Projet (Equipe 2) Catégorisation d'articles

KREIS Amaël, TRILLO Baptiste, DJEBBOUR Elias, and BERHIL Ilyes

Université de Versailles Saint-Quentin-en-Yvelines

Abstract. Dans ce projet, nous avons exploré des techniques de classification supervisée et de clustering non supervisée appliquée à un large corpus d'articles de presse provenant de Medium.com. Le dataset, contenant plus de 190 000 articles, a été prétraité à l'aide de techniques de nettoyage de texte, vectorisation TF-IDF et réduction de dimension.
Pour la classification, nous avons testé plusieurs algorithmes, notamment les arbres de décision, les SVM et les KNN, en optimisant leurs hyperparamètres et en comparant leurs performances à l'aide de matrices de confusion et métriques d'évaluation (précision, rappel, F1-score). Nous avons également exploré des techniques d'ensemble learning comme le bagging pour améliorer la robustesse des modèles.
En parallèle, nous avons appliqué des méthodes de clustering telles que K-Means et DBSCAN afin de regrouper les articles par similarité thématique, analysant ainsi la structure sous-jacente du dataset. Une analyse des clusters obtenus a permis d'identifier des relations sémantiques entre les catégories d'articles.
Les résultats montrent que la classification souffre d'une forte confusion entre des catégories sémantiquement proches, ce qui a conduit à la mise en place d'un regroupement des catégories similaires pour améliorer les performances. Ces travaux mettent en lumière les défis et opportunités liés au traitement automatique du langage naturel (NLP) appliqué à la catégorisation de contenus journalistiques.

Keywords: Apprentissage supervisé, Clustering, Traitement du langage naturel (NLP), Classification d'articles, Machine Learning, TF-IDF, LDA, K-Means, SVM, Arbre de décision, Bagging, Medium.com, Analyse de texte

1 Introduction

1.1 Contexte et motivation

Avec la croissance du contenu numérique, l'analyse automatique des textes devient un enjeu majeur dans divers domaines tels que la recommandation d'articles et l'organisation de bases de connaissances. Medium.com, une plateforme de blogging populaire, regroupe des articles couvrant une multitude de sujets allant de la technologie à la littérature. Analyser ces articles permettrait d'améliorer leur catégorisation automatique et de faciliter la navigation pour les utilisateurs.

Dans ce projet, nous avons travaillé sur un dataset de plus de 190 000 articles publiés sur Medium, avec pour objectif de classifier ces textes selon leurs principales thématiques et d'identifier des regroupements naturels via des méthodes de clustering.

1.2 Problématique et défis

La catégorisation d'articles présente plusieurs défis:

- **Similarité entre des catégories :** Plusieurs catégories peuvent avoir des sujets très similaires, par exemple "Blockchain" et "Cryptocurrency" qui sont deux sujets étroitement liés. Ceci peut entraîner des erreurs de classifications.
- **Grande diversité de sujets :** Medium couvre un large éventail de thématiques, rendant la classification plus difficile.
- **Traitement du langage naturel :** le prétraitement est essentiel pour obtenir des représentations numériques exploitables.

Nous nous posons donc la question suivante : **comment classifier efficacement les articles de Medium et identifier des regroupements thématiques pertinents malgré la complexité du dataset ?**

1.3 Méthodologie

Pour répondre à cette problématique, nous avons adopté une approche en plusieurs étapes :

- **Prétraitement des données :** nettoyage du texte, suppression des stopwords, vectorisation en TF-IDF.
- **Réduction de dimension :** utilisation de l'Analyse Linéaire Discriminante (LDA) ou la Principal Direction Analysis (PDA) pour optimiser l'espace de représentation.
- **Classification supervisée :** entraînement de plusieurs modèles (Knn, Arbres de décision, SVM...).
- **Clustering non supervisé :** application de K-Means et DBSCAN pour identifier des groupes de sujets.
- **Analyse des résultats et interprétation :** comparaison entre les approches supervisées et non supervisées.

Ce rapport détaillera chaque étape du projet, en mettant en avant les performances des modèles et les défis rencontrés.

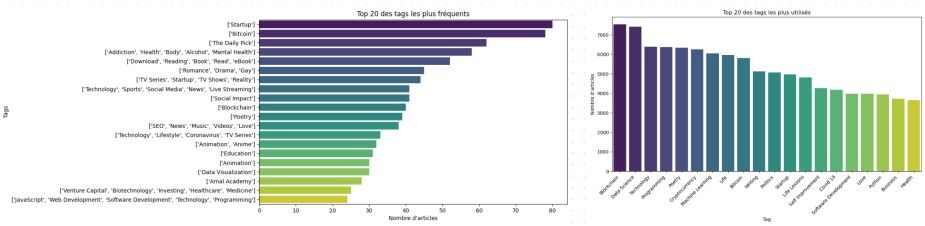
2 Analyse du jeu de données

`35% Ilyes, 35% Elias, 15% Baptiste et 15% Amaël`

Nous avons essayé de visualiser le dataset pour savoir par la suite sur quelle

partie travailler car notre dataset étant très grand, il faudra le réduire. C'est pour cela que nous avons regardé comment était organisé les articles. Il faut vérifier si certains textes n'en ont pas car il faudra donc les supprimer.

Voici maintenant des graphes montrant les tags existants :



On peut voir ici qu'il a beaucoup de tags présents (76k en tout) donc on peut voir qu'il y a beaucoup de textes qui ont plusieurs tags. On voudra donc enlever les textes qui n'ont pas de tags et faire du tri pour garder seulement ceux qui ne possèdent pas les 20 tags les plus présents. Ceci va nous permettre de réduire le nombre de textes pour ne garder que les plus pertinents pour nous à analyser.

3 Prétraitement des Données

`35% Amaël, 35% Baptiste, 15% Elias et 15% Ilyes`

Avant d'appliquer les algorithmes de classification, les textes ont subi plusieurs étapes de prétraitement afin d'assurer une meilleure qualité des données et d'améliorer les performances des modèles.

3.1 Nettoyage et Normalisation

Les articles du dataset ont été prétraités en appliquant les transformations suivantes :

- Suppression des balises HTML et des caractères spéciaux.
- Conversion en minuscules pour éviter la distinction entre les majuscules et minuscules.
- Suppression des mots vides (stopwords) pour éliminer les termes non-informatifs.
- Application du **stemming** ou de la **lemmatization** afin de ramener les mots à leur racine commune et de réduire la dimensionnalité du vocabulaire.

3.2 Vectorisation et Réduction de Dimension

Une fois les textes nettoyés, ils ont été transformés en représentations numériques par **TF-IDF (Term Frequency-Inverse Document Frequency)**, permettant de capturer l'importance des termes dans les articles.

4 Classification des Articles

`75% Amaël et 25% Ilyes`

Afin de réduire la dimensionnalité des données et d'optimiser l'efficacité des classifieurs, une analyse discriminante linéaire (**LDA - Linear Discriminant Analysis**) a été appliquée.

Plusieurs algorithmes de classification supervisée ont été testés sur les données transformées. L'objectif était de classifier les articles en fonction des thématiques identifiées.

4.1 Modèles de Classification

Les modèles suivants ont été appliqués et comparés :

- **K-Nearest Neighbors (KNN)** : testé avec différentes valeurs de k pour trouver le meilleur paramètre.
- **Support Vector Machine (SVM)** : optimisé avec plusieurs noyaux (linéaire, RBF, polynomial).
- **Decision Tree** : permettant d'exploiter les structures arborescentes pour la classification.
- **Random forest** : ensemble d'arbres de décision optimisé via le nombre d'arbres et leur profondeur.

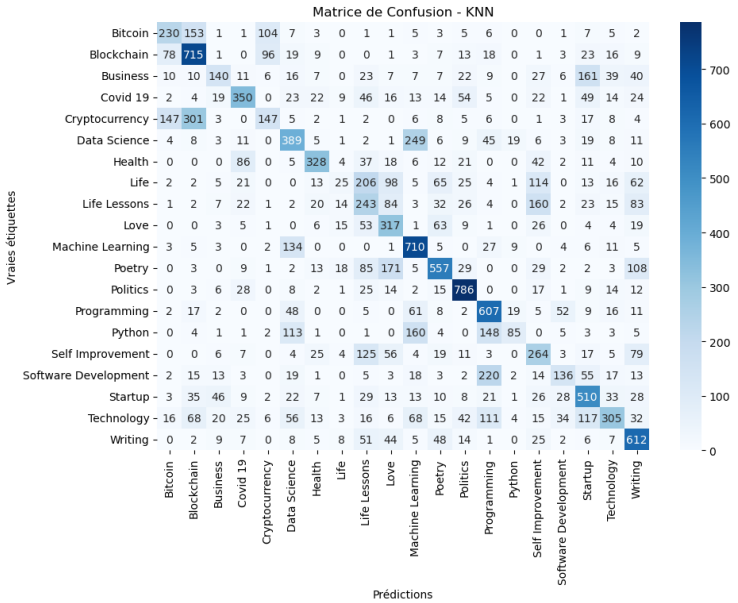


Fig. 1. Matrice de confusion

On obtient une précision de 0.63, ce résultat est plutôt bon sachant que nous avons 20 catégories différentes. L'analyse de la matrice de confusion montre une forte confusion entre certaines catégories proches, comme "Blockchain" et "Cryptocurrency", ou "Artificial Intelligence" et "Machine Learning".

4.2 Amélioration par Regroupement des Catégories

Afin de réduire les erreurs de classification et d'améliorer la précision globale, nous avons décidé de **regrouper certains tags similaires** en catégories plus générales. Par exemple, les catégories *Blockchain*, *Cryptocurrency* et *Bitcoin* ont été fusionnées sous un seul label **"Crypto"**. De même, les catégories *Data Science*, *Machine Learning* et *Artificial Intelligence* ont été réunies sous **"AI & Data"**. Cette approche a permis d'améliorer la distinction entre les classes et de réduire les erreurs de classification observées dans les matrices de confusion initiales.

4.3 Optimisation et Évaluation

Chaque modèle a été optimisé par une recherche d'hyperparamètres via **GridSearchCV**, et les performances ont été évaluées à l'aide des métriques suivantes :

- **Précision (Accuracy)**
- **F1-score, Recall, Precision** pour chaque classe
- **Matrice de confusion** pour visualiser les erreurs de classification

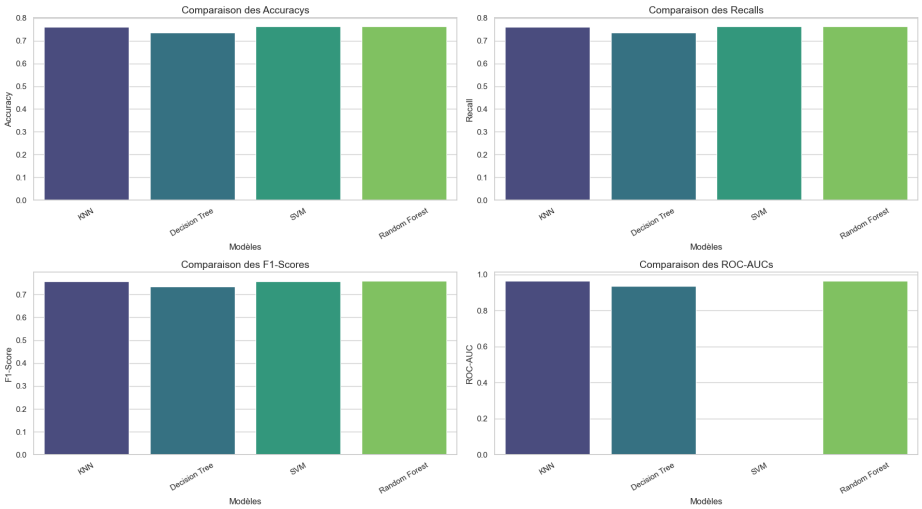


Fig. 2. Résultats de la classification

On peut voir que Knn, SVM et Random forest donnent des résultats assez similaires avec des précisions autour de 0.76 ce qui est 10% de plus par rapport au Knn sans grouper les tags, ce qui est une nette amélioration.

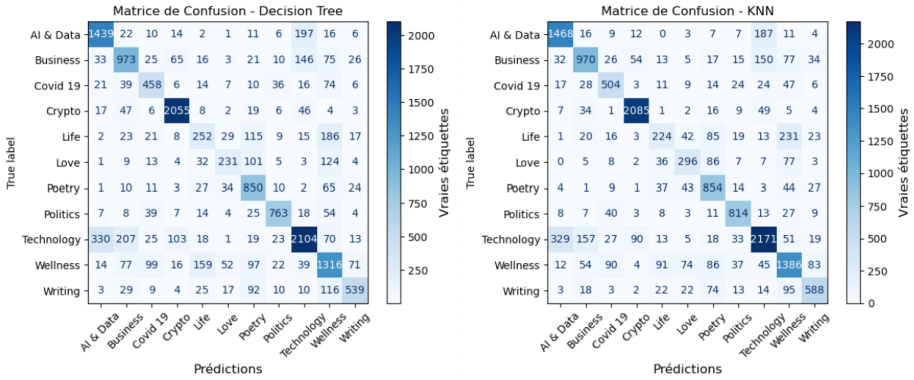


Fig. 3. Matrice de confusion du Decision tree et Knn

Les résultats pour Knn, SVM et Random forest sont très similaire, par contre en comparant knn et decision tree on voit que les catégories ou decision tree a le plus de mal sont "Technology" et "Politics". Peut-être qu'avec plus d'étages, le decision tree serait plus performant sur ces catégories.

Nous n'avons pas eu le temps d'appliquer des méthodes d'ensemble sur ces algorithmes, mais peut-être que du boosting pourrait améliorer les résultats.

4.4 Deep learning

Nous avons testé 2 algorithmes de Deep Learning :

- **Convolutional Neural Networks** : utilisés principalement en vision par ordinateur, les CNN ont été adaptés au traitement du langage en identifiant des motifs locaux dans les séquences de mots.
- **Transformer** : un Transformer est une architecture de réseau de neurones basée sur des mécanismes d'attention, particulièrement efficace pour traiter des données séquentielles comme le texte, permettant d'analyser les relations à longue distance entre les éléments d'une séquence.

Pour ces deux algorithmes, l'entraînement a été fait sur un dataset réduit de 64 571 articles qui possèdent au moins un des 15 tags les plus présents dans le dataset. Cela a permis de réduire les durées d'entraînement qui pouvaient être très longues.

CNN: Les CNN, initialement conçus pour l'analyse d'images, ont été adaptés au traitement du texte en capturant des motifs récurrents dans les séquences de mots. Chaque texte est converti en une suite de nombres, permettant aux filtres de convolution d'extraire des expressions ou structures importantes. Ensuite, une couche de pooling sélectionne les informations essentielles avant qu'une dernière couche ne classe le texte dans une catégorie spécifique.

Nous avons défini les paramètres suivants :

- **Nombre maximum de mots dans le vocabulaire** : 20 000.
- **Longueur maximum d'une séquence** : 200.
- **Nombre de couches de convolution** : 1. (Une deuxième couche n'améliore pas les résultats dans notre cas mais ralentit l'entraînement.)
- **Pourcentage de Dropout** : 60% permet d'éviter le sur-apprentissage en désactivant aléatoirement des neurones à chaque étape d'apprentissage.)
- **Nombre d'étapes d'apprentissage:** 10.
- **Taille du batch:** 64.

Transformer: Spécialisé dans le traitement du texte, le Transformer est aujourd'hui une référence dans ce domaine en raison de sa capacité à capturer le contexte global d'une séquence. Cependant, il nécessite un volume important de données pour l'entraînement et une puissance de calcul conséquente, rendant son utilisation coûteuse en temps et en ressources.

Nous avons défini les paramètres suivants :

- **Longueur maximum des séquences** : 32 (réduite car augmente beaucoup trop le temps d'entraînement.
- **Taille du batch** : 64.
- **Modèle pré-entrainé** : DistilBert (essentiel pour palier au manque de données pour l'entraînement et plus rapide que Bert)

Conclusion: Le CNN et le Transformer ont tous deux atteint une précision d'environ 54 %. Toutefois, l'entraînement du CNN prend en moyenne 20 minutes, contre 563 minutes pour le Transformer, soit un rapport de temps d'entraînement très avantageux en faveur du CNN. Ce résultat montre que, dans notre configuration et avec les ressources disponibles, le CNN offre un excellent compromis entre précision et rapidité. Bien que le Transformer soit conçu pour capturer des relations complexes dans les textes, l'absence d'un grand volume de données d'entraînement et les contraintes de calcul ont limité son avantage. Cependant, bien que le CNN soit plus rapide que le Transformer pour une performance équivalente, les algorithmes classiques étudiés précédemment (KNN, SVM et Decision Tree) restent plus efficaces dans notre cas. En effet, ils offrent de meilleurs résultats, tout en étant moins exigeants en ressources et plus adaptés à notre volume de données. Ainsi, dans notre situation actuelle, les méthodes classiques demeurent les plus adaptées, rendant les modèles de Deep Learning moins pertinents dans ce contexte.

5 Clustering des articles

`75% Baptiste et 25% Elias`

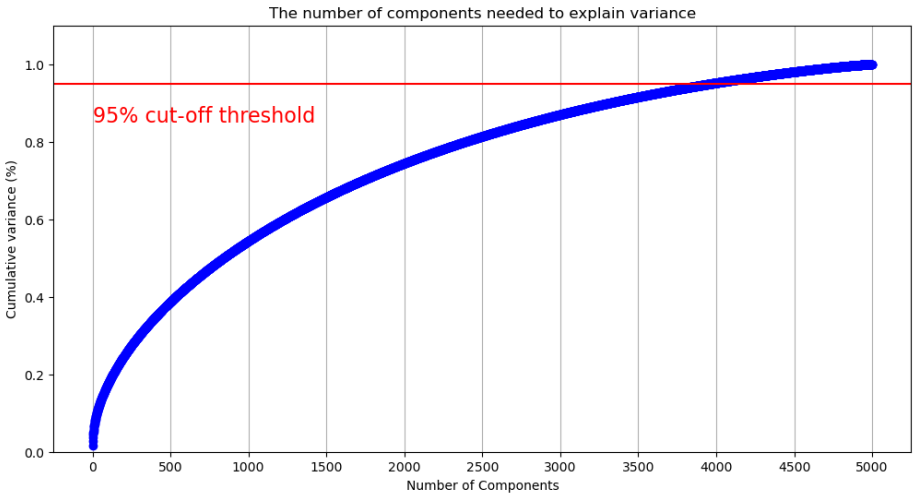
Le but de cette section est d'explorer la méthode de clustering pour la comparer avec celle de classification. On passera par différentes étapes non supervisées comme le PCA et différents algorithmes de clustering pour voir si nous allons pouvoir retrouver les différents groupes de textes.

5.1 Feature Selection

Après avoir appliqué les différentes étapes de texte transformation et de texte preprocessing, on passe à la feature selection qui passera par le PCA pour rester dans la logique du non supervisé.

Pour commencer, il faut trouver le bon nombre de feature à sélectionner pour appliquer TF-IDF car trop de feature prendra trop de temps aux algorithmes de clustering et pas assez de feature faussera les algorithmes. On va donc trouver combien de mots couvrent 95% ou 99% du corpus des textes. On trouve donc que l'on a 25 628 mots et 225 409 mots qui couvrent respectivement 95% et 99% du corpus. Mais malheureusement, l'algorithme TF-IDF ne peut pas être appliqué avec 25 000 mots donc on se limitera à seulement 5 000.

On détermine ensuite le nombre de composants qui sont nécessaires pour l'algorithme de PCA. Une méthode existe utilisant le cumul de la variance expliquée qui doit être supérieur à 95%. Ce graphique montre combien de composants, il faut pour appliquer PCA.



Il faut donc un nombre de composants pour un PCA optimisé de 3 970. On enregistre donc les 3 970 composants de PCA dans un nouveau dataset avec les différents tags liés au texte.

Mais après avoir essayé plusieurs algorithmes de clustering, il y a trop de composants donc on va prendre 19 composants.

5.2 Data Mining

échec de PCA Après avoir essayé d'optimiser les différentes valeurs pour appliquer une feature selection non supervisée, on se retrouve à réduire tout cela pour que nos algorithmes soient applicables sur nos machines. Pour certains algorithmes de clustering, il faut trouver le bon nombre de clusters à déterminer pour que les algorithmes soient efficaces. On va donc appliquer la méthode du coude et la méthode de la silhouette pour trouver le bon nombre de clusters. Après avoir appliqué ces méthodes, on trouve un nombre de clusters qui vaut 2 ou 3 si on prend respectivement tout le corpus ou seulement 1000 textes (ce qui n'est pas beaucoup).

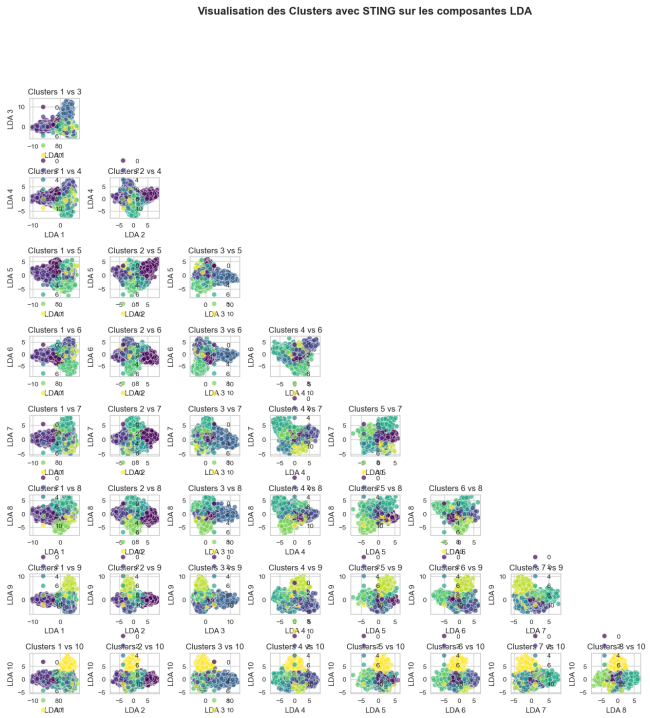
En essayant d'appliquer plusieurs algorithmes de clustering en analysant les différents graphiques, on ne peut pas distinguer 2 ou 3 clusters lorsque l'on essaie de visualiser tous les composants des PCA. On va donc essayer d'utiliser les composants de LDA qui ont été utilisés pour la classification car la classification donne de bons résultats.

Clustering avec LDA En appliquant à nouveau la méthode du coude, nous avons déterminé que le nombre optimal de clusters est 11.

Nous avons ensuite testé plusieurs algorithmes de clustering, en sélectionnant au moins un par catégorie abordée en cours :

- Partitioning clustering approaches : **K-means**
- Hierarchical clustering approaches : **AGNES**, **DIANA** et **BIRCH**
- Density based clustering approaches : **DBSCAN** et **OPTICS**
- Grid based clustering approaches : **STING** et **WaveCluster**

Pour les algorithmes qui avaient besoin du nombre de clusters, ils ont été exécutés avec 11 et pour les autres qui avaient besoin d'autres valeurs, des boucles pour trouver les valeurs optimales ont été faites. Après application des algorithmes, nous avons essayé de visualiser les clusters avec les différents algorithmes mais nous avons le même problème qu'avec PCA. Certains algorithmes qui ont leur nombre de clusters fixés trouvent des clusters que l'on ne peut pas voir en 2D et pour les autres soit il y a des clusters qui n'ont aucun sens ou alors nous n'avons pas le bon nombre de clusters. (notamment pour **OPTICS** et **Mean Shift**)



6 Conclusion

Dans l'objectif de retrouver des tags pour classifier les documents, nous avons testé deux types d'algorithmes : supervisés et non supervisés. Le but était de comparer les deux méthodes pour savoir laquelle était la plus efficace pour la classification. La classification, bien que très efficace grâce à la connaissance préalable des catégories, possède des désavantages comme certaines catégories trop vague peuvent être mélangées avec d'autres et rendre donc la classification fausse. Pour le clustering, le but était de retrouver les groupes de textes ayant des liens et pouvoir retrouver les noms par la suite. La taille du corpus et les résultats incohérents des algorithmes ont rendu le clustering inefficace pour cette tâche. Il était donc difficile de trouver des clusters pour que les textes aillent entre eux. La méthode de classification avec les algorithmes supervisés sont donc la manière optimisée pour faire la catégorisation d'articles même avec les quelques petits désavantages qu'elle propose.