Une nouvelle perspective dans la résolution du défi de classification Analyse de l'appartenance politique des orateurs du Défi Fouille de Texte (DEFT)

Mikhail Biriuchinskii, Alina Myasnikova, Luísa Batista

Université Paris X, TAL M2

Abstract

Cet article présente une étude réalisée dans le cadre du cours d'Apprentissage Artificiel pour les étudiants suivant la formation en Traitement Automatique des Langues. L'objectif de ce travail est de réaliser la tâche de la classification et trouver le meilleur algorithme. Le code, ainsi que les documents de supports peuvent être trouvés sur le répertoire Github dédié.

1 Introduction

La tâche 3 du Défi Fouille de Texte (DEFT) de l'année 2009 se concentre sur une entreprise ardue : la détermination précise de l'appartenance politique des orateurs. Dans un contexte parlementaire où l'expression des opinions est supposée être claire et distincte, il est fascinant de constater les résultats mitigés obtenus dans cette tentative d'assignation des bonnes affiliations partisanes aux intervenants.

Lors de l'édition originale de ce défi, l'équipe de Montréal s'est distinguée en parvenant à compléter cette tâche. En utilisant l'algorithme des k plus proches voisins, les chercheurs ont obtenu les meilleurs résultats en termes de f-mesure, atteignant 33,90%.

Ce défi, désormais datant de plus de dix ans, a vu l'évolution des outils et le développement de nouvelles méthodes. Notre initiative de reprendre cette tâche vise à explorer ces avancées, cherchant à surpasser les scores antérieurs en utilisant les méthodologies actuelles et émergentes.

2 Observations sur le corpus

Le corpus sélectionné pour cette étude présente une diversité d'interventions parlementaires représentatives des différents partis politiques. Les extraits reflètent des prises de position variées.

Cependant, ce qui complique la tâche d'attribution politique, c'est la nature même des partis, qui affichent une large gamme d'orientations idéologiques, allant du centrisme libéral au progressisme écologiste, en passant par le centre droit et la gauche unitaire. Même en réalisant une étude de fréquences avec le module 'WordCloud' (Appendix 1), il s'avère évident que les interventions ne soient pas si variées, car ils tous utilisent les mêmes mots, et abordent les sujets sujets tels que l'aide humanitaire, les politiques agricoles, les budgets alloués, la problématique du VIH/sida et les migrations.

3 Préparation des données

Face à un corpus initial comprenant 19 370 textes pour chaque langue dans l'ensemble d'entraînement, ainsi que 12 917 textes dans l'ensemble de test, stockés dans des fichiers XML distincts, notre équipe a pris l'initiative de convertir ces données vers un format plus pratique et efficace pour le traitement ultérieur.

La décision a été prise d'adopter un format de dictionnaire Python pour chaque langue, une structure organisée de la manière suivante :

```
{
  'data_en': array([['texte', 'étiquette'], ..., ['texte', 'étiquette']], dtype='<...'),
  'data_fr': array([['texte', 'étiquette'], ..., ['texte', 'étiquette']], dtype='<...'),
```

'data_it': array([['texte', 'étiquette'], ..., ['texte', 'étiquette']], dtype='<...')
}

Nous avons converti nos données dans le format de dictionnaire Python pour chaque langue, puis utilisé Spacy pour éliminer les mots vides, lemmatiser le corpus et le mettre en minuscules. Ce processus a pris environ 15 heures pour assurer la qualité et la cohérence des données avant la classification.

4 SGD Classifieur

Le classifieur Stochastic Gradient Descent est un algorithme que nous avons choisi en premier à cause de sa capacité à traiter efficacement des ensembles de données volumineux, une caractéristique particulièrement avantageuse étant donné la taille importante de notre corpus, bien qu'inférieur à 100 000 textes.

Malgré sa moindre performance par rapport à d'autres algorithmes tels que la régression logistique, le SGD offre des avantages significatifs pour notre cas d'utilisation spécifique - création de classifieur multilabel multilingue.

4.1 Comparaison des paramètres

Les résultats de notre analyse ont montré que, de manière inattendue, les performances avec le corpus nettoyé étaient légèrement inférieures à celles du corpus brut. En particulier, l'utilisation de TfidfVectorizer a montré des performances légèrement meilleures avec le corpus nettoyé (figure 1).

Lors de l'examen des traits les plus discriminants identifiés par le TfidfVectorizer, une tendance s'est manifestée. La majorité des caractéristiques semblaient être associées à des valeurs aberrantes telles que "12h59", "1323" ou "0ostituzione", suggérant ainsi un possible surapprentissage.

En même temps, indépendamment de la nature du corpus, la méthode CountVectorizer a généré des résultats plus robustes et cohérents tout au long de nos expérimentations.

Type	Accuracy
Corpus brut CountVectorizer	0.76
Corpus brut TfidfVectorizer	0.61
Corpus nettoyé CountVectorizer	0.75
Corpus nettoyé TfidfVectorizer	0.62

Tableau 1.

4.2 Hyperparamètres

Après avoir identifié le type de corpus et la méthode de vectorisation les plus performants, nous avons entrepris de déterminer les hyperparamètres les plus efficaces pour l'algorithme. À cette fin, nous avons employé GridSearchCV de scikit-learn :

	Paramètres
CountVectoriser	max_features=11802
SGD	alpha=0.01, early_stopping=False, epsilon=0.1,eta0=0.01, max_iter=1586, loss="modified_huber", n_jobs=-1, learning_rate="adaptive", penalty='l2',shuffle=False

Tableau 2.

Après avoir appliqué ces paramètres, nous les avons comparés avec les paramètres par défaut du Corpus brut CountVectorizer (voir Annexe 2).

4.3 Conclusions pour le SGDClassifier

L'utilisation de la méthode GridSearch n'a pas permis de trouver les meilleurs paramètres pour améliorer le modèle utilisant CountVectorizer avec le corpus brut (0,76 de précision contre 0,58 en Accuracy). Malgré cela, même avec ses paramètres de base, cet algorithme surpasse celui choisi par l'équipe de Montréal (D. Forest et al.) lors du défi. En détail, la classe 'GUE-NGL' atteint une précision de 0,78, la rendant plus facilement identifiable, alors que 'PSE' obtient 0,73, étant moins performant dans ce contexte. Ces résultats présentent de légères différences par rapport à ceux de l'équipe de Montréal, avec des valeurs minimales de précision respectivement à 0,11 pour 'ELDR' et maximales à 0,63 pour 'PSE'. Ainsi, le classifieur SGD a surpassé l'algorithme des k plus proches voisins dans cette tâche spécifique.

5 Kernel Approximation

En cas de performances moins satisfaisantes, nous envisagions d'explorer l'option de kernel approximation. Bien que le SGD ait donné des résultats encourageants, notre désir d'expérimenter d'autres classifieurs nous a conduit

à considérer en premier lieu l'approche de kernel approximation, explorant ainsi ses variables dans un contexte non linéaire.

5.1 Techniques utilisées

Nous avons évalué le modèle avec des données prétraitées, que nous avons désignées sous le nom de 'clean', ainsi qu'avec des données non prétraitées, appelées 'dirty'. De plus, nous avons examiné les performances du modèle en utilisant les techniques de *CountVectorizer* et de *TfidfVectorizer*. Au cours de ces expérimentations, divers paramètres et attributs ont été ajustés dans le but d'optimiser les résultats, comme 'n_composantes' et 'gamma'. Par ailleurs, nous avons incorporé deux méthodes de kernel approximation, à savoir la fonction de base radiale (RBF) et la méthode de Nystroem, pour élargir davantage les perspectives de notre analyse.

5.2 Paramètrage

Nous avons opté pour les paramètres que la machine pouvait prendre en charge efficacement. Nous avions précédemment expérimenté le modèle de Kernel avec des données différentes, constatant que la configuration d'un attribut "n_components" plus élevé (comme à 10 000) pouvait potentiellement améliorer les scores, car il détermine le nombre de composants dans lesquels les caractéristiques sont transformées. Cependant, à chaque tentative avec nos propres données, la machine rencontrait des limitations. Par conséquent, nous avons pris la décision de maintenir la valeur de cet attribut à 1 000, assurant ainsi une performance stable et réussie du noyau, malgré la limitation observée.

5.3 Analyse

En ce qui concerne la disparité des résultats entre les données prétraitées et non prétraitées, elle s'est avérée minime, ne démontrant pas de différence considérable. Cependant, une distinction significative de performances a été observée entre l'utilisation de CountVectorizer et de TfidfVectorizer. Le tableau suivant (Tableau 3) expose cette expressivité.

Expérience	**Technique**	**Accuracy**	**F-score**
Kernel Approximation RBF (clean)	CountVectorizer	0.3151	0.2325
Kernel Approximation RBF (clean)	TfidfVectorizer	0.3776	0.2975
Kernel Approximation Nystroem (clean)	CountVectorizer	0.2888	0.1397
Kernel Approximation Nystroem (clean)	TfidfVectorizer	0.4158	0.3745

Tableau 3.

Cette variation entre les vectoriseurs s'est maintenue de manière notable dans les deux méthodes de kernel approximation, que ce soit avec la fonction de base radiale (RBF) ou la méthode de Nystroem.

6 Arbres de décision

De manière similaire à ce que nous avons fait pour kernel approximation, nous avons testé les arbres de décision avec les données nettoyées et les données non traitées. Cependant, nous n'avons utilisé que CountVectorizer, ce qui nous laisse penser que l'utilisation de TfidfVectorizer pourrait entraîner des scores encore plus satisfaisants.

6.1 Résultats

À notre grande surprise, même avec l'utilisation exclusive de CountVectorizer, nous avons obtenu des résultats très satisfaisants, avec une précision et un rappel très équilibrés, conduisant à une accuracy assez élevée de 0,74, avec les données nettoyées et non nettoyées. De plus, l'utilisation des arbres de décision ne nécessite pas le réglage minutieux de paramètres et de multiples essais. Cela nous a rendu le processus bien plus simple et rapide, et, étonnamment, avec des résultats nettement meilleurs.

7 SVM

Le SVM a été choisi en raison de ses propriétés qui le rendent particulièrement adapté à cette tâche.

1. **Haute dimensionnalité des données textuelles :** Les discours parlementaires sont représentés comme des vecteurs de termes, conduisant à des espaces de caractéristiques de grande dimension. Le SVM est performant dans ces situations.
2. **Capacité à gérer des espaces non linéaires :** Les textes sont difficiles à différencier. Notre hypothèse est telle qu'un modèle linéaire ne sera pas efficace dans ce contexte. Le SVM a un paramètre « kernel » qui permet de choisir un noyau polynomial, rbf ou sigmoïde au lieu de celui linéaire.
3. **Robustesse malgré les données peu volumineuses :** Même avec des ensembles de données où le nombre de caractéristiques est plus élevé que le nombre d'échantillons, le SVM demeure robuste. Cette caractéristique est pertinente, car notre corpus entier dépasse à peine 30 000 échantillons.

7.1 Expériences

En raison de capacités limitées de nos machines, il a été décidé de mener les expériences sur le corpus anglais uniquement. Pour cette partie, le prétraitement de données a été simplifié (NLTK, au lieu de SpaCy, listes de tokens au lieu de dictionnaires), car les ressources de la machine n'étaient pas suffisantes pour accomplir la tâche autrement. Vu que les corpus sont parallèles, il n'y a pas de raison de croire que les résultats dans les deux autres langues auraient une différence significative.

7.2 L'hypothèse initiale

Les meilleurs résultats peuvent être obtenus avec un corpus prétraité (lowercasing, tokenisation, suppression des mots vides, stemming – le tout avec NLTK, car cela nécessite moins de ressources que SpaCy), vectorisé avec TfIdfVectorizer (car il prend en compte l'importance de chaque terme contrairement à CountVectorizer) et avec un kernel rbf ou polynomial. Nous avons mené 5 expériences.

Numéro de l'expérience	**Corpus brut ou nettoyé**	**Vectorizer**	**Kernel**
1	nettoyé	TfIdf	RBF
2	nettoyé	TfIdf	Polynomial
3	nettoyé	Count	RBF
4	nettoyé	TfIdf	Sigmoïde
5	brut	TfIdf	Polynomial

Tableau 4.

À la suite des tests nous pouvons constater que les meilleurs résultats ont été obtenus au cours de la deuxième expérience, comme présumé. Nous avons testé le même modèle sur un corpus brut et les scores ont été inférieurs à ceux avec un corpus nettoyé. Nous avons également essayé de diviser nous-mêmes le corpus en train et test pour voir si les résultats dépendent des proportions, mais nous n'avons remarqué aucune différence significative.

Le modèle SVM propose également d'autres paramètres qui peuvent être ajustés pour améliorer la catégorisation, notamment gamma qui contrôle la largeur de la frontière de décision, la marge (C), le degré pour le kernel polynomial (une des pistes dans notre cas), les poids pour les classes déséquilibrées et une tolérance pour la convergence du critère d'arrêt. Le module scikit-learn dispose d'une fonction GridSearchCV qui teste les différents paramètres en mesurant leurs scores pour voir quelles valeurs sont les plus adaptées aux données, mais il n'était pas possible de mener à bien cette expérience sur nos machines en raison de leurs capacités limitées.

8 Conclusion

Après une série d'expériences visant à déterminer le modèle le plus efficace, nos analyses ont révélé que les algorithmes les plus adaptés à la tâche sont : SGD, SVM et arbres de décision. L'utilisation de ces modèles sur le corpus prétraité permet d'atteindre l'exactitude au niveau de 0,74 et plus. Les options à envisager pour améliorer la qualité de la catégorisation sont : le changement / l'ajout d'hyperparamètres, l'étude plus fine des traits discriminants et l'augmentation de la taille du corpus.

References

Collectif. 2009. DEFT2009, Actes du cinquième DÉfi Fouille de Textes. Proceedings of the Fifth DEFT Workshop, édition numérique.

Shihab Elbagir, Jing Yang. 2018. *Sentiment Analysis of Twitter Data Using Machine Learning Techniques and Scikit-learn*. ACAI '18: Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence, Article No. 57, Pages 1–5. DOI: 10.1145/3302425.3302492.

A. Yousaf et al. 2021. *Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD)*. IEEE Access, vol. 9, pp. 6286-6295, 2021, doi: 10.1109/ACCESS.2020.3047831.

M. Cardaioli, P. Kaliyar, P. Capuozzo, M. Conti, G. Sartori and M. Monaro. 2020. *Predicting Twitter Users' Political Orientation: An Application to the Italian Political Scenario*. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), The Hague, Netherlands, 2020, pp. 159-165, doi: 10.1109/ASONAM49781.2020.9381470.

A. Patle and D. S. Chouhan. 2013. *SVM kernel functions for classification*. International Conference on Advances in Technology and Engineering (ICATE), Mumbai, India, 2013, pp. 1-9, doi: 10.1109/ICAdTE.2013.6524743

Roman, I., Santana, R., Mendiburu, A. et al. 2021. *In-depth analysis of SVM kernel learning and its components.* Neural Comput & Applic 33, 6575–6594. https://doi.org/10.1007/s00521-020-05419-z

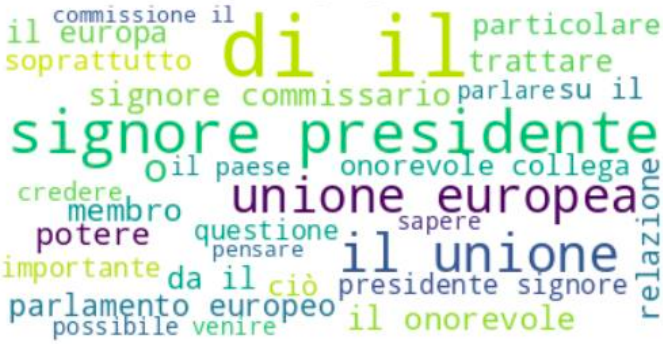
Kernel Approximation. ScikitLearn. Disponibile sur : https://scikit-learn.org/stable/modules/kernel_approximation.html (Consulté le : 26 décembre 2023).

A Appendice

[data_train_en]

[data_train_fr]

[data_train_it]



B Appendice

Paramètres par défaut, CountVect.

	precision	recall	f1-score	support
ELDR	0.74	0.67	0.70	4017
GUE-NGL	0.78	0.80	0.79	5379
PPE-DE	0.77	0.80	0.78	13713
PSE	0.73	0.74	0.73	10881
Verts-ALE	0.74	0.69	0.72	4755
accuracy			0.76	38745
macro avg	0.75	0.74	0.75	38745
weighted avg	0.75	0.76	0.75	38745

Paramètres de SearchGrid

	precision	recall	f1-score	support
ELDR	0.62	0.35	0.45	4017
GUE-NGL	0.57	0.70	0.63	5379
PPE-DE	0.59	0.70	0.64	13713
PSE	0.55	0.54	0.55	10881
Verts-ALE	0.60	0.37	0.46	4755
accuracy			0.58	38745
macro avg	0.59	0.53	0.54	38745
weighted avg	0.58	0.58	0.57	38745