UNIVERSITÉ CÔTE D'AZUR

Université Côte d'Azur

MASTER INFORMATIQUE
Parcours INTELLIGENCE ARTIFICIELLE

Algorithme de recommendation - Projet TATIA

Authors :
BOULLI Marouan & HADDOU Amine

Mentor :
CABRIO Elena

Rapport d'alternance

HADDOU Amine

26 janvier 2024

Table des matières

1 Recommendation d'article

Comme sujet de projet, nous avons fais le choix d'implémenter un alogrithme de recommendation d'article en fonction du dernier article lu par l'utilisateur. Le fonctionnement de cet algorithme se décomposera en deux étapes principales : classifier l'article et fournir un classement d'articles similaires au dernier lu.

Concernant la classification, nous travaillerons avec 5 catégories d'articles différentes. Ce choix a été imposé par le jeu de donnée trouvée mais ces catégories englobent un ensemble d'article riche et divers. Les catégories sont *buisness*, *entertainment*, *politics*, *sport* and *tech*.

Une fois l'article classifiée, l'algorithme proposera un classement des 3 articles les plus similaires dans notre jeu de donnée. On basera notre classement sur un calcul de similiarité de l'article lu comparé aux autres articles de la même catégorie.

On pourrait alors se demander quelle est l'utilité de classifier l'article. Dans un contexte d'une large base de donnée, il est important de prendre en compte le prix et consommation de calcul de similiarité sur un grand ensemble d'articles différents. De plus, on souhaite dans le cadre de notre projet que les articles recommendés appartiennent à la même catégorie que le dernier article lu par l'utilisateur. On implémentera l'algorithme dans cette logique.

Enfin, on devra valider notre algorithme pour en juger l'effficacité. Pour se faire, nous baserons la validation sur deux métriques : l'*accuracy* et le ration de mot clés similaires entre les deux articles.

2 Jeu De Donnée

Comme base de données pour ce projet, nous avons choisi le jeu de donnée d'article de la BBC disponible sur Kaggle.

Le jeu de donnée est décomposée en deux ensembles de données : un ensemble de *training* contenant 1490 articles, et un second de *testing* contenant 735. La différence entre les deux ensembles est dans la classification des articles. Les articles du premier ensemble sont labellisés (classifiés) alors que ceux du second ne le sont pas. Pour cette raison, nous nous limiterons à travailler avec le premier ensemble pour le moment. Nous n'utiliserons le second que lors de la validation pour tester les performances de l'algorithme finale.

Il est ensuite important de vérifier que le jeu de donnée n'est pas biaisé en surreprésentant (resp. sous-représentant) une catégorie. Or, notre jeu de donné est bien équilibre avec des catégories qui représentent de $17.5\% - 23.3\%$ de l'ensemble des données figure 1.

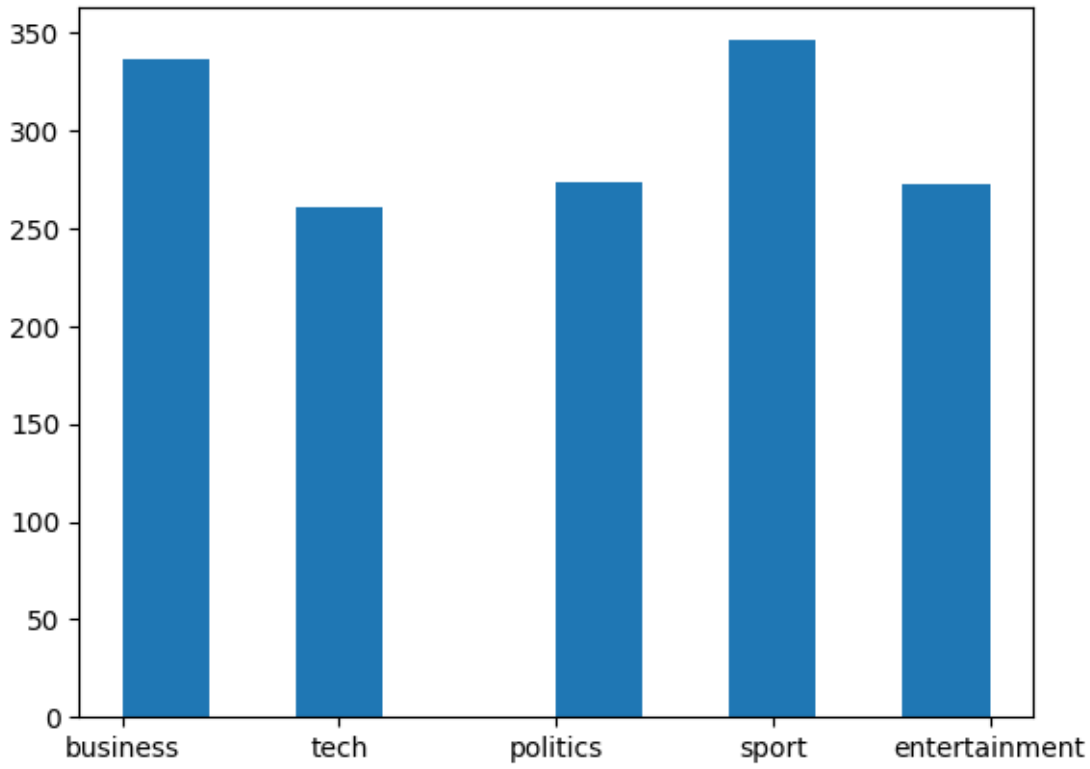

FIGURE 1 – Proportion des catégories d'article

2.1 Preprocessing

Avant d'entamer l'implémentation de l'algorithme, il est essentiel de débuter par le prétraitement de notre jeu de données. Cette phase revêt une importance cruciale, car elle conditionnera la qualité de nos données et, par conséquent, celle de notre algorithme.

Il convient de souligner, dans un premier temps, que notre jeu de données est prêt à être traité. Les modèles de machine learning présentent des performances médiocres lorsqu'ils sont confrontés à des données textuelles. L'objectif principal est de surmonter cette contrainte en apportant des modifications subtiles à notre base de données.

Nous commencerons par éliminer la ponctuation des articles, car elle ne contribue pas à la valeur ajoutée de la classification. Dans cette démarche, nous recherchons le sens global plutôt que le sens par phrase. Toutefois, nous procéderons également à la suppression des mots dépourvus de sens sémantique, tels que les articles et les propositions. Voici un exemple des modifications apportées :

worldcom ex-boss launches defence lawyers defending former worldcom chief bernie ebbers against a battery of fraud charges have called a company whistleblower as their first witness. [..]
worldcom exboss launches defence lawyers defending former worldcom chief bernie ebbers battery fraud charges called company whistleblower first witness cynthia [..]

Enfin, toujours avec l'idée de nous intéresser seulement au sens global de l'article, nous allons réaliser une lemmatisation. Ce procédé nous permettra de mettre en évidence

les mots clés des articles, normaliser le texte et améliorer la précision sémantique des modèles de machine learning.
Nous allons utiliser le lemmatizer *WordNetLemmatizer*, entrainé sur la base de donnée *WordNet*, pour lemmatiser notre jeu de donnée. Appliqué sur l'exemple précédent, nous obtenons le résultat suivant :

worldcom exboss launch defence lawyer defending former worldcom chief bernie ebbers battery fraud charge called company whistleblower first witness cynthia

Enfin, on vectorisera les "nouveaux" articles afin d'obtenir un jeu de donnée vectoriels. Un type de donnée très largement utilisé pour le machine learning. Cette étape sera réalisé dans un pipeline qu'on présente dans la section suivante.

3 Entrainement des modèles

Afin de classifier les articles, nous allons tester des modèles de classification. Mais il nous reste avant celà une dernière étape : couper notre jeu de donnée en deux parties. La première sera utilisée pour l'entrainement et la seconde pour les testes. Il est habituellement recommendé, pour un jeu de donnée de notre envergure, de garder 77% du data set pour l'entrainement et le reste pour le training. C'est ce qu'on appliquera ici.

3.1 Support Vector Machine Classifier

Le premier modèle choisi est le Support Vector Machine Classifier. Ce modèle est généralement utilisé sur des data set de taille moyennes (quelques milliers) mais il se trouve qu'il peut également être très performant sur de plus petits jeu de donnée de qualité.

3.1.1 Pipeline

Pour l'entrainement, on crée deux pipeline permettant d'automatiser une séries de tâches. La pipeline se chargera ici de

- Vectoriser les articles grâce à la classe `CountVectorizer` fourni par *Scikit-learn*
- Tfid transformer : calcule le poids des mots dans chaque phrase en fonction de leur fréquence dans le corupus.
- Entrainement du modèle SVC sur le jeu de donnée modifié.
- Vectoriser les articles grâce à la classe `CountVectorizer` fourni par *Scikit-learn*
- *Doc2Vec* : une méthode donnant un poids aux mots en se basant sur des réseaux de neurones.
- Entrainement du modèle SVC sur le jeu de donnée modifié.

3.1.2 Validation

Pour valider un modèle, nous allons nous baser sur les métrics proposés par Scikit-learn : l'accuracy et la précision principalement. En utilisant les deux pipelines, on obtient les mêmes résultats sauf pour l'accuracy. On gagne 1% de plus en score en utilisant *Doc2Vec* 97% contre 98%.

3.2 K-neighbhors Classifier

Le second modèle choisi est le K-neighbhors Classifier. Ce modèle peut performer correctement sur notre dataset, mais il est généralement plus lent que le précédent.

3.2.1 Pipeline

Pour l'entrainement, on reprend les deux pipeline présenter précédemment.

3.2.2 Validation

En utilisant les deux pipelines, on obtient de moins bons résultats en utillisant le *TFIDF* (95% d'accuracy). Alors qu'en utilisant le *Doc2Vec*, on obtient des résultats similaires à ceux du SVM entrainé avec une "pipeline *TFIDF*".

4 Recommendation d'articles