

Etude comparative de vecteurs pour l'identification du parti politique d'interventions parlementaires

Pauline Degez and Florian Philippe and Valentine Fleith

43004729@parisnanterre.fr

40007675@parisnanterre.fr

43017467@parisnanterre.fr

Abstract

Dans cette étude, nous avons exploré trois techniques de vectorisation différentes (TF-IDF, Doc2Vec et embeddings BERT) ainsi que quatre algorithmes d'apprentissage automatique (Random Forest, SVM, Régression Logistique et Perceptron) pour réaliser la tâche 3 de la cinquième édition du défi DEFT'09 (Défi Fouille de Texte 2009). Notre objectif était de comparer les performances des différents types de vectorisation dans une tâche de classification de texte, et de nous confronter aux résultats de 2009 (0.33 f-mesure). En définitive, le modèle SVM avec des vecteurs TF-IDF s'est révélé le plus performant (0.47). Les résultats obtenus avec les embeddings BERT, en revanche, se sont avérés plutôt décevants, bien qu'ils pourraient être améliorés par un fine-tuning pour mieux prendre en compte la spécificité du corpus.

1 Introduction

La cinquième édition du défi Fouille de Textes (DEFT) porte sur la fouille d'opinions sur des corpus multilingues. Trois tâches ont été proposées, dans trois langues : le français, l'anglais et l'italien. Cet article se concentre sur la 3ème tâche, dont l'objet est l'identification automatique du parti politique d'appartenance de chacun des intervenants dans un corpus de débats parlementaires européens. Il s'agit d'une tâche de classification à 5 classes: Verts-ALE, GUE-NGL, PSE, ELDR et PPE-DE.

Le but de nos expériences sera ainsi de trouver un/des classifieur(s) permettant de réaliser cette tâche. Pour ce faire, nous utiliserons les algorithmes de Machine Learning implementés dans la bibliothèque Python scikit-learn.

1.1 Travaux présentés en 2009

En 2009, un seul participant a soumis un travail pour la tâche 3 ; la Présentation de l'édition

Parti	ELDR	GUE-NGL	PPE-DE	PSE	Verts/ALE
F-mesure	0.21	0.37	0.47	0.37	0.25

Table 1: Moyennes des F-mesures par parti politique.

2009 [1]¹ évoque, pour expliquer cela, les faibles résultats des logiciels sur cette tâche, bien que conformes à ceux que des humains obtiendraient manuellement. L'équipe de l'Université de Montréal (D. Forest and al.) a obtenu en moyenne les f-mesures présentées dans la Table 1. En moyenne, cela donne donc une f-mesure 0.331.

1.2 Notre approche et travaux antérieurs

Pour ce travail, notre approche a été comparative sur plusieurs niveaux. Tout d'abord, nous comparons différents classifieurs : Random Forest, Régression logistique, Perceptron et Support Vector Machine. De plus, nous testons aussi différentes vectorisations du corpus sur l'ensemble de ces modèles: TF-IDF, Doc2Vec, et des Bert embeddings.

Plusieurs travaux de recherches explorent les comparaisons entre performances des modèles selon les techniques de vectorisation utilisées. Nous pouvons par exemples évoquer ceux de P. Joseph et S. Y. Yerima [2]² en 2022, qui compare les performances des N-grams, TF-IDF, Sac de mots, Word2Vec, Doc2Vec, etc. Leur objectif est de comparer l'impact de la vectorisation sur la précision des modèles. Dans leur article, les modèles Doc2Vec et TF-IDF démontrent de bons résultats, nous allons ainsi les tester dans notre expérience. Nous décidons d'ajouter à ces deux dernier les embeddings de BERT afin d'avoir trois

¹Actes du cinquième défi fouille de texte, DEFT2009, Paris, France, 22 juin 2009

²P. Joseph and S. Y. Yerima, "A comparative study of word embedding techniques for SMS spam detection," 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 2022, pp. 149-155,

techniques variées : une méthode statistique, une méthode fondée sur un ANN classique et une sur un Transformer.

2 Méthode

2.1 Dataset

Nous utilisons le dataset fournit pour la tâche 3 de l'édition 2009 de DEFT. Au moment de faire des statistiques descriptives, nous nous sommes rendus compte que le corpus présentait des doublons, et ce majoritairement dans la partition test.

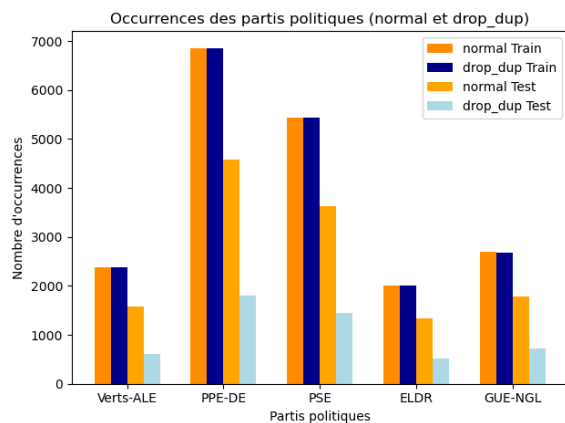


Figure 1: Nombre d'interventions par parti par partition test/train, dans le corpus original et dans la version sans doublons

Après suppression des doublons, la partition prévue (40/60) est changée : elle est maintenant de 20/80³. Nous avons envisagé de réimplémenter le partitionnement prévu, mais avons renoncé pour deux raisons : refaire le partitionnement nous éloigne, encore, du corpus initial, et les résultats de quelques modèles sur un corpus repartitionné étaient proches des résultats sur cette partition 20/80. Par ailleurs, la répartition des classes est déséquilibrée : les classes PPE-DE et PSE sont plus grandes et forment à elles deux 63,5 % du corpus. Ceci devra être pris en compte dans le prétraitement.⁴

2.2 Prétraitements

Le texte des interventions a été soumis à un pré-traitement simple :

- (1) Suppression de la ponctuation
- (2) Unification de la casse en minuscules

³0.79 pour le train et 0.21 pour le test

⁴la figure correspond au train, mais la répartition est sensiblement la même dans le test

Statistique	Train	Test
Moyenne	3871.4	1021.2
STD	2149.1	569.7
Min	2005.0	525.0
1er quartile	2376.0	615.0
Médiane	2687.0	715.0
3eme quartile	5431.0	1448.0
Max	6858.0	1803.0

Table 2: Nombre d'intervention des partis par partition

Répartition des partis politiques dans le corpus train

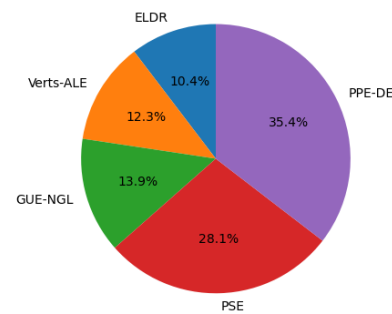


Figure 2: Répartition des interventions par parti dans la partition train sans doublons

(3) Tokenisation⁵

Pour résoudre le problème de déséquilibre des classes, nous avons opté pour le *downsampling* afin d'obtenir des classes relativement équilibrées, en utilisant la fonction *resample* de la bibliothèque *scikit-learn*. Nous avons choisi de downsampler les deux classes majoritaires en fonction de la médiane du nombre de documents sur toutes les classes. Après *downsampling*, les classes PPE-DE et PSE ne représentent plus que 43% du corpus, ayant chacune été ramenée autour de 21,5% du corpus

2.3 Les différents classifieurs testés

Nous avons sélectionné quatre modèles classiques de machine learning, à savoir Random Forest, Support Vector Machine (SVM), Régression Logistique et Perceptron.

Le **Random Forest** est une forêt d'arbres de décision qui utilisent le bagging pour améliorer la qualité des prédictions. L'algorithme fonctionne par une combinaison des résultats de multiples ar-

⁵Une lemmatisation avec la bibliothèque SpaCy a été envisagée, mais ce corpus multilingue aurait nécessité le chargement de 3 modèles linguistiques différents et ralenti d'autant le temps de traitement

Parti	Train	Test
ELDR	2531 (16,1%)	525 (16,2%)
PPE-DE	3402 (21,63%)	715 (22%)
GUE-NGL	3402 (21,63%)	687 (21,2%)
PSE	3402 (21,63%)	693 (21,4%)
Verts-ALE	2990 (19%)	614 (19%)
Total	15727	3235

Table 3: Nombre d'interventions par parti par partition pour une langue après *downsampling*, exemple de l'italien

bres construits sur des sous-échantillons pris aléatoirement dans le corpus.

Le **SVM**, quant à lui, repose sur l'idée de trouver un hyperplan optimal pour séparer les classes qui garantit une marge maximale. L'algorithme utilise une fonction noyau pour passer transformer les données et ajouter un espace de dimension supplémentaire, ce qui permet de résoudre des problèmes de classification non linéaire tout en utilisant tout de même un classifieur linéaire.

La **Régression Logistique** modélise la probabilité qu'un échantillon appartienne à une classe en utilisant une fonction sigmoïde. Elle nécessite une relation linéaire entre les *features* et la classe.

Enfin, le **Perceptron** est un modèle de réseau de neurones linéaire qui ajuste les poids d'un seul neurone en fonction de l'erreur de prédiction, optimisé via une descente de gradient.

Ces quatre modèles ont été sélectionnés car complémentaires : ils permettent d'aborder le problème sous différents angles, avec des approches plus adaptées à la linéarité et d'autres à la non linéarité.

2.4 Les différents vecteurs testés

Nous avons choisi dans notre étude de comparer les résultats obtenus sur une tâche de classification en utilisant 3 techniques de vectorisation différentes.

La vectorisation **TF-IDF**⁶ (*Term Frequency-Inverse Document Frequency*) est la méthode la seule méthode non neuronale que nous présentons, en ce qu'elle se fonde sur des statistiques. La fréquence d'apparition de chaque mot dans un document est divisée par sa fréquence d'apparition dans le corpus, permettant de donner plus d'importance aux mots significatifs dans leurs documents d'apparition.

La vectorisation **Doc2Vec**⁷ qui génère un *embed-*

⁶Implémentée avec la fonction `tfidfvectorizer` de `scikit-learn`

⁷Implémentée à l'aide de la bibliothèque `gensim`

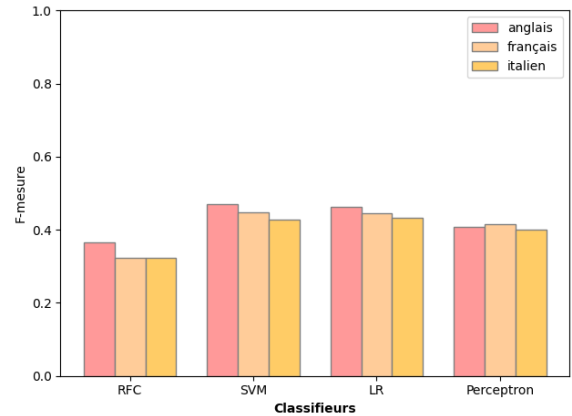


Figure 3: Résultats obtenus avec les vecteurs TF-IDF.

ding pour chaque document. Ces vecteurs sont l'output d'un réseau de neurones à mémoire distribuée. Pour cette étude, nous avons choisi de générer des vecteurs Doc2Vec à 100 dimensions⁸ avec une fenêtre glissante de 5 mots et d'ignorer les mots n'apparaissant pas au moins 3 fois.

La vectorisation avec **BERT** multilingue⁹ se base sur les réseaux de neurones mais s'appuie sur une architecture transformer caractérisée par ses mécanismes d'attention. Contrairement à Doc2Vec, BERT produit des *embeddings* au niveau des tokens, qui peuvent être agrégés pour obtenir un vecteur document. Nous avons généré des vecteurs à 768 dimensions (profondeur de la couche de sortie finale du modèle pré-entraîné).

3 Résultats

Comme mentionné en introduction, nous avons testé trois méthodes de vectorisation : TF-IDF, doc2vec et BERT. Nous avons utilisé ces vecteurs pour entraîner et comparer 4 modèles différents : Régression logistique, SVM, Random Forest et Perceptron.

3.1 Vecteurs TF-IDF

Les meilleurs résultats que nous avons obtenus sont ceux avec une vectorisation tf-idf. On peut voir sur la figure ?? que nous avons obtenu au mieux : 0.46 en anglais avec un SVM, 0.43 en italien avec une régression linéaire et 0.44 en français avec un SVM.

⁸Bien que des dimensions plus élevées puissent améliorer la capacité de représentation, nous avons choisi cette configuration pour ne pas saturer nos ressources matérielles.

⁹`bert-base-multilingual-uncased`

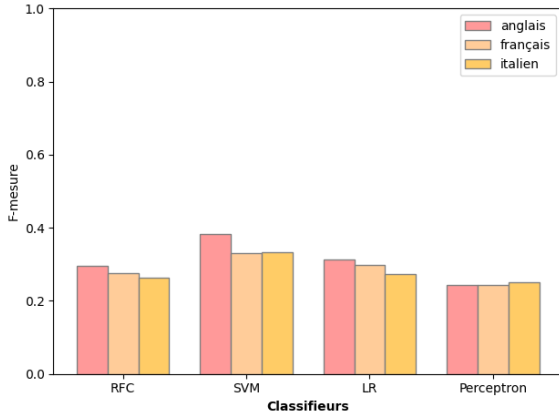


Figure 4: Résultats obtenus avec les vecteurs doc2vec.

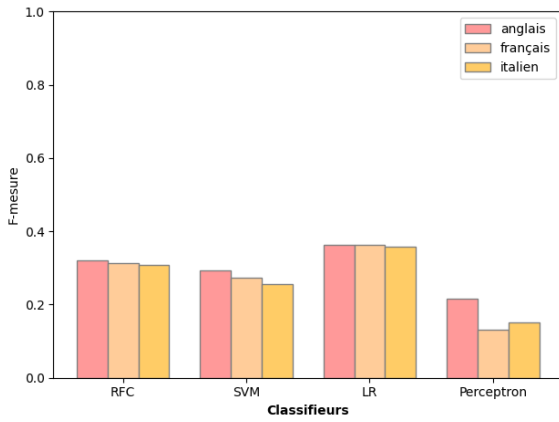


Figure 5: Résultats obtenus avec les vecteurs BERT.

3.2 Vecteurs doc2vec

Comme on peut le voir sur la figure ??, avec les vecteurs doc2vec, c'est le SVM qui obtient les meilleurs résultats dans les trois langues.

3.3 Vecteurs BERT

La figure ?? montre qu'avec les vecteurs BERT, la régression logistique obtient les meilleurs résultats dans les trois langues.

3.4 Comparaison des méthodes de vectorisation

Le tableau ?? montre les résultats obtenus avec les trois méthodes de vectorisation. On peut voir que la méthode TF-IDF est la plus performante dans les trois langues.

Le meilleur résultat a été obtenu sur l'anglais avec un SVM et des vecteurs TF-IDF, avec un score de 0.47. La figure ?? montre la matrice de confusion correspondante.

	TF-IDF	DOC2VEC	BERT
Français	0.44	0.33	0.36
Anglais	0.47	0.38	0.36
Italien	0.43	0.33	0.35

Table 4: Comparison of different models

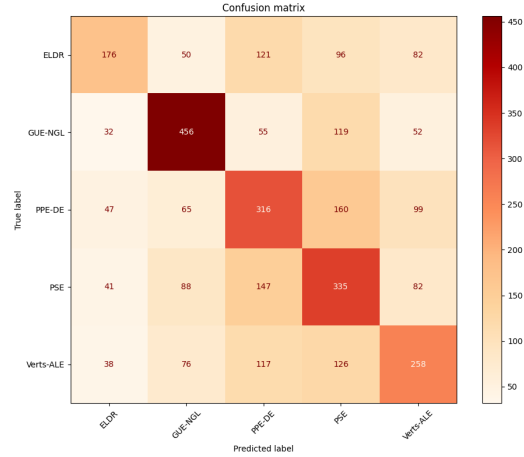


Figure 6: Matrice de confusion pour l'anglais avec des vecteurs TF-IDF et un SVM.

4 Discussion et Conclusion

Ces résultats montrent que, sur ce corpus, les vecteurs qui donnent les meilleurs classifieurs sont finalement les plus simples, les vecteurs TF-IDF. Cela peut s'expliquer de plusieurs façons. Tout d'abord, les données étudiées donnent une explication de ces performances : les données de discours du parlement européens sont de nature spécialisée, et contiennent nécessairement des mots spécialisés au domaine. Or, Bert est entraîné sur un corpus général. Il faudrait le fine-tuner afin d'obtenir de meilleurs résultats. En revanche, le vectoriseur tf-idf s'adapte aux mots de vocabulaire spécifique à ce type de document. Par ailleurs, on peut noter que la taille très réduite des documents peut être à l'origine de la mauvaise qualité des vecteurs obtenus, non seulement avec BERT mais aussi avec doc2vec. De plus, les classifieurs que nous lançons sur nos données sont des classifieurs simples, linéaires ou arbres. Or, les vecteurs complexes générés par réseaux de neurones pourraient nécessiter un réseau de neurones pour la classification ensuite; cela pourrait constituer une perspective pour poursuivre notre expérience. Nous pourrions également imaginer des approches hybrides, qui combindraient les features TF-IDF avec des embeddings BERT pour capturer autant d'informations que nécessaires.

5 Bibliographie

Grouin, Arnulphy, Berthelin, El Ayari, García-Fernandez, Grappy, Hurault-Plantet, Paroubek, Robba, Zweigenbaum. 2009. [Présentation de l'édition 2009 du Défi Fouille de Textes \(DEFT'09\)](#) In *Proceedings of the Fifth DEFT Workshop*, DEFT2009, Paris, France, 22nd June 2009. Pages 35-50.

Joseph, Prashob and Yerima, Suleiman Y. 2022. [A comparative study of word embedding techniques for SMS spam detection](#). In *Proceedings of the 14th International Conference on Computational Intelligence and Communication Networks (CICN)*. Al-Khobar, Saudi Arabia, 2022. Pages 149-155.