

CLUSTERING ET ANALYSE DE SENTIMENTS DES CRITIQUES DE LIVRES AMAZON

Rapport Mini Projet Big Data

Réalisé par :

MARYEME HASSANI
OUIDAD TARIF
EL BACHIR DRISSI HASSANI

Encadrant :

MOHAMED BAKHOUYA
ABDERRAHMANE
AQACHTOUL

Table des matières

1	Introduction Générale	2
2	Description du problème et importance	3
2.1	Description du problème :	3
2.2	Objectifs du Projet :	3
2.3	Contexte et motivation :	4
3	Détails du Dataset et Étapes de Prétraitement	4
3.1	Détails du Dataset :	4
3.2	Défis liés au Dataset :	5
3.3	Étapes de Prétraitement :	5
3.3.1	Mahout :	5
3.3.1.1	Importation des Données dans HDFS	5
3.3.1.2	la commande seqdirectory :	5
3.3.1.3	la commande seq-sparse :	7
3.3.1.4	Stockage dans HDFS :	8
3.3.2	Python :	8
3.3.2.1	Tokenisation :	8
3.3.2.2	Vectorisation :	8
4	Sélection et Implémentation de l'Algorithme	10
4.1	Choix de l'Algorithme	10
4.2	Implémentation de l'Algorithme :	10
4.2.1	Mahout	10
4.2.1.1	Initialisation des centroïdes	10
4.2.1.2	Exécution de K-Means	11
4.2.1.3	Sortie des Résultats	12
4.2.2	Python	13
4.2.2.1	Chargement des Données	13
4.2.2.2	Exécution de K-Means	13
4.2.2.3	Visualisation des Clusters	13
4.2.2.4	Sortie des Résultats	14
5	Défis Rencontrés et Solutions Apportées	15
6	Conclusion	16
7	Bibliographies	17

1 Introduction Générale

Le Big Data fait référence à l'ensemble des technologies, outils et méthodes utilisés pour traiter et analyser des volumes massifs de données, souvent dans des formats non structurés. Ces données peuvent provenir de diverses sources, telles que des réseaux sociaux, des capteurs, des transactions en ligne, ou encore des critiques de produits, comme c'est le cas dans ce projet. L'analyse de telles quantités de données pose des défis importants, notamment en termes de stockage, de traitement et d'analyse rapide. Pour répondre à ces besoins, plusieurs technologies ont été développées, notamment MapReduce, Mahout, et des systèmes de stockage distribués tels que HDFS.

MapReduce, un modèle de programmation, permet de traiter de grandes quantités de données de manière parallèle et distribuée sur un cluster. Il se compose de deux étapes : Map divise les données en morceaux traités indépendamment, et Reduce regroupe les résultats pour produire un résultat final. Utilisé avec des plateformes comme Hadoop, il permet de traiter efficacement de grandes quantités de données en répartissant les tâches sur plusieurs nœuds, optimisant ainsi les performances.

Apache Mahout est une bibliothèque open-source pour le machine learning sur de grandes données. Elle offre des algorithmes comme K-Means, utilisés pour des tâches telles que le clustering de critiques de livres, et fonctionne bien avec Hadoop pour le traitement distribué des données massives.

Le Hadoop Distributed File System (HDFS) est un système de stockage distribué conçu pour gérer de grandes quantités de données, assurant leur disponibilité grâce à la réplication sur plusieurs nœuds. HDFS est utilisé pour stocker les critiques de livres et faciliter leur traitement parallèle par des algorithmes de machine learning.

D'autres technologies comme Apache Spark, Apache Flink, et Apache Kafka sont également utilisées pour le traitement et l'analyse en temps réel de données massives. Spark permet un traitement plus rapide en mémoire, Flink gère les flux de données en temps réel, et Kafka aide à la gestion des flux massifs de données.

2 Description du problème et importance

2.1 Description du problème :

Dans ce projet, l'objectif principal est d'appliquer des techniques d'analyse de données à un ensemble de critiques de livres pour en extraire des informations significatives. Nous avons choisi d'utiliser un algorithme de clustering, plus précisément le K-Means, pour regrouper ces critiques en clusters homogènes selon des critères sémantiques similaires.

Les données utilisées dans ce projet proviennent de critiques de livres collectées en ligne, plus spécifiquement de Amazon. Le dossier contenant les critiques de livres comporte 1 631 fichiers, chacun représentant une critique individuelle. Ces critiques peuvent être très variées dans leur forme et leur contenu, rendant leur traitement complexe. En effet, la gestion d'un grand nombre de critiques sous forme de texte brut nécessite des techniques avancées de machine learning pour en extraire des tendances et des groupements pertinents.

Avant d'être utilisées, les données ont été prétraitées pour éliminer les informations inutiles telles que le contenu web, la ponctuation et les chiffres. De plus, toutes les lettres majuscules ont été converties en minuscules, les mots ont été réduits à leur racine (stemming), et les mots vides (stop words) ont été supprimés. Ces étapes sont essentielles pour améliorer la qualité du modèle de clustering basé sur l'algorithme K-Means.

2.2 Objectifs du Projet :

- **Clustering de critiques de livres** : Comment regrouper efficacement un grand nombre de critiques (potentiellement des centaines de milliers) de manière à ce que les critiques similaires se retrouvent dans le même groupe ?
- **Gestion de données massives** : Le dataset utilisé est assez volumineux pour que des techniques de calcul haute performance (HPC) et de stockage distribué comme HDFS soient nécessaires pour le traitement à grande échelle.

2.3 Contexte et motivation :

Ce projet revêt une grande importance pour plusieurs raisons :

- **Utilisation de données volumineuses** : Avec la croissance exponentielle des données textuelles disponibles en ligne, notamment à travers les critiques de livres, il devient nécessaire de pouvoir gérer, traiter et analyser efficacement ces informations. Le problème de l'analyse de données massives se pose pour de nombreuses entreprises qui cherchent à tirer parti de l'énorme volume de données non structurées générées sur Internet.

- **Amélioration de l'expérience utilisateur** : L'objectif final de ce projet est de pouvoir classifier et catégoriser les critiques de manière automatique. Cela permettrait à une plateforme de recommandations de livres de mieux orienter ses utilisateurs vers des livres qui correspondent à leurs préférences, en se basant sur des similitudes sémantiques dans les critiques.

- **Exploitation du calcul haute performance (HPC)** : L'analyse des grandes quantités de données nécessite des capacités de calcul importantes. L'utilisation de HDFS permet de stocker et de traiter de grands ensembles de données de manière distribuée, ce qui optimise les performances de l'algorithme. L'utilisation de clusters HDFS permet de travailler de manière plus efficace avec des données réparties sur plusieurs nœuds et d'assurer une évolutivité du système.

- **Application du Machine Learning** : Ce projet démontre l'application de méthodes de machine learning dans un contexte concret, en l'occurrence le clustering de données textuelles. En apprenant à classifier les critiques de livres de manière autonome, l'algorithme peut être utilisé pour des applications pratiques, telles que des systèmes de recommandation automatisés ou des analyses de tendances dans des domaines variés (éducation, commerce, divertissement).

3 Détails du Dataset et Étapes de Prétraitement

3.1 Détails du Dataset :

Le dataset utilisé dans ce projet provient de critiques de livres collectées sur Amazon. Il contient 1 631 fichiers, chaque fichier représentant une critique individuelle. Ces critiques sont au format texte et encodées en UTF-8. Le dataset est varié, reflétant différents styles d'écriture, sentiments, et structures.

3.2 Défis liés au Dataset :

1 - **Nature non structurée** : Les critiques varient considérablement en longueur, vocabulaire et style, rendant leur analyse directe difficile.

2 - **Volume important** : Avec plus de 1 600 fichiers, la gestion et le traitement nécessitent des stratégies de stockage et de calcul efficaces, notamment avec l'utilisation de systèmes distribués comme HDFS.

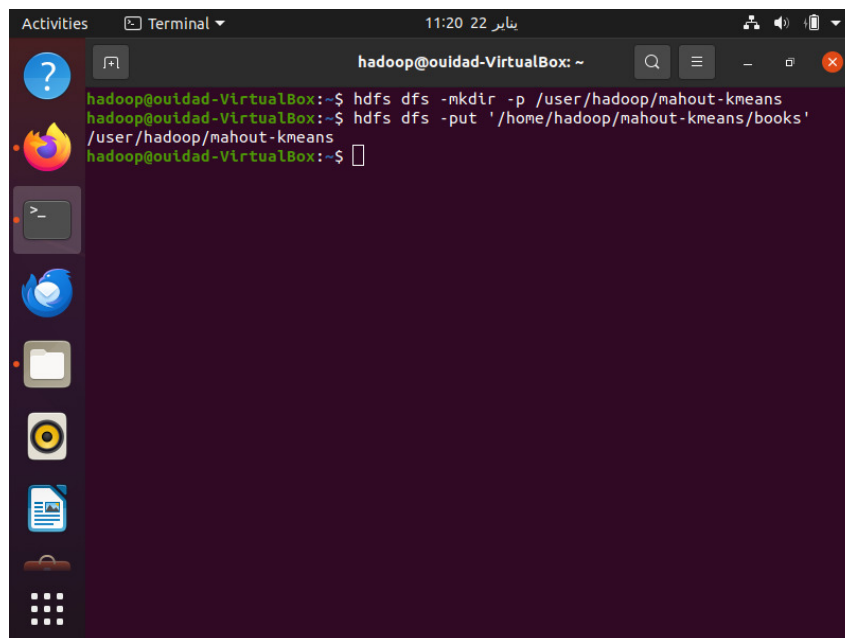
3.3 Étapes de Prétraitement :

Pour préparer le dataset à l'analyse, les étapes de prétraitement suivantes ont été réalisées :

3.3.1 Mahout :

3.3.1.1 Importation des Données dans HDFS

Afin d'exécuter l'algorithme K-Means avec Apache Mahout, il est essentiel de stocker les données dans HDFS (Hadoop Distributed File System) pour permettre un traitement distribué efficace.



```

hadoop@ouidad-VirtualBox: ~
hadoop@ouidad-VirtualBox:~$ hdfs dfs -mkdir -p /user/hadoop/mahout-kmeans
hadoop@ouidad-VirtualBox:~$ hdfs dfs -put '/home/hadoop/mahout-kmeans/books'
/user/hadoop/mahout-kmeans
hadoop@ouidad-VirtualBox:~$
  
```

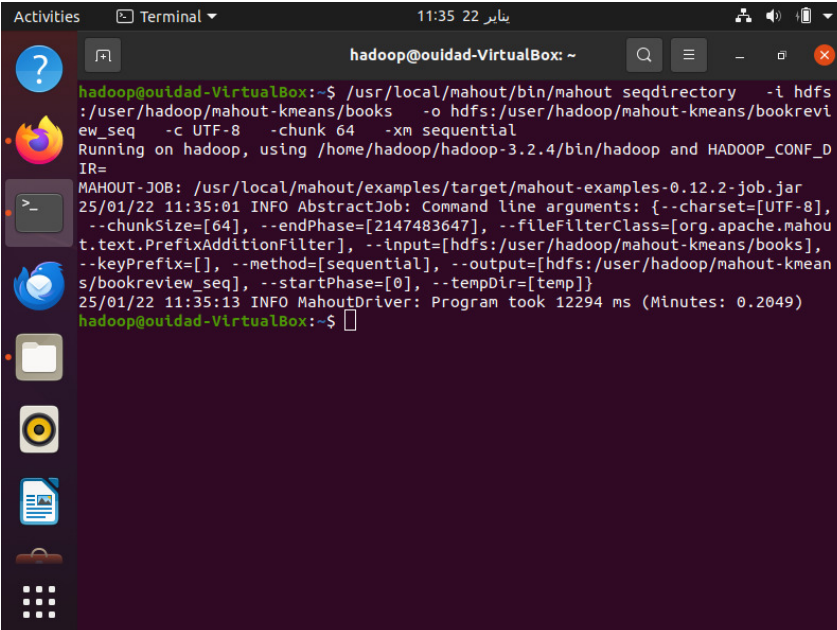
Nous avons commencé par créer un répertoire dans HDFS où les données seront stockées :

3.3.1.2 la commande seqdirectory :

Nettoyage des données : La commande `seqdirectory` joue un rôle essentiel dans la phase de prétraitement. Elle supprime les éléments inutiles tels que le contenu web non pertinent, les URL, et les balises HTML, qui peuvent interférer avec les étapes d'analyse. Ce nettoyage garantit que seules les données textuelles significatives sont conservées, ce qui améliore la précision des résultats ultérieurs.

Tokenisation : Cette commande segmente les données textuelles en mots individuels, appelés tokens, en supprimant les espaces et en identifiant chaque unité lexicale. Cette étape est cruciale pour les tâches de vectorisation et de modélisation, car elle permet d'analyser les textes à un niveau granulaire et facilite l'extraction des caractéristiques.

Conversion au format SequenceFile : Une fois les données nettoyées et tokenisées, `seqdirectory` les convertit en un format binaire appelé `SequenceFile`. Ce format est conçu pour optimiser le stockage et le traitement de grandes quantités de données dans des environnements distribués, comme Hadoop. Il permet également à Mahout de traiter efficacement les textes pour des tâches avancées, telles que la classification ou le clustering.



```

hadoop@ouidad-VirtualBox: ~
hadoop@ouidad-VirtualBox:~$ /usr/local/mahout/bin/mahout seqdirectory -i hdfs
:/user/hadoop/mahout-kmeans/books -o hdfs://user/hadoop/mahout-kmeans/bookrevi
ew_seq -c UTF-8 -chunk 64 -xm sequential
Running on hadoop, using /home/hadoop/hadoop-3.2.4/bin/hadoop and HADOOP_CONF_D
IR=
MAHOUT-JOB: /usr/local/mahout/examples/target/mahout-examples-0.12.2-job.jar
25/01/22 11:35:01 INFO AbstractJob: Command line arguments: [--charset=UTF-8],
--chunkSize=[64], --endPhase=[2147483647], --fileFilterClass=[org.apache.mahou
t.text.PrefixAdditionFilter], --input=[hdfs://user/hadoop/mahout-kmeans/books],
--keyPrefix=[], --method=[sequential], --output=[hdfs://user/hadoop/mahout-kmean
s/bookreview_seq], --startPhase=[0], --tempDir=[temp]]
25/01/22 11:35:13 INFO MahoutDriver: Program took 12294 ms (Minutes: 0.2049)
hadoop@ouidad-VirtualBox:~$

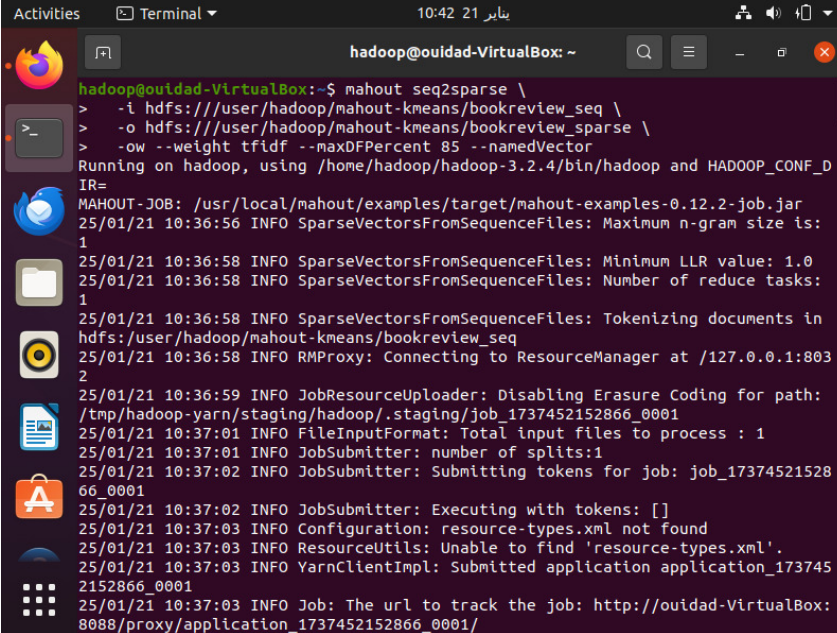
```

3.3.1.3 la commande seq-sparse :

Normalisation du texte : La commande seq2sparse commence par convertir tout le texte en minuscules, ce qui garantit une uniformité dans le traitement des données. Elle applique ensuite un stemming, qui réduit les mots à leur racine commune (par exemple, "courir" devient "cour"), afin de regrouper les variantes grammaticales d'un même mot et d'améliorer la pertinence de l'analyse.

Extraction des caractéristiques : Cette étape consiste à créer une matrice de fréquence des termes pondérée à l'aide de la méthode TF-IDF (Term Frequency-Inverse Document Frequency). Ce mécanisme attribue un poids élevé aux mots significatifs qui apparaissent fréquemment dans un document mais rarement dans les autres, permettant de capturer leur importance relative dans l'ensemble des données.

Vectorisation : Une fois les caractéristiques extraites, seq2sparse transforme les données textuelles en vecteurs numériques. Ces vecteurs, qui représentent les documents sous forme mathématique, sont directement exploitables par les algorithmes de clustering ou de classification de Mahout. Cette transformation rend les données prêtes pour des analyses avancées, telles que le regroupement ou la segmentation.



```

hadoop@ouidad-VirtualBox:~$ mahout seq2sparse \
> -i hdfs:///user/hadoop/mahout-kmeans/bookreview_seq \
> -o hdfs:///user/hadoop/mahout-kmeans/bookreview_sparse \
> -ow --weight tfidf --maxDFPercent 85 --namedVector
Running on hadoop, using /home/hadoop/hadoop-3.2.4/bin/hadoop and HADOOP_CONF_D
IR=
MAHOUT-JOB: /usr/local/mahout/examples/target/mahout-examples-0.12.2-job.jar
25/01/21 10:36:56 INFO SparseVectorsFromSequenceFiles: Maximum n-gram size is:
1
25/01/21 10:36:58 INFO SparseVectorsFromSequenceFiles: Minimum LLR value: 1.0
25/01/21 10:36:58 INFO SparseVectorsFromSequenceFiles: Number of reduce tasks:
1
25/01/21 10:36:58 INFO SparseVectorsFromSequenceFiles: Tokenizing documents in
hdfs:///user/hadoop/mahout-kmeans/bookreview_seq
25/01/21 10:36:58 INFO RMProxy: Connecting to ResourceManager at /127.0.0.1:803
2
25/01/21 10:36:59 INFO JobResourceUploader: Disabling Erasure Coding for path:
/tmp/hadoop-yarn/staging/hadoop/.staging/job_1737452152866_0001
25/01/21 10:37:01 INFO FileInputFormat: Total input files to process : 1
25/01/21 10:37:01 INFO JobSubmitter: number of splits:1
25/01/21 10:37:02 INFO JobSubmitter: Submitting tokens for job: job_17374521528
66_0001
25/01/21 10:37:02 INFO JobSubmitter: Executing with tokens: []
25/01/21 10:37:03 INFO Configuration: resource-types.xml not found
25/01/21 10:37:03 INFO ResourceUtils: Unable to find 'resource-types.xml'.
25/01/21 10:37:03 INFO YarnClientImpl: Submitted application application_173745
2152866_0001
25/01/21 10:37:03 INFO Job: The url to track the job: http://ouidad-VirtualBox:
8088/proxy/application_1737452152866_0001/

```



```

hadoop@ouidad-VirtualBox: ~
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=330386
File Output Format Counters
  Bytes Written=330386
25/01/21 10:42:01 INFO HadoopUtil: Deleting hdfs://user/hadoop/mahout-kmeans/boo
kreview_sparse/partial-vectors-0
25/01/21 10:42:01 INFO MahoutDriver: Program took 304829 ms (Minutes: 5.0804833
333333335)
hadoop@ouidad-VirtualBox:~$ hdfs dfs -ls /user/hadoop/mahout-kmeans/bookreview_
sparse
Found 7 items
drwxr-xr-x - hadoop supergroup          0 2025-01-21 10:39 /user/hadoop/mahou
t-kmeans/bookreview_sparse/df-count
-rw-r--r-- 1 hadoop supergroup      43153 2025-01-21 10:38 /user/hadoop/mahou
t-kmeans/bookreview_sparse/dictionary.file-0
-rw-r--r-- 1 hadoop supergroup      46353 2025-01-21 10:39 /user/hadoop/mahou
t-kmeans/bookreview_sparse/frequency.file-0
drwxr-xr-x - hadoop supergroup          0 2025-01-21 10:40 /user/hadoop/mahou
t-kmeans/bookreview_sparse/tf-vectors
drwxr-xr-x - hadoop supergroup          0 2025-01-21 10:41 /user/hadoop/mahou
t-kmeans/bookreview_sparse/tfidf-vectors
drwxr-xr-x - hadoop supergroup          0 2025-01-21 10:37 /user/hadoop/mahou
t-kmeans/bookreview_sparse/tokenized-documents
drwxr-xr-x - hadoop supergroup          0 2025-01-21 10:38 /user/hadoop/mahou
t-kmeans/bookreview_sparse/wordcount
hadoop@ouidad-VirtualBox:~$

```

3.3.1.4 Stockage dans HDFS :

Après le nettoyage, la normalisation, la tokenisation et la vectorisation des données :

- Le dataset préparé est stocké dans le système de fichiers distribué Hadoop (HDFS).
- Ce stockage permet un calcul distribué lors des tâches de clustering ou d'autres analyses avec Mahout.

3.3.2 Python :

3.3.2.1 Tokenisation :

Avec Python, la tokenisation est effectuée en utilisant des outils comme `nltk.word-tokenize` ou `TfidfVectorizer` de la bibliothèque `scikit-learn`. Le texte d'entrée est segmenté en mots individuels ou tokens, en décomposant chaque phrase en unités lexicales. Cette étape est cruciale pour la vectorisation et permet une analyse des données à un niveau détaillé.

3.3.2.2 Vectorisation :

Une fois les caractéristiques extraites, les données textuelles sont transformées en vecteurs numériques à l'aide de la sortie du `TfidfVectorizer`. Ces vecteurs représentent les documents sous une forme mathématique, les rendant adaptés à des analyses avancées telles que le clustering ou la classification. Les vecteurs résultants sont généralement stockés dans un format de matrice creuse, optimisant ainsi l'utilisation de la mémoire et l'efficacité des calculs.

```
import os
from sklearn.feature_extraction.text import TfidfVectorizer

# Créer un vecteur TF-IDF à partir des textes
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(texts) # Transformer les textes en une matrice de caractéristiques

print(f"Shape of the TF-IDF matrix: {X.shape}") # Afficher la taille de la matrice
```

Shape of the TF-IDF matrix: (1631, 4348)

4 Sélection et Implémentation de l'Algorithme

4.1 Choix de l'Algorithme

Le choix de l'algorithme K-Means pour ce projet a été motivé par plusieurs considérations :

Scalabilité : K-Means est efficace sur le plan computationnel et adapté au traitement de grands ensembles de données, ce qui le rend compatible avec des systèmes distribués comme Hadoop et Mahout.

Simplicité : Cet algorithme est simple à comprendre et à implémenter, se concentrant sur le regroupement de données similaires en fonction de leurs vecteurs de caractéristiques.

Distance Cosinus : La distance cosinus a été sélectionnée comme mesure de similarité pour capturer efficacement la proximité angulaire entre vecteurs, particulièrement utile pour les données textuelles de haute dimension.

Nature non supervisée : K-Means ne nécessite pas de données étiquetées, ce qui le rend idéal pour le clustering des critiques textuelles non structurées, où les catégories ne sont pas définies à l'avance.

4.2 Implémentation de l'Algorithme :

4.2.1 Mahout

Les données vectorisées résultant du prétraitement ont été chargées dans HDFS pour leur utilisation par Mahout. Cette étape de téléchargement garantit que l'infrastructure Hadoop peut exploiter pleinement les capacités du système distribué pour l'analyse des données.

4.2.1.1 Initialisation des centroïdes

Les centroïdes initiaux jouent un rôle crucial dans le clustering K-Means. Deux approches principales ont été utilisées :

Aléatoire : Les centroïdes sont choisis aléatoirement parmi les points de données. Bien que rapide, cette méthode peut parfois conduire à une mauvaise convergence.

k-means++ : Cette méthode améliore la sélection initiale des centroïdes pour maximiser la distance entre eux, accélérant ainsi la convergence et améliorant la qualité du clustering final.

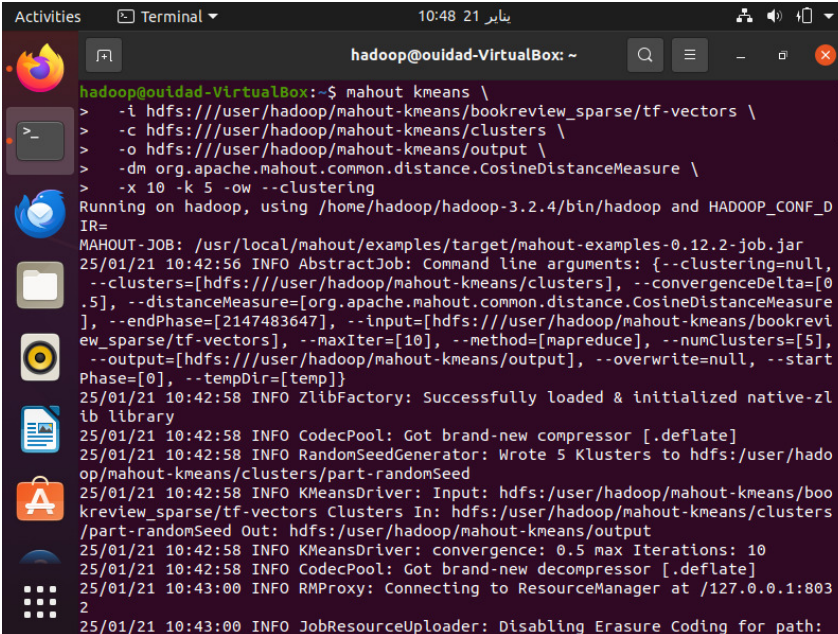
- Apache Mahout prend en charge l'implémentation de k-means++, qui est activée par défaut. Ceux-ci sont stockés dans un répertoire HDFS spécifié

par l'utilisateur.

4.2.1.2 Exécution de K-Means

L'algorithme K-Means a été lancé à l'aide de Mahout, en spécifiant plusieurs paramètres clés ; La commande Mahout comprenait le chemin des vecteurs d'entrée, le nombre de clusters (fixé à 5), les seuils de convergence pour les vecteurs et les centroïdes (0,01), ainsi que le nombre maximal d'itérations (10 000). La distance cosinus a été utilisée comme mesure de similarité pour garantir des regroupements précis.

Grâce à la gestion distribuée de Hadoop, chaque tâche a été répartie sur les nœuds du cluster, exploitant ainsi le parallélisme pour accélérer le calcul tout en maintenant une tolérance aux pannes.



```

hadoop@ouidad-VirtualBox: ~
$ mahout kmeans \
> -i hdfs:///user/hadoop/mahout-kmeans/bookreview_sparse/tf-vectors \
> -c hdfs:///user/hadoop/mahout-kmeans/clusters \
> -o hdfs:///user/hadoop/mahout-kmeans/output \
> -dm org.apache.mahout.common.distance.CosineDistanceMeasure \
> -x 10 -k 5 -ow --clustering
Running on hadoop, using /home/hadoop/hadoop-3.2.4/bin/hadoop and HADOOP_CONF_D
IR=
MAHOUT-JOB: /usr/local/mahout/examples/target/mahout-examples-0.12.2-job.jar
25/01/21 10:42:56 INFO AbstractJob: Command line arguments: {--clustering=null,
--clusters=[hdfs:///user/hadoop/mahout-kmeans/clusters], --convergenceDelta=[0
.5], --distanceMeasure=[org.apache.mahout.common.distance.CosineDistanceMeasure
], --endPhase=[2147483647], --input=[hdfs:///user/hadoop/mahout-kmeans/bookrevi
ew_sparse/tf-vectors], --maxIter=[10], --method=[mapreduce], --numClusters=[5],
--output=[hdfs:///user/hadoop/mahout-kmeans/output], --overwrite=null, --start
Phase=[0], --tempDir=[temp]}
25/01/21 10:42:58 INFO ZlibFactory: Successfully loaded & initialized native-zl
ib library
25/01/21 10:42:58 INFO CodecPool: Got brand-new compressor [.deflate]
25/01/21 10:42:58 INFO RandomSeedGenerator: Wrote 5 Klusters to hdfs://user/hado
op/mahout-kmeans/clusters/part-randomSeed
25/01/21 10:42:58 INFO KMeansDriver: Input: hdfs://user/hadoop/mahout-kmeans/boo
kreview_sparse/tf-vectors Clusters In: hdfs://user/hadoop/mahout-kmeans/clusters
/part-randomSeed Out: hdfs://user/hadoop/mahout-kmeans/output
25/01/21 10:42:58 INFO KMeansDriver: convergence: 0.5 max Iterations: 10
25/01/21 10:42:58 INFO CodecPool: Got brand-new decompressor [.deflate]
25/01/21 10:43:00 INFO RMProxy: Connecting to ResourceManager at /127.0.0.1:803
2
25/01/21 10:43:00 INFO JobResourceUploader: Disabling Erasure Coding for path:

```

4.2.1.3 Sortie des Résultats

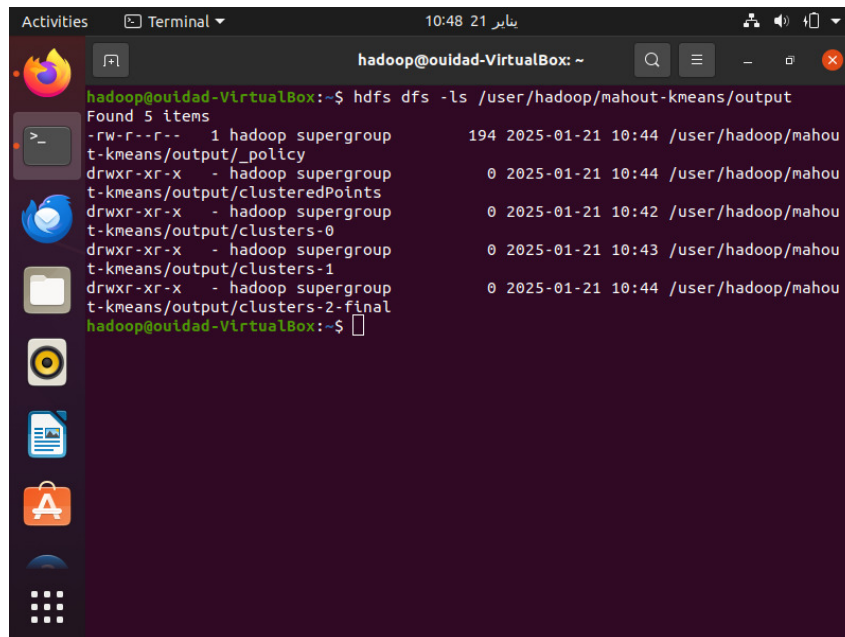
les résultats finaux du clustering ont été enregistrés dans deux répertoires principaux sur HDFS :

Attributions des clusters :

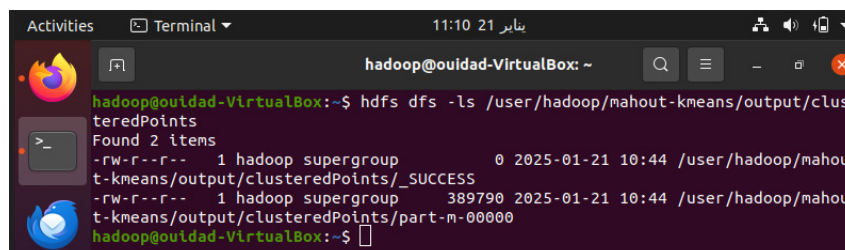
Contient les affectations de chaque point de données (critique) à un cluster spécifique. Ces résultats ont ensuite été extraits pour une analyse locale en utilisant la commande `hadoop fs -get`. Emplacement : `/user/hadoop/kmeans-output`.

Centroïdes finaux :

Stocke les coordonnées des centroïdes finaux pour chaque cluster. Emplacement : `/user/hadoop/kmeans-clusters`.



```
hadoop@ouidad-VirtualBox: ~
hadoop@ouidad-VirtualBox:~$ hdfs dfs -ls /user/hadoop/mahout-kmeans/output
Found 5 items
-rw-r--r-- 1 hadoop supergroup      194 2025-01-21 10:44 /user/hadoop/mahout-kmeans/output/_policy
drwxr-xr-x - hadoop supergroup      0 2025-01-21 10:44 /user/hadoop/mahout-kmeans/output/clusteredPoints
drwxr-xr-x - hadoop supergroup      0 2025-01-21 10:42 /user/hadoop/mahout-kmeans/output/clusters-0
drwxr-xr-x - hadoop supergroup      0 2025-01-21 10:43 /user/hadoop/mahout-kmeans/output/clusters-1
drwxr-xr-x - hadoop supergroup      0 2025-01-21 10:44 /user/hadoop/mahout-kmeans/output/clusters-2-final
hadoop@ouidad-VirtualBox:~$
```



```
hadoop@ouidad-VirtualBox: ~
hadoop@ouidad-VirtualBox:~$ hdfs dfs -ls /user/hadoop/mahout-kmeans/output/clusteredPoints
Found 2 items
-rw-r--r-- 1 hadoop supergroup      0 2025-01-21 10:44 /user/hadoop/mahout-kmeans/output/clusteredPoints/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 389790 2025-01-21 10:44 /user/hadoop/mahout-kmeans/output/clusteredPoints/part-m-00000
hadoop@ouidad-VirtualBox:~$
```

4.2.2 Python

Dans cette section, l'algorithme K-Means est implémenté en Python à l'aide de la bibliothèque scikit-learn :

4.2.2.1 Chargement des Données

Les données vectorisées issues de l'étape de prétraitement sont chargées dans Python sous forme de matrice, prête pour l'analyse. Ces vecteurs représentent les documents dans un espace numérique prêt pour l'analyse par clustering.

```
from google.colab import drive
drive.mount('/content/drive')

# Assurez-vous de spécifier le bon chemin vers vos fichiers texte
text_files_path = '/content/drive/MyDrive/books/'

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

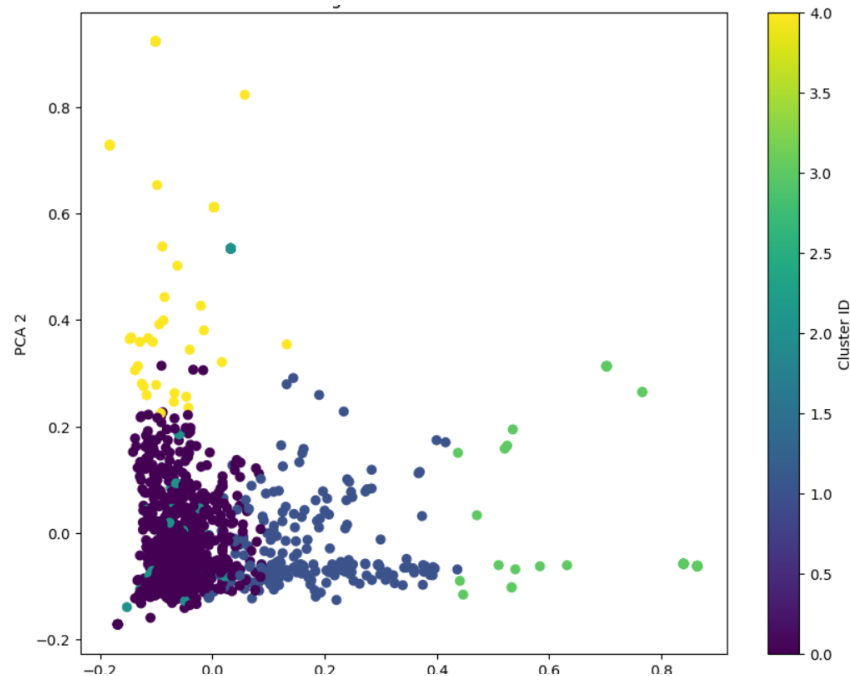
4.2.2.2 Exécution de K-Means

L'algorithme K-Means est appliqué en définissant un nombre de clusters. Chaque document est analysé et assigné à un cluster en fonction de sa proximité avec les centroïdes calculés. Cette étape permet non seulement d'exécuter le regroupement, mais aussi d'identifier directement à quel cluster appartient chaque document, établissant ainsi des groupes thématiques ou similaires.

```
File: 1573.txt is in cluster 0
File: 1563.txt is in cluster 0
File: 1574.txt is in cluster 0
File: 1576.txt is in cluster 0
File: 158.txt is in cluster 2
File: 1577.txt is in cluster 0
File: 1575.txt is in cluster 0
File: 1562.txt is in cluster 0
File: 1569.txt is in cluster 0
File: 1571.txt is in cluster 0
File: 1579.txt is in cluster 0
File: 1581.txt is in cluster 0
File: 1564.txt is in cluster 1
```

4.2.2.3 Visualisation des Clusters

Pour mieux comprendre la distribution des clusters, une réduction de dimensionnalité avec PCA (Principal Component Analysis) est réalisée, suivie d'une visualisation des données :



4.2.2.4 Sortie des Résultats

Cette étape permet d'extraire les mots les plus représentatifs de chaque cluster. Par exemple, le Cluster 0 met en avant des termes liés à Trump et deal, suggérant des sujets politiques ou économiques. Le Cluster 1 inclut des mots comme insight et presid, indiquant des discussions politiques. D'autres clusters révèlent des thèmes variés comme l'inspiration, l'amour, ou des conseils pratiques. Ces termes clés offrent une vision claire des sujets principaux de chaque groupe.

```
Cluster 0:
trump book read deal donald busi like man good love

Cluster 1:
great book read make insight trump presid america man busi

Cluster 2:
excel read book insight donald trump understand inspir high recommend

Cluster 3:
great book read love realli trump everyon enjoy condit defin

Cluster 4:
good read book man insight expect make buy trump deal
```

5 Défis Rencontrés et Solutions Apportées

Incompatibilité entre les Versions de Mahout et Hadoop :

L'un des principaux défis a été l'incompatibilité entre les versions de Mahout et Hadoop. Certains outils de Mahout nécessitent des versions spécifiques de Hadoop pour fonctionner correctement. Ce problème a été résolu en vérifiant soigneusement les versions compatibles dans la documentation officielle et en alignant les configurations en conséquence.

Conversion de Fichiers CSV en SequenceFile :

La conversion des fichiers CSV en SequenceFile, format requis par Mahout pour le traitement distribué, s'est révélée complexe. Des erreurs liées à l'encodage ou à des formats incohérents ont été corrigées en appliquant une validation stricte des fichiers d'entrée avant la conversion.

Limitations de la Mesure de Distance Cosinus :

L'utilisation de la distance cosinus pour mesurer la similarité entre les critiques a parfois montré ses limites dans les cas où les critiques étaient très courtes ou contenues des termes rares. Pour atténuer ce problème, un poids TF-IDF a été appliqué lors de la vectorisation pour réduire l'impact des termes non significatifs et accentuer les mots-clés pertinents.

Analyse des Résultats sur un Grand Volume :

La taille importante des résultats générés, notamment les centroïdes et les attributions de clusters, a compliqué leur analyse. Ce problème a été abordé en extrayant un sous-ensemble représentatif des données pour une inspection manuelle.

6 Conclusion

Ce projet a permis de mettre en œuvre un processus de clustering efficace pour analyser et regrouper les critiques textuelles en exploitant la puissance de Hadoop et Mahout. À travers des étapes méthodiques de préparation des données, de vectorisation, et de clustering, nous avons réussi à identifier des groupes cohérents reflétant des similarités entre les critiques. Malgré des défis tels que l'incompatibilité des versions, la conversion complexe des fichiers, et la gestion de données massives, des solutions robustes ont été développées pour garantir un traitement optimal.

Les résultats obtenus offrent des perspectives intéressantes pour une analyse approfondie des thématiques sous-jacentes à chaque cluster, avec des applications potentielles dans des domaines tels que l'analyse de sentiment, la segmentation de marché ou la personnalisation des services.

Ce projet démontre l'importance d'une infrastructure big data bien conçue pour le traitement de grands volumes de données textuelles, tout en soulignant les avantages d'une collaboration efficace entre les outils open source pour résoudre des problèmes complexes.

7 Bibliographies

[1] <https://mahout.apache.org/documentation/users/clustering/k-means-clustering.html>

[2] <https://github.com/mehulkatara/K-meansApacheMahout>

[3] <https://hadooptutorial.weebly.com/k-means-clustering.html>

[4] <https://github.com/netocosta/HadoopMahout>