Rapport de Projet : Analyse de Données Massives

L'Écosystème Hadoop et Spark pour la Détection de "Fake News"

Mohamed Dhia Eddine Thabet
Mohamed Aziz Dridi
École Nationale d'Ingénieurs de Carthage (ENICarthage)

3ème année cycle d'ingénieurs informatique – Classe B
Année Universitaire 2025-2026

26 octobre 2025

Table des matières

1. Introduction

L'avènement du Big Data a transformé notre capacité à traiter l'information. Cependant, l'augmentation du volume s'est accompagnée d'une augmentation de la désinformation ("Fake News"). Ce projet vise à mettre en œuvre une chaîne de traitement de données complète pour analyser et comparer les caractéristiques linguistiques des articles de presse authentiques et des articles de désinformation.

L'objectif principal est de comparer quatre technologies fondamentales de l'écosystème Big Data : **Hadoop MapReduce**, **Apache Pig**, **Apache Hive** et **Apache Spark**.

1.1 Problématique et Jeu de Données

Nous avons utilisé le jeu de données "Fake and Real News" disponible sur Kaggle[1]. Ce corpus contient environ 45 000 articles de presse, séparés en deux fichiers :

- `True.csv` : Plus de 21 000 articles vérifiés.
- `Fake.csv` : Plus de 23 000 articles de désinformation.

Chaque article comprend un titre, un corps de texte, un sujet et une date. Notre analyse se concentre sur le **corps du texte** pour y trouver des signatures lexicales distinctives.

1.2 Objectif d'Analyse

L'objectif central est de réaliser une analyse de fréquence de mots (WordCount) sur l'ensemble du corpus, en séparant les résultats par catégorie ("FAKE" ou "TRUE").

2. Environnement Technique et Défis Rencontrés

La mise en place de l'environnement de développement pour ce projet a constitué un défi significatif, principalement dû à l'utilisation d'une Machine Virtuelle (VM) Cloudera Quickstart basée sur une version obsolète de CentOS (CentOS 6).

2.1 Problématiques de la VM Cloudera

Peu après le début du projet, il est apparu que l'environnement fourni présentait plusieurs limitations majeures :

- **Gestionnaire de Paquets `yum` Inopérant :** CentOS 6 étant en fin de vie ("End-of-Life"), ses dépôts de paquets officiels ont été archivés ou supprimés. Le gestionnaire `yum` était donc incapable de télécharger ou d'installer de nouveaux logiciels, y compris des outils de base comme `unrar` (nécessaire pour décompresser l'archive du dataset) ou `git`.
- **Erreurs de Certificats SSL :** Les tentatives de réparation de `yum` en le redirigeant vers les serveurs d'archives ont échoué en raison de problèmes de validation des certificats SSL. La chaîne de confiance de cette vieille distribution n'était plus compatible avec les certificats modernes.

1. https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

2.2 Solutions et Contournements

Plusieurs stratégies ont été employées pour surmonter ces obstacles :

1. **Réparation des Dépôts `yum` :** Des manipulations manuelles des fichiers de configuration (`.repo`) ont été nécessaires pour pointer vers les URLs des archives (`vault.centos.org`, `archives.fedoraproject.org`).
2. **Désactivation de la Vérification SSL :** Pour contourner les erreurs de certificat, la vérification SSL a dû être désactivée globalement dans la configuration de `yum` (`sslverify=false` dans `yum.conf`).
3. **Forçage du Protocole HTTP :** En dernier recours, les URLs des dépôts ont été modifiées pour utiliser `http` au lieu de `https`, évitant ainsi complètement la négociation SSL.
4. **Utilisation de la Machine Hôte :** Pour certaines tâches (comme la décompression initiale de l'archive `.rar`), il a été plus simple d'utiliser la machine hôte (exécutant un OS moderne) et de transférer les fichiers résultants (`.csv`) vers la VM via `scp` (Secure Copy).

Ces étapes, bien que fastidieuses, ont permis de rendre l'environnement de la VM suffisamment fonctionnel pour installer les prérequis manquants et exécuter les différentes briques logicielles (Hadoop, Pig, Hive, Spark Shell).

2.3 Gestion de Version avec Git

Malgré l'impossibilité d'installer `git` directement sur la VM, le code source du projet (scripts Python, Java, PigLatin, HiveQL) a été développé sur la machine hôte et géré via un dépôt Git. Cela a permis un suivi des versions et une organisation claire du code. Les fichiers étaient ensuite transférés vers la VM pour exécution via `scp` ou des dossiers partagés (dans le cas d'une alternative Docker qui a été envisagée). L'utilisation de Git garantit également la disponibilité publique et la reproductibilité du projet. Le code source est disponible sur GitHub : `https://github.com/diya-thabet/FakeNews-BigData-Detection.git`

3. Préparation des Données (ETL)

L'analyse de données textuelles brutes issues de fichiers CSV présente un défi majeur. Les virgules, guillemets et sauts de ligne à l'intérieur du texte des articles rendent le parsing par des outils distribués (comme MapReduce) complexe et sujet aux erreurs.

Pour résoudre ce problème, une étape de pré-traitement (ETL - Extract, Transform, Load) a été nécessaire :

1. **Extraction :** Les fichiers `Fake.csv` et `True.csv` ont été lus.
2. **Transformation :** Un script Python a été utilisé pour extraire uniquement le label (”TRUE” ou ”FAKE”) et le corps de l'article. Le tout a été nettoyé et formaté en un seul fichier `articles.tsv`. Ce format utilise une tabulation (`\t`) comme séparateur, un caractère beaucoup plus fiable pour le traitement de texte libre.
3. **Chargement :** Le fichier `articles.tsv` (approx. 110MB) a été chargé sur le système de fichiers distribué HDFS, le rendant accessible à MapReduce, Pig, Hive et Spark.

4. Mission 2 : Implémentation MapReduce (Java)

Le paradigme MapReduce est le fondement historique de l'écosystème Hadoop. Il divise un problème en deux phases distinctes, exécutées en parallèle sur un cluster.

- **Phase Map (Mappeur) :** Le Mappeur lit le fichier `articles.tsv` ligne par ligne. Pour chaque ligne, il nettoie le texte (minuscules, suppression de la ponctuation) et le "tokenise" (sépare en mots). Il émet ensuite une paire clé-valeur où la clé est composite (`<Label, Mot>`) et la valeur est `1`.
  Exemple de sortie : <("FAKE", "trump"), 1>
- **Phase Reduce (Réducteur) :** Après une phase de tri et de mélange ("Shuffle"), le Réducteur reçoit toutes les valeurs pour une clé identique. Sa seule tâche est de les agréger (ici, en faire la somme) pour obtenir le compte total.
  Exemple d'entrée : <("FAKE", "trump"), [1, 1, 1, ...]>
  Exemple de sortie : <("FAKE", "trump"), 73933>

Conclusion : MapReduce est extrêmement robuste et scalable, mais s'avère très "verbeux" : il a nécessité trois fichiers Java distincts (Mapper, Reducer, Driver) pour une tâche conceptuellement simple.

5. Mission 3 : Abstraction de Haut Niveau (Pig & Hive)

Pig et Hive ont été créés pour simplifier le développement sur Hadoop en masquant la complexité de MapReduce.

5.1 Apache Pig (Le Dataflow)

Pig utilise un langage de script (PigLatin) qui décrit un *flux de traitement de données* (dataflow). Notre tâche de WordCount, qui nécessitait des centaines de lignes en Java, a été réduite à un script d'une dizaine de lignes :

1. **LOAD :** Charger `articles.tsv` depuis HDFS.
2. **FOREACH...GENERATE :** Appliquer le nettoyage, la **TOKENIZE** (tokenisation) et le **FLATTEN** (aplatissement des mots en lignes).
3. **FILTER :** Supprimer les mots trop courts (stop words).
4. **GROUP BY :** Regrouper par clé composite `(label, mot)`.
5. **FOREACH...GENERATE :** Appliquer la fonction **COUNT** sur chaque groupe.
6. **ORDER BY / DUMP :** Trier et afficher les résultats.

Pig a automatiquement converti ce script en jobs MapReduce optimisés. Il s'est avéré être l'outil le plus intuitif et le plus efficace pour cette tâche spécifique de NLP.

5.2 Apache Hive (L'Entrepôt de Données SQL)

Hive fournit une interface SQL pour interroger des données sur HDFS. Pour notre analyse, nous avons créé une table externe pointant vers notre dossier `hive_input` (contenant `articles.tsv`).

Nous avons ensuite pu réimplémenter le WordCount en pur SQL (HiveQL) grâce à des fonctions avancées :

- `LATERAL VIEW explode(split(text_body, ' '))` : Cette combinaison de fonctions est le cœur du ”Map”. Elle prend le corps du texte, le divise en un tableau de mots (`split`), puis transforme ce tableau en lignes individuelles (`explode`).
- `GROUP BY label, cleaned_word` : C’est la phase ”Reduce”, qui regroupe les lignes identiques pour le comptage.

Hive s’est montré très puissant, prouvant qu’il pouvait gérer des tâches ETL complexes au-delà des simples agrégations de type ”Business Intelligence”.

6. Mission 4 : Implémentation Modernisée (Spark)

Apache Spark est l’évolution moderne de MapReduce. Sa force principale est le traitement *in-memory* (en mémoire vive), le rendant jusqu’à 100 fois plus rapide que MapReduce, qui écrit sur le disque à chaque étape.

L’implémentation a été réalisée en PySpark sur Google Colab, contournant les limitations de la VM pour cette partie. La logique est basée sur les **RDD (Resilient Distributed Datasets)**, des collections de données immuables et distribuées.

La chaîne de traitement était la suivante :

1. **baseRDD :** Créé en chargeant `articles.tsv` depuis le stockage (‘sc.textFile(...)‘).
2. **pairedRDD :** Créé en appliquant une transformation `.flatMap(...)`. Cette seule étape a réalisé le parsing, le nettoyage, la tokenisation et la création des paires `<clé, 1>` (le ”Map”).
3. **countsRDD :** Créé en appliquant `.reduceByKey(...)`. Cette transformation a regroupé et additionné toutes les valeurs pour chaque clé (le ”Reduce”).
4. **Action :** L’action `.take(100)` a finalement déclenché l’ensemble du calcul (évaluation paresseuse) et retourné les 100 premiers résultats.

Spark combine la puissance bas-niveau de MapReduce avec l’élégance et la concision des API fonctionnelles, tout en offrant des performances largement supérieures.

7. Analyse des Résultats

L’exécution du job Spark a fourni une liste des mots les plus fréquents pour chaque catégorie. Cette liste, bien que simple, révèle des signatures linguistiques très claires et convaincantes.

7.1 Tableau Comparatif (Top 15 des Mots)

Voici un extrait du Top 15 des termes uniques les plus fréquents pour chaque catégorie, après filtrage des ”stop words” les plus courants (comme ”about”, ”their”, ”more”, etc.).

7.2 Interprétation des Résultats

Trois observations majeures et convaincantes ressortent de ce tableau :

1. **Le Point Commun (Trump) :** Le mot ”trump” domine les deux catégories. C’est attendu, car l’essentiel des articles (vrais et faux) a été rédigé durant l’élection et la présidence de Donald Trump. Cependant, il est notable que les articles ”FAKE” le mentionnent **plus de 73% plus souvent** que les articles ”TRUE”.

Articles "FAKE" (Désinformation)			**Articles "TRUE"** (Authentiques)		
Rang	Mot	Fréquence	Rang	Mot	Fréquence
1	trump	73 933	1	trump	42 601
2	people	25 963	2	**reuters**	**28 404**
3	president	25 586	3	president	25 548
4	**clinton**	**18 011**	4	state	18 757
5	**obama**	**17 813**	5	government	17 980
6	**donald**	**17 215**	6	states	17 639
7	news	14 126	7	house	16 407
8	**hillary**	**13 565**	8	also	15 952
9	white	12 778	9	united	15 572
10	time	12 728	10	republican	15 292
11	state	12 525	11	people	15 117
12	against	11 029	12	told	14 244
13	**media**	**10 982**	13	could	13 705
14	campaign	10 571	14	last	12 614
15	house	10 556	15	washington	12 143

TABLE 1 – Comparaison des 15 mots les plus fréquents (hors stop words courants).

2. **La Signature "FAKE" (Personnalisation et Émotion) :** La liste "FAKE" est dominée par des noms propres : `clinton`, `obama`, `donald`, `hillary`. Cela suggère que la désinformation est hautement **personnalisée** et se concentre sur des figures politiques spécifiques pour générer de l'engagement. De plus, la présence de `media` et `news` indique que les articles "FAKE" sont souvent auto-référentiels et cherchent à discréditer les sources d'information traditionnelles.
3. **La Signature "TRUE" (Institution et Source) :** La découverte la plus significative est le mot `reuters` à la 2ème place des articles "TRUE". Notre jeu de données "TRUE" provient en grande partie de l'agence de presse Reuters. Les articles citent leur propre source ("(Reuters) - ..."), ce qui devient une signature involontaire mais puissante de l'authenticité. De plus, des mots comme `government`, `state`, `republican`, `house` et `washington` montrent un vocabulaire beaucoup plus **institutionnel** et formel, centré sur les processus et les lieux de pouvoir plutôt que sur les individus.

8. Conclusion Générale

Ce projet a permis de réaliser une chaîne complète d'analyse Big Data, de l'ETL à l'analyse finale, malgré des défis techniques importants liés à l'environnement initial. La comparaison des outils a démontré une claire évolution des paradigmes :

- **MapReduce (Java)** reste le moteur fondamental, mais sa complexité le rend impraticable pour un développement rapide.
- **Pig et Hive** ont prouvé leur valeur en tant qu'outils d'abstraction essentiels. Pig s'est révélé supérieur pour le traitement de texte (NLP/ETL), tandis que Hive offre la puissance familière de SQL pour l'agrégation.
- **Apache Spark** s'impose comme la solution moderne, combinant la vitesse du traitement *in-memory* avec la flexibilité d'une API riche, capable de gérer aussi bien le SQL, le streaming que le dataflow.

L’analyse des résultats, bien que simple, a suffi à révéler une distinction claire entre le vocabulaire de la désinformation (personnalisé, émotionnel) et celui des nouvelles authentiques (institutionnel, factuel et sourcé). La persévérance face aux problèmes d’environnement a également permis de renforcer la compréhension pratique de la configuration et des dépendances de l’écosystème Hadoop.