جامعة عبد المالك السعدي
Université Abdelmalek Essaâdi

IASD
INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNEES

كلية العلوم و التقنيات بطنجة
Faculté des Sciences et Techniques de Tanger

UNIVERSITÈ ABDELMALEK ESSAADI

Master IA et Science de Données

Projet : Big Data

Sujet :

Prédiction de sentiments en temps réel des flux de données de réseau social Twitter

Réalisé Par :

MAHRI AYMANE
SABBAHI MOHAMED AMINE

Encadre Par :

Prof. EL YUSUFI Yasyn

Année universitaire : 2023/2024

Table of contents:

Table of figures:

1. Introduction :

Ce projet a été réalisé dans le cadre du Mini Projet Big Data à l'Université Abdelmalek Essaadi, Faculté des Sciences et Techniques de Tanger, Département Génie Informatique. L'objectif principal était de développer une application Web permettant de prédire en temps réel les sentiments des tweets en utilisant l'API Apache Kafka Streams. La prédiction des sentiments est essentielle pour diverses applications, telles que l'analyse de marché, la détection des tendances, et l'amélioration de l'expérience utilisateur sur les plateformes sociales.

L'importance croissante des réseaux sociaux comme source de données massives offre des opportunités uniques pour l'analyse en temps réel. En exploitant les technologies de streaming et de machine learning, ce projet vise à transformer des données non structurées en informations précieuses et exploitables. En intégrant des outils modernes comme Spark, Flask, et MongoDB, nous avons créé une architecture robuste et évolutive capable de traiter, analyser et visualiser les sentiments des utilisateurs de Twitter en temps réel. Cette approche permet non seulement une meilleure compréhension des comportements des utilisateurs, mais également une réponse plus rapide aux tendances émergentes.

2. Architecture Implémentée :

L'architecture mise en œuvre pour ce projet est illustrée dans le diagramme ci-dessous. Cette architecture repose sur plusieurs composants clés orchestrés via Docker, permettant une analyse des sentiments en temps réel à partir des flux de données Twitter.

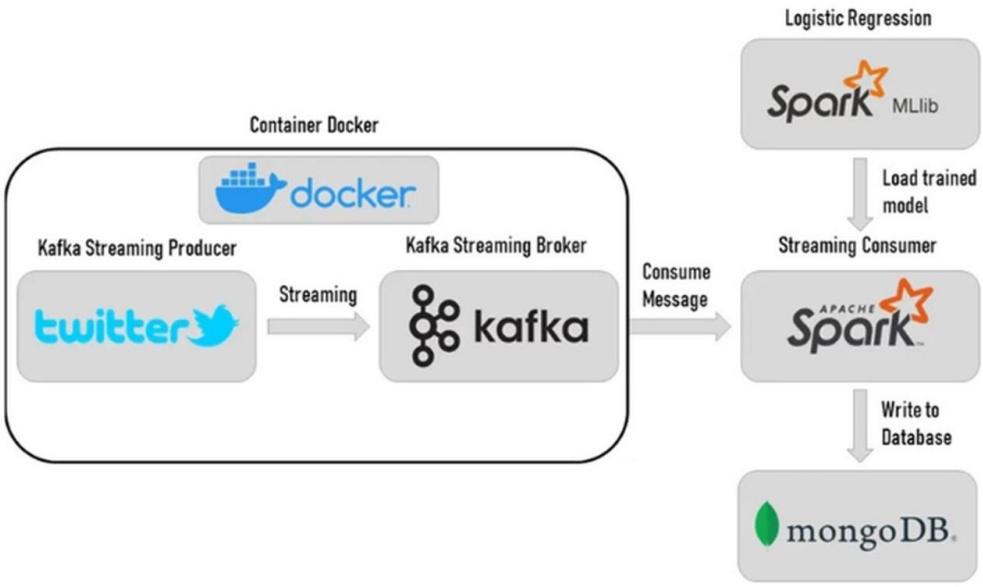

Figure 1: Architecture de projet implementer

3. Description de l'Architecture :

3.1 Producteur de Streaming Kafka :

Les donnees Twitter : Les données de tweets sont récupérées en temps réel à partir d'un fichier csv «twitter_training.csv». Ces données brutes sont envoyées au producteur de streaming Kafka.

Docker : Le producteur et consommateur de streaming aussi Spark sont encapsulé dans un conteneur Docker pour assurer une portabilité et une gestion simplifiée.

3.2 Courtier de Streaming Kafka :

Kafka : Kafka reçoit les flux de données de Twitter et les distribue aux consommateurs inscrits. Il agit comme un système de messagerie qui stocke et distribue les messages (tweets) aux consommateurs en temps réel.

3.3 Consommateur de Streaming :

Apache Spark : Spark est utilisé pour consommer les messages de Kafka. Le modèle de Régression Logistique préalablement entraîné avec Spark MLlib est chargé pour prédire le sentiment des tweets.

Modèle de Régression Logistique : Ce modèle est utilisé pour analyser et prédire les sentiments des tweets en temps réel.

3.4 Base de Données :

MongoDB : Les résultats de la prédiction des sentiments sont écrits dans une base de données MongoDB. Cette base de données est utilisée pour stocker les résultats des analyses et permettre leur visualisation via l'application Web.

4. Jeu de Données Utilisé :

Pour ce projet, nous avons utilisé un ensemble de données provenant de Twitter, spécifiquement conçu pour l'analyse des sentiments. Les caractéristiques du jeu de données sont résumées dans le tableau ci-dessous :

Ensemble de Données	Fichier	Nombre de Lignes	Colonnes
Ensemble d'Apprentissage	**twitter_training.csv**	69,491	Tweet ID, Entity, Sentiment (Labels), Tweet content
Ensemble de Validation	**twitter_validation.csv**	998	Tweet ID, Entity, Tweet content

5. Prétraitement et Nettoyage des Données :

Le prétraitement et le nettoyage des données sont des étapes cruciales pour garantir la qualité des modèles de machine learning. Nous avons utilisé la librairie PySpark pour le prétraitement, en combinant les outils NLTK pour le traitement du langage naturel. Les étapes de nettoyage comprenaient la suppression des doublons, le traitement des valeurs manquantes, et la normalisation des textes pour une meilleure cohérence des données.

Voici les techniques que nous avons utilisées :

2.1 Nettoyage du Texte :

- **Suppression des Caractères Non Alphabétiques :** Nous avons nettoyé le contenu des tweets pour supprimer tous les caractères non alphabétiques, en utilisant la fonction regexp_replace.
- **Conversion en Minuscules :** Tous les caractères ont été convertis en minuscules pour uniformiser le texte.
- **Filtrage des Tweets Vides :** Nous avons éliminé les tweets qui sont devenus vides après le nettoyage.

2.3 Tokenization :

- Le texte nettoyé a été divisé en tokens (mots) à l'aide du Tokenizer de PySpark.

2.4 Suppression des Mots Vides :

- Nous avons supprimé les **stop words** (mots vides) des tokens pour réduire le bruit dans les données et améliorer la qualité des caractéristiques.

2.2 Conversion des Étiquettes de Sentiment en Indices Numériques :

- Les étiquettes de sentiment ont été converties en indices numériques pour faciliter l'entraînement des modèles. Les étiquettes **'Positive'**, **'Negative'**, **'Neutral'** et **'Irrelevant'** ont été respectivement mappées aux valeurs 1, 2, 0 et 3.

2.5 Vectorisation des Caractéristiques :

- **CountVectorizer :** Nous avons utilisé le CountVectorizer pour transformer les mots filtrés en une matrice de comptage de termes, ce qui a permis de convertir le texte en caractéristiques numériques exploitables par les modèles de machine learning.
- **IDF (Inverse Document Frequency) :** Nous avons appliqué l'IDF pour pondérer les termes fréquents moins importants et les termes rares plus significatifs dans le texte.

6. Entraînement des Modèles :

Pour l'entraînement des modèles, nous avons utilisé la librairie SparkML lib pour entraîner quatre algorithmes de machine learning. Les résultats obtenus sont résumés dans le tableau ci-dessous :

Modèle	Précision
Régression Logistique	81.8%
Arbre de Décision	34%
Forêt d'Arbres Décisionnels	41%
Naive Bayes	79%

L'entraînement a été effectué sur un ensemble de données comprenant 69,491 tweets, avec des labels de sentiments (Négatif, Positif, Neutre, Irrélevant). Le modèle de Régression Logistique a montré la meilleure performance avec une précision de 81.8%.

7. Déploiement et Visualisation en Temps Réel (Dashboard) :

L'application Web a été déployée en utilisant Flask pour le backend et Chart.js pour la visualisation des données en temps réel. Cette architecture repose sur Docker et Docker Compose pour orchestrer les différents services, y compris Apache Kafka pour le streaming des données et MongoDB pour la sauvegarde des résultats.

7.1 Déploiement de l'Application Web :

Backend avec Flask :

- **Flask** a été utilisé pour créer le serveur web et gérer les requêtes HTTP. Il sert également de pont entre les données stockées dans MongoDB et la visualisation front-end.

Orchestration avec Docker et Docker Compose :

- **Docker :** tous les composants de l'application, y compris Kafka, Spark et Flask, sont conteneurisés pour garantir la portabilité et une gestion simplifiée, à l'exception de MongoDB qui était sur notre machine locale.
- **Docker Compose :** Utilisé pour orchestrer les différents conteneurs, assurant leur interconnexion et leur démarrage synchronisé.

7.2 Visualisation en Temps Réel :

Chart.js pour la Visualisation :

- **Chart.js** a été utilisé pour visualiser les données de sentiment en temps réel. Cette bibliothèque JavaScript permet de créer des graphiques interactifs et dynamiques directement dans le navigateur.

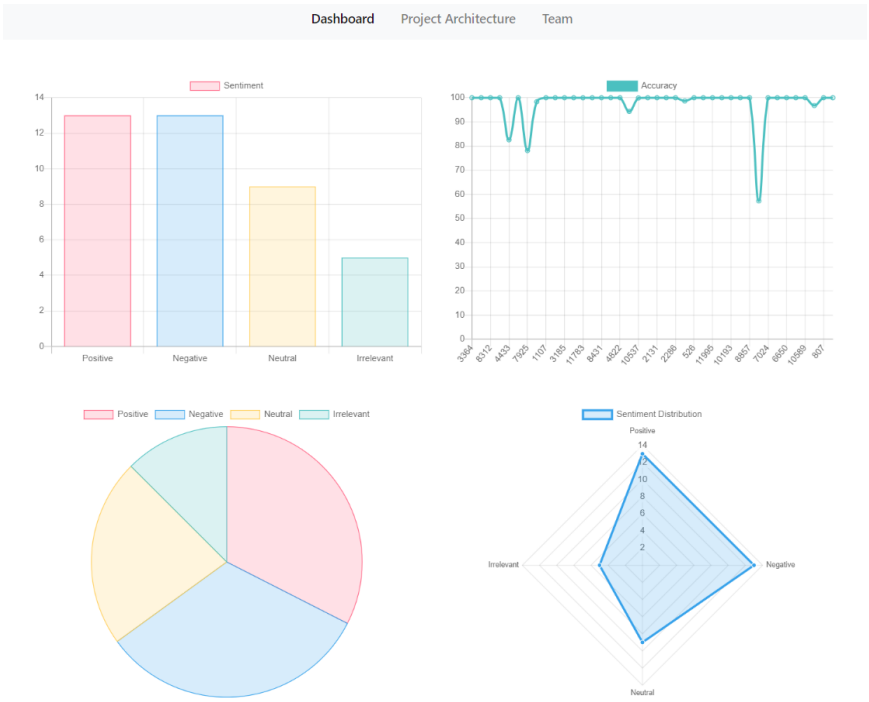

Figure 2: Tableau de bord de visualisation en temps reel.

Mise à Jour Automatique :

- **Le tableau de bord** de l'application Web se met à jour toutes les 3 secondes en récupérant les nouvelles données de la base MongoDB. Cela permet une analyse instantanée des tendances sur Twitter, offrant des informations en temps réel sur les sentiments des utilisateurs.

Interaction avec MongoDB :

Les résultats de la prédiction des sentiments, générés par le modèle de machine learning, sont écrits dans MongoDB. Flask interroge cette base de données régulièrement pour mettre à jour le tableau de bord, garantissant ainsi que les données visualisées sont toujours actuelles.

```
_id: ObjectId('664a55c385090dadf237e6ab')
Tweet_ID : 3364
Entity : "Facebook"
Tweet_Content : "I mentioned on Facebook that I was struggling for motivation to go for…"
Predicted_Sentiment : "Negative"
Accuracy : 100
```

```
_id: ObjectId('664a55cf85090dadf237e6ac')
Tweet_ID : 352
Entity : "Amazon"
Tweet_Content : "BBC News - Amazon boss Jeff Bezos rejects claims company acted like a …"
Predicted_Sentiment : "Positive"
Accuracy : 100
```

```
_id: ObjectId('664a55d785090dadf237e6ad')
Tweet_ID : 8312
Entity : "Microsoft"
Tweet_Content : "@Microsoft Why do I pay for WORD when it functions so poorly on my @Sa…"
Predicted_Sentiment : "Negative"
Accuracy : 100
```

Figure 3: Stockage de donnees sur MongoDB.

8. Outils Utilisés:

Pour réaliser ce projet, nous avons utilisé une combinaison d'outils et de technologies modernes, chacun jouant un rôle crucial dans l'ensemble de l'architecture. Voici les principaux outils et technologies utilisés :

Catégorie	Outils/Technologies	Description
Technologies et Outils	Kafka	Utilisé pour le streaming des données en temps réel.
	Spark	Utilisé pour le traitement et l'analyse des données.
	Docker	Utilisé pour la conteneurisation des services, assurant une portabilité et une gestion simplifiée.
	MongoDB	Utilisé pour la persistance des résultats de l'analyse des sentiments.
Langages de Programmation	Python	Langage principal utilisé pour le développement de l'application.
	JavaScript	Utilisé pour le développement du frontend et la visualisation des données avec Chart.js.
Frameworks et Librairies	Flask	Utilisé pour développer le backend de l'application web.
	Chart.js	Utilisé pour la visualisation des données en temps réel.
Librairies Python	kafka-python	Utilisé pour interagir avec Apache Kafka depuis les scripts Python.
	pymongo	Utilisé pour interagir avec MongoDB.
	pyspark	Utilisé pour le traitement des données et l'entraînement des modèles de machine learning avec Spark.
	pandas	Utilisé pour la manipulation et l'analyse des données.
	flask	Utilisé pour développer le backend de l'application web.
Dépôt de Code	GitHub	Héberge le code source du projet, facilitant la collaboration et le suivi des modifications du code.

9. Conclusion Générale:

Ce projet démontre l'efficacité de l'utilisation d'Apache Kafka pour le streaming de données en temps réel et l'application de modèles de machine learning pour la prédiction des sentiments sur les réseaux sociaux. Le déploiement via Docker a facilité l'orchestration des services nécessaires, et l'utilisation de Flask et Chart.js a permis de créer une interface utilisateur intuitive pour la visualisation des résultats. Les performances variées des modèles soulignent l'importance du choix de l'algorithme en fonction des besoins spécifiques du projet.

En conclusion, ce projet a permis de mettre en pratique des concepts avancés de big data et de machine learning, tout en offrant une solution pratique pour l'analyse en temps réel des sentiments sur les réseaux sociaux. Les compétences acquises et les technologies employées dans ce projet ouvrent la voie à de nombreuses applications potentielles dans divers domaines, tels que l'analyse de marché, la détection des tendances et l'amélioration de l'expérience utilisateur sur les plateformes numériques.