UNIVERSITÉ DE MONTPELLIER
FACULTÉ DES SCIENCES

Classification de documents d'opinions

Méthodes des sciences de la donnée

Groupe de travail :

Baptiste Darnala
Eliott Duverger
Thomas Di Giovanni
Pierre Van Iseghem

Année universitaire : 2018/2019

1 Introduction

L'objectif de ce projet consiste au triage de divers documents d'opinions par le biais de classifieurs. Le corpus à analyser est composé de 10000 commentaires de films et il est accompagné d'un fichier csv qui labélise leur polarité.
Afin de réaliser cette classification, nous avons tout d'abord effectué divers pré-traitements sur les documents. Nous avons ensuite essayé plusieurs classifieurs pour voir lesquels nous semblaient les plus adaptés sans autre modification que les pré-traitements.
Puis, nous avons choisi de nous concentrer sur deux classifieurs (SVC et RFC) sur lesquels nous avons réalisé des grid searchs afin de rechercher leurs meilleurs hyper-paramètres respectifs.
En modifiant les paramètres du vectorizeur, nous sommes parvenus à une précision de 91% pour SVC et 87% pour RFC sur notre jeu de données.
Dans ce rapport, nous détaillerons dans une premier temps les choix adoptés et dans un second temps, les améliorations apportées à ces classifieurs.

2 Recherche des meilleurs pré-traitements

2.1 Différents pré-traitement

Suppression des caractères non-ASCII :
Pour commencer les pré-traitements, nous avons d'abord jugé utile de supprimer les caractères non ASCII. En effet, il nous a semblé que très peu d'utilisateurs utilisaient ce genre de caractère. Nous les avons donc supprimés car ce ne sont pas des caractères lourds de sens qui sont souvent envoyés par erreur par les utilisateurs qui ont commentés les films.

Suppression des contractions :
La langue anglaise se compose, tout comme en français, de formes composées. Notamment sur les négations. Il est donc important de retirer ces contractions afin d'obtenir des littéraux terminaux et de pouvoir travailler sur un texte propre.
Un des exemples le plus commun, le mot "don't". Avec ce pré-traitement il est transformé en "do not" ce qui va normaliser nos données et nous permet de travailler avec ces marques de négations qui ont du sens. Cela devrait aider à inverser la polarité du mot qu'il précède.

Suppression des majuscules :
En général, les majuscules sont utilisées pour les débuts de phrases et pour les noms propres. Dans notre cas, il existe un autre usage des majuscules : crier à l'écrit.
Et dans un nombre non négligeable de commentaires, on peut aussi remarquer que le spectateur utilise les majuscules de cette façon. Cependant, nous avons décidé de les supprimer : d'une part pour réduire le nombre de mots différents et donc gagner du temps d'exécution. Et d'autre part, parce que les spectateurs les utilisent pour accentuer leurs propos. Passer les caractères en minuscule ne changera donc pas le sens du commentaire mais perdra juste en poids.

Suppression de la ponctuation et substitution des nombres :
Encore une fois, le but des pré-traitements est de réduire le nombre de mots différents afin d'avoir une vitesse d'exécution raisonnable. Nous avons estimé que les signes de ponctuation pouvaient être ambivalents et qu'il n'était pas forcément judicieux de les garder.
De même, afin de travailler sur des données textes et réduire le nombre de mots, nous avons choisi de remplacer les nombres par leur forme écrite.

Suppression des stopwords :
Les commentaires étant écrits en langage naturel, il existe un grand nombre de mots qui servent de liant à une grammaire syntaxique mais qui n'ont pas de signification propre (et donc pas de polarité). Nous les avons donc supprimés afin de gagner un grand temps de calcul sans perdre de précision sur le classifieur.
Pour se faire, nous avons importé la liste de stop-words du module NLTK. Seul problème : la liste de stop-words que nous avons trouvé contenait certains mots (comme "not") que nous voulions absolument garder, afin de ne pas perdre le sens de la phrase. Nous avons donc créé notre propre liste de "go-words" (à retirer des stop-words).
De plus, nous avons choisi d'ajouter des mots tels que "movie" ou "popcorn", qui sont des mots récurrents dans nos commentaires mais n'apportent que très peu d'informations sur l'opinion de l'auteur.

Lemmatisation du texte :
La forme lemmatisée d'un mot correspond à sa forme la plus basique dans la langue associée. Un mot sous la forme de lemme est donc la racine de ce mot, c'est à dire un mot sans genre ni nombre et pour les verbes, à l'infinitif. C'est un pré-traitement utile et important pour rassembler les mots ayant le même sens et donc réduire le nombre total des mots différents, tout en gardant la polarité du commentaire.

Suppression des verbes :
Une autre idée a été proposée dans le groupe : supprimer les verbes. En effet, dans le but d'avoir de meilleurs résultats, retirer les verbes des commentaires et se concentrer sur les noms ou les adjectifs peut être une bonne idée.
Malheureusement, après une série de tests, cette suppression a principalement mené à une perte de précision pour nos classifieurs, nous avons préféré ne pas la conserver.

Dans la suite du rapport, la totalité des tests ont été effectués avec les pré-traitements notés précédemment. A noter que même si notre meilleur classifieur utilise ces pré-traitement, il a d'abord été testé sans, mais aussi avec une partie de ces pré-traitements pour finalement trouver la meilleure combinaison possible.

3 Le choix des classifieurs

Après la réalisation des pré-traitements, nous nous sommes intéressés aux classifieurs possiblement utiles pour la classification des documents. Pour cela, nous avons effectué des batteries de tests en utilisant la technique du kFold qui consiste à diviser le jeu de test en plusieurs parties pour obtenir des précisions différentes.

Nous avons appliqué cette méthode sur les classifieurs suivants : LogisticRegression, KNeighbors, DecisionTree, GaussianNB, SVC et RandomForest.
Le choix de ces classifieurs a été justifié par plusieurs raisons. En effet, une grande partie de ces classifieurs ont été étudiés en cours, il était donc plus simple de les appréhender et d'améliorer leur performances. Ensuite, nous nous sommes beaucoup renseigné sur le site scikit-learn[1] qui propose à la fois une explication pour le fonctionnement des classifieurs, mais aussi une explication pour leurs paramètres. De ce fait, nous avons pu faire plus aisément notre choix, notamment grâce à une carte[2] proposant des classifieurs selon le type de donnée et la méthode appliquée. Dans notre cas, voulant faire une classification de données textuelles, nous avons essentiellement travaillé sur SVC.

Les résultats que nous avons obtenus nous ont poussés à choisir de travailler sur les classifieurs SVC et RandomForest. En effet, ils obtenaient les meilleurs résultats avec une précision comprise entre 73% et 77% avec des paramètres par défaut. Les autres classifieurs, quant-à-eux, obtenaient des résultats moins importants, voir même mauvais pour certains.

4 Grid search

Afin d'améliorer la précision des classifieurs précédemment choisis, nous avons ensuite utilisé une grid search afin d'essayer différentes combinaisons d'hyper-paramètre et de déterminer la meilleure possible.
Pour une grande partie des paramètres, les valeurs obtenus étaient celles proposées par défaut pour les classifieurs. Seule une petite partie des paramètres ont renvoyé une valeur différente que celle proposée par défaut :

Random forest :

- n_estimators : 285

n_estimator est le nombre d'arbres de décision dans la Random Forest. Avoir un nombre d'arbres conséquent permet d'éviter certains problèmes des arbres de décision comme le sur-apprentissage.

SVC :

- kernel = 'linear'

Le kernel égal à linear signifie qu'il y a une plus grande flexibilité sur les fonctions de perte. Ce qui est censé être avantageux quand on dispose d'une grande quantité de données. Même si nos 10000 commentaires ne sont pas grand chose comparé aux immenses training sets utilisés dans la recherche et dans le privé, nous avons pensé qu'il était judicieux de donner ce paramètre.

[1] https://scikit-learn.org/stable/
[2] https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

5 Paramètres secondaires

df_min :
Le paramètre df_min du kFold se base sur le TFIDF des mots présents dans le corpus. Il filtre tous les mots ayant une fréquence inférieure à celui-ci. Ainsi, il permet d'une part de réduire grandement le nombre de mots à prendre en compte dans le corpus et donc de réduire le temps de calcul, et d'autre part, il permet d'améliorer légèrement la précision en retirant des mots très peu fréquents et donc, avec peu d'intérêts du fait de leur rareté.
Afin de déterminer les valeurs optimales pour df_min, nous avons fonctionné par essais jusqu'à ce que la précision ne fluctue plus.

df_max :
Le paramètre df_max correspond à l'opposée de df_min. De la même façon, df_max nous permet de filtrer les mots (et groupes de mots) apparaissant trop souvent et ayant donc un poids trop important. Dans notre cas, df_max est fixé à 0.9, une fréquence inexistante pour les mots. Autrement dit, tous les mots ayant une fréquence supérieure à df_min sont pris en compte.

k fold : n_split = 20
Diviser les jeu d'apprentissage en plusieurs sous-jeux et les mélanger permet d'améliorer la classification car on ne va pas toujours l'entraîner sur les mêmes données. Cela permet de ne trop entrainer le classifieur sur les mêmes données et donc ne pas biaisé le classification vis à vis de certains mots qui ne pourrait être vu que dans le jeu de test.

n_gram : (1, 3)
Il nous permet de créer des groupes de un à trois mots, afin d'avoir un point de vue plus global dans la sémantique de la phrase. Cela permet de prendre en compte les négations, afin d'éviter de perdre le sens d'un groupe de mots et de mal interpréter sa polarité (eg. "happy" et "not happy").

FOREST : df_min 0.087
df_max = 0.9
n_split 25
n_gram = 1,3

SVC : df_min 0.01
df_max 0.9
n_split 20
n_gram 1,3

Finalement, le classifieur que nous avons choisi est SVC, au vu des résultats que nous avons obtenu après avoir trouvé ce qui nous semblait être les meilleurs hyper-paramètres pour chacun des deux classifieurs.

6 Résultats du challenge

Sur le jeux de commentaires mis à disposition pour le challenge, le classifier SVC que nous avons choisis a obtenu une précision de 88.85%. Nous avons donc obtenu un moins bon résultat que celui sur nos données d'apprentissage. Il peut y avoir plusieurs raison à cela.
Tout d'abord, il peut exister des mots sur lesquelles le classifieur n'a jamais été entraîné et donc peut donner une réponse fausse, ou peut tout aussi bien avoir été trop entraîné sur le premier jeu de test ce qui peut biaiser les résultats.
Le problème de l'ironie dans les commentaires peut toujours aussi être un problème qu'il est parfois très dur à déceler. Surtout si par exemple, après l'entraînement, certains mots sont assignés par le classifieur comme des mots qui donnent une réponse positive mais qui dans le nouveau jeu de test sont plus utilisé ironiquement.

Pour améliorer nos résultats, nous aurions pu rechercher de meilleurs pré-traitements sur les commentaires qui auraient pu être plus simple et plus efficace à analyser par les classifieurs. Nous aurions pu approfondir le travail sur d'autres classifieurs qui auraient pu, au finale, être plus efficaces. Nous aurions pu aussi affiner la recherche sur de meilleurs hyper-paramètres pour les classifieurs.