Rapport Projet Big Data

Intitulé : Analyse en temps réel des sentiments des utilisateurs de twitter sur les vaccins anti-covid.

Réalisé par :

- GHOMSI KONGA Serge
- SANOU Désiré
- WAFFA PAGOU Brondon

Introduction

Alors que la campagne de vaccination anti-covid bat son plein, nous remarquons une grande hésitation des populations vis-à-vis des vaccins. Toutefois, le degré d'hésitation est difficilement mesurable. Le projet se veut un outil d'analyse des avis des utilisateurs de twitter sur les vaccins. Pour ce faire, nous avons entrepris de mettre en place une chaîne complète de collecte, de traitement et de visualisation des tweets relatifs au vaccins. Cette chaîne permet de répondre à diverses questions: les sentiments sont-ils majoritairement positifs? quelle région est plus favorable/opposée aux vaccins ? et bien d'autres. Le rapport fait un tour d'horizon des différentes technologies utilisées pour ce projet, les difficultés rencontrées, les résultats obtenus ainsi que les pistes d'amélioration de la plateforme mise en place.

I. Architecture des services

La chaîne suit une architecture assez complexe combinant plusieurs technologies big data comme l'illustre la figure 1 ci-après:

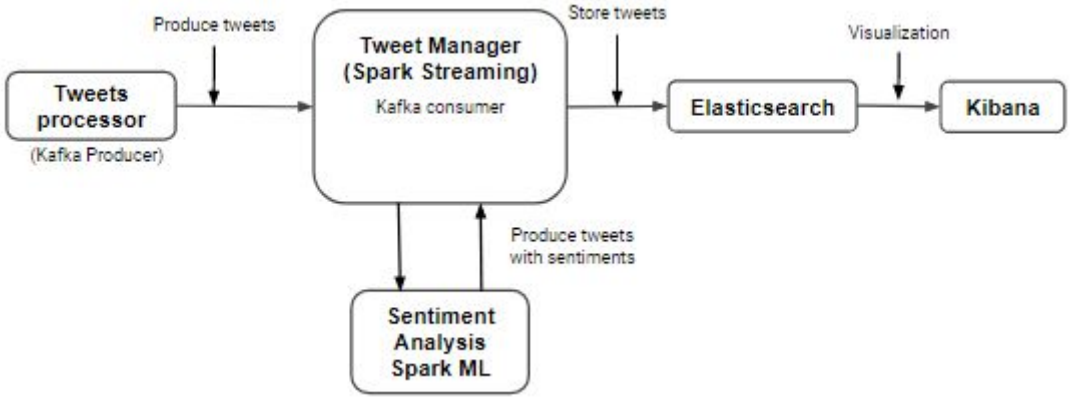

Figure 1: Architecture des services

Dans ce rapport, nous présentons le rôle de chaque service en suivant la chronologie d'intervention des services (gauche vers la droite).

I.1. Tweets Processor: Kafka producer

Ce composant est responsable de la collecte des tweets. Nous utilisons la composante Streaming de l'API *Twitter4J* pour collecter en temps réel, les nouveaux tweets filtrés sur un *query tag*. Il va démarrer un listener qui va écouter l'API et renvoyer un paquets de tweets toutes les 5 secondes.

Avec Kafka, on définit un Topic (une sorte d'identifiant) du Producer mis en place. Le Producer va enregistrer chaque paquet de tweets collectés, préalablement transformés, et l'envoyer ensuite sur une adresse Ainsi il sera possible de récupérer aisément et d'identifier les données venant d'un topic spécifique sur cette adresse et de les consommer. Ce fonctionnement défini par Kafka permet la mise en place de plusieurs Producers, afin de collecter des données de diverses sources.

I.2. Tweet Manager

Les données collectées par le producteur sont traitées par le Consumer(consommateur), qui n'est rien d'autre que *Spark Streaming.* Dans ce tunnel et à l'aide d'un outil (Kafka.Util), on crée un Stream dirigé vers l'adresse où le Producer avait envoyé les données et on les charge selon le Topic défini. Ce procédé est répété toutes les 10 secondes. Les données reçues sous forme de batch de tweets sont traitées afin de déterminer la géolocalisation des auteurs des tweets.

I.3. Sentiment Analysis Spark ML

Pour l'analyse des sentiments, deux solutions ont été explorées concernant le modèle à utiliser pour les prédictions: Utilisation d'un modèle pré-entrainé ou création d'un modèle.

I.3.1. Création d'un modèle

Le but est d'avoir un modèle que l'on peut mettre à jour afin qu'il réponde à la réalité.

Avec StreamingKmeans de Spark Mllib, il est possible de créer un modèle Kmeans dont la mise à jour des centroïdes se fait à la volée, dès que le JavaDstream des données d'entraînement est mis à jour. Cette option est bien l'idéal car elle nous affranchit du codage en dur du nouvel entraînement. En revanche, StreamingKmeans ne pouvait pas être intégré au Pipeline de vectorisation des textes que nous avons mis en place à cause de l'instabilité de la librairie *org.apache.spark.mllib.*

Pour contourner ce problème, le modèle de Kmeans de Spark ML a été créé avec une mise à jour explicite des centroïdes périodiquement. Le modèle obtenu a par contre des résultats mitigés avec une silhouette autour de zéro (sachant que les valeurs sont comprises entre -1 et 1). Les meilleurs modèles ayant des valeurs autour de 1.

Étant donné ces résultats, on ne pouvait pas se baser uniquement sur ce modèle pour avoir des prédictions acceptables; raison pour laquelle nous avons opté pour un modèle pré-entraîné pour la suite de l'étude.

I.3.2. Utilisation d'un modèle pré-entraîné

Afin de mieux apprécier le fonctionnement de la chaîne de suivante et avoir des représentations fiables, l'API de Stanford a été utilisée pour effectuer l'analyse des sentiments sur les tweets. CoreNLP est une librairie Java développée par Stanford permettant le traitement des données textuelles dont l'analyse des sentiments. CoreNLP supporte

actuellement 6 langues : l'anglais, l'arabe, le chinois, le français, l'allemand et l'espagnol. Le processing des données textuelles est basé sur un pipeline qui fait office d'interface.

I.4. ElasticSearch

ElasticSearch est un moteur de recherche et d'analyse RESTful distribué, et conçu pour répondre à une multitude de cas d'utilisation. C'est ainsi, qu'il nous a permis tout d'abord de centraliser le stockage de nos données, plus préciséments tweets et sentiments des utilisateurs, ensuite d'assurer une recherche ultra-rapide, et enfin effectuer des analyses aussi puissantes que scalables.Tout ceci, s'étant effectué à l'aide de la création d'un index "tweetml" de type "_doc", pour le stockage de nos données.

I.5. Kibana

Kibana est une interface utilisateur gratuite et ouverte permettant de visualiser les données Elasticsearch et de naviguer dans la Suite Elastic. Il nous a donc permis de visualiser le nombre total de tweets, "le tweets by language" à l'aide de l'histogramme, le "TweetMap" de nos données, "tweets count by sentiment" à l'aide du camembert .

II. Déploiement de l'architecture

La plateforme a été mise en place sur un PC physique, qui présente les caractéristiques suivantes : Processeur : Inter ® Core(™) i7-6500U CPU @ 2.50GHz 2.59GHz, Ram : 16Go, Système Ubuntu 20.04 LTS 64 bits.
Nous utilisons Eclipse comme IDE et Java version "1.8.0_221".

II.1. Installation

Ci-dessous la liste des outils installés avec les configurations par défaut.

- Apache Spark : Spark-2.2.0-bin-hadoop2.7
- Apache Kafka : kafka_2.11-2.1.0
- nginx 1.18.0
- Elasticsearch 7.10.2
- Kibana 7.10
- Apache Maven 3.6.3

II.2. Développement

Nous avons créé deux Maven projects dans Eclipse : kafka-twitter-producer & kafka-twitter-consumer.

a. kafka-twitter-producer

- Ajout des dépendances maven. Nous citons quelques unes : twitter4j-stream: 4.0.6, kafka-clients: 2.1.0, slf4j-simple: 1.7.25,
- Création d'une classe Sérialisable *TweetBean* qui implémente l'interface à l'objet tweet et d'une classe Geocoder pour la gestion de la géolocalisation.
- Création de la classe *TweetProducer* qui implémente la collecte et l'enregistrement des tweets.

Pour l'exécution, il faudrait préalablement démarrer le serveur Zookeeper de Kafka et ensuite exécuter la classe *TweetProducer.*

b. kafka-twitter-consumer

- Ajout également des dépendances nécessaires: spark-streaming_2.11: 2.2.1, spark-streaming-kafka-0-8_2.11: 2.2.0, elasticsearch-spark-20_2.11 : 7.10.2,...
- Création d'une classe sérialisable *TwitterAnalysisBean,* qui implémente l'interface à l'objet tweet et contient le tunnel d'analyse de sentiment de l'API de Standford.
- Création d'une autre classe sérialisable *TwitterAnalysisML* qui contient notre modèle *KMeans* d'analyse de sentiment avec Spark ML.
- Création d'une classe *TweetsConsumer* qui implémente l'ingestion des données par Spark Streaming et le stockage dans ElasticSearch.

Pour l'exécution, se positionner dans le répertoire du projet et exécuter les commandes suivantes dans le terminal ceci après avoir démarré Kibana et Elasticsearch:

$ *mvn clean compile assembly:single*

$ spark-submit --class valdom.project.kafka_twitter_consumer.TweetsConsumer --master local[4] target/kafka-twitter-consumer-0.0.1-SNAPSHOT-jar-with-dependencies.jar

- Ouvrez Kibana UI (localhost:5601) pour visualiser les données.

III. Résultat final

Pour le Consumer, nous avons choisi de répartir nos jobs sur 4 nœuds. La figure 2 présente un résumé des différentes exécutions. Les batches de données contiennent en moyenne 90 tweets ingérés. Et Dix tweets sont en moyenne enregistrés par seconde. La figure 3 nous présente ces résultats.

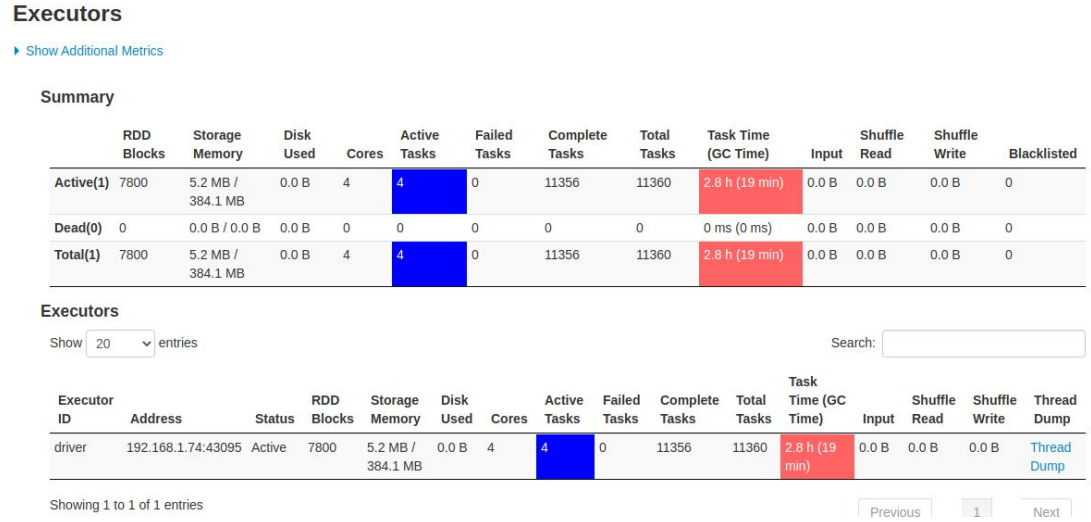

Figure 2: Résumé des exécutions sur les 4 nœuds.

La plateforme que nous avons mise en place permet de répondre aux questions que l'on se pose sur les sentiments de la twittosphère vis-à-vis du vaccin. Ces résultats peuvent être visualisés sur des tableaux de bords contenant des diagrammes circulaires et une carte du monde. Ainsi, on peut aisément observer que la majeure partie des tweets se concentrent dans le monde Occidental (Europe + Amérique de Nord) tandis que l'Afrique et le moyen orient sont les moins expressives sur les vaccins. Cela peut non seulement s'expliquer par la faible présence de ces derniers sur Twitter ou la relative faiblesse de cas dans ces régions du monde. Autre constat, plus de 80% des tweets ont un avis négatif sur les vaccins (figure 4).

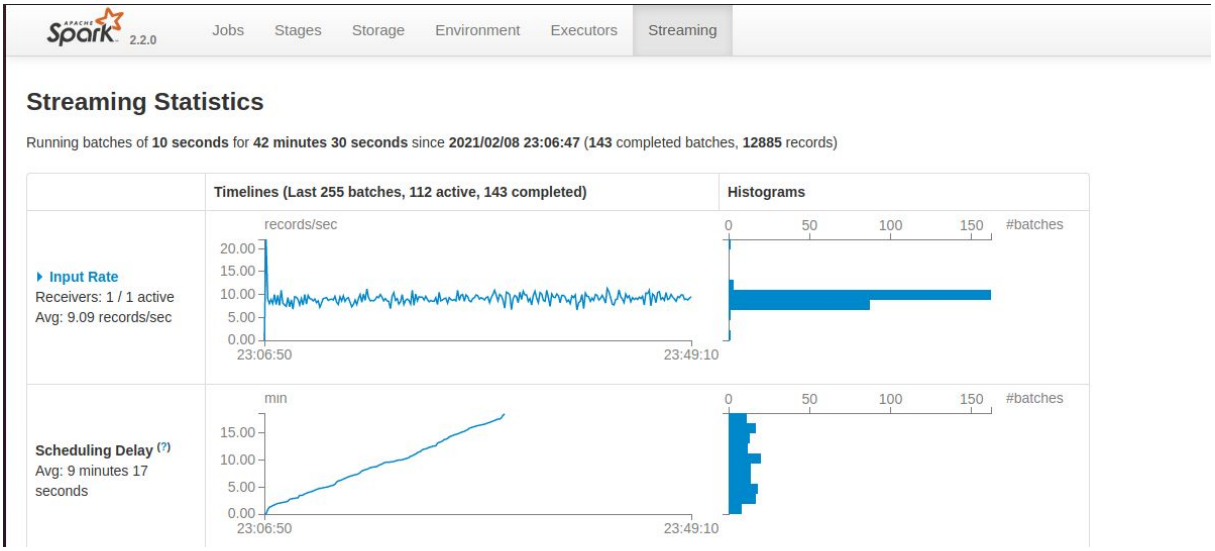

Figure 3: Statistiques du Streaming

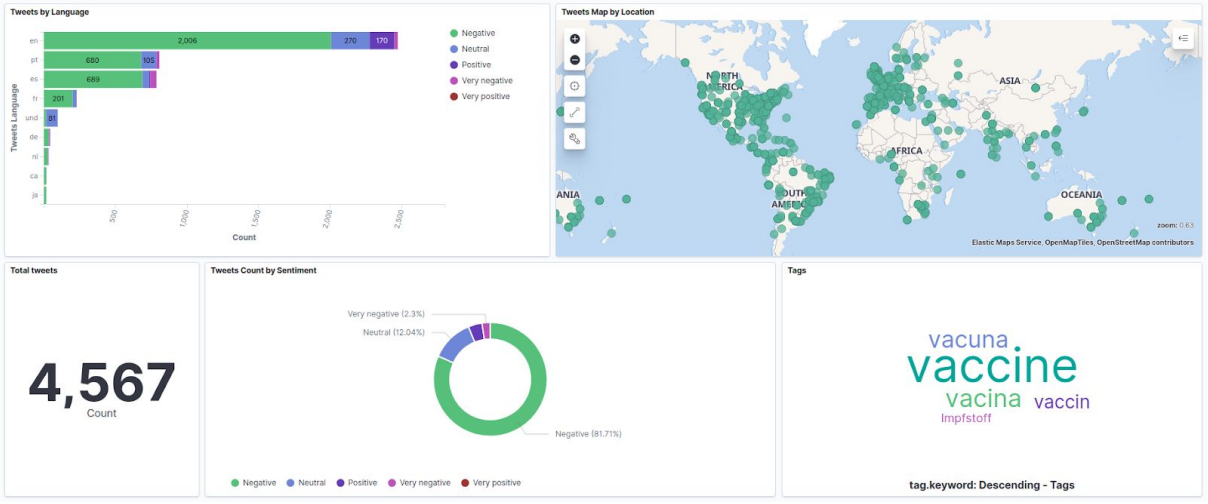

Figure 4: Kibana dashboard

Conclusion

Ce projet, en plus de son aspect académique, a une utilité pratique pour réaliser des sondages indirects sur les sentiments des utilisateurs de Twitter. Par ailleurs, il peut être facilement étendu à d'autres études car il suffit de lui attribuer d'autres sources de données par la création d'autres producers et/ou consumers.

L'étude a, en outre, permis de voir également que les librairies de Machines Learning sont assez limitées en termes de diversité d'algorithmes comparativement aux librairies dédiées comme Tensorflow.

Les résultats obtenus peuvent être améliorés car Twitter n'est certainement pas la seule source de données, ni la plateforme la plus utilisée. De ce fait, se baser uniquement sur les tweets pour donner une conclusion est assez trompeuses. Tout compte fait, une large frange de la population mondiale (au moins 70%) n'est pas très enthousiaste à l'idée de se faire vacciner.