Détection d'auteur dans un corpus de discours

Antoine Cadiou, Vincent Jouve, Jean Soler

Master 1 Données,Apprentissage et Connaissances @ Sorbonne Université
UE RITAL 2019-2020, Encadrant : Vincent Guigue

http://dac.lip6.fr/master/enseignement/rital/

Résumé- Compte-rendu du projet de l'UE « RITAL » du Master 1 DAC. Etant donné un corpus de discours donnés par les présidents Jacques Chirac et François Mitterrand, notre objectif est de trouver un pré-traitement, un post-traitement et un modèle de classification qui permet de dire le plus précisément possible quel est le président qui a donné un discours donné.

I. INTRODUCTION

La détection d'auteurs est un sujet important du traitement automatique du langage, de nos jours il peut être utile dans de nombreux domaines, par exemple pour détecter des « fake news » ou de l'usurpation d'identité. Dans ce projet, nous avons travaillé sur la différenciation des discours donnés par Jacques Chirac et François Mitterrand. Nous avons mené une campagne d'expériences afin de trouver les différents pré/post-traitements qui correspondaient le mieux à nos données d'apprentissages et de tests. Aussi, nous avons comparé plusieurs approches de classification (à la fois supervisées et non-supervisées).

II. PRÉ-TRAITEMENTS DES DONNÉES

Les données d'apprentissages sont fortement déséquilibrées (~80% Chirac / 20% Mitterrand), nous avons équilibré nos exemples d'apprentissage en récupérant l'intégralité des 'Mitterrand' (~7500) et en prenant autant de Chirac. On se retrouve ainsi avec un dataset équilibré d'environ 15000 exemples, ce qui est idéal pour entraîner le classifieur (ainsi, la précision des prédictions aura un sens à évaluer, là où, si on avait laissé le dataset déséquilibré, avec un classifieur naïf qui renvoie toujours 'Chirac' on aurait eu une précision de 80% en cross-validation, mais cela n'aurait pas été pertinent du tout car on ne connaît pas la proportion des classes dans le dataset de test).

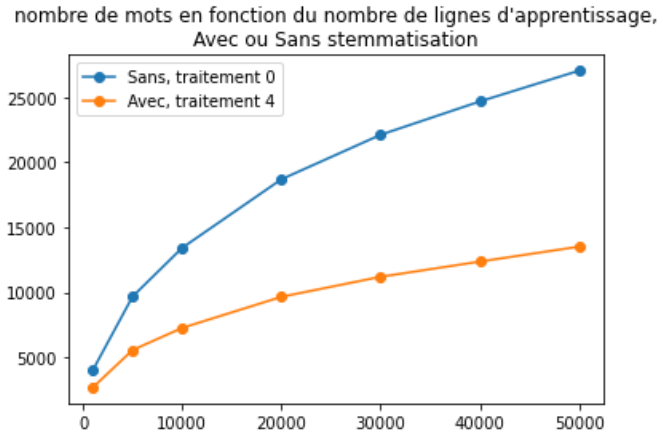
Nous avons mis au point une fonction de traitement des données qui permet, au choix, de :

- Mettre tout en minuscule ou laisser le texte tel qu'il est, avec les majuscules
- Enlever les stopwords grâce à la liste de la librairie NLTK
- Stemmatiser les mots, c'est-à-dire réduire les mots à leur racine commune. (La racine n'étant pas une entrée du dictionnaire, à la différence de la Lemmatisation).

Nous avons donc défini 6 traitements possibles que l'on va comparer pour choisir le plus pertinent :

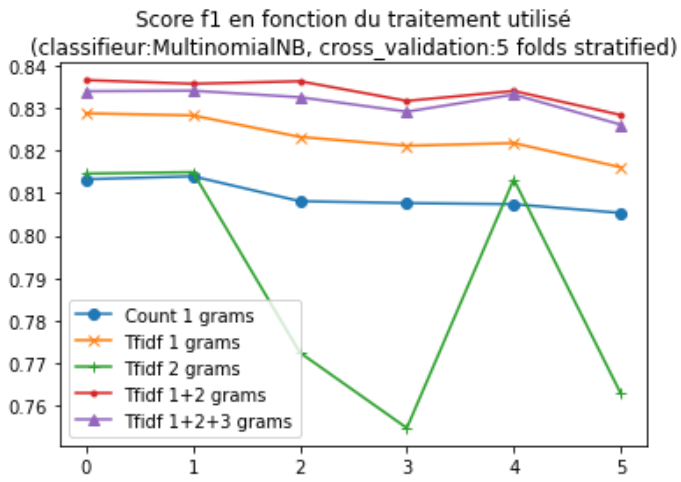
Traitement N°	Mettre en minuscules	Enlever les stopwords	Stemmatisation
0	Non	Non	Non
1	Oui	Non	Non
2	Non	Oui	Non
3	Oui	Oui	Non
4	Oui	Non	Oui
5	Oui	Oui	Oui

En fonction du traitement appliqué et de la taille d'apprentissage (nombre de lignes utilisées dans le fichier d'apprentissage, sur 57000 environ), nous observons que le nombre de mots diminue de façon conséquente (presque d'un facteur 2) si on applique une stemmatisation ou non :



(Graphe 1)

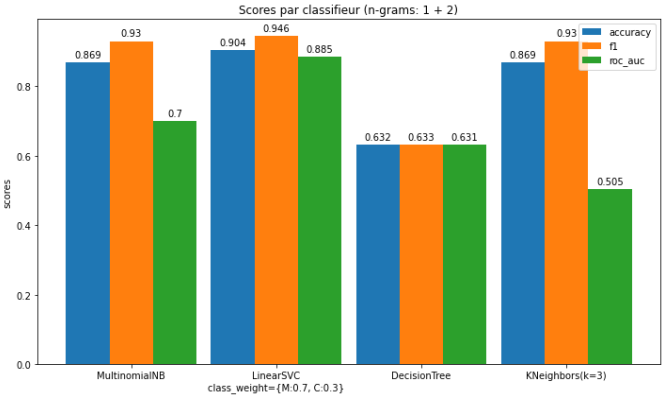
Une fois que nous avons appliqué un traitement, il était nécessaire de réfléchir à la structure de données à utiliser.
Faire des sparse matrix (grâce à la librairie SKLEARN) était un choix judicieux car dans une matrice de comptage des mots par exemple, il y aurait eu beaucoup de zéros qui auraient pris de la place en mémoire, la sparse matrix convient donc ici.
Nous nous sommes demandés s'il était préférable de faire un simple comptage ou bien un TFIDF, pour ce faire nous avons fait quelques tests avec un classifieur MultinomialNB.



Ce graphe montre que, pour un classifieur MultinomialNB, le traitement TFIDF donne de meilleurs scores f1 que le traitement avec un simple comptage des occurrences des mots. Ce graphe met également légèrement en évidence que les résultats sont meilleurs lorsqu'on laisse les stopwords (de manière générale, lorsqu'on les retire, nos résultats sont moins bons). Cependant le fait de stemmatiser (traitement 4) ou non (traitements 0 et 1) apporte peu de différences. Mais il est tout de même préférable d'utiliser la stemmatisation car il y a nettement moins de mots dans la sparse matrix (c.f graphe 1).
Ici, nous avons aussi analysé les façons de séparer les mots pour faire notre sac de mots. Ainsi, nous avons voulu voir pour quels n-grams (en faisant varier les n) nous obtenions les meilleurs résultats. En conclusion, le pré-traitement que nous appliquerons pour la suite est le traitement 4 + les uni-grams + les bi-grams pour obtenir des résultats à la fois efficaces et rapides.

III. Classifications, Entrainments et predictions

Nous avons expérimenté plusieurs types de classifieurs supervisés :

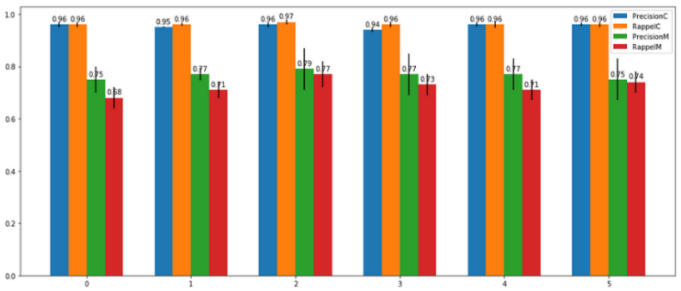


Nous observons que le classifieur LinearSVC obtient de bons résultats, notamment pour le score roc_auc qui s'améliore significativement par rapport à un MultinomialNB. La précision du LinearSVC s'élève à 90,4% sur un dataset d'apprentissage équilibré. Ces scores ont été obtenus en procédant par cross-validation sur le dataset complet d'entraînements(les classes sont complètement déséquilibrées) avec 5 folds dans lesquels les proportions des classes sont conservées (stratified).

Nous avons également essayé une approche non-supervisée du problème. En effet, si l'on suppose qu'il existe 2 clusters qui, dans l'idéal, représentent les 2 auteurs, alors il nous suffirait de regarder de quel centroïde notre exemple est le plus proche, et nous pourrions procéder par classification de la sorte.
Ainsi, nous avons implémenté un K-moyenne, mais les résultats sont très variables et sont difficiles à évaluer. En effet, d'une exécution à une autre les clusters sont très différents (dû à des convergences différentes selon les initialisations aléatoires).
Après entraînement de l'algorithme, on s'aperçoit qu'il a classé environ 12000 exemples dans le même cluster, ce qui n'est pas le résultat recherché car, sur 15000 exemples, on espère obtenir 2 clusters de tailles 7500 chacun.
On a alors conclu que l'utilisation d'un classifieurs supervisé était à préférer.

IV. Modèle de Markov caché

Nous avons également fait un modèle basé sur une chaine de Markov cachée. Comme il n'est pas possible de prendre les phrases comme une observation en elle-même puisque le nombre de phrases possible est beaucoup trop grand et il n'y a pas assez de données. Donc pour que le modèle fonctionne on découpe les phrases en mot et on garde un indicateur pour pouvoir reconstruire la phrase à la fin, ici on prend (idbloc,idligne). Pour chaque mot on lui attribue la même étiquette que la phrase dont il provient et nous considérons un bloc comme une séquence. Ainsi nous avons une base d'apprentissage qui est constitué de plusieurs séquences, pour chaque séquence nous avons les mots et leur étiquette associé. Pour apprendre le modèle on considère comme états cachés les présidents (ici Chirac et Mitterrand). Puis pour la prédiction il suffit de donner une séquence au modèle et d'effectuer un décodage, on se retrouve avec les étiquettes sur les mots. Pour donner un résultat pour la phrase on reconstruit la phrase avec l'indicateur, puis on compte combien de prédiction il y a pour Chirac et combien pour Mitterrand puis on attribue à la phrase la classe Chirac s'il est présent en majorité ou Mitterrand si c'est lui qui obtient le plus de mot. Ce qui est bien avec un HMM c'est qu'il n'y a pas de problème de lissage à la fin car le modèle donne le résultat pour le bloc et comme il est peu probable d'avoir un transition Chirac Mitterrand ou Mitterrand Chirac. On ne prédit jamais CMCMC par exemple.
On obtient par cross-validation (apprentissage sur 90% et test sur 10%) les scores de précision rappel suivant, sur chaque classe:



On peut voir qu'il y a peu de différence selon le traitement. L'erreur correspond à l'écart-type sur les 10 tests. Les résultats sont plutôt corrects la précision et le rappel sont proches de 1. On voit donc que le modèle n'a pas pris la facilité de donner toujours la même classe sinon la métrique rappelM serait à 0. Ici on peut se dire que le meilleur traitement est le 1 donc juste de passer en minuscule car les valeurs sont élevées et l'écart-type est petit.

V. POST-TRAITEMENTS

Nous observons que les données sont sous formes de blocs. Il est intéressant d'utiliser cette information car on peut donc en déduire qu'une personne qui parle ne dis pas qu'une ligne mais plusieurs à la suite. Donc on fait un post traitement qui agit comme un K plus proches voisins dans l'idée de lisser les décisions probablement fausses et donc de créer des blocs réguliers qui ont plus de sens. Nous avons pris un k=9 Sachant que Mitterrand apparaît bien moins souvent que Chirac, nous avons mis un coefficient de 2.3 pour le nombre d'occurrences des lignes de Mitterrand.

VI. CONCLUSION ET FUTURS TRAVAUX

Concernant le pré-traitement, nous pouvons dire que les stopwords ont leur importance dans les discours, aussi il est intéressant d'étudier les unigrams et les bigrams ensembles pour de meilleurs résultats. La stemmatisation et la suppression des majuscules n'apportent pas une réelle différence dans les résultats mais on diminue significativement le nombre de mots après stemmatisation.
Concernant les méthodes de classifications, après avoir essayé de la classification supervisée et non-supervisée, nous préférerons le modèle de Markov caché qui est une méthode très robuste et qui nous a permis d'atteindre 93% de bonnes classifications dans le rendu du projet.
Avec le modèle de Markov, il est inutile d'appliquer le post-traitement car on a déjà des blocs lisses. Le post-traitement est utile dans le cas d'une classification avec le LinearSVC par exemple, on obtient alors des résultats bien moins probants qu'avec le modèle de Markov (75% de bonnes classifications dans le rendu du projet).

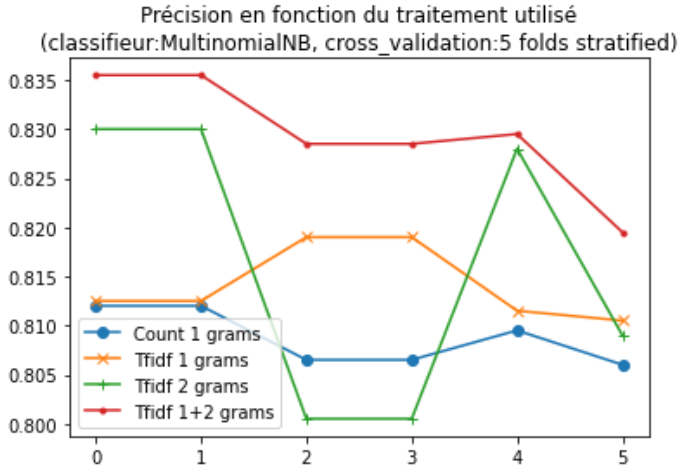
Il pourrait être judicieux dans de futures expériences de modifier la fonction de calcul des distances dans le K-moyenne pour obtenir de résultats plus convaincants en classification non-supervisée. Aussi, probablement qu'en faisant du Part-Of-Speech tagging dans le prétraitement on pourrait sélectionner des n-grams plus pertinents ?

Détection des sentiments dans des revues de films

Nous sommes partis des mêmes bases que pour la détection des auteurs concernant le pré-traitement des données mais cette fois-ci, les données sont en anglais et donc cela a été pris en compte pour la stemmatisation et pour les stopwords. Aussi, les données d'apprentissage sont déjà équilibrées donc on base nos tests sur l'ensemble des données disponibles (les 1000 exemples positifs et le 1000 exemples négatifs).
Le but ici est de détecter le sentiment d'une revue, de façon binaire : positif ou négatif.

I. PRÉ-TRAITEMENTS DES DONNÉES

Nous avons donc le même tableau des traitements que la problématique précédente, et voici les premiers résultats sur la précision d'un algorithme MultinomialNB en fonction des différents pré-traitements :

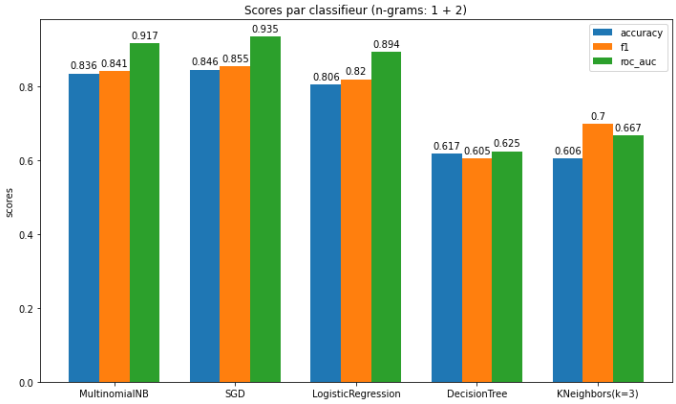


(Graphe 1)

On remarque ici que les traitement 0 et 1 sont meilleurs que les autres. Aussi, la sélection des uni-grams et des bi-grams dans notre sac de mots a un impact important et les résultats sont nettement plus élevés.
Nous allons donc dans la suite des expériences sélectionner le traitement 0 + les uni-grams + les bi-grams afin d'obtenir la meilleure précision possible.

II. CLASSIFICATIONS SUPERVISÉE

Nous avons expérimenté plusieurs types de classifieurs supervisés
(c.f graphe 2)
Nous observons que le classifieur SGD obtient les meilleurs résultats. La précision du SGD s'élève à 84,6% sur un dataset d'apprentissage équilibré. Ces scores ont été obtenus en procédant par cross-validation, avec 5 folds dans lesquels les proportions des classes sont conservées (stratified).



(Graphe 2)

III. CLASSIFICATION NON-SUPERVISÉE

Nous avons voulu faire un K-moyenne en espérant obtenir 2 clusters représentant les 2 polarités de sentiments. Les conclusions sont les mêmes que pour la détection d'auteur. Cependant, bien que les résultats n'étaient pas convaincants, nous avons voulu avoir un retour sur les mots dominants dans chaque cluster :

Nous observons que la plupart des mots prédominants sont les mêmes dans les 2 clusters (ex : 'film', 'movie', 'character',…) ce qui est normal car ils font partis du lexique employé dans

les revues. Cependant les 2 clusters contiennent des termes intéressants et discriminants (ils sont encadrés dans les nuages de mots). Mais leurs poids ne sont pas assez pris en compte lors de la catégorisation de nouveaux termes. Pour continuer dans cette voie, il faudrait faire de l'extraction de candidats-termes qui définissent le mieux chacune des deux polarités.

IV. CONCLUSION

La meilleure solution que nous pouvons apporter est à nouveau avec de la classification supervisée, en utilisant un classifieur SGD, il nous permet d'obtenir un taux de précision de 85%.