Master en Sciences et Techniques (MST)

Filière : Systèmes d'information Décisionnels et Imagerie (SIDI)

Module: Big Data technologies

Mini-Projet

Classification des articles utilisant Mahout

Réalisé par :

Harafi Mohamed

El Azzouzi Mohamed

Encadré par :

Mr. Mohamed Bakhoya

AU: 2024/2025

Table des matières

Introduction

La classification de texte est une tâche essentielle en traitement automatique des langues (TAL), visant à attribuer une étiquette ou une catégorie à un texte donné. Cette tâche trouve des applications dans divers domaines tels que :

- La détection de spams.
- L'analyse des sentiments.
- La catégorisation de documents.

Problématique : Comment Mahout sur Hadoop permet-il une classification de textes efficace, rapide et évolutive sur de grands volumes de données ?

1. Présentation de Hadoop

1.1. Contexte

Hadoop est un Framework open-source développé par la Fondation Apache, conçu pour le stockage et le traitement de grandes quantités de données de manière distribuée. Inspiré des travaux de Google sur le système de fichiers distribué (GFS) et le modèle de programmation MapReduce, Hadoop a été créé pour répondre aux besoins croissants de traitement de données massives dans divers secteurs.

1.2. Composants Principaux

L'architecture de Hadoop est composée de plusieurs modules clés qui collaborent pour assurer le stockage et le traitement efficaces des données à grande échelle :

- Hadoop Distributed File System (HDFS) : Système de fichiers distribué conçu pour stocker de très grandes quantités de données sur des clusters de machines standard. HDFS suit une architecture maître/esclave, où un NameNode (nœud maître) gère l'espace de noms du système de fichiers et régule l'accès aux fichiers, tandis que les DataNodes (nœuds esclaves) gèrent le stockage des données sur les nœuds du cluster. Les fichiers sont divisés en blocs, généralement de 128 Mo, et chaque bloc est répliqué sur plusieurs DataNodes pour assurer la tolérance aux pannes.
- MapReduce : Modèle de programmation et moteur de traitement qui permet le traitement parallèle de grandes quantités de données. Une tâche MapReduce est divisée en deux phases principales :

  Phase Map : Les données d'entrée sont divisées en fragments, et une fonction Map est appliquée à chaque fragment pour produire une série de paires clé/valeur intermédiaires.

  Phase Reduce : Les paires clé/valeur intermédiaires sont regroupées par clé, et une fonction Reduce est appliquée pour produire les résultats finaux.

Ce modèle simplifie le développement d'applications distribuées en gérant automatiquement la parallélisation, la distribution des données et la tolérance aux pannes.

- **YARN (Yet Another Resource Negotiator)** : Cadriciel de gestion des ressources de cluster introduit dans Hadoop 2.0. YARN sépare les fonctions de gestion des ressources et de planification des tâches du moteur MapReduce, permettant à Hadoop de prendre en charge diverses applications de traitement de données, y compris des modèles autres que MapReduce.

1.3. Avantages pour l'Apprentissage Automatique

L'architecture de Hadoop offre plusieurs avantages significatifs pour les applications d'apprentissage automatique :

- **Scalabilité :** La capacité à ajouter facilement des nœuds supplémentaires au cluster permet de gérer des volumes de données croissants sans compromettre les performances.
- **Tolérance aux pannes** : La réplication des données sur plusieurs nœuds assure la continuité des opérations en cas de défaillance matérielle ou logicielle.
- **Flexibilité :** Avec YARN, Hadoop peut exécuter divers types de traitements de données, y compris des algorithmes d'apprentissage automatique, en utilisant différents modèles de programmation.
- **Coût-efficacité :** L'utilisation de matériel standard et de logiciels open-source réduit les coûts associés au stockage et au traitement de grandes quantités de données.

2. Introduction à Apache Mahout

2.1. Objectifs et Historique

Apache Mahout est un projet de la Fondation Apache visant à fournir des implémentations évolutives d'algorithmes d'apprentissage automatique. Initialement conçu pour fonctionner avec Hadoop via le modèle MapReduce, Mahout a évolué pour inclure des algorithmes adaptés à des environnements distribués, notamment en s'intégrant avec Apache Spark pour des performances améliorées.

2.2. Algorithmes Supportés

Mahout propose une variété d'algorithmes pour différentes tâches d'apprentissage automatique :

- **Filtrage Collaboratif :** Techniques de recommandation basées sur les préférences des utilisateurs, utilisées notamment dans les systèmes de recommandation de produits ou de contenus.

- **Clustering :** Regroupement de données similaires en clusters, utile pour des applications telles que la segmentation de clients ou l'analyse de textes.
- **Classification** : Assignation de catégories prédéfinies à des données, employée dans des domaines comme la détection de spams ou la reconnaissance d'images.

2.3. Intégration avec Hadoop

Mahout utilise le modèle MapReduce de Hadoop pour exécuter des algorithmes d'apprentissage automatique de manière distribuée, permettant ainsi de traiter de vastes ensembles de données efficacement. Cette intégration facilite le déploiement d'algorithmes sur des clusters Hadoop, tirant parti de la puissance de calcul distribuée pour accélérer le traitement.

2.4. Avantages de Mahout

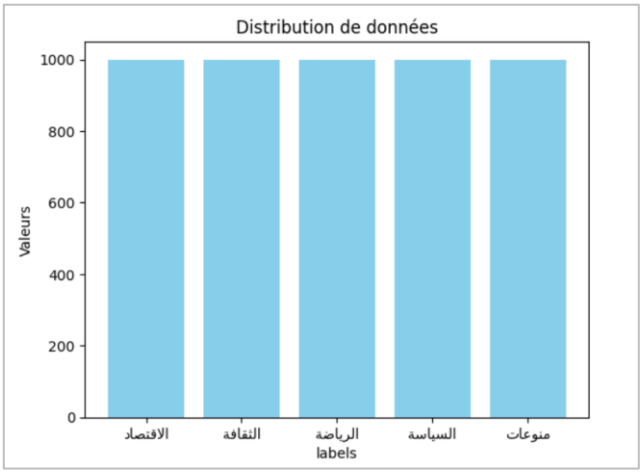
- **Évolutivité** : Conçu pour traiter de grandes quantités de données en parallèle, Mahout est capable de gérer des ensembles de données volumineux en répartissant les tâches sur plusieurs nœuds du cluster.
- **Bibliothèque Riche :** Offre une variété d'algorithmes prêts à l'emploi pour diverses tâches d'apprentissage automatique, réduisant ainsi le temps de développement et facilitant l'expérimentation.
- **Communauté Active :** Soutenu par une communauté dynamique qui contribue à son développement continu, Mahout bénéficie de mises à jour régulières et d'une documentation riche, facilitant son adoption et son utilisation.

3. Exploration et Prétraitement des Données

3.1. Exploration du Dataset:

	text	label
0	جب دسم جمع كثير كوميدي بعض طرب مفاجات ثير خري ...	0
1	سجل ممثل مغرب سعيد غساو حضور ضمن فيلم سينماء ج...	0
2	ستعد مطرب نسيم محمد اطلاق عمل غناء جديد عنوان ...	0
3	اط قيصر غناء عرب نجم عراق اظم ساهر عبر اذاع مي...	0
4	عبدالال وسحاب خبار مغرب اطار احتفال شعب مغرب ع...	0
...	...	...
4995	ضنت قرع دور مجموع اقصاء افريق مءهل نسخ قادم اس ...	4
4996	حيد معد دنيا نبخت صغير واصل دريب حراس مرمي جري...	4
4997	وم ظهر صول جديد مسلسل ضيح مبار ره قدم مغشوش سا...	4
4998	مقتال منضم مولود وجد قال ستفاد تجرب كوديم قال ...	4
4999	رابح حشلاف دخل جواء طول عالم زين ختبر حظوظ الا...	4

5000 rows × 2 columns



3.2. Prétraitement des Données

```
df['text'] = df['text'].apply(remove_stop_words)
df['text'] = df['text'].apply(Arabic_Light_Stemmer)
df['text'] = df['text'].apply(normalizeArabic)
df['text'] = df['text'].apply(Removing_non_arabic)
df['text'] = df['text'].apply(Removing_numbers)
df['text'] = df['text'].apply(Removing_punctuations)
df['text'] = df['text'].apply(remove_extra_Space)
df['text'] = df['text'].apply(remove_hashtages_and_mentions)
df['text'] = df['text'].apply(replace_random_text)
```

3.2.1. Suppression des mots vides (Stop words)

Les **mots vides** sont des termes très fréquents dans une langue qui apportent peu de valeur informative pour les tâches de classification. Par exemple, en arabe, des mots comme "من" (de), "إلى" (à), "في" (dans) sont souvent supprimés.

- **Raison** : Ces mots sont des bruits dans les données et n'aident pas à la distinction entre les classes.
- **Méthode** : L'utilisation des bibliothèques comme nltk pour récupérer la liste des mots vides en arabe et les filtrer du texte.

3.2.2. Stemming léger (Arabic Light Stemmer)

Le **stemming** est le processus qui consiste à réduire un mot à sa racine. Par exemple, "يكتب" (écrire) et "كتابة" (écriture) seront réduits à la racine "كتب".

- **Raison** : Le stemming permet de regrouper différentes formes d'un mot sous une seule forme, ce qui réduit la complexité des données.
- **Méthode** : En arabe, des outils comme **Camel Tools** ou **Farasa** peuvent être utilisés pour effectuer un stemming léger.

3.2.3. Normalisation du texte arabe

La **normalisation** consiste à uniformiser les variations d'un même caractère. Par exemple, en arabe, certaines lettres ont plusieurs formes (ex : "أ" et "ة" ,"ا" et "ه").

- **Raison** : Cette étape permet de réduire la variabilité du texte et d'éviter que le modèle ne considère des variantes comme des mots distincts.
- **Méthode** : La normalisation inclut la transformation des caractères "آ" ,"إ" ,"أ" en "ا" et la transformation de "ة" en "ه".

3.2.4. Suppression des caractères non arabes (Removing_non_arabic)

Les caractères **non arabes** incluent les lettres latines, les chiffres, ou d'autres symboles qui ne sont pas pertinents pour l'analyse du texte arabe.

- **Raison** : Ces caractères peuvent ajouter du bruit aux données et nuire à la performance du modèle.
- **Méthode** : On les supprime en vérifiant si chaque caractère appartient à l'intervalle Unicode réservé à l'arabe (U+0600 à U+06FF).

3.2.5. Suppression des nombres (Removing_numbers)

Les **nombres** peuvent parfois ne pas avoir de valeur informative dans des tâches de classification de texte, sauf si la présence de chiffres est un facteur clé (par exemple, dans des tâches de prévision des prix ou de classification d'annonces).

- **Raison** : Les chiffres ne sont pas toujours pertinents pour le modèle, à moins qu'ils ne soient expressément utiles.
- **Méthode** : Les nombres sont supprimés par un filtrage basé sur les chiffres.

3.2.6. Suppression des ponctuations (Removing_punctuations)

Les **ponctuations** (comme ".", "!", "?", etc.) sont des éléments de structuration du texte mais n'ajoutent pas d'information pour la classification de texte.

- **Raison** : Les ponctuations peuvent gêner le modèle en introduisant de la variabilité dans les données sans apporter d'informations utiles pour la classification.
- **Méthode** : Elles sont supprimées en utilisant des fonctions de traduction ou des expressions régulières.

3.2.7. Suppression des espaces supplémentaires (remove_extra_Space)

Il peut y avoir des **espaces supplémentaires** entre les mots ou au début/fin des phrases, créant des incohérences.

- **Raison** : Ces espaces inutiles peuvent perturber le traitement du texte et créer des erreurs lors de l'entraînement du modèle.
- **Méthode** : Les espaces supplémentaires sont supprimés avec des fonctions simples comme strip() et replace().

3.2.8. Suppression des hashtags et mentions (remove hashtags and mentions)

Les **hashtags** (ex: #hashtag) et **mentions** (ex: @utilisateur) ne sont généralement pas pertinents pour l'analyse de texte sauf si vous souhaitez spécifiquement analyser les réseaux sociaux ou la propagation de certains sujets.

- **Raison** : Ces éléments peuvent perturber la compréhension du texte, surtout si les hashtags ou mentions ne sont pas analysés spécifiquement.
- **Méthode** : Les hashtags et mentions sont supprimés à l'aide d'expressions régulières qui détectent le caractère "#" ou "@".

3.2.9. Remplacement de texte aléatoire (replace_random_text)

Dans certains cas, les **mots ou phrases aléatoires** peuvent apparaître dans le dataset en raison de problèmes de collecte de données, de spam ou d'erreurs de saisie.

- **Raison** : Ces éléments peuvent être nuisibles pour l'apprentissage, introduisant des biais ou des erreurs dans le modèle.
- **Méthode** : Remplacer les textes indésirables par un placeholder ou les supprimer directement.

4. Prétraitement des données en mahout

Avant de procéder à la classification de textes avec Apache Mahout, il est essentiel de suivre une série d'étapes de prétraitement pour transformer les documents bruts en une forme appropriée pour l'analyse.

4.1. Collecte et Organisation des Documents

Organisez les documents dans une hiérarchie de dossiers où chaque sous-dossier représente une catégorie spécifique

4.2. Conversion des Documents en Séquences

Transformer les documents textuels en séquences de mots, en appliquant des opérations telles que la conversion en minuscules, la suppression des caractères spéciaux, et éventuellement le filtrage des mots vides (stop words).

Commande :

```
hadoop@elazzouzi-mohamed-VirtualBox:~$ mahout seqdirectory \
  -i data/ \
  -o sequences/ \
  -c UTF-8 \
  --chunkSize 64
```

-i : Chemin vers le répertoire contenant les documents organisés par catégorie.

-o : Chemin vers le répertoire de sortie où les séquences seront stockées.

-c : Encodage des fichiers d'entrée (par défaut UTF-8).

--chunkSize : Taille des segments de lecture en Mo (par défaut 64).

Résultat:

4.3. Analyse des Séquences et Vectorisation

Convertir les séquences de mots en vecteurs numériques en appliquant des techniques telles que la pondération TF-IDF et la normalisation.

Commande :

```
hadoop@elazzouzi-mohamed-VirtualBox:~$ mahout seq2sparse \
  -i sequences/ \
  -o vectors/ \
  --maxDFPercent 85 \
  --minSupport 2 \
  --weight TFIDF \
  --norm 2 \
  --namedVector \
  --analyzerName org.apache.lucene.analysis.standard.StandardAnalyzer
```

-i : Chemin vers le répertoire des séquences générées précédemment.

-o : Chemin vers le répertoire de sortie pour les vecteurs.

--maxDFPercent : Pourcentage maximal de documents dans lesquels un terme peut apparaître (pour filtrer les termes trop fréquents).

--minSupport : Nombre minimal de documents dans lesquels un terme doit apparaître pour être inclus.

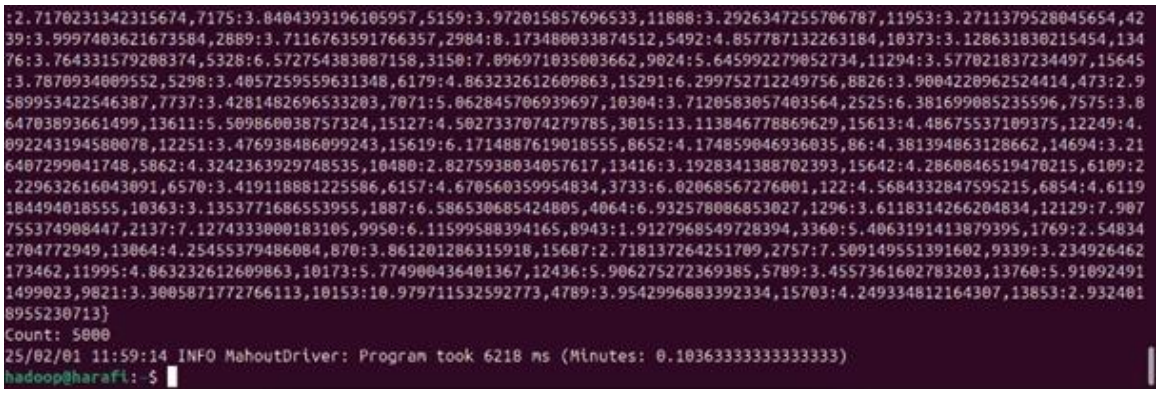
--weight : Schéma de pondération à utiliser (par exemple, TFIDF).

--norm : Type de normalisation à appliquer (par exemple, norme L2).

--namedVector : Inclure les noms des vecteurs pour une meilleure lisibilité.

--analyzerName : Classe de l'analyseur Lucene à utiliser pour le tokenization.

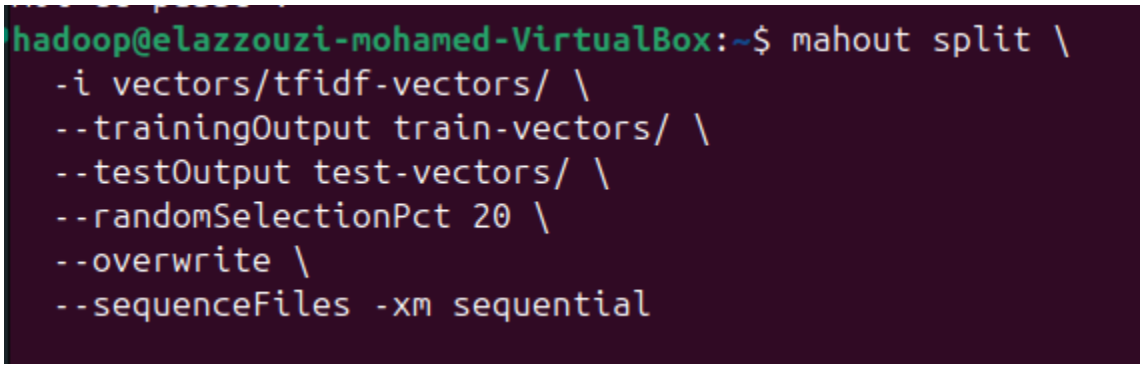
Résultat:



4.4. Division des Données en Ensembles d'Entraînement et de Test

Séparer les vecteurs en ensembles d'entraînement et de test pour évaluer les performances du modèle.

Commande :



-i : Chemin vers le répertoire des vecteurs TF-IDF.

--trainingOutput : Répertoire de sortie pour les vecteurs d'entraînement.

--testOutput : Répertoire de sortie pour les vecteurs de test.

--randomSelectionPct : Pourcentage de données à utiliser pour l'ensemble de test.

--overwrite : Écraser les fichiers existants si nécessaire.

--sequenceFiles : Indiquer que les fichiers sont au format séquence.

-xm : Mode d'exécution (par exemple, séquentiel).

5. Modélisation et entraînement du modèle

Dans le cadre de la modélisation d'un système de classification d'articles avec Apache Mahout, nous avons utilisés un algorithme de classification supervisé : Naïve Bayes. Cet algorithme présente des avantages et des inconvénients spécifiques.

5.1. Naïve Bayes

a. Définition

Le Naïve Bayes est un algorithme probabiliste basé sur le théorème de Bayes, supposant l'indépendance des caractéristiques entre elles. Il est particulièrement adapté à la classification de textes et est reconnu pour sa simplicité et son efficacité, notamment dans les filtres anti-spam.

Avantages

- **Simplicité** : Facile à implémenter et à interpréter.
- **Efficacité** : Nécessite peu de données pour l'entraînement et est performant sur de grands ensembles de données.
- **Rapidité** : Les phases d'apprentissage et de classification sont rapides.

Inconvénients

- **Hypothèse d'indépendance** : Suppose que les caractéristiques sont indépendantes, ce qui n'est pas toujours le cas en pratique.
- **Sensibilité aux données rares** : Peut être influencé par des caractéristiques peu fréquentes, affectant ainsi la précision.

b. Phase d'entraînement

Pour préparer vos données d'entraînement en utilise la commande suivante :

```
$MAHOUT_HOME/bin/mahout split \
  -i ${WORK_DIR}/articles-vectors/tfidf-vectors \
  --trainingOutput ${WORK_DIR}/articles-train-vectors \
  --testOutput ${WORK_DIR}/articles-test-vectors \
```

--randomSelectionPct 40 --overwrite --sequenceFiles -xm sequential

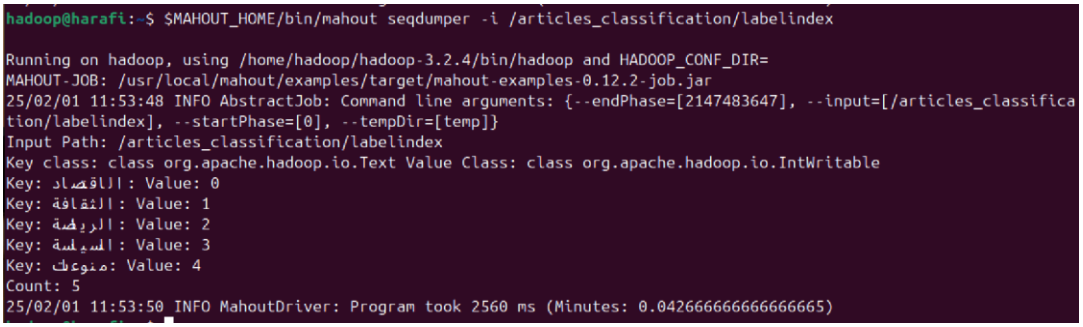
Cette commande est utilisée pour diviser l'ensemble de vecteurs TF-IDF en un ensemble d'entrainement et de test, selon le pourcentage spécifié.

Après on passe à la phase d'entraînement et en utilise la commande suivante

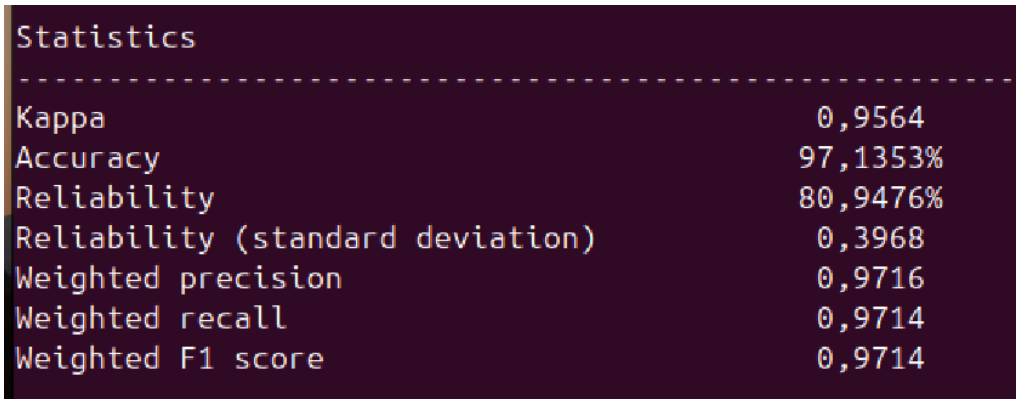
```
$MAHOUT_HOME/bin/mahout  trainnb \
    -i ${WORK_DIR}/articles-train-vectors \
    -o ${WORK_DIR}/model \
    -li ${WORK_DIR}/labelindex
```

Le fichier label index est un *Hadoop Sequence File* qui associe a chaque classe une valeur numérique, Il est utilisé après l'entraînement, notamment lors de la classification de nouvelles données avec le modèle entraînement.

Pour visualiser ce fichier on peut utiliser la commande ***seqdumper***, et le résultat sera de la forme :



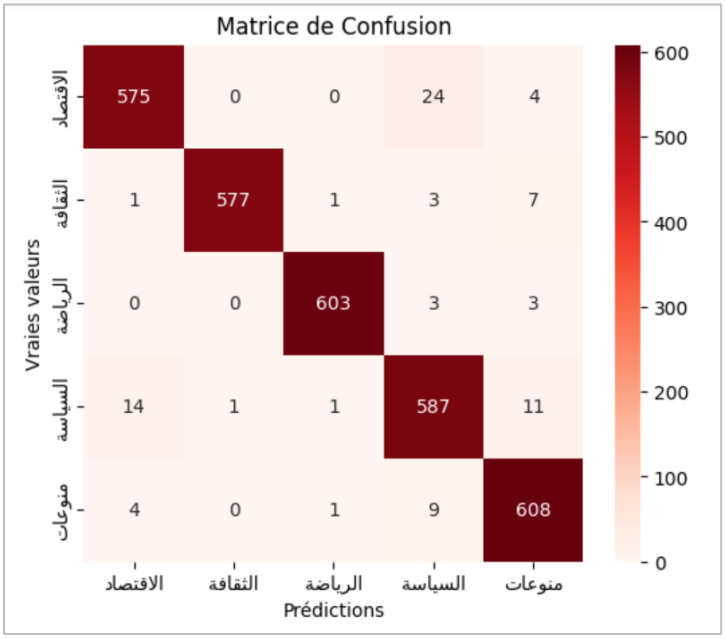
Résultat de classification



Résultats de classification sur les données d'entrainement de Naïve Bayes

```
Summary
-------------------------------------------------------
Correctly Classified Instances          :       2950       97,1353%
Incorrectly Classified Instances        :         87        2,8647%
Total Classified Instances              :       3037

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       <--Classified as
575     0       0       24      4        |  603         a     = الاقتصاد
1       577     1       3       7        |  589         b     = الثقافة
0       0       603     3       3        |  609         c     = الرياضة
14      1       1       587     11       |  614         d     = السياسة
4       0       1       9       608      |  622         e     = منوعات
```



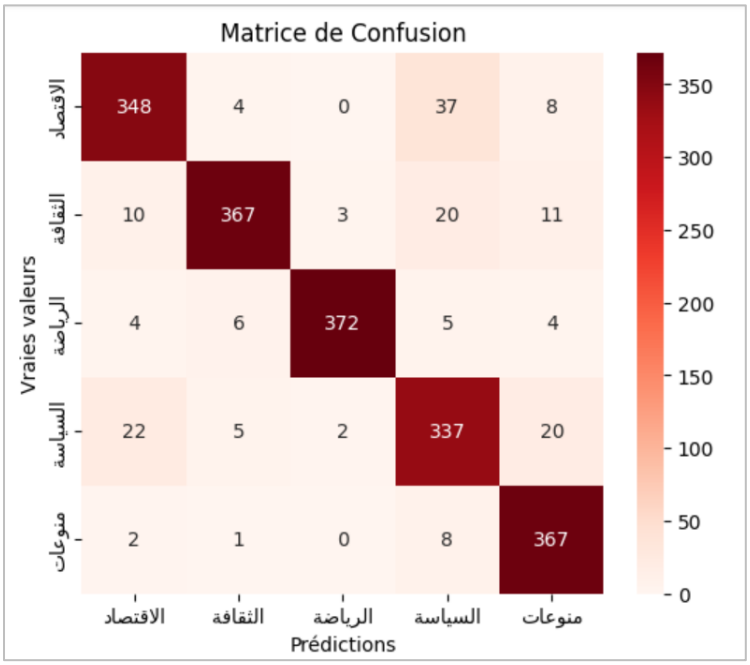
c. Phase de test

Résultats de classification sur les données test de Naïve Bayes

```
Statistics
-------------------------------------------------------
Kappa                                     0,878
Accuracy                                 91,2379%
Reliability                              76,0814%
Reliability (standard deviation)          0,3749
Weighted precision                        0,9147
Weighted recall                           0,9124
Weighted F1 score                         0,9128
```

```
Summary
-------------------------------------------------------
Correctly Classified Instances          :       1791       91,2379%
Incorrectly Classified Instances        :        172        8,7621%
Total Classified Instances              :       1963

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       <--Classified as
348     4       0       37      8        |  397         a     = الاقتصاد
10      367     3       20      11       |  411         b     = الثقافة
4       6       372     5       4        |  391         c     = الرياضة
22      5       2       337     20       |  386         d     = السياسة
2       1       0       8       367      |  378         e     = منوعات
```



Conclusion

L'utilisation de Hadoop pour la classification de textes est une solution puissante pour traiter de grands volumes de données de manière distribuée et évolutive. Bien que cela permette d'améliorer la scalabilité et de réduire les coûts, la mise en œuvre peut être complexe et nécessiter des ajustements pour optimiser les performances et le traitement des données. Pour des projets à grande échelle, Hadoop est un excellent choix, mais il demande une bonne maîtrise technique et une gestion adéquate des ressources.