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 de différents classifieurs et vecteurs dans une tâche de classification de textes, et de nous confronter aux résultats de 2009 (0.33 f-mesure sur le français). 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 possiblement en raison du corpus, mais pourraient peut-être être améliorés par un *fine-tuning*.

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 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.

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.

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 différentes vectorisations du corpus sur l'ensemble de ces modèles: `TF-IDF`, `Doc2Vec`, et des `Bert embeddings`.

Plusieurs travaux de recherche explorent les comparaisons entre les 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 comparent les performances des `N-grams`, `TF-IDF`, `Sac de mots`, `Word2Vec`, `Doc2Vec`, etc. Leur objectif était 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 avons donc décidé de les inclure dans nos expériences. Nous avons ajouté à ces deux derniers les *embeddings* de `BERT` afin d'avoir trois techniques variées : une méthode statistique, une méthode fondée sur un ANN classique et une sur

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

[2]P. 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,

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 rendu compte que le corpus présentait de nombreux doublons.

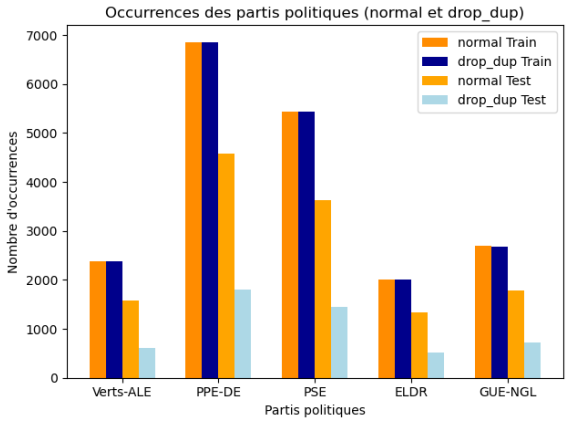

Figure 1: Nombre d'interventions par parti par partition test/train, dans le corpus original et dans la version sans doublons

Après suppression des doublons, la partition prévue n'est plus respectée : elle est passée de 40/60 à 20/80[3]. Nous avons envisagé de rétablir le partitionnement prévu, mais avons renoncé pour deux raisons. Tout d'abord, refaire le partitionnement nous aurait éloigné encore plus du corpus initial. De plus, des expériences préliminaires sur quelques modèles n'ont montré que peu de changement avec les résultats de la partition 20/80.
Par ailleurs, la répartition des classes est déséquilibrée : les classes PPE-DE et PSE sont plus fournies et forment à elles deux 63,5 % du corpus. Ceci sera pris en compte dans les prétraitements.[4]

2.2 Prétraitements

Le texte des interventions a été soumis à un prétraitement simple :

(1) Suppression de la ponctuation
(2) Unification de la casse en minuscules
(3) Tokenisation[5]

[3] 0.79 pour le train et 0.21 pour le test

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

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

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'interventions des partis par partition

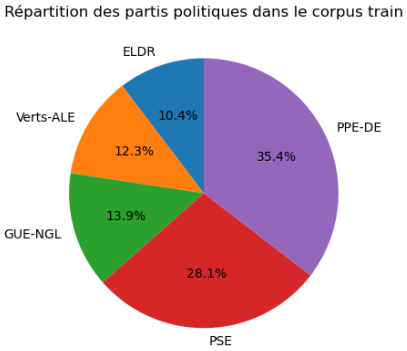

Figure 2: Répartition des interventions par parti dans la partition train sans doublons

Pour résoudre le problème du déséquilibre des classes, nous avons opté pour le *downsampling* en utilisant la fonction *resample* de la bibliothèque `sckit-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.

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*, example de l'italien

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 utilise le *bagging* pour améliorer la qualité des prédictions. L'algorithme fonctionne par une combinaison des résultats de multiples arbres 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 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 selectionnés car complémentaires : ils permettent d'aborder le problème sous différentes 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 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] génère un *embedding* pour chaque document. Ces vecteurs sont *l'ouput* d'un réseau de neurones à mémoire distribuée. Pour cette étude, nous avons choisi de générer des vecteurs `Doc2Vec` à 100 dimensions[8] avec une fenêtre glissante de 5 mots et d'ignorer les mots n'apparaissant pas au moins 3 fois.

La vectorisation avec **`BERT`** multilingue[9] 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-entrainé).

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.

Clf	RFC	SVM	Perceptron	LR
Anglais	0.37	**0.47**	0.41	0.46
Français	0.34	**0.45**	0.42	0.44
Italien	0.33	**0.43**	0.40	0.43

Table 4: F-mesure des classifieurs par langue avec vectorisation `TF-IDF`

Clf	RFC	SVM	Perceptron	LR
Anglais	0.29	**0.39**	0.22	0.31
Français	0.26	**0.34**	0.25	0.28
Italien	0.27	**0.33**	0.22	0.28

Table 5: F-mesure des classifieurs par langue avec vectorisation `Doc2Vec`

Clf	RFC	SVM	Perceptron	LR
Anglais	0.32	**0.32**	0.22	0.36
Français	0.31	**0.27**	0.13	0.36
Italien	0.31	**0.26**	0.15	0.36

Table 6: F-mesure des classifieurs par langue avec vectorisation `BERT`

En ce qui concerne les vecteurs, les meilleurs résultats observés ont été obtenus à partir de `TF-IDF` : le meilleur classifieur a une f-mesure de 0.47

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

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

[8]Bien 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*

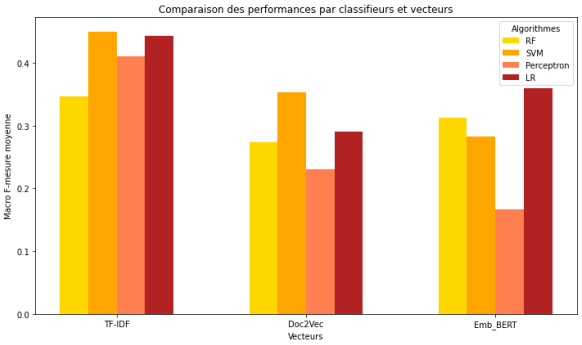

Figure 3: Comparaison des vecteurs et classifieurs.

avec ces vecteurs, tandis que le meilleur à partir de Doc2Vec est à 0.39 et à partir de BERT à 0.36. De façon générale, les résultats ont tendance à être supérieurs avec cette méthode de vectorisation. Bien que BERT et Doc2Vec démontrent des performances similaires, cette dernière méthode est celle qui performe le mieux des deux.

Globalement, les résultats sont relativement homogènes entre les langues. Avec le TF-IDF, la moyenne sur l'anglais de tous les modèles est à 0.47, 0.44 pour le français et 0.43 pour l'italien. Celui qui performe le moins bien est le Random Forest (f-mesure moyenne à 0.32 sur l'anglais).

Au niveau des classifieurs, on constate relativement peu de disparités entre leurs performances. De manière générale, le meilleur classifieur est le SVM (en moyenne à 0.39 sur l'anglais en combinant toutes les méthodes de vectorisation), toutefois relativement proche de la Régression logistique (f-mesure moyenne à 0.37).

4 Discussion et Conclusion

Les résultats montrent que, sur ce corpus, les vecteurs qui donnent les meilleurs classifieurs sont finalement les moins complexes et les moins denses, c'est-à-dire, les vecteurs TF-IDF. Cela peut s'expliquer de plusieurs façons. Une première raison peut être la nature même des données étudiées : les données de discours du parlement Européens sont spécialisées, et contiennent nécessairement des mots relatifs au domaine. Or, BERT est entrainé sur un corpus général. On pourrait envisager de le *fine-tuner* afin d'obtenir de meilleurs résultats. [10] En revanche, le vectoriseur TF-IDF s'adapte au vocabulaire juridique spécifique à ce type de document. Par ailleurs, on peut supposer que la taille relativement 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 sous forme arbres de décision. Or, les vecteurs complexes générés par réseaux de neurones pourraient en nécessiter un égaklement pour la classification ensuite; cela pourrait constituer une perspective intéressante pour poursuivre notre expérience. Nous pourrions également imaginer des approches hybrides, qui combineraient les *features* TF-IDF avec des *embeddings* BERT, pour capturer autant d'informations que nécessaire. De plus, afin d'améliorer les résultats, nous avons envisagé de lancer une Grid Seach pour tuner les hyperparamètres de chacun des classifieurs, mais après quelques tests, le coût computationnel n'a pas su être compensé par une amélioration suffisante de la f-mesure.[11]

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.

[10] Il existe également un modèle spécialisé sur le domaine légal, nommé LEGAL-BERT, mais il n'est entraîné que sur l'anglais. Cela pourrait toutefois constituer une recherche ultérieure.

[11] Nous l'avons laissée dans le code malgré tout et il est toujours possible de la faire tourner sur l'ensemble des classifieurs *(cf.* README.md*)*.