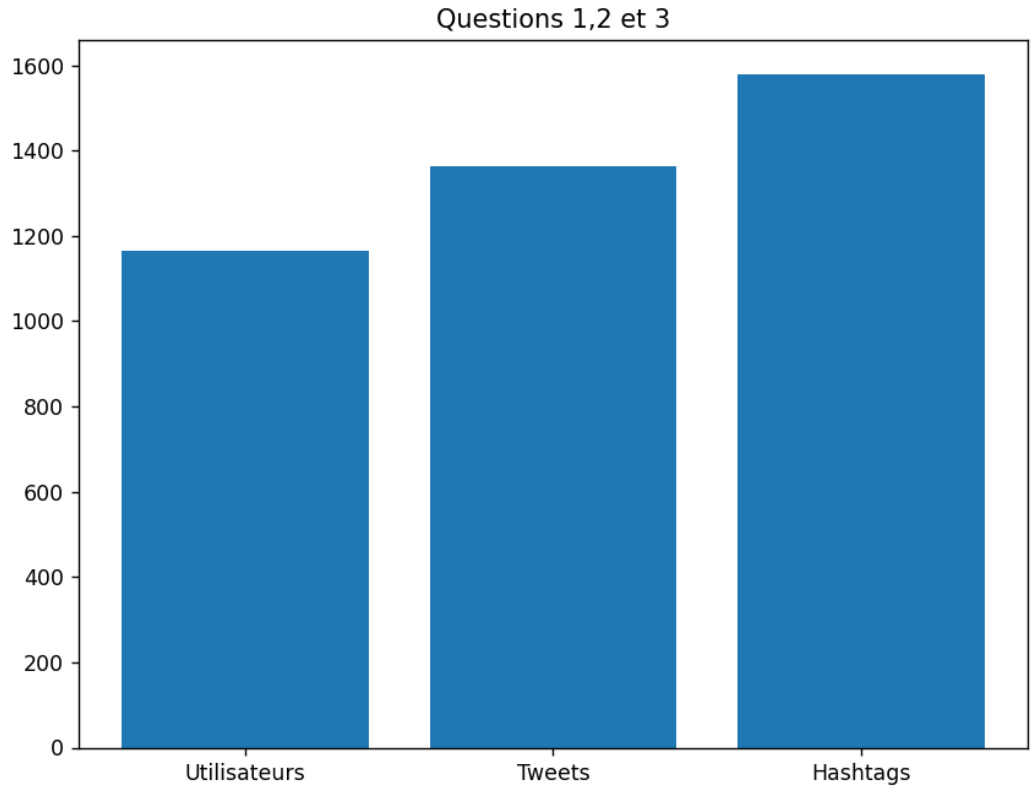
- **Analyse des relations :** En employant des requêtes complexes, le gestionnaire dévoile les réseaux de relations, identifiant par exemple les followers mutuels, les utilisateurs suivis et les followers de 'Spinomade'. Cela répond à des questions sur les structures sociales et l'interconnectivité entre les utilisateurs.
- **Intégration MongoDB-Neo4j :** Le gestionnaire utilise l'intégration entre MongoDB et Neo4j pour enrichir les requêtes Neo4j avec des données utilisateur provenant de MongoDB, offrant ainsi une vue complète des interactions et des relations.

En combinant les fonctionnalités de **mongodb_manager.py** et **neo4j_manager.py**, notre application peut répondre de manière exhaustive aux questions posées par le projet. Ces gestionnaires facilitent non seulement la récupération et l'analyse des données mais aussi leur visualisation et leur compréhension dans le contexte des réseaux sociaux numériques et de l'engagement des utilisateurs.

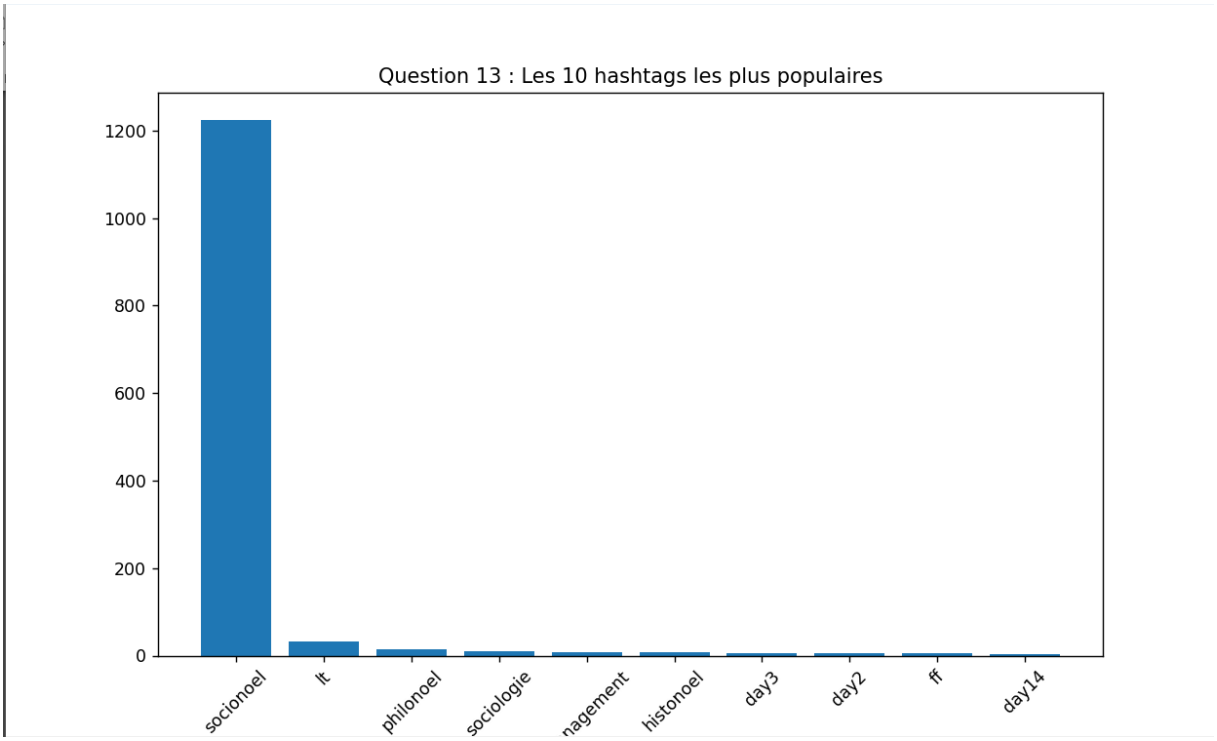
3. Analyse des résultats obtenus

L'analyse des données recueillies et générées par notre application offre un aperçu significatif du comportement et des interactions des utilisateurs sur la plateforme sociale analysée.

• Analyse des données MongoDB



Nombre d'utilisateurs, de tweets et de hashtags



Les 10 hashtags les plus populaires

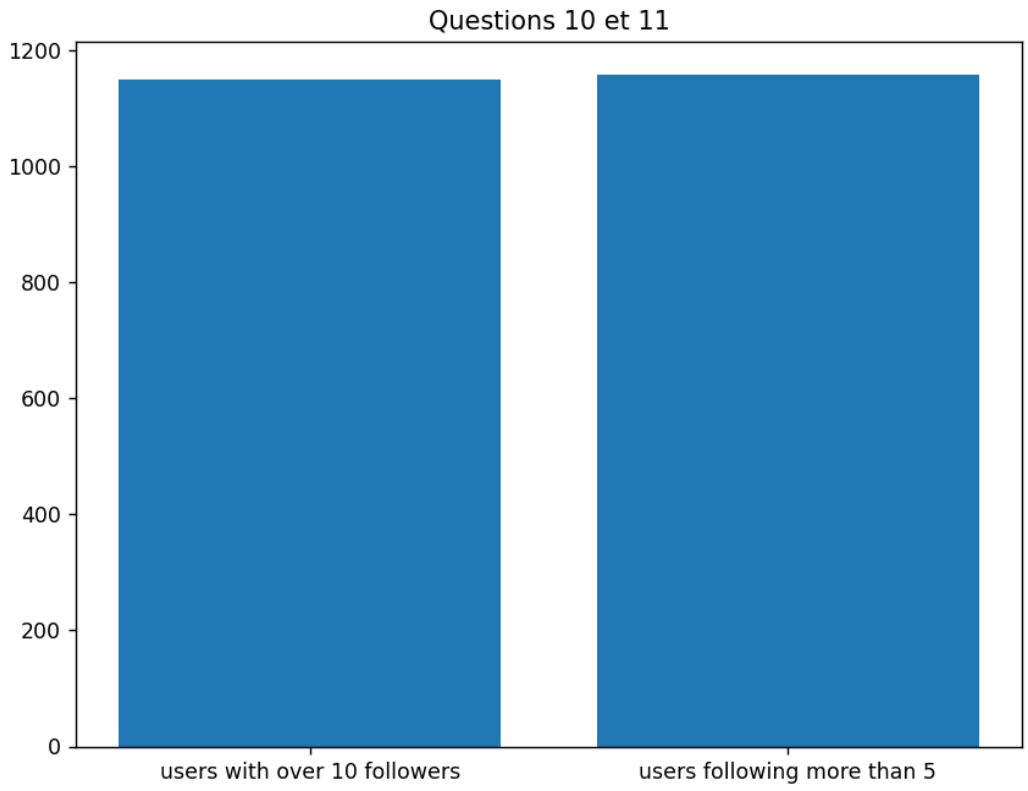
La première visualisation met en évidence le nombre d'utilisateurs, de tweets et de hashtags. Avec respectivement environ 1200, 1400 et 1500, les données indiquent une plateforme active où les hashtags légèrement surpassent les tweets et les utilisateurs en termes de quantité. Ce déséquilibre peut suggérer une forte culture de tagging, importante pour le ciblage et la découverte de contenu.

Les 10 hashtags les plus populaires, dominés de manière écrasante par '#socionoel' avec 1225 mentions, révèlent des tendances saisonnières ou des événements spécifiques captivant l'attention des utilisateurs. La chute drastique de mentions des hashtags suivants, avec '#lt' ne recueillant que 33 mentions, indique un intérêt concentré, probablement autour d'un événement unique ou d'une campagne spécifique, laissant peu de place pour des thèmes diversifiés dans la conversation générale.

Les 10 tweets les plus populaires

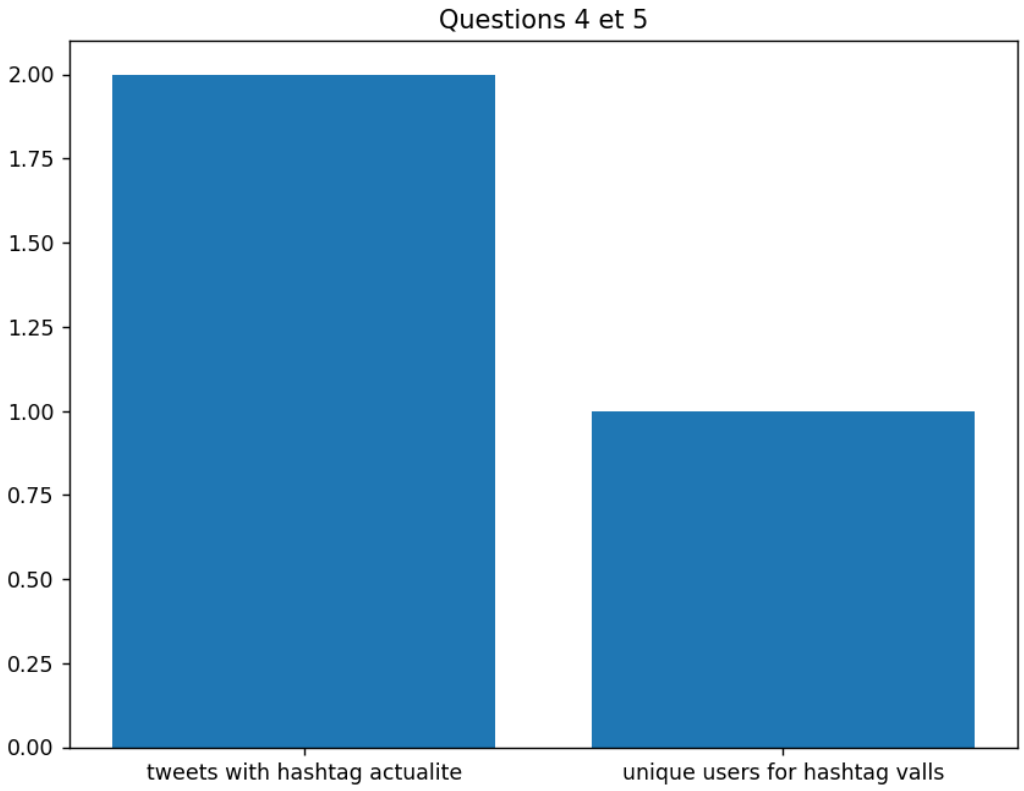
Le tweet le plus populaire semble être en espagnol et parle de se "désintoxiquer" après une campagne, probablement politique vu le contexte habituel de ce terme. L'auteur du tweet recommande un livre de Gregorio Morán, ce qui suggère que le livre pourrait offrir un certain recul ou une critique pertinente.

La visualisation indique qu'il domine nettement en termes de popularité, avec un nombre de favoris qui le distingue des autres tweets listés. Les neuf autres tweets semblent avoir une popularité plus modérée et relativement comparable entre eux.



Q10 et Q11

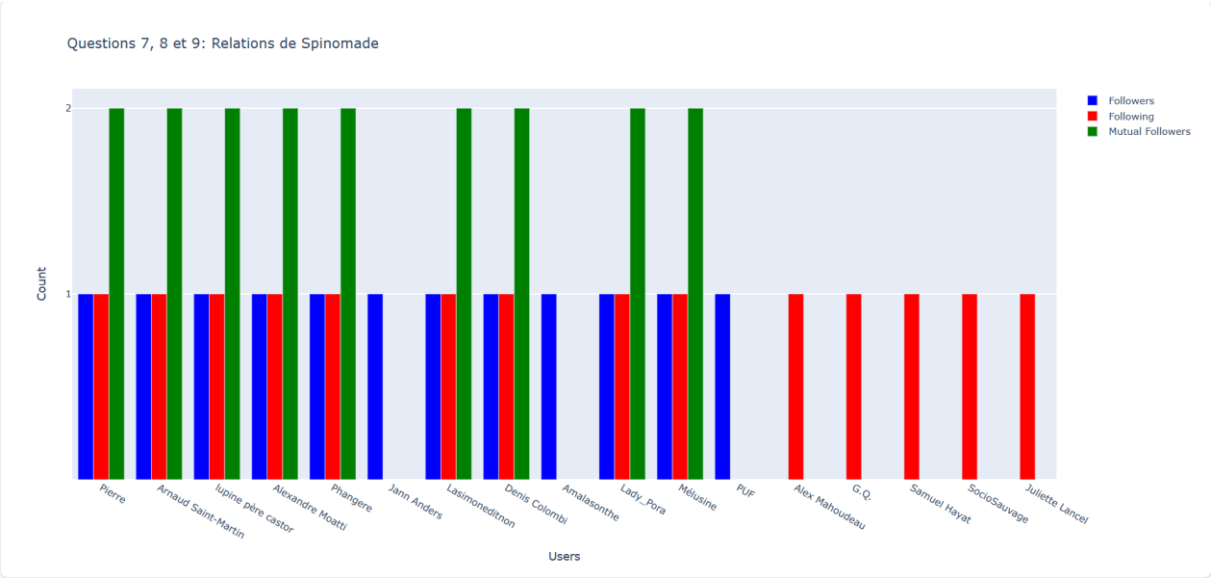
L'examen du nombre de followers illustre la structure de l'influence sur la plateforme. Une majorité d'utilisateurs ayant plus de 10 followers pointe vers une stratification où certains comptes agissent comme des pôles d'influence. Inversement, une prépondérance d'utilisateurs suivant plus de 5 personnes indique un réseau étendu et actif, avec une tendance à la recherche de diversité dans les sources d'information et de contenu.



Q4 et Q5

L'analyse des tweets par hashtag révèle les centres d'intérêt dominants au sein de la communauté. A travers le graphe nous pouvons clairement voir à quel point l'hashtag actualité est beaucoup plus populaire que l'hashtag valls.

- Analyse des données Neo4j



Relations de Spinomade

Les relations de Spinomade, illustrées par le dernier graphique, offrent un aperçu fascinant des interactions sociales. La distinction entre les followers (bleu), les followings (rouge) et les relations mutuelles (vert) permet de comprendre les dynamiques d'influence et d'engagement. Notamment, certains utilisateurs, bien que suiveurs, ne sont pas suivis en retour, ce qui indique une hiérarchie ou une disparité dans les relations d'influence.

- Intégration des données et influenceurs principaux

En fusionnant les données recueillies à partir de MongoDB et Neo4j, une corrélation significative émerge entre le volume de contenu produit, tel que les tweets et les hashtags, et la structure des réseaux sociaux, notamment les suivis et les followers. Cette relation souligne comment les interactions et les connexions entre utilisateurs sont influencées par le contenu qu'ils génèrent et partagent. En outre, l'analyse révèle que les principaux vecteurs d'influence, marqués par leur large base de followers, jouent un rôle crucial dans l'orientation du dialogue et des tendances au sein de la communauté. Leur capacité à mobiliser, à informer et à influencer est particulièrement visible lorsqu'on considère l'impact significatif de comptes de personnalités publiques et d'entités telles que 'Libération' et 'Manuel Valls'. Ces derniers ne se contentent pas de participer au discours ; ils le façonnent, mettant en évidence le rôle prépondérant que les institutions et les figures politiques jouent dans la sphère des médias sociaux.

4. Requêtes utilisées

• Script d'initialisation de la base de données Neo4j

Le script d'initialisation joue un rôle crucial dans la préparation de notre environnement de base de données Neo4j, permettant de structurer efficacement les données pour des analyses ultérieures. Voici une explication détaillée du processus :

```
neo4j_custom > init_neo4j.py > ...
from py2neo import Graph

# Connexion à Neo4j
uri = "neo4j+s://dc15bb96.databases.neo4j.io"
user = "neo4j"
password = "XZLEUi-SSuZkCSafSdxd7qtJJGHRDTmVRZNBzDAYc58"
graph = Graph(uri, auth=(user, password))

# Import des données des utilisateurs et de leurs relations "follow"
query_follow = """
LOAD CSV WITH HEADERS FROM 'https://raw.githubusercontent.com/Eetff4422/project_nosql/main/neo4j_custom/docs/tw_user_follow.csv' AS row
WITH row WHERE row.sourceIdUser IS NOT NULL AND row.targetIdUser IS NOT NULL AND trim(row.sourceIdUser) <> "" AND trim(row.targetIdUser) <> ""
MERGE (user1:User {id: row.sourceIdUser})
MERGE (user2:User {id: row.targetIdUser})
MERGE (user1)-[:FOLLOWS]->(user2);
"""
graph.run(query_follow)

# Import des données des tweets, des utilisateurs, et des relations "retweet"
query_retweet = """
LOAD CSV WITH HEADERS FROM 'https://raw.githubusercontent.com/Eetff4422/project_nosql/main/neo4j_custom/docs/tweet_retweet.csv' AS row
MERGE (tweet:Tweet {id: row.idTweet})
MERGE (user:User {id: row.idUser})
MERGE (originalTweet:Tweet {id: row.idRetweet})
MERGE (user)-[:POSTED]->(tweet)
MERGE (tweet)-[:RETWEET_OF]->(originalTweet);
"""
graph.run(query_retweet)

print("Les données ont été importées dans Neo4j avec succès.")
```

Init_neo4j.py

Dans ce script, la connexion à la base de données Neo4j est établie via des identifiants sécurisés. Ensuite, deux séries d'importations de données sont réalisées :

- Utilisateurs et leurs relations de suivi (Follow) : Les données sont importées depuis un fichier CSV accessible en ligne. Chaque ligne représente une relation de suivi entre deux utilisateurs. Les entités User sont créées dans la base de données Neo4j, avec des relations FOLLOWS indiquant qui suit qui.
- Tweets, utilisateurs, et relations de retweet : De manière similaire, les informations concernant les tweets et les interactions entre utilisateurs sont importées. Les entités Tweet sont associées aux utilisateurs qui les ont postés, et les relations RETWEET_OF connectent les tweets originaux à leurs retweets.

Ce processus d'initialisation garantit que la base de données est bien structurée et prête à être utilisée pour l'analyse. Chaque étape est conçue pour refléter fidèlement les dynamiques réelles des utilisateurs et de leurs interactions sur la plateforme sociale, permettant ainsi des analyses précises des réseaux sociaux et des flux de contenu.

- ## Initialisation de la base de données MongoDB

```python
import requests
import csv
from pymongo import MongoClient

def init_mongodb():
    # Connexion à MongoDB
    client = MongoClient("mongodb+srv://frangiessono1:ohADwBLcW5IIgQnl@cluster442.cadx49i.mongodb.net/")
    db = client.dbnowsql  # Nom de votre base de données

    # URLs des fichiers CSV sur GitHub
    files_urls = [
        "https://raw.githubusercontent.com/Eetff4422/project_nosql/main/mongodb/docs/tw_user.csv",
        "https://raw.githubusercontent.com/Eetff4422/project_nosql/main/mongodb/docs/tweet.csv",
        "https://raw.githubusercontent.com/Eetff4422/project_nosql/main/mongodb/docs/tweet_hashtag.csv"
    ]

    # Noms des collections à créer pour chaque fichier CSV
    collections_names = ["tw_user", "tweet", "tweet_hashtag"]

    def download_and_import_csv(url, collection_name):
        response = requests.get(url)
        decoded_content = response.content.decode('utf-8')
        if collection_name == 'tw_user':
            cr = csv.reader(decoded_content.splitlines(), delimiter='\t')
        else:
            cr = csv.reader(decoded_content.splitlines(), delimiter=',')
        header = next(cr)
        data = [dict(zip(header, row)) for row in cr]

        collection = db[collection_name]
        collection.insert_many(data)
        print(f"Imported {len(data)} records into {collection_name}.")

    for url, name in zip(files_urls, collections_names):
        download_and_import_csv(url, name)
```

Init_mongodb.py

Ce script automatise le processus d'importation de données depuis des fichiers CSV situés sur GitHub vers une base de données MongoDB. Pour chaque fichier CSV défini par son URL, le script crée une nouvelle collection dans MongoDB et y importe les données, en ajustant le délimiteur en fonction du format de chaque fichier CSV. Une fois les données importées, le script affiche le nombre d'enregistrements ajoutés pour chaque collection.

- ## MongoDB

- Comptage des Entités : La base de données MongoDB est interrogée pour compter le nombre d'utilisateurs, de tweets et de hashtags présents. Cette étape sert de fondation pour analyser l'activité et l'engagement sur la plateforme.
- Analyse des Hashtags : Des requêtes d'agrégation sont utilisées pour identifier les 10 hashtags les plus populaires, offrant un aperçu des sujets les plus discutés ou des campagnes en cours.
- Détermination de la Plus Longue Discussion : Une série de requêtes filtre les tweets qui initient des discussions et calcule la longueur de ces discussions pour trouver la plus longue, révélant ainsi les fils de conversation les plus engagés.

- ## Neo4j

- Analyse des Relations d'Utilisateurs : Des requêtes Cypher sont exécutées pour explorer les relations entre les utilisateurs, telles que le nombre de followers et de followings, et pour identifier les réseaux sociaux d'influenceurs clés.
- Identification des Discussions et des Hashtags : Les données de Neo4j sont intégrées avec MongoDB pour enrichir l'analyse des discussions et des hashtags, permettant une vue d'ensemble des dynamiques de conversation et des thèmes populaires.

- Main

- main_check_database()

Cette fonction vérifie si les bases de données MongoDB et Neo4j existent déjà. Elle retourne deux booléens indiquant l'existence de la base de données MongoDB et de la base de données Neo4j, respectivement. Cela permet d'éviter de réinitialiser ou de recréer des bases de données qui existent déjà.

- init_mongodb()

Initialise la base de données MongoDB. Si la base de données n'existe pas ou est vide, cette fonction crée la base de données et remplit les collections en important les données depuis des fichiers CSV ou toute autre source de données spécifiée.

- init_neo4j()

Similaire à init_mongodb(), mais pour Neo4j. Elle initialise la base de données Neo4j, en créant les nœuds, les relations, et en peuplant la base avec les données initiales nécessaires pour les requêtes et analyses futures

- main_update_document_fields()

Cette fonction est appelée pour nettoyer et mettre à jour les types de champs dans les documents de MongoDB. Cela inclut la conversion des types de données erronés (comme transformer des chaînes de caractères en nombres où c'est nécessaire) et le nettoyage des valeurs nulles ou incorrectes.

- connect()

Les méthodes connect() pour les gestionnaires MongoDB et Neo4j établissent des connexions avec leurs bases de données respectives. Ces connexions sont utilisées pour exécuter des requêtes et des opérations de base de données.

- execute_queries()

Ces méthodes exécutent un ensemble de requêtes prédéfinies sur les bases de données MongoDB et Neo4j. Elles récupèrent des données spécifiques nécessaires pour l'analyse, la visualisation ou d'autres traitements.

- Visualizer(mongo_data, neo4j_data)

Cet objet est responsable de la visualisation des données. Il prend en entrée les résultats des requêtes exécutées sur MongoDB et Neo4j et utilise diverses techniques de visualisation pour représenter ces données graphiquement.

- visualize_mongodb_data(), visualize_neo4j_data()

Ces méthodes de l'objet Visualizer génèrent des graphiques et des visualisations à partir des données récupérées des bases de données MongoDB et Neo4j, respectivement. Elles permettent de mieux comprendre les données à travers des représentations visuelles.

- generate_report(mongo_data, neo4j_data)

Cette fonction crée un rapport basé sur les données récupérées de MongoDB et Neo4j. Le rapport peut inclure des analyses, des résumés, et des insights extraits des données, fournissant une vue d'ensemble utile et des conclusions basées sur les données.

- close()

Les méthodes close() pour les gestionnaires MongoDB et Neo4j ferment proprement les connexions aux bases de données. Ceci est une bonne pratique pour libérer des ressources et éviter des problèmes de connexions ouvertes inutilement.

5. Difficultés rencontrées et solutions adoptées

La manipulation des grandes quantités de données a présenté plusieurs défis, notamment en termes de performance et de cohérence des données. Des problèmes tels que la redondance des données et les requêtes longues ont nécessité l'optimisation des indices et l'amélioration des scripts d'interrogation pour une récupération plus efficace des données.

Une autre difficulté a été l'intégration des données entre Neo4j et MongoDB, nécessitant une synchronisation précise pour maintenir l'intégrité et la pertinence des analyses. Des solutions telles que l'emploi de références croisées et de conventions de nommage cohérentes ont permis de lier les données des deux systèmes de manière logique et efficace.

En outre, la complexité des requêtes Cypher et la nécessité d'une compréhension approfondie de la structure de graphe ont été surmontées par une phase d'apprentissage intensif et la consultation de la documentation et des forums spécialisés. Cette approche a permis d'affiner les requêtes pour extraire des informations significatives tout en maintenant des temps de réponse acceptables.

6. Conclusion

Ce projet illustre l'efficacité de l'utilisation combinée de MongoDB et Neo4j pour explorer des ensembles de données complexes, reflétant les interactions et les comportements des utilisateurs sur une plateforme sociale spécifique. L'analyse a mis en évidence des compétences essentielles dans la manipulation, l'interrogation et la visualisation de données non structurées, compétences de plus en plus recherchées dans l'ère des big data. La capacité à déchiffrer des réseaux d'interaction et à identifier des tendances de contenu offre des perspectives précieuses pour des applications stratégiques et marketing.

Le projet souligne également l'importance d'une approche agile et de la capacité à apprendre continuellement face aux technologies de données émergentes. Les insights tirés de cette étude démontrent la valeur de la science des données dans la compréhension des dynamiques numériques et soulignent le besoin actuel de professionnels qualifiés dans ce domaine. En fin de compte, cette expérience enrichit notre compréhension dans le domaine du numérique et affine nos compétences en préparation aux défis futurs dans le monde des données.