Tâche 3 de l'édition 2009 du DÉfi Fouille de Texte (DEFT): apprentissage de classification par parti politique d'interventions au parlement européen.

GOURVES Manon & BESSAC Lucile

Abstract

La cinquième édition du défi fouille de textes (DEFT) a eu lieu en 2009 et portait sur la fouille d'opinion. Une des tâches (tâche 3) du défi était d'**identifier le parti politique d'appartenance d'un parlementaire à partir d'interventions contenues dans un corpus multilingue** (français, anglais, italien) composé de débats parlementaires européens. Cette tâche s'était à l'époque révélée difficile et les méthodes proposées avaient alors obtenu de mauvais résultats.

Cet article présente les résultats obtenus pour cette même tâche réalisée en 2024 sur le **corpus français** à l'aide des outils les plus répandus aujourd'hui et vise à déterminer si oui ou non ces derniers permettent d'obtenir de meilleurs résultats.

1 Introduction

Lors de l'édition 2009 du DEFT, la tâche d'identification du parti politique d'un orateur avait obtenu des résultats jugés médiocres et il avait été conclu que "même dans un contexte parlementaire où les opinions sont supposées s'exprimer clairement, il n'est pas facile d'attribuer le bon parti à l'orateur.". Mais est-ce toujours le cas en 2024 ?

Dans cet article, nous reproduirons dans un premier temps les méthodes utilisées en 2009, puis nous explorerons des algorithmes, des vectorisations et des approches actuelles avant de comparer nos résultats à ceux obtenus en 2009.

2 Corpus

Le corpus utilisé est un corpus en français d'interventions parlementaires au Parlement européen.

Ce corpus a été constitué en récupérant les archives de séances parlementaires qui se sont tenues **entre 1999 et 2004**. Chaque intervention est enrichie de méta-données dont le nom du groupe politique européen dont le parlementaire qui s'exprime relève.

Les classes du corpus sont réparties comme suit, avec **19370 documents dans le corpus de train** et 12917 documents dans le corpus de test :

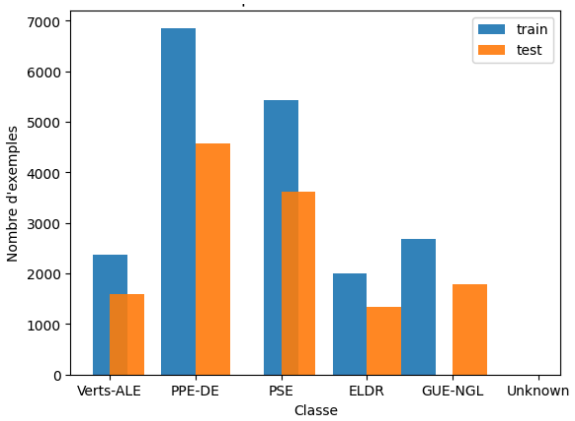

***Fig. 1** : Répartition des classes dans les corpus de train et test.*

3 Le travail de 2009

La méthode de 2009 identifie des mots discriminants, représentés par leur **fréquence** ou leur **fréquence pondérée par IDF**, pour décrire les documents soumis à l'algorithme des **k plus proches voisins** ou au **classifieur naïf bayésien**.

Elle cherche la meilleure combinaison de nombre de traits discriminants, technique de vectorisation et algorithme de classification.

Après suppression des stop-words et lemmatisation, **les meilleures performances ont été obtenues avec 10 000 mots discriminants représentés par leur fréquence et k=2 pour k-NN.**

Les résultats de 2009 suggèrent un surapprentissage au moment de l'entraînement car les mots discriminants en entraînement se révèlent moins efficaces en test. De plus, les performances dépendent du nombre de traits discriminants : plus il y en a, meilleurs sont les résultats.

Afin de tester cette hypothèse, nous reproduirons d'abord l'expérience de 2009, puis expérimenterons de nouveaux classifieurs facilement implémentables et accessibles en 2024, puis combinerons ces derniers à une nouvelle méthode de vectorisation.

4 Notre méthodologie

4.1 Reproduction de l'expérience de 2009

Nous avons décidé dans un premier temps de reproduire la méthodologie suivie par Forest et al. afin d'assurer une comparaison fiable. Nous avons travaillé sur le même corpus, limité aux données en français. La vectorisation, l'implémentation des modèles et l'optimisation des paramètres a été réalisée avec SciKit-Learn.

Pour la vectorisation, nous avons retenu la méthode TF-IDF en raison de ses avantages par rapport à la méthode CountVectorizer qui se limite seulement à la fréquence brute des mots tandis que TF-IDF ajuste le poids des mots en fonction de leur fréquence dans un document et de leur importance globale dans le corpus. **TF-IDF est particulièrement performant pour la classification textuelle** car il atténue l'impact des mots trop fréquents et peu informatifs tout en mettant l'accent sur les mots discriminants du corpus.

À partir de cette vectorisation, nous avons entraîné les mêmes algorithmes que dans l'étude originale, à savoir le **k-plus proches voisins** (KNN) et un **Naïve Bayes**, afin d'évaluer leurs performances sur les données françaises.

Ne connaissant pas la variante exacte du Bayesien Naïf employée, nous avons opté pour le **Multinomial Naïve Bayes** (MNB) car il correspond mieux à la nature de nos données. En plus, nous avons optimisé les paramètres avec GridSearchCV, bibliothèque qui n'existait pas en 2009.

Lors de l'étude de Forest et al. les meilleurs résultats obtenus ne dépassent pas une **f-mesure de 0.34**. Dans leurs tests, l'algorithme des k plus proches voisins a toujours généré de meilleurs résultats que le classifieur bayésien naïf.

Dans notre cas, l'algorithme KNN obtient de moins bons résultats qu'en 2009 avec une f-mesure de 0.24. Cependant, le Multinomial Naïve Bayes atteint une f-mesure de 0.53, ce qui est une augmentation considérable par rapport à 2009. C'est l'optimisation des hyperparamètres facilitée par GridSearchCV qui nous a permis d'obtenir de meilleures performances.

4.2 Test de nouveaux algorithmes

Étant donné qu'une f-mesure de 0.53 reste insuffisante, nous avons cherché à améliorer nos résultats en explorant **d'autres algorithmes** qui n'étaient pas utilisés dans l'étude originale.

4.2.1 Complement Naive Bayes

Le premier algorithme est une variante du Multinomial Naïve Bayes. Il s'agit du Complement Naïve Bayes, conçu pour mieux gérer les ensembles de données avec des **classes déséquilibrées**, nous avons estimé qu'il pourrait être adapté à nos données. Cependant, les résultats obtenus sont inférieurs à ceux du MNB, avec une F-mesure de **0.46**.

4.2.2 SVM

Nous avons choisi d'essayer le SVM en second algorithme car il est efficace pour gérer les tâches de classification textuelle où les frontières entre les classes peuvent être subtiles. Avec une f-mesure de **0.528**, il donne des résultats très proches du MNB.

4.2.3 Logistic Regression

Notre troisième algorithme est Logistic Regression. C'est un modèle simple mais il est généralement performant sur des tâches de classification. Cependant, ces performances peuvent êtres limitées lorsqu'il y a un déséquilibre de classes. Toutefois, comme le Complement Naïve Bayes, conçu pour gérer ses déséquilibres, n'a pas surpassé MNB, nous avons décidé d'essayer la régression logistique.

Logistic Regression obtient une f-mesure de **0.50**, ce qui ne permet pas de dépasser le Multinomial Naïve Bayes.

4.2.4 Random Forest

Nous avons également fait des essais avec Random Forest, cependant le coût de calcul était très élevé pour des résultats très peu performants.

4.3 Test de nouvelles vectorisations

Les données utilisées sont riches lexicalement et contextuellement, soulignant l'importance des relations sémantiques entre mots. Par exemple, le mot *écologiste* peut indiquer une affiliation aux “Verts/ALE”, mais dans un contexte comme *ces emmerdeurs d’écologistes*, il pourrait refléter un parti opposé. Pour capturer ces nuances, nous avons intégré des méthodes tenant compte du contexte et des relations sémantiques.

4.3.1 Tf-idf Vectoriser et ngrams

Notre première méthode utilise la vectorisation TF-IDF avec un argument n-gram=(1,2) pour rendre compte des relations locales entre les mots. À part pour le KNN, tous les classifieurs ont donné de bons résultats. SVM est en tête avec une accuracy de 0.722 et une F-mesure de **0.718**, ce qui indique un bon équilibre entre la précision et le rappel. L’utilisation d’une vectorisation TF-IDF avec *n-grams=(1,2)* a permis de capturer davantage de relations contextuelles entre les mots. Cette méthode de vectorisation contribue à une amélioration globale des performances des modèles.

	Modèle	Accuracy	Précision	Rappel	F-mesure
0	KNeighbors	0.343423	0.802203	0.343423	0.244658
1	Multinomial Naive Bayes	0.676008	0.776424	0.676008	0.662563
2	Complement Naive Bayes	0.662305	0.793927	0.662305	0.652196
3	SVM	0.722459	0.733349	0.722459	0.718334
4	Logistic Regression	0.704421	0.719379	0.704421	0.699318

Table. 1 : *Rapport de classification des algorithmes avec TF-IDF et n_grams=(1,2)*

4.3.2 Word2Vec

Nous avons testé Word2Vec, absent des travaux de Forest et al. (2009) car introduite en 2013. Word2Vec génère des représentations continues de mots capturant leurs relations sémantiques et contextuelles.

Cependant, les classifieurs bayésiens, incompatibles avec des vecteurs pouvant contenir des valeurs négatives, ont été écartés au profit de SVM et de la Régression Logistique, adaptés à cette vectorisation.

Pour classifier les documents, nous avons calculé la moyenne des vecteurs de mots de chaque document. Nous obtenons malheureusement des résultats médiocres avec **0.06** de F-mesure pour la Régression Logistique et 0.09 de F-mesure pour le SVM.

	Modèle	Accuracy	Précision	Rappel	F-mesure
0	Logistic Regression W2V	0.146783	0.047783	0.178559	0.064073
1	SVM W2V	0.178757	0.064591	0.227903	0.098982

Table. 2 : *Rapport de classification des algorithmes avec Word2Vec*

Après réflexion, l'utilisation de Doc2Vec aurait été plus appropriée pour produire directement des vecteurs de documents, ce qui pourrait expliquer nos résultats décevants avec Word2Vec.

4.3.3 Doc2Vec

Nous avons donc testé Doc2Vec qui vectorise directement les documents.

Les résultats se sont effectivement améliorés mais restent décevants avec **0.2** de F-mesure pour la Régression Logistique et 0.19 pour le SVM.

	Modèle	Accuracy	Précision	Rappel	F-mesure
0	Logistic Regression D2V	0.340094	0.222185	0.230094	0.202776
1	SVM D2V	0.343810	0.207357	0.224468	0.188640

Table. 3 : *Rapport de classification des algorithmes avec Word2Vec*

Après recherches, il semble que les embeddings produits par Word2Vec ou Doc2Vec sont souvent **mieux exploités avec des modèles complexes** comme des réseaux de neurones par exemple. En revanche, des classifieurs linéaires comme SVM ou Logistic Regression fonctionnent mieux avec des représentations simples et discriminantes comme celles produites par TF-IDF. Il n’était donc peut-être pas pertinent d’essayer d’utiliser des word embeddings.

5 Résultats et conclusion

Les meilleurs résultats sont obtenus en combinant une vectorisation tf-idf prenant en compte le contexte avec des ngrams et un SVM comme algorithme de classification.

Nous obtenons une accuracy de 0.722 et une F-mesure de 0.718.

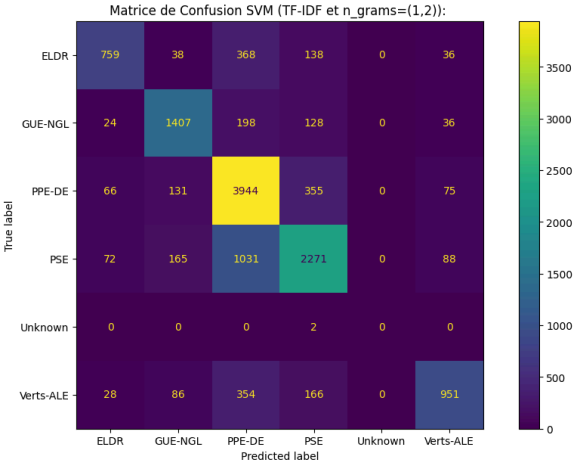

***Fig. 2** : Matrice de confusion de la meilleure combinaison : SVM (TF-IDF et n_grams=(1,2))*

Les paramètres de l'algorithme offrant ces résultats sont les suivants : *C*: 1, *class_weight*: None, *dual*: True, *max_iter*: 700.

Nous pouvons conclure qu'aujourd'hui nous avons librement accès à des outils faciles à utiliser permettant d'obtenir de meilleurs résultats pour la tâche de classification par parti politique d'interventions au parlement européen.

En effet, si en 2009 la meilleure f-mesure obtenue était d'environ 0.33, les outils aujourd'hui disponibles permettant d'atteindre un score de 0.718.

Notre étude s'est limitée à l'implémentation d'algorithmes relativement peu complexes mais il serait intéressant d'étudier les performances de différents réseaux neuronaux sur cette même tâche.

Références

Dominic Forest avec la collaboration de Astrid van Hoeydonck, Danny Létourneau et Martin Bélanger. 2009. *Impacts de la variation du nombre de traits discriminants sur la catégorisation des documents.*

C. Grouin, B. Arnulphy, J-B. Berthelin, S. El Ayari, A. García-Fernandez, A. Grappy, M. Hurault - Plantet, P. Paroubek, I. Robba et P. Zweigenbaum. 2009. *Présentation de l'édition 2009 du DÉfi Fouille de Textes (DEFT'09).*

Documentation Naive Bayes scikit-learn https://scikit-learn.org/stable/modules/naive_bayes.html

Documentation KNN scikit-learn https://scikit-learn.org/stable/modules/neighbors.html#classification

Corpus d'apprentissage : https://deft.lisn.upsaclay.fr/corpus/deft09_parlement_appr.xml.tar.gz

Corpus de test : https://deft.lisn.upsaclay.fr/corpus/deft09_parlement_test.xml.tar.gz

Corpus de référence : https://deft.lisn.upsaclay.fr/corpus/deft09_parlement_ref.tar.gz

Site du DEFT (Défi Fouille de Textes) : https://deft.lisn.upsaclay.fr/

Page du DEFT 2009 : https://deft.lisn.upsaclay.fr/actes/2009/