RAPPORT TAL
Projet 2020-2021

Classification de documents

Etudiant 1 : GIANG Cécile
N° étudiant : 3530406

Etudiant 2 : KHALFAT Célina
N° étudiant : 28716860

TABLE DES MATIÈRES

CHIRAC OU MITTERRAND

Introduction

Nous disposons d'une base de 54413 phrases tirées de discours de Chirac et de Mitterrand. Cette base est étiquetée par les labels 'C' ou 'M', indiquant le président ayant prononcé chaque phrase. L'objectif est de mettre en place un classifieur capable de labelliser une autre base de phrases également prononcées par Chirac ou Mitterrand (27162 phrases).

Analyse préliminaire

Nous commençons notre étude par une analyse préliminaire de notre base d'apprentissage.
Nous remarquons tout d'abord que la base est fortement déséquilibrée, puisque 49890 des phrases proviennent de Chirac, contre 7523 phrases pour Mitterrand (soit 86.9% données étiquetées 'C' contre 13.1% de données étiquetées 'M'). Nous craignons qu'un tel déséquilibre pousse notre classifieur à ne prédire que la classe majoritaire ('C') ; il faudra donc soit modifier un peu notre base pour la balancer un peu mieux, soit modifier notre modèle afin qu'il puisse s'adapter à ce déséquilibre.

Un wordcloud sur les phrases propres à Chirac et à Mitterrand ne nous permettent pas de différencier à vue d'œil les discours des deux présidents, même en supprimant les stopwords. Voici les wordclouds pour Chirac à gauche, pour Mitterrand à droite :

Nous nous proposons donc de refaire un wordcloud en prenant cette fois-ci en compte les odds ratios afin de déterminer les mots les plus discriminants pour chaque classe.

Nous nous apercevons ainsi par exemple que Chirac mentionne beaucoup les mots ‘France’, ‘économie’, ‘social’, ‘engagement’, ‘développement’ et leurs dérivées tandis que Mitterrand parle plus de ‘guerre’, ‘Europe’, ‘industrialisation’, etc…

Il est aussi intéressant de noter que Chirac, contrairement à Mitterrand, utilise beaucoup des mots incitant la confiance et cohésion. En particulier parmi les mots les plus discriminants pour lui, on retrouve ‘solidarité’, ‘respect’, ‘valeur’, ‘coeur’, ‘amitié’, ‘ensemble’ tandis que cette tendance n’est pas du tout présente chez Mitterrand.

Pré-processing de la base d’apprentissage

Cette phase est primordiale afin de réduire la dimensionnalité de notre base lorsqu’elle sera mise sous forme liste de vecteurs sur le vocabulaire.

Les pré-traitements que nous considérons sont notamment :

- La mise en minuscule
- La normalisation afin de supprimer les accents et les caractères non normalisés
- La suppression des chiffres et de la ponctuation
- Le stemming
- L’élimination des stopwords

Ces pré-traitements s’effectuent par paramétrage (True ou False) d’une fonction dédiée au processing du corpus. Il s’agira pour nous de retrouver par grid search la/les combinaisons optimisant la métrique choisie sur la base d’apprentissage en validation croisée. Sachant que le modèle de machine learning choisi, le paramètre de régularisation et le type de représentation des phrases seront aussi des paramètres à optimiser, nous devons déjà faire une pré-sélection sur ces pré-traitement afin de réduire le nombre de combinaisons à tester. Nous choisissons de fixer la suppression des chiffres et de la ponctuation à True : les discours ont été prononcés mais dans la base, les phrases ont été écrites de manière neutre (la ponctuation n’apporte donc pas d’information supplémentaire).

Modèles de machine learning

Nous choisissons de tester trois modèles de machine learning : un SVM linéaire (LinearSVC), un modèle Naive Bayes (MultinomialNB) et une régression logistique (LogisticRegression). En particulier, sklearn propose également un paramètre de régularisation C pour les modèles de régression linéaire et SVM que nous nous attacherons à optimiser.

Choix de la métrique à optimiser

Par défaut, les scores calculés pour un modèle de sklearn sont des taux de bonne classification (accuracy). Cette métrique n'est cependant pas du tout adaptée à notre cas, puisqu'un classifieur ne prédisant que la classe majoritaire (qui représente ici plus de 80% de nos données) aura une accuracy supérieure à 0.8… Nous choisissons donc d'optimiser le score F1 sur la classe minoritaire (Mitterrand).

Première campagne d'expériences

Nous effectuons un premier grid search sur les pré-traitement ainsi que sur les modèle de machine learning décrits plus hauts. Le type de représentation des phrases (vectorisation par comptage ou par tf-idf) ansi que les types de n-grams sont également des hyperparamètres à prendre en compte.

	Language	Line	Lower	Remove digit	Remove punctuation	Remove stopwords	Normalize	Stemming	N-gram range	Vectorizer	Model	C	Accuracy score	F1 score	ROC-AUC score	Average score
755	french	None	False	True	True	False	True	True	(1, 2)	Counter	Logistic Regression	70	0.905544	0.574113	0.871335	0.783664
1945	french	None	True	True	True	False	False	True	(1, 2)	Counter	Logistic Regression	90	0.904586	0.572971	0.871630	0.783063
1746	french	None	True	True	True	False	False	False	(1, 2)	Counter	Logistic Regression	80	0.906589	0.572763	0.872183	0.783845
353	french	None	False	True	True	False	False	True	(1, 2)	Counter	Logistic Regression	10	0.907129	0.571728	0.877641	0.785499
360	french	None	False	True	True	False	False	True	(1, 2)	Counter	Logistic Regression	80	0.904847	0.571694	0.872273	0.782938
1938	french	None	True	True	True	False	False	True	(1, 2)	Counter	Logistic Regression	20	0.906519	0.571593	0.876695	0.784936

Les scores f1 obtenus sont assez faibles. Les meilleurs scores obtenus tournent autour de 0.57, ce qui est assez mauvais. Nous relevons d'abord les paramètres permettant d'obtenir les meilleures performances. Il est à noter que nous ne prenons pas directement la combinaison de paramètres donnant le meilleur score, mais étudions les 50 meilleures combinaisons pour convenir d'une agrégation de ces paramètres qui nous parait optimale. Ainsi par exemple, les meilleures combinaisons ont leurs paramètres stemming à True et no_stopwords à False, mais un grand nombre de combinaisons produisant des scores similaires (de l'ordre de 0.56-0.57) ont leurs paramètres stemming à False et no_stopwords à True. Nous choisissons de prendre ces derniers afin de réduire la dimensionnalité de nos matrices (suppression de stopwords).

Ainsi, nous décidons des paramètres suivants :

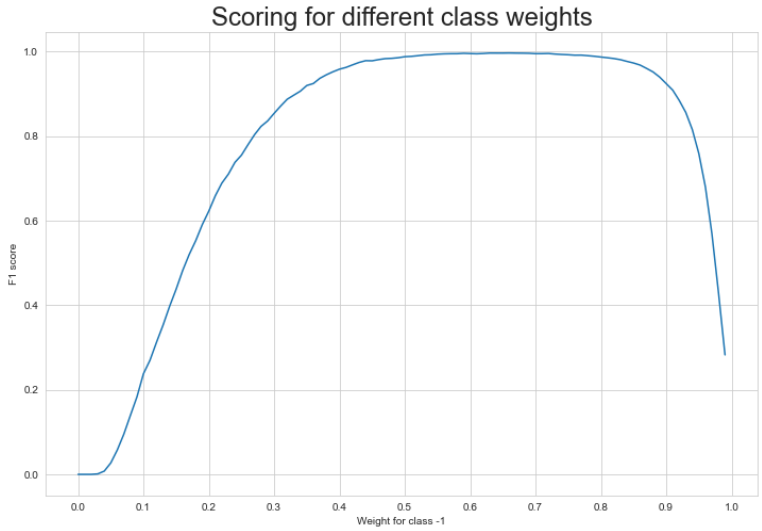
- Pré-traitements : pas de mise en minuscule, suppression des chiffres et de la ponctuation, pas de normalisation, pas de stemming, suppression des stopwords
- Représentation vectorielle : comptage, avec un n-gram range à (1,2) – prise en compte des unigrammes et bigrammes.
- Modèle : régression logistique avec C entre 10 et 100.

Deuxième campagne d'expériences

Comme nous l'avons précisé plus haut, les données sont fortement déséquilibrées et la majorité d'entre elles appartient à la classe Chirac. Nous risquons ainsi d'apprendre un classifieur ne prédisant presque que la classe majoritaire. Deux choix s'offrent à nous :

1) Sur-échantillonner la classe minoritaire, sous-échantilloner la classe minoritaire, ou faire un mélange des deux. Nos expériences n'ont pas été très concluantes, les résultats obtenus sont moins bons que sur la base d'origine
2) Changer la fonction de coût pour pénaliser plus fortement les mauvaises prédictions sur la classe minoritaire. Le module sklearn propose pour cela pour chacun de ses modèle un paramètre class_weight nous permettant d'attribuer un ratio de pénalité sur les différentes classes.

Nous lançons une deuxième campagne d'expériences afin de déterminer les poids idéaux, en fixant certains des paramètres déjà optimisés (les pré-traitements). Nous laissons la représentation vectorielle et le modèle comme hypermaramètres, et nous trouvons que mettre la classe -1 (Mitterrand) à ~0.6 et la classe 1 à ~0.4 nous donne les meilleurs résultats. Cette fois-ci la représentation vectorielle à préférer est le tf-idf, toujours avec un modèle de régression logistique et un C plus faible. Nous le fixons à 20.



TROISIEME CAMPAGNE D'EXPERIENCE

Nous avons observé précédemment que les phrases des présidents étaient rangées par blocs dans la base d'apprentissage. Nous faisons l'hypothèse que les données de test suivent la même tendance et nous nous proposons de procéder à un lissage des résultats afin d'obtenir également des blocs sur nos prédictions. C'est un choix justifiable puisque nos classifieurs auront tendance à moins bien prédire les phrases de la classe minoritaire, chaque prédiction -1 peut alors se révéler importante si elle n'est pas isolée. Nous procédons à trois lissages :

- Premier lissage : chaque prédiction devient le signe de la somme pondérée de son voisinage, en accordant un poids plus important aux voisins valant -1. Le nombre de voisins pris en compte est un hyperparamètre que nous fixons à 12 après optimisation par tests
- Deuxième lissage : cette fois-ci nous nous intéressons aux labels isolés (ie entourés par deux labels de l'autre classe), ou aux séquences de labels différentes (-1,1,-1,1,-1). Dans le premier cas nous fixons la classe du label isolé aux labels qui l'entourent, dans le deuxième cas nous prenons le label majoritaire dans le voisinage en conflit
- Troisième lissage : Nous remarquons un effet de bord sur les labels prédits en apprentissage : il arrive que les séquences de -1 soient plus courtes que prévues car les labels en bordure de séquence, bien qu'originellement prédites comme -1, ont été moyennées à 1. Nous proposons donc un dernier lissage qui consiste à regarder les voisins directs en début et en fin de séquences de -1, si leurs prédictions étaient originellement à -1 (avant lissage). Si oui on met à -1 les suites de 1 originellement à -1 dans le voisinage direct des bords de séquences de -1.

La liste des labels générés nous paraît bien lissée, le bruit observé est maintenant éliminé. Ce sont ces paramètres que nous avons utilisé afin de générer le fichier de labels 'preds_labels.txt'.

REVUES FILMS: POSITIVES OU NEGATIVES ?

Introduction

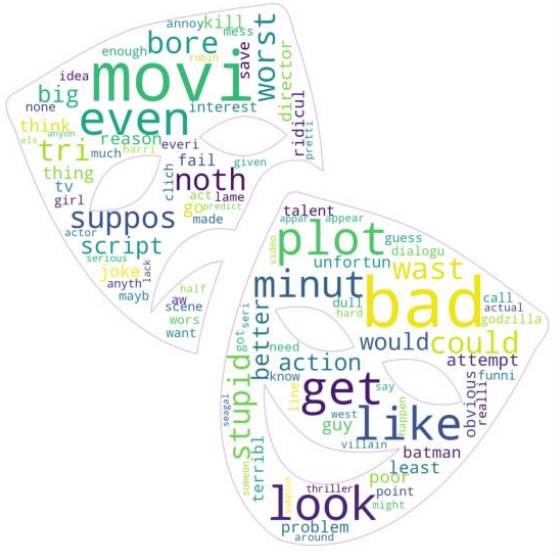
Nous disposons d'une liste de revues de films provenant de la base de données iMDB. Cette liste, composée de 1000 revues positives et de 1000 revues négatives, nous servira de base d'apprentissage. Elle est étiquetée par les labels 0 (avis négatif) ou 1 (avis positif). Comme pour la tâche précédente, notre objectif est de mettre en place un classifieur capable de labelliser un corpus d'avis composé de 25 000 revues.

Analyse préliminaire

Nous commençons notre étude par une analyse préliminaire de notre base d'apprentissage.
Nous remarquons d'abord que contrairement à la base de donnée présidentielle, la base d'apprentissage iMDB est parfaitement équilibrée. Nous n'aurons donc pas à nous soucier de balancer la base pour éviter un sous-apprentissage sur une classe minoritaire.

Comme sur les données présidentielles, un simple wordcloud sur chaque classe de revues ne permet pas de conclure grand-chose, même avec une vectorisation td-idf et en supprimant les stopwords. Plutôt qu'un vocabulaire spécifique à la polarité de la classe, nous retrouvons plutôt le champ lexical lié aux films ('character', 'movie', 'director', etc…).

Nous nous intéressons maintenant au wordcloud de chaque classe sur les odds ratios afin de déterminer les mots les plus discriminants pour chaque classe.



Nous nous apercevons ainsi par exemple que les mots qui permettent le mieux de distinguer la classe négative sont ceux liés à l'ennui ('bore', 'annoy'), 'bad', 'waste', 'nothing', etc… contre des mots à connotation positive comme 'great', 'beautiful', 'best' ou 'well'.
Il est également intéressant de noter que les films où l'on tue ('kill'), ceux centrés autour de 'godzilla' ou 'batman' sont assez largement détestés, alors que les films parlant d' 'alien', les films 'disney' ou les longs-métrages de 'spielberg' sont en général appréciés.

Pré-processing de la base d'apprentissage

Les pré-traitements que nous considérons sont notamment :

- Ajouter un marqueur quand on a une suite de ponctuations (' ????!!?', '.....'), ces marqueurs étant intéressants car permettant d'augmenter la polarité d'une revue
- Dupliquer les mots complètement en majuscule afin d'augmenter leur poids dans la revues ('SO GOOD !!' sera très clairement positif)
- Sélectionner seulement la première phrase, la dernière phrase ou l'intégralité d'une revue
- La mise en minuscule
- La normalisation afin de supprimer les accents et les caractères non normalisés
- La suppression des chiffres et de la ponctuation
- Le stemming
- L'élimination des stopwords

Afin de limiter le nombre de combinaisons à tester par GridSearch, nous choisissons de fixer la suppression des chiffres et l'élimination des stopwords à True.

Modèles de machine learning

Nous choisissons de tester trois modèles de machine learning : un SVM linéaire (LinearSVC), un modèle Naive Bayes (MultinomialNB) et une régression logistique (LogisticRegression). En particulier, sklearn propose également un paramètre de régularisation C pour les modèles de régression linéaire et SVM que nous nous attacherons à optimiser.

Choix de la métrique à optimiser

Comme nous n'avons pas affaire avec des données déséquilibrées, le score d'accuracy est un bon indicateur de qualité sur nos prédictions.

Campagne d'expériences

Nous effectuons un premier grid search sur les pré-traitement ainsi que sur les modèle de machine learning décrits plus hauts. Le type de représentation des phrases (vectorisation par comptage ou par tf-idf) ansi que les types de n-grams sont également des hyperparamètres à prendre en compte.

	Language	Line	Lower	Mark punctuation	Duplicate upper	Remove digit	Remove punctuation	Remove stopwords	Normalize	Stemming	N-gram range	Vectorizer	Model	C	Accuracy score
452	english	NaN	True	False	True	True	True	True	False	False	(1, 2)	Tf-Idf	Linear SVC	40	0.8640
874	english	NaN	False	True	True	True	True	True	False	False	(1, 1)	Tf-Idf	Linear SVC	80	0.8630
1162	english	NaN	False	True	True	True	True	True	True	False	(1, 1)	Tf-Idf	Logistic Regression	80	0.8620
513	english	NaN	True	False	True	True	True	True	False	False	(1, 1)	Tf-Idf	Logistic Regression	60	0.8620
884	english	NaN	False	True	True	True	True	True	False	False	(1, 2)	Tf-Idf	Linear SVC	40	0.8600
777	english	NaN	True	False	True	True	True	True	True	True	(1, 2)	Tf-Idf	Linear SVC	60	0.8600

Les scores d'accuracy obtenus en apprentissage sont assez bons. Les meilleurs scores obtenus tournent autour de 0.86 en cross validation. Comme pour les données présidentielles, nous relevons les paramètres permettant d'obtenir les meilleures performances. Il est à noter que nous ne prenons pas directement la combinaison de paramètres donnant le meilleur score, mais étudions les 50 meilleures combinaisons pour convenir d'une agrégation de ces paramètres qui nous parait optimale. Ainsi par exemple, les meilleures combinaisons ont leurs paramètres stemming à False, mais la plupart des meilleures combinaisons produisant des scores similaires (de l'ordre de 0.84-0.86) ont leur paramètres True.

Ainsi, nous décidons des paramètres suivants :

- Pré-traitements : mise en minuscule, suppression des chiffres et de la ponctuation, normalisation, stemming, suppression des stopwords, duplication des mots en majuscule, pas de marqueur sur les ponctuations multiples, sélection de l'intégralité du texte
- Représentation vectorielle : tf-idf, avec un n-gram range à (1,2) - prise en compte des unigrammes et bigrammes.
- Modèle : modèle SVM linéaire, avec C à 40

Ce sont ces paramètres que nous avons utilisé afin de générer le fichier de labels 'imdb_labels.txt'.