



UNIVERSITÉ CÔTE D'AZUR 



DIGITAL SYSTEMS
FOR HUMANS
GRADUATE SCHOOL & RESEARCH



FORMASUP
PROVENCE - ALPES - CÔTE D'AZUR



CFR
É P U R E
MÉDITERRANÉE

Université Côte d'Azur

MASTER INFORMATIQUE
Parcours INTELLIGENCE ARTIFICIELLE

Algorithme de recommandation - Projet TATIA

Authors :

BOULLI Marouan & HADDOU Amine

Mentor :

CABRIO Elena

Rapport d'alternance

HADDOU Amine

18 janvier 2024

Table des matières

1	Recommandation d'article	2
2	Jeu De Donnée	2
2.1	Preprocessing	3
3	Entrainement des modèles	4
3.1	C-Support Vector Classification	4
3.1.1	Pipeline	4
3.1.2	Validation	4

1 Recommendation d'article

Comme sujet de projet, nous avons fait le choix d'implémenter un algorithme de recommandation d'article en fonction du dernier article lu par l'utilisateur. Le fonctionnement de cet algorithme se décomposera en deux étapes principales : classer 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 données trouvées mais ces catégories englobent un ensemble d'articles riche et divers. Les catégories sont *business*, *entertainment*, *politics*, *sport* and *tech*.

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

On pourrait alors se demander quelle est l'utilité de classer l'article. Dans un contexte d'une large base de données, il est important de prendre en compte le prix et consommation de calcul de similarité sur un grand ensemble d'articles différents. De plus, on souhaite dans le cadre de notre projet que les articles recommandés 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'efficacité. Pour se faire, nous baserons la validation sur deux métriques : l'*accuracy* et le ratio de mots clés similaires entre les deux articles.

2 Jeu De Données

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

Le jeu de données est décomposé 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 final.

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

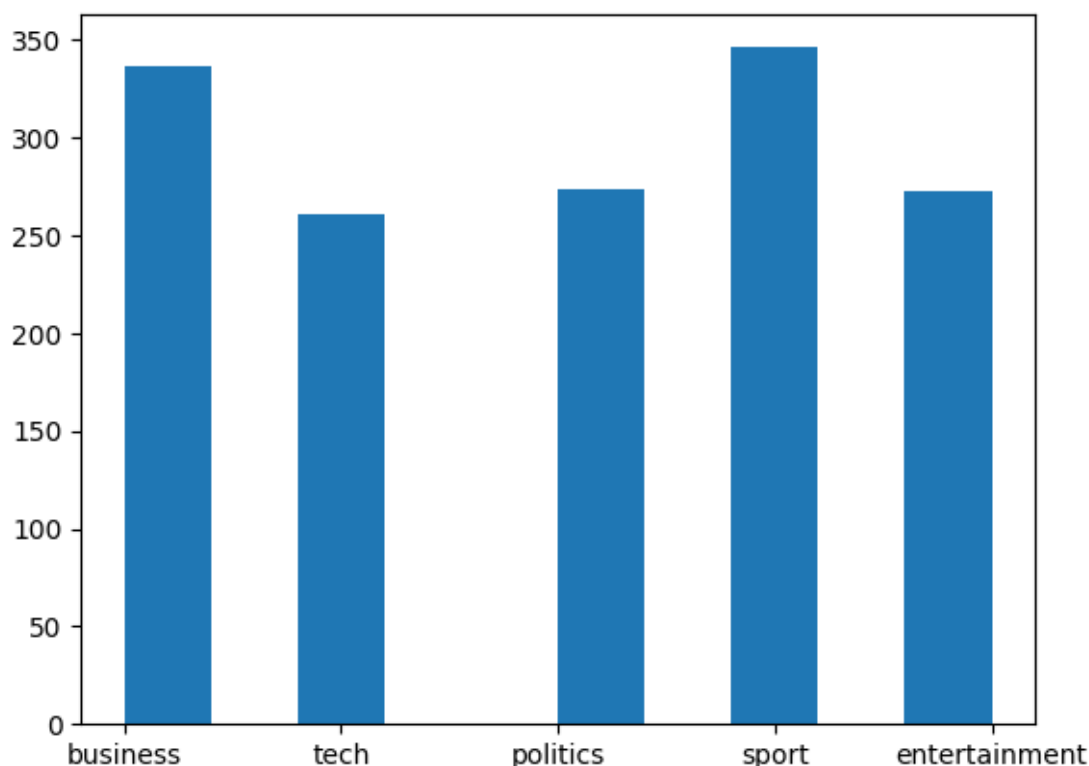


FIGURE 1 – Proportion des catégories d'article

2.1 Preprocessing

Avant d'entamer l'implémentation de l'algorithme, il est essentiel de débiter 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*, entraîné 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 Entraînement 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'entraînement et la seconde pour les testes. Il est habituellement recommandé, pour un jeu de donnée de notre envergure, de garder 77% du data set pour l'entraînement et le reste pour le training. C'est ce qu'on appliquera ici.

3.1 C-Support Vector Classification

Le premier modèle choisi est le Support Vector Classification. 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'entraînement, on crée une 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*
- Tfidf transformr : j'ai oublié ce que ça fait
- Entraînement du modèle SVC sur le jeu de donnée modifié.

3.1.2 Validation