Rapport pour le projet du Module Terminologie et Stylométrie
Par Mr Gaël Lejeune
M2 Langue et informatique
Projet réalisé par : Ouali Jughurta et Fintan Herlihy

Introduction :

La classification de documents par sentiment est une tâche intrinsèquement liée au domaine du traitement automatique du langage : en effet, des faits linguistiques permettent de caractériser un texte pour qu'il nourrisse ensuite un algorithme de classification. Ces faits linguistiques peuvent être de natures si diverses, qu'il est parfois difficile de s'attarder à déterminer lesquels peuvent être pertinents pour la classification par sentiment. Il y a donc nécessité, dans certains cas, de passer par des méthodes plus purement statistiques.

Dans les domaines de la recherche d'information et de la fouille de texte, plusieurs méthodes existent pour ce qui est, par exemple, de déterminer la similarité d'un document à un autre, ou de récupérer des documents correspondant à une requête donnée. Pour ces tâches, la méthode de la pondération des termes, (qu'on peut définir par le classement de termes dépendant de leur fréquence d'apparition dans un ensemble de documents, ou corpus), permet de faire face à la problématique qu'engendre la loi de Zipf : elle stipule que la fréquence d'apparition d'un terme dans un corpus est inversement proportionnelle à son rang. Selon cette loi, les mots les communément utilisés sont peu importants pour les tâches mentionnées plus haut, alors que les mots plus rares sont plus utiles, permettant par exemple de mieux caractériser un document et s'il appartient à une catégorie spécifique. Plusieurs différents types de pondérations existent, qui peuvent être employées pour différentes finalités selon la tâche à laquelle on s'intéresse. Nous nous intéresserons, dans notre cas, à la tâche de classification de tweets en français selon leur polarité, en utilisant deux méthodes de pondération, à savoir, Okapi BM25 et TF-IDF.

1 Description du jeu de données

1.1 Présentation du jeu de données :

Le jeu de données que nous allons utiliser dans le cadre de ce projet est un fichier json composé de tweets en langue française classer sous quatre labels : objectif, positive, négative et mix, chaque classe est disponible dans un fichier spécifique à elle, la classe mix est constitué d'un mélange des trois autres classes. Le jeu de données d'entrainement est composé de 3906 tweets, celui de test 976 tweets. Ainsi nous avons un jeu de données multi classes.

Le train set est composé de 93187 Tokens, pour avoir connaissance de la distribution de ces derniers,

nous avons calculer la fréquence de chaque token afin de connaitre mieux notre corpus.

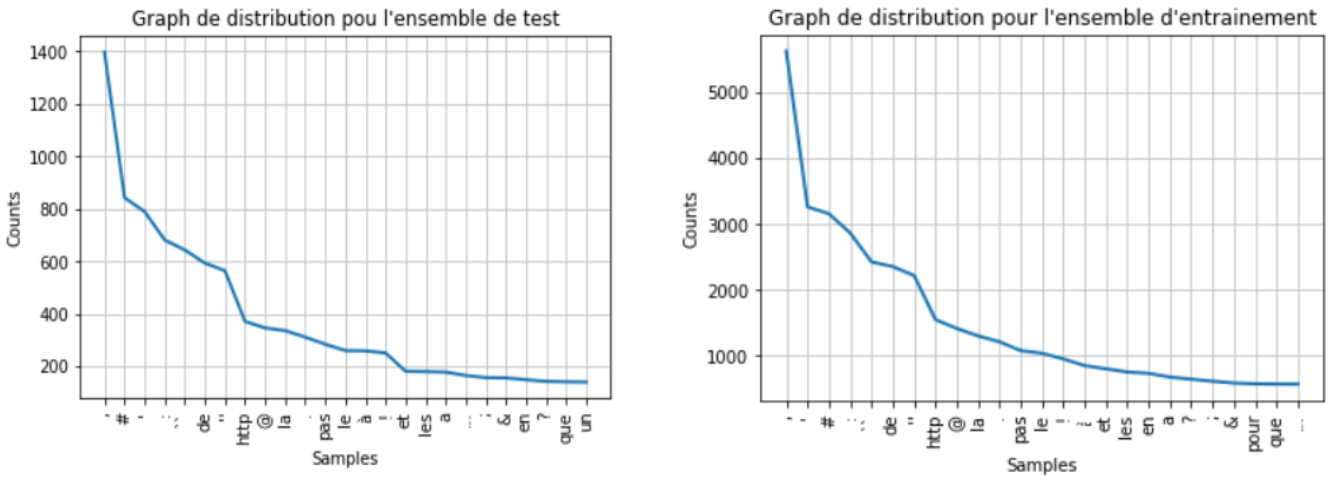

Figure 1 Graphe de Distribution

Nous pouvons observer qu'un nombre important de tokens sont des ponctuations et des mots outils, ce graphe montre la distribution des 25 premiers tokens. Le « http » indique une présence considérable de liens dans le corpus, ce qui est relativement normal car ces données sont extraites d'un réseau social. Cependant, ce problème peut être éviter par une simple pondération TF-IDF, qui va donner plus de poids aux mots les moins fréquents (plus rare) par contre, dans le cas où un sac de mots est utilisé, cette distribution impactera les résultats.

Sentiments	Dataset entrainement	Dataset test
Positive	494	123
Negative	1268	318
Objective	1643	411
Mixed	501	124
Totale	3906	976

Le tableau ci-dessous indique le nombre de tweets pour chaque classe de sentiment dans l'ensemble d'entrainement et de test. La classe objective est la mieux représentée dans les deux ensembles, les classes mixes et positive sont les moins représentés.

1.2 Caractéristique des tweets :

- La longueur des tweets : la longueur maximale d'un tweet est de 140 caractères, c'est la spécifité de ce réseau social ainsi les utilisateurs en tendance à exprimer leurs sentiments en utilisant des mots qui ont une valeur sémantique importante. Dans le cadre de notre jeu de données, la longueur des tweets varie entre 28 et 140, cependant il existe certains tweets qui

contiennent des liens, ces derniers sont représentés comme une seule unité(token).

- Le hachtags : l'hashtag est beaucoup utilisé sur tweeter, c'est le symbole clé qui permet la diffusion de l'informations, suivie de quelques mots qui encapsule une information.

- Style d'écriture : comme dans tous les réseaux sociaux, le langage utilisé sur tweeter est plutôt familier, on remarque qu'il y a une forte utilisation d'abréviation, code urbain et répétition de lettre soit pour exprimer un sentiment ou une chose d'une manière explicite, cela peut être également dû à une erreur de frappe ou d'orthographe.

- L'arobase « @ » : ce symbole est disponible dans l'intégralité des tweets, il permet de déterminer le nom ou le pseudo de l'utilisateur.

2 - Les méthodes :

Nous présentons la Baseline qui à été mise en place pour servir de modèle de base, ensuite nous expliquons méthodologies utilisées pour les différentes techniques de représentation vectorielle ainsi que les pré-traitements.

2.1 Baseline :

Nous avons procédé dans un premier temps à la création d'une Baseline à partir de laquelle nous allons améliorer nos modèles, en utilisant différentes méthodes et paramètres. Suite à l'importation des deux ensemble (entrainement et test), nous avons effectué une représentation vectorielle simple pour rendre les valeurs qualitatives en valeur numérique dans le but de représenter les données d'entrainement en vecteurs et les transposer dans une matrice pour être intelligible par la machine. Par la suite, nous avons créé plusieurs modèles avec des algorithmes de classification (RandomForest, LinearSVM, Knn, Naive baiyes). Comme pour les données d'entrainement, une représentation vectorielle a été réalisé sur les données du test.

2.2 Vectorisation TF-IDF

TF-IDF est un algorithme de pondération, c'est une combinaison de deux formules de TF (la fréquence des termes) et IDF (la fréquence inverse des documents). Cette combinaison permet de donner plus de poids aux mots les moins fréquents car ces derniers sont considérés plus discriminant. Ainsi, tf-idf va nous permettre de résoudre le problème de la fréquence de la ponctuation et des mots vides dans la distribution des tokens dans le jeu de données.

2.3 Un contre le reste « OnevsReste » :

C'est une technique utilisée dans les processus ou problématique de classification multi-classes, cette

méthode permet la création d'un classifieur unique pour chaque classe ainsi, il sera par la suite plus facile à acquérir davantage de connaissance dans l'apprentissage pour chaque classe en inspectant les autres classificateurs propres à chaque classe. Etant donné que nous avons une un ensemble de données multi-classes nous allons expérimenter cette méthode

2.4 Pré-traitement des données :

Cette étape dépend des données que nous allons utiliser et de l'observation de ces derniers, nous avons décidé de supprimer la ponctuation qui représente une grande quantité de token, les mots vides et la lemmatisation afin de réduire les formes fléchis des verbes. Pour réaliser cela, nous avons utilisé la bibliothèque de NLP Spacy.
Les pré-traitements effectués :

- Suppression de la ponctuation
- Suppression des valeurs numériques
- Suppression des urls
- Suppression des mots vides
- Lemmatisation (pour réduire les formes fléchis)
- Normalisation (mettre les données au minuscule)

2.5 Effets de pré-traitement sur les données

Traitement	Nombre de tokens
Sans prétraitement	93187
Ponctuation	81823
Numérique	80353
URLs	78678
Mots vides	44058

Nous pouvons observer dans ce tableau que l'application de pré-traitements diminue considérablement le nombre de jetons dans l'ensemble de données, cette réduction de bruit impactera positivement ou négativement les résultats de classifications, Après la combinaison de plusieurs de ces traitements, nous avons déduit que dans notre cas la suppression des mots vides et les liens impacte négativement nos résultats, de ce fait nous avons opté pour la meilleure combinaison.

2.6 Okapi BM25

Okapi BM 25 (Online Keyword Access to Public Information Best Match 25) est, au contraire de TF-IDF, une méthode de pondération de documents pour estimer leur pertinence par rapport à une recherche donnée, qui a été développée par des chercheurs britanniques au début des années 80 dans le cadre du développement de catalogues bibliothécaires. Elle peut être expliquée par la formule suivante :

$$p_{ij} = L_{ij} G_i = \left(\frac{f_{i,j}\,(k_1+1)}{k_1((1-b)+b(\frac{ld\,j}{ld\,moy}))+f_{i,j}}\right) F4$$

Où :
p ij est le poids du terme *i* dans un document *j*
L ij est le poids local du terme *i* dans un document *j*
G i est le poids global du terme *i*
f i,j est la fréquence du terme *i* dans un document *j*
k est une valeur arbitraire de lissage
F4 est un score de pertinence
ld est la longueur du document (*moy* est la moyenne de longueur d'un document)

L'implémentation [note pied de page avec lien] de cet méthode de pondération que nous avons mis à emploi prend en entrée un objet TfidfTransformer de la bibliothèque d'apprentissage automatique scikit-learn [note pied de page] pour ensuite le transformer en représentation Okapi BM25.

3 Résultats et perspectives :

Dans cette section nous allons exposer les résultats obtenus dans le cade de ce projet de classification et les éventuelles perspectives que nous souhaitons apporter prochainement.

3.1 **Résultats**

Le tableau ci-dessous montre les résultats obtenus avec la Baseline, nous avons récupéré les «`weighted avg`» et l'exactitude « accuracy ». le taux d'exactitude est relativement proche entre les différents modèle, KNN affiche le plus bas résultats contrairement à Logtic Regression qui affiche le meilleure résultats.

	Précision	Rappel	F score	Accuracy
RandomForst	**0.60**	**0.61**	**0.60**	**0.61**
SVM	**0.60**	**0.61**	**0.61**	**0.61**
KNN	**0.51**	**0.50**	**0.48**	**0.50**
Naive Bayes	**0.63**	**0.63**	**0.60**	**0.63**
LogisticRegression	**0.63**	**0.65**	**0.63**	**0.65**

Nous allons observer en détail les résultats par classe du modèle LogisticRegression, il est remarquable que la classe mix affiche le taux le plus bas avec une précision de 0.31 et un rappel de 0.13, on remarque que notre modèle trouve des difficultés cette classe contrairement à la classe objective qui affiche une précision de 0.75.

	precision	recall	f1-score	support
mix	0.31	0.13	0.18	124
neg	0.59	0.69	0.64	318
obj	0.75	0.79	0.77	411
pos	0.61	0.61	0.61	123
accuracy			0.65	976
macro avg	0.57	0.55	0.55	976
weighted avg	0.63	0.65	0.63	976

Ce deuxième tableau affiche les résultats obtenus avec la vectorisation TF-IDF sans pré-traitements

	Précision	Rappel	F score	Accuracy
RandomForst(Tf-IDF)	**0.58**	**0.61**	**0.58**	**0.61**
SVM (Tf-IDF)	**0.64**	**0.66**	**0.64**	**0.66**
KNN (Tf-IDF)	**0.61**	**0.63**	**0.60**	**0.63**
Naive Bayes (Tf-IDF)	**0.55**	**0.60**	**0.54**	**0.60**
LogisticRegr (Tf-IDF)	**0.61**	**0.64**	**0.60**	**0.64**

Cette pondération a apporté une net amélioration dans le taux d'exactitude pour la majorité des modèles, notamment pour knn avec une différence de 0.13 mais n'empêche que les résultats sont relativement bas

Le troisième tableau montre les résultats avec la vectorisation TF-IDF avec des données prétraités et le « one versus reste ».

	Précision	Rappel	F score	Accuracy
RandomForst(Tf-IDF)	**0.58**	**0.61**	**0.58**	**0.61**
+ OneVersusRest	0.58	0.61	0.58	0.61
SVM (Tf-IDF)	**0.64**	**0.66**	**0.64**	**0.66**
+ OneVersusRest	0.65	0.67	0.65	0.67
KNN (Tf-IDF)	**0.61**	**0.63**	**0.60**	**0.63**
+ OneVersusRest	0.57	0.59	0.57	0.59
Naive Bayes (Tf-IDF)	**0.55**	**0.60**	**0.54**	**0.60**
+ OneVersusRest	0.56	0.62	0.57	0.62
LogisticRegr (Tf-IDF)	**0.61**	**0.64**	**0.60**	**0.64**
+ OneVersusRest	0.65	0.64	0.60	0.64

L'utilisation de la méthode « one versus reste » n'a pas apporté une augmentation considérable dans le taux d'évaluation

3. 2 **Conclusion :**

Nous avons un apprentissage déséquilibré, ce problème se produit généralement lorsqu'il y a plus d'instances dans certaines classes que d'autres, ce qui est le cas de notre ensemble jeu de données (voir le tableau de l'ensemble de jeu de données). D'autant plus, nous avons une classe mixes qui regroupe des instances mélangées à savoir des tweets objective, positive et négative. De ce fait, les algorithmes de classifications ont tendance à ignorer les classes les moins représentés, autrement dit les petites classes.

3.3 **Perspective** :

Nous souhaitons appliquer l'échantillonnage synthétique car les classes que nous avons sont extrêmement déséquilibrés par exemple, la classe objective a 3 fois plus de tweets que la classe positive et mixes d'autant plus, cette dernière est difficilement classifiée par les algorithmes.

L'implémentation de Okapi BM25 pour comparer les résultats avec d'autres méthodes de vectorisation.

Répartition des taches

Taches réalisées par Ouali jugurtha :

- Implémentation du code avec Jupyter notebook
- Section description du jeu de données : Présentation du jeu de données, caractéristique du jeu de données
- Section méthodes : Baseline, Vectorisation TF-IDF, OneVersusRest, Prétraitement des données.
- Section Résultats et perspectives

Taches réalisées par Fintan Herlihy :

- Introduction
- Section méthodes : Okapi BM25