# Classification de résumés d'articles scientifiques à partir de la classification des revues

Léo Gaillard, Lucas Anki, Pascal Cuxac

# Classification de résumés d'articles scientifiques à partir de la classification des revues

Léo Gaillard*, Lucas Anki*, Pascal Cuxac*

*Inist-CNRS (UAR 76), 2 rue Jean Zay, 54500 Vandœuvre-lès-Nancy
[prénom].[nom]@inist.fr,

## 1 Introduction

Pour réaliser un modèle qui permette de classifier des résumés d'articles scientifiques selon la classification Science-Metrix [1], nous disposons d'articles classés en fonction du domaine scientifique de la revue dans laquelle ils sont parus : nous ne pouvons pas nous servir de ces données en l'état pour réaliser un modèle de classification de documents. Pour ce faire, nous avons constitué un corpus de plus de 2,5 millions de documents. Après vectorisation (en utilisant Bert, Devlin et al. (2019)) nous avons appliqué l'algorithme des KPPV en utilisant la bibliothèque Faiss, (Johnson et al. (2019)) pour écarter les documents supposés mal classés afin d'obtenir un jeu de données. Cet algorithme a l'avantage d'être simple et efficace, d'après Guo et al. (2003). L'ensemble de nos codes peut être retrouvé sur notre Github [2].

## 2 Préparation des données

Nous possédons $N > 2700000$ résumés d'articles scientifiques ainsi que la classification Science-Metrix de la revue dont ils sont issus. Nous avons choisi de sélectionner les données ayant le plus de chance d'être correctement classées en utilisant un algorithme des KPPV.

Nous utilisons le modèle Bert pour obtenir nos embeddings (de dimension 768 : soit $(M_{nk})_{n,k\in[\![1,N]\!]\times[\![1,768]\!]}$ où $n$ est le document représenté et $k$ la k-ème coordonnée de l'embedding. On calcule la matrice C des distances cosinus, connue efficace pour comparer des données textuelles (jugée bonne mais perfectible par Li et Han (2013)) :

$$\forall n, m \in [\![1, N]\!] \times [\![1, N]\!], C_{nm} = 1 - \frac{\sum_{k=1}^{768} M_{nk} M_{mk}}{\sqrt{\sum_{k=1}^{768} {M_{nk}}^2} \sqrt{\sum_{k=1}^{768} {M_{mk}}^2}}$$

Une fois cette matrice obtenue, nous pouvons calculer pour chaque classe $l_{i\in[1,174]}$ les documents les plus pertinents à regarder. En appliquant l'algorithme des KPPV sur la matrice C des distances cosinus calculées, on peut ordonner pour chaque classe les documents ayant le plus de documents de même label qu'eux parmi leurs k plus proches voisins. Nous avons choisis un k différent pour chaque classe $l_i$ et avons pris $k_{l_i} = Card\{n, n \in l_i\} + 1$. Le risque

1. https://www.science-metrix.com/fr/classification/
2. https://github.com/Inist-CNRS/ws-data/tree/master/sciencemetrix-classification

de prendre un $k$ trop petit est de ne pas pouvoir départager les meilleurs documents sur les classes sur-représentées.

Une fois cette liste ordonnée obtenue, nous choisissons pour chaque classe $l_i$ constituée de $s_{l_i}$ documents, le nombre suivant de documents : $\max(3000, |p \times s_{l_i}|)$ (où $|.|$ est la fonction partie entière, et $0 \leq p \leq 1$ la proportion de documents à conserver pour constituer le jeu d'entraînement). Nous prenons au maximum les 3000 meilleures données pour limiter le déséquilibre des classes sur-représentées à l'entraînement (nous avons fait varier le nombre maximum de données à prendre sans différences majeures : 3000 est fixé pour l'instant).

Nous obtenons ainsi nos données. Nous les séparons en tirant uniformément 20 % des documents de ce corpus pour constituer le jeu de test, les 80 % restants constituent les données d'entraînement.

# 3 Résultats

Pour évaluer cette méthode, nous créons trois jeux de données d'entraînement et de test :

— Le premier jeu de données $D_1$ est constitué en prenant $p = 0.3$.

— Le deuxième jeu de données $D_2$ est constitué en posant $p = 0.7$. Nous avons plus de données pour les classes de tailles petites et normales mais elles sont plus dispersées.

— Le troisième jeu de données $D_3$ est constitué sans prendre en compte le classement des documents obtenus après KPPV et servira de jeu de données de référence lors de la comparaison des *accuracies*. L'équilibre des classes est préservé.

Nous avons utilisé deux bibliothèques pour entraîner nos modèles : XGBoost, en gardant l'embedding Bert utilisé dans la partie de sélection des données et FastText, utilisant le doc2vec proposé par défaut. Nous comparons les résultats dans le tableau 1.

| | fastText | XGBoost |
|---|---|---|
| $D_3$ | 0.58 | 0.53 |
| $D_2$ | 0.76 | 0.74 |
| $D_1$ | **0.84** | **0.82** |

TAB. 1 – *accuracy obtenue en fonction des bibliothèques d'entraînement utilisées et de la proportion prise au moment du kppv.*

On observe que plus l'on restreint le jeu de données en utilisant notre méthode de sélection, plus l'*accuracy* augmente. Pour $p < 0.3$, l'augmentation ne semble plus significative.

# 4 Conclusion

En conclusion, la sélection des données par KPPV améliore significativement la performance du modèle sortant (*accuracy* multipliée par plus de $1.5$). Nous avons choisi de générer les données de *test* avec le même protocole que le corpus d'entraînement. Il serait intéressant de comparer ces modèles sur un ensemble de documents correctement labélisés. Au vu de la répartition de nos données dans chacune des classes, il serait intéressant d'augmenter les données des petites classes (par requête ou par génération de données).

## Références

Devlin, J., M.-W. Chang, K. Lee, et K. Toutanova (2019). Bert : Pre-training of deep bidirectional transformers for language understanding.

Guo, G., H. Wang, D. Bell, Y. Bi, et K. Greer (2003). Knn model-based approach in classification. In R. Meersman, Z. Tari, et D. C. Schmidt (Eds.), *On The Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE*, Berlin, Heidelberg, pp. 986–996. Springer Berlin Heidelberg.

Johnson, J., M. Douze, et H. Jégou (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data 7*(3), 535–547.

Li, B. et L. Han (2013). Distance weighted cosine similarity measure for text classification. In H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, et X. Yao (Eds.), *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, Berlin, Heidelberg, pp. 611–618. Springer Berlin Heidelberg.

## Summary

We present here how to build a dataset to train a Science-Metrix scientific domain classification model based on scientific article abstracts, knowing that we only have access to the Science-Metrix domain of the journal in which the article appeared.