Master 1 Langue et Informatique

M1SOL023

Méthodologie de la Recherche en Langue et Informatique

Rapport de projet final
(Classification automatique de textes)

Enseignante :
Laurence Devillers
Nour El Houda Ben Chaabene
Etudiante :
Shilin Xie
21102552
30 Décembre 2022

Tableau de matière

Introduction

La numérisation a changé la façon dont nous traitons et analysons l'information. La quantité d'informations sur le Web augmente désormais de façon exponentielle, avec des pages Web, des e-mails, des revues, des livres électroniques, du matériel d'étude, des actualités et des pages de médias sociaux, tous inondés d'informations contenant beaucoup de texte. Afin de pouvoir faire face rapidement au volume croissant, le traitement automatisé devient de plus en plus important.

Le projet de Méthodologie de la Recherche en Langue et Informatique vise à mettre en place une solution permettant de catégoriser automatiquement des textes en fonction de leur contenu. La classification de texte est l'application de traitement automatiques des langues la plus basique et la plus pratique, qui est appliquée à la reconnaissance du spam, à la classification des sentiments, à la classification des sujets et à d'autres tâches. La tâche consiste à attribuer un label à chaque texte en fonction des mots et des phrases qu'il contient.

Dans ce projet, la méthode basée sur l'apprentissage automatique est principalement utilisée : utiliser l'expérience passée pour apprendre automatiquement des règles au lieu de définir manuellement des règles. La classification de l'apprentissage automatique apprend à partir d'un ensemble d'apprentissage, en créant un mappage d'un espace d'entrée X vers un espace de sortie Y (valeurs discrètes). Les classificateurs basés sur l'apprentissage automatique utilisent des exemples étiquetés comme données d'entraînement pour apprendre l'association entre les mots/phrases et les étiquettes, c'est-à-dire les catégories. Selon différentes catégories de sortie (étiquettes), il peut être divisé en classification binaire, multi-classification (les deux classifications je ferai dans le projet) et classification multi-étiquettes.

Les données traitées dans ce projet seront des textes en langage naturel en format électronique, provenant de différentes sources. Nous utiliserons des outils d'apprentissage automatique pour entraîner des modèles de classification de textes qui sera capable de prédire le label d'un texte en fonction de son contenu.

Pour les Données du projet, j'ai choisi le corpus French Twitter Sentiment Analysis pour faire une classification binaire sur les identifiants des sentiments (positif et négatif) et le corpus Brown (version avec POS et sans POS) pour réaliser une multi-classification de quinze catégories fines et une multi-classification de trois grandes catégories.

Pour entamer le travaille, j'utiliserai le langage Python pour écrire le code, car la librairie de python contient de nombreux packages pouvant être appelés, ce qui est très pratique et rapide. Principalement je vais appeler la module sklearn directement et utilisez l'algorithme à l'intérieur, tous les processus serai réalisés sur VScode, un éditeur de code source multiplateforme développé par Microsoft.

Le plan de ce projet consiste en plusieurs étapes. Tout d'abord, je vais préparer les données en nettoyant et en structurant les textes. Ensuite, je vais entraîner des modèles de classification de textes en utilisant des algorithmes d'apprentissage automatique supervisé. Enfin, je évaluerai la performance du modèle en utilisant des métriques adéquates et nous effectuerons des tests pour vérifier sa fiabilité.

Etat de l’art

En ce qui concerne les techniques de classification de textes, il existe actuellement trois étapes : la classification par apprentissage automatique (machine learning), la classification par apprentissage profond (deep learning) et la classification par attention, dont les idées centrales sont de se concentrer respectivement sur la probabilité, la globalité et l'attention. Dans ce projet, j'utilise principalement la classification par apprentissage automatique, et j'impliquerai la classification binaire et la multi-classification, où les algorithmes représentatifs de la classification binaire et de la multi-classification par apprentissage automatique sont : *Perceptron* (une méthode de classification linéaire qui peut directement obtenir une fonction discriminante linéaire), *Bayes Multinomial* (en supposant que la distribution de probabilité obéit à une distribution multinomiale simple, principalement utilisé pour la classification de caractéristiques discrètes, en classification de textes avec des mots comme granularité de calcul), *Régression Logistique* (mesure la relation entre une variable dépendante d'une catégorie et une ou plusieurs variables indépendantes en estimant les probabilités à l'aide de fonctions logistiques/sigmoïdes), *Machine à Vecteurs de Support* (extrait le meilleur hyperplan ou la meilleure ligne qui sépare deux classes), *Arbre de Décision* (un processus de classification des données par le biais d'un ensemble de règles), *Random Forest* (un algorithme intégré qui est du type Bagging et qui rend le résultat global du modèle avec une précision et une performance de généralisation élevées en combinant plusieurs classificateurs faibles et le résultat final par vote ou en prenant la moyenne) etc. Les algorithmes mentionnés ci-dessus sont également ceux que j'utiliserai dans mon projet, et j'ajouterai également un algorithme de classification par apprentissage profond, le *Perceptron Multicouche* (un réseau de neurones artificiels à action directe), afin de comparer les avantages des différents classificateurs.

Avant de me lancer dans ce projet, je me suis beaucoup renseigné sur Internet, pour voir ce que d'autres avaient fait pour des tâches similaires et comment ils avaient utilisé les sept modèles de classification et les paramètres d’ajustement.

Une tâche similaire à ma tâche de classification binaire pour l'analyse des sentiments positifs et négatifs des tweets français est le jeu de données *IMDB Dataset of 50K Movie Reviews*, qui demande également de classer le sentiment de chaque critique comme positif ou négatif. J'ai remarqué quelques implémentations qui utilisent la plupart des mêmes classificateurs que moi mais qui les traitent différemment, comme le comptage des mots positifs et négatifs à haute fréquence comme base pour un jugement ultérieur et l'utilisation de *gensim.models.Word2Vec* pour vectoriser le texte[1] ou pour un exemple plus détaillé de nettoyage de données textuelles[2].

La tâche de classification des données *BBC Full Text Document Classification* qui compte 2226 textes et cinq catégories, est similaire à la tâche de multi-classification des données de corpus *Brown*. L'un des types de techniques les plus similaires à ma tâche actuelle est cette mise en œuvre

[1] https://www.kaggle.com/code/ soumighosh99/imdb-dataset-bagofwords-tfidf-and-word2vec-model

[2] https://www.kaggle.com/code/anishnaskar/text-classification-using-multinomial-naive-bayes

d'un modèle de Régression Logistique pour classer des vecteurs de mots avec différentes caractéristiques prises en compte[3].

Il est également possible d'utiliser de nombreux algorithmes différents ou des transformations d'extraction de caractéristiques pour le même ensemble de données afin d'obtenir des résultats différents, par exemple dans la classification de la dichotomie des sentiments sur Twitter en français, j'ai vu qu'un auteur avait publié son travaille *CamemBERT for French Tweets classification*[4], il a utilisé modèle CamemBERT (un modèle linguistique de pointe pour le français basé sur le modèle RoBERTa) pour mettre en œuvre la classification des textes françaises.

Le même jeu de données peut également être utilisé pour classer les éléments qui le composent selon des besoins différents. Le jeu de données de corpus Brown, par exemple, est plus souvent utilisé comme matériel pour l'entraîne[5]ment de modèles de classification d'annotations lexicales(POS), alors que je l'utilise comme matériel pour le modèle de classification de domaines décrit dans l'article.

Il est important de noter que, dans le domaine de la classification de données, il n'existe pas de solution unique qui convient à tous les problèmes. Il est donc souvent nécessaire de faire des essais et des erreurs et de comparer différentes approches pour trouver la solution la plus efficace pour chaque problème spécifique.

---

[3] https://www.kaggle.com/code/gauravsb/text-classification-in-machine-learning

[4] https://www.kaggle.com/code/houssemayed/camembert-for-french-tweets-classification/notebook

Jeu de données

French Twitter Sentiment Analysis

C'est un ensemble de données français avec les étiquettes pour l'analyse des sentiments (données traduites de l'anglais vers le français.[6]

Il s'agit d'un ensemble de données qui est stocké dans un fichier csv, il y a 2 classes, un commentaire Twitter est une instance ,donc sur un total de 1 526 724 instances, les mots de ces instances totalisent 252 741 mots, et deux classes : positif (valeur 1) et négatif (valeur 0) dans l'ensemble de données. Parmi elles, 755 120 instances sont réparties en positif, 715 604 instances sont divisées en négatif. Voici les longueurs des vecteurs formés en fonction des différentes caractéristiques:

	Unigram	Bi-gram	Tri-gram
Avec stop-words	19 373 177	38 226 814	55 631 117
Sans stop-words	19 373 092	38 226 643	55 630 860

Corpus Brown pour multi-classification de quinze catégories fines

Le type d'ensemble de données est un fichier csv[7], qui contient 15 classes : *adventure, belles_lettres, editorial, fiction, government, hobbies, humor, learned, lore, mystery, news, religion, reviews, romance, science_fiction*. Il y a un total de 57 340 textes, et le nombre de mots sans doublon dans l'ensemble des données est de 55 982. Ce dataset contient du texte avec des annotations POS et du texte brut, je comparerai les résultats des tests formés par ces deux ensembles de texte.

Nombre d'articles possédés par catégorie :

Catégorie	Nombre
religion	354
lore	224
learned	263
mystery	25
adventure	89
government	223
editorial	314

[6] https://www.kaggle.com/datasets/hbaflast/french-twitter-sentiment-analysis

[7] https://www.kaggle.com/datasets/nltkdata/brown-corpus

humor	250
hobbies	429
belles_lettres	499
romance	320
news	300
fiction	82
reviews	285
science_fiction	112

Voici les longueurs des vecteurs formés en fonction des différentes caractéristiques:

Avec marque POS:	Unigram	Bi-gram	Tri-gram
Avec stop-words	889 937	1 812 617	2 686 375
Sans stop-words	512 062	978 828	1 391 398
Sans marque POS:	**Unigram**	**Bi-gram**	**Tri-gram**
Avec stop-words	1 435 713	3 276 694	5 154 585
Sans stop-words	991 438	2 227 475	3 435 181

Corpus Brown de NLTK pour multi-classification de trois grandes catégories

Ce jeu de données est directement importé du module NLTK, donc le jeu de données est différent du marron ci-dessus, il a une taille plus petite et moins d'étiquettes. Il y a un total de 309 textes (instances) dans cet ensemble de données, qui sont divisés en 9 classes : *editorial, reviews, news, adventure, mystery, fiction, romance, learned, government.*

Je les reclasse directement en trois grandes catégories: news=[editorial, reviews, news], books=[adventure, mystery, fiction, romance], sciences=[learned, government].Dans la catégorie news, il y a 88 documents avec 202 862 mots, dans la catégorie books, il y a 111 documents avec un total de 265 021 mots et dans la catégorie sciences, il y a 110 documents avec un total de 252 005 mots.

Voici les longueurs des vecteurs formés en fonction des différentes caractéristiques:

	Unigram	Bi-gram	Tri-gram
Avec stop-words	237 879	758 404	1 347 775
Sans stop-words	197 294	504 898	827 474

Pour l'ensemble d'entraînement de chaque ensemble de données, je l'ai configuré pour utiliser 70 % des données pour l'entraînement, et 30 % des données seront utilisées pour les tests, et afin de garantir que le programme exécute le même ensemble d'entraînement à chaque fois et jeu de test, j'ai mis *random_state* à zéro.

Méthode

Afin d'obtenir un modèle avec une grande précision et une compréhension plus approfondie de divers paramètres de chaque algorithme, je vais ajuster divers paramètres dans les fonctions des classificateurs et effectuer différents pré-traitements sur les données de texte. En général, je ferai quatre plans pour chaque ensemble de données : 1. Utiliser le texte brut, 2. Transformer les lettres en minuscules 3. Supprimer les mots vides 4. Transformer les lettres en minuscules et supprimer les mots vides. Pour les textes anglais et français, j'ai utilisé les listes de mots vides correspondantes.

Avant de former un classificateur d'apprentissage automatique, il y a un problème qui doit être résolu, et c'est l'extraction de caractéristiques. Les ordinateurs ne comprennent pas le texte comme nous, ils ne comprennent que les 0 et les 1. Par conséquent, la première étape de la formation d'un classificateur TAL consiste à convertir le texte en une représentation vectorielle numérique. L'une des méthodes d'incorporation de texte les plus couramment utilisées est le sac de mots, ce modèle considère le texte comme un ensemble de mots multiples et non ordonnés, indépendamment de la grammaire et de l'ordre des mots. Il peut également être considéré comme un modèle d'espace vectoriel dont le mot est l'unité de base. Cependant, dans des scénarios pratiques, l'ordre des mots est très important et peut affecter le sens d'une phrase. Par conséquent, nous devons préserver l'ordre des mots dans les caractéristiques. Avec une collection de N caractéristiques, le texte peut être représenté comme un vecteur en utilisant le modèle du sac de mots. Au fur et à mesure que N augmente, le nombre de caractéristiques qui peuvent être extraites augmente de façon exponentielle, et l'espace des caractéristiques augmente de façon exponentielle. La fréquence de ces caractéristiques d'ordre supérieur est également relativement faible, ce qui n'est pas très utile pour la classification et affecte directement l'efficacité et la complexité du traitement ultérieur. Par conséquent, pour les tâches générales de classification de texte, N de 3 est suffisant, et les caractéristiques binaires et monadiques sont utilisées pour éviter l'ajustement excessif. Donc, j'utiliserai Unigram, Bi-gram et Tri-gram pour extraire des caractéristiques et générer des vecteurs à partir des données textuelles en utilisant différentes méthodes de pré-traitement.

Afin de sélectionner les caractéristiques importantes de l'ensemble de données, j'ai utilisé le *TfidfVectorizer* pour l'implémenter. Le *TfidfVectorizer* est équivalent à l'utilisation d'une combinaison du *CountVectorizer* (qui sert à transformer les documents texte en une matrice de comptage clairsemée) et du *TfidfTransformer* (qui ajoute des poids TF-IDF à la matrice).

Une fois que les représentations vectorielles de tous les documents texte étiquetés sont générées, elles peuvent être utilisées pour former un classificateur. Un vecteur de documents texte est transmis au classificateur avec la catégorie correcte. Une fois que le modèle est formé pour répondre aux critères de performance requis, il peut être utilisé pour faire des prédictions précises. La même méthode d'extraction de caractéristiques est utilisée pour créer des représentations vectorielles de nouveaux documents texte, puis les modèles de classification utilisent ces vecteurs de caractéristiques pour prédire la catégorie du document.

J'ai importé sept modèles d'algorithmes d'apprentissage automatique de sklearn, à savoir : *Perceptron, MultinomialNB, LogisticRegression, LinearSVC, DecisionTreeClassifier, RandomForestClassifier* et *Perceptron multicouche*. Pour les paramétrages de chaque classificateur, je les ai modifiés un par un à l'avance jusqu'à ce que les meilleurs paramétrages du modèle pour ces données soient testés. Par contre, je n'ai pas utilisé *Decision tre*e et *Perceptron Multicouche* dans le corpus Twitter, ses données de 2 et 3 gramme sont trop volumineuses, j'ai attendu 678 minutes et il n'est toujours pas sorti, et d'après le test de ce modèle sur d'autres données, ses performances ne sont pas très exceptionnelles, alors j'ai les abandonné. Parmi les différents classificateurs bayésiens, le Bayésien Multinomial est le plus adapté au traitement de texte, c'est pourquoi je l'ai choisi. Avant la tâche, je pense que les arbres de décision et les forêts aléatoires peuvent ne pas être très adaptés à la classification de texte, surtout pour les données de texte qui ont un somme considérable, puis à la fin je peux vérifier si mon hypothèse est correcte.

Pour mesurer les performances de chaque classificateur, j'ai importé classification_report de sklearn.metrics pour voir la relation entre les prédictions du classificateur et les résultats réels, également j'enregistré le temps qu'il met à s'exécuter.

Résultats

French Twitter Sentiment Analysis

Les cinq méthodes qui ont les taux de précisions plus haut :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule
Logistic Regression	0.8106275299224048	3	FALSE	TRUE
linear_svc	0.8102105157439228	3	FALSE	TRUE
linear_svc	0.8101908658611671	3	TRUE	TRUE
linear_svc	0.8097804016436035	3	TRUE|FALSE	FALSE
Logistic Regression	0.8092171050046068	3	TRUE	TRUE

Les fonctionnalités utilisées par chaque classificateur pour obtenir sa plus grande précision :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule	Temps_execution(secondes)
Logistic Regression	0.8106275299224048	3	FALSE	TRUE	148.44856333732605
linear_svc	0.8102105157439228	3	FALSE	TRUE	30.660584926605225
MultinomialNB	0.7884471789318324	3	TRUE	TRUE	1.9161920547485352
Perceptron	0.7838207232030182	3	TRUE	FALSE	4.7606189250946045
Random Forest	0.5619931967739259	1	TRUE	FALSE	213.1473913192749

Dans cette tâche de classification binaire de texte français, à l'exception de *Random Forest*, d'autres modèles sont considérés comme étant entraînés avec des fonctionnalités Tri-gram pour obtenir les meilleurs résultats. Et les minuscules sont plus importantes que la suppression des mots vides. La *Logistic Regression* et *linear_svc* ont la précision la plus élevée, tandis que la précision de la forêt aléatoire n'est que de 56 %, le temps d'exécution est le plus long et il y a un grand écart avec les autres classificateurs. Plus, la précision de l'ensemble d'apprentissage et de l'ensemble de test ne sont généralement pas très différentes pour chaque modèle, sauf que *Perceptron* a une différence de 20% dans la précision des deux ensembles sous certaines conditions.

Comme mentionné précédemment, la vitesse de *Decision Tree* et de *Perceptron multicouche* est trop lente. Je n'ai formé ces deux modèles qu'une seule fois et j'ai obtenu les résultats:

Classificateur	**Accuracy**	**N-gram**	**Enlever stop-words**	**En minuscule**	Temps_execution(secondes)
Decision Tree	0.5641	1	FALSE	FALSE	125.07898688316345
Perceptron multicouche	0.7942	1	FALSE	FALSE	473.66553020477295

On peut voir que les performances de *l'arbre de décision* et de *la forêt aléatoire* sont similaires et ne conviennent pas à de telles tâches. Mais le résultat de *Perceptron multicouche* est très bon. Compte tenu de l'importance de Tri-gram pour la précision du modèle précédent, en supposant que je puisse utiliser cette fonctionnalité dans *Perceptron multicouche*, il a de bonnes chances d'atteindre un taux de précision de plus de 80%, mais le temps d'attente sera très long.

Dans cet ensemble de données, il existe un phénomène courant de *Overfit*, en particulier pour *Perceptron* et *linear_svc*, le taux de précision le plus élevé de leur ensemble d'apprentissage est de 98 %, tandis que dans l'ensemble de test, ils ne peuvent atteindre que le taux de précision le plus élevé d'environ 50 %. Pour les modèles qui fonctionnent mal dans l'ensemble de test, leur précision dans l'ensemble d'apprentissage ne sera pas particulièrement élevée. Par exemple, la précision de *l'arbre de décision* dans l'ensemble d'apprentissage n'est que de 2 % supérieure à celle de l'ensemble de test.

Corpus Brown de NLTK pour multi-classification de trois grandes catégories

Les cinq méthodes qui ont les taux de précisions plus haut :

Classificateur	**Accuracy**	**N-gram**	**Enlever stop-words**	**En minuscule**
linear_svc	0.967741935483871	1	TRUE	FALSE
linear_svc	0.956989247311828	1	FALSE	TRUE| FALSE
Perceptron		2	FALSE	TRUE
Logistic Regression		1	FALSE	TRUE| FALSE
linear_svc	0.946236559139785	2	FALSE	TRUE| FALSE
		1	TRUE	TRUE

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule
linear_svc	0.9354838709677419	3	FALSE	TRUE| FALSE
Perceptron		2	TRUE	TRUE
		2	FALSE	FALSE
		1	TRUE	TRUE
		1	FALSE	FALSE
linear_svc	0.9247311827956989	2	TRUE	TRUE| FALSE
Random Forest		1	FALSE	TRUE
Perceptron		1	TRUE	FALSE
Perceptron multicouche		1	TRUE	FALSE

Dans le tableau on peut constater qu'il est possible d'obtenir la même précision en utilisant différentes fonctionnalités et différents modèles.

Les fonctionnalités utilisées par chaque classificateur pour obtenir sa plus grande précision :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule	Temps_execution(secondes)
linear_svc	0.967741935483871	1	TRUE	FALSE	0.017364978790283203
Perceptron	0.956989247311828	2	FALSE	TRUE	0.032349109649658 2
Logistic Regression	0.956989247311828	1	FALSE	TRUE| FALSE	0.3043060302734375/0.25968098640441895
Random Forest	0.9247311827956989	1	FALSE	TRUE	0.35959911346435547
Perceptron multicouche	0.9247311827956989	1	TRUE	FALSE	2.6827070713043213
MultinomialNB	0.9032258064516129	1	TRUE	TRUE| FALSE	0.0054950714111328125/0.006679058074951172
Decision Tree	0.8172043010752689	1	FALSE	TRUE	0.045735120773315 43

Pour les résultats de cet ensemble de données, nous pouvons voir que le taux de précision de six classificateurs peut atteindre plus de 90 %, et l'écart de performance n'est pas trop grand. Seul le taux de précision de l'arbre de décision est de 81,7 %, ce qui est le meilleur. Résultat : une différence d'environ 15 %. Pour Ngram, il n'est pas nécessaire d'utiliser Tri-gram pour mieux performer. La suppression des mots vides est plus importante pour obtenir une précision optimale que l'unification des mots en lettres minuscules.

La précision de l'ensemble d'apprentissage dans cet ensemble de données est généralement de 4 à 10 % supérieure à celle de l'ensemble de test, et les trois modèles les plus performants et *Perceptron multicouche o*nt même une précision de 100 %.

Corpus Brown pour multi-classification de quinze catégories fines

---

En considérant les annotations POS :

Les cinq méthodes qui ont les taux de précisions plus haut :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule
linear_svc	0.5415068015347053	2	TRUE	FALSE
linear_svc	0.5413905359841878	2	FALSE	FALSE
linear_svc	0.5412161376584118	2	FALSE	TRUE
linear_svc	0.5411580048831531	2	TRUE	TRUE
linear_svc	0.535344727357284	1	FALSE	FALSE

Les fonctionnalités utilisées par chaque classificateur pour obtenir sa plus grande précision :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule	Temps_execution(secondes)
linear_svc	0.5415068015347053	2	TRUE	FALSE	4.658125877380371
Perceptron	0.48703639111731195	2	TRUE	FALSE	1.1430649757385254
Logistic Regression	0.47058481571910243	1	TRUE	TRUE	7.986335039138794
Perceptron multicouche	0.43169398907103823	1	TRUE	FALSE	30.148328065872192
MultinomialNB	0.3249040809208232	1	TRUE	TRUE	0.38156580924987793

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule	Temps_execution(secondes)
Random Forest	0.1926520172073015	2	FALSE	FALSE	10.47055697441101
Decision Tree	0.1926520172073015	2	FALSE	FALSE	5.070909023284912

À partir de ce résultat, on peut voir que la tâche de multi-classification *linear_svc* pour cet ensemble de données a d'excellentes performances, en particulier lors de l'utilisation des fonctionnalités Bi-gram. Perceptron et la régression logistique ont une précision similaire et sont également de bons classificateurs, et le temps d'exécution de Perceptron est encore très court. *Perceptron multicouche* et *MultinomialNB* ont des performances médiocres, mais le temps d'exécution de *MultinomialNB* est le plus court parmi tous les classificateurs. *La forêt aléatoire* est aussi mauvaise que *l'arbre de décision*, et la même précision est obtenue dans les mêmes conditions de fonctionnalité, et le temps d'exécution de *la forêt aléatoire* est le double de celui de *l'arbre de décision*. D'une manière générale, dans la multi-classification de cet ensemble de données, les deux caractéristiques du bi-gramme et de la suppression des mots vides sont plus importantes.

Sans en considérant les annotations POS :

Les cinq méthodes qui ont les taux de précisions plus haut :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule
linear_svc	0.5264504127427043	1	TRUE| FALSE	TRUE| FALSE
linear_svc	0.5227299151261481	2	TRUE| FALSE	TRUE| FALSE
linear_svc	0.509010580165097	3	TRUE| FALSE	TRUE| FALSE
Logistic Regression	0.47942099755842343	1	TRUE| FALSE	TRUE| FALSE
Perceptron	0.4658179281478898	2	TRUE| FALSE	TRUE| FALSE

Les fonctionnalités utilisées par chaque classificateur pour obtenir sa plus grande précision :

Classificateur	Accuracy	N-gram	Enlever stop-words	En minuscule	Temps_execution(secondes)
linear_svc	0.5264504127427043	1	TRUE| FALSE	TRUE| FALSE	1.605010747909546
Logistic Regression	0.509010580165097	3	TRUE| FALSE	TRUE| FALSE	99.713886022556775
Perceptron	0.4658179281478898	2	TRUE| FALSE	TRUE| FALSE	1.2092349529266357
Perceptron multicouche	0.44082083478665274	1	TRUE| FALSE	TRUE| FALSE	27.342951059334143
MultinomialNB	0.35984187885129637	1	TRUE| FALSE	TRUE| FALSE	0.38156580924987793
Decision Tree	0.16439948843157773	1	FALSE	FALSE	0.9701449871063232
Random Forest	0.13951866062085805	1	TRUE| FALSE	TRUE| FALSE	6.336454868316665

D'après ce résultat, dans les données qui ne considèrent pas POS, la casse des mots et la présence ou l'absence de mots vides n'ont aucun effet lorsque les mots sont convertis en vecteurs, car il est indiqué dans les six classificateurs si ces caractéristiques sont considérées ou non. Le taux de précision est constant et la caractéristique la plus importante est Ngram. *Linear_svc* est toujours le classificateur avec le taux de précision le plus élevé, mais il n'est pas aussi bon que les performances dans l'ensemble de données compte tenu du POS, et les performances de la régression logistique sont pires que lors de l'examen de l'ensemble de données du POS, mais parce que le meilleur résultat est l'utilisation de Tri-gram, en conséquence ça marche lentement. La précision de *Perceptron multicouche* et *MultinomialNB* est plus élevée dans cette classification, et la précision des autres classificateurs n'est pas aussi élevée que dans l'ensemble de données précédent qui ne tient pas compte du POS.

Dans cet ensemble de données, il existe un phénomène courant de *Overfit*, en particulier pour *Perceptron* et *linear_svc*, le taux de précision le plus élevé de leur ensemble d'apprentissage est de 98 %, tandis que dans l'ensemble de test, ils ne peuvent atteindre que le taux de précision le plus élevé d'environ 50 %. Pour les modèles qui fonctionnent mal dans l'ensemble de test, leur précision dans l'ensemble d'apprentissage ne sera pas particulièrement élevée. Par exemple, la précision de *l'arbre de décision* dans l'ensemble d'apprentissage n'est que de 2 % supérieure à celle de l'ensemble de test.

De plus, pour chaque traitement d'entité et le classificateur auquel il s'applique, il existe une matrice de confusion correspondante pour critiquer les détails de classification de chaque classe. En raison du grand nombre, elle ne peut pas être affichée ici. Veuillez référer au fichier de code Afficher dans .

Conclusion et Perspectives

L'objectif principal de ce project est de comprendre en détail les avantages et les inconvénients de chaque algorithme de classification utilisé et le rôle de chaque paramètre, afin qu'à l'avenir, nous sachions effectuer une série de pré-traitements sur le texte et sélectionner le modèle de classification pour obtenir le résultat souhaité.

Selon le classificateur formé pour chacune des différentes caractéristiques, grâce à la matrice de confusion correspondante, nous pouvons connaître la précision et le taux de rappel et le score F1 pour chaque catégorie dans chaque modèle. Cependant, les modèles ne sont pas parfaits. Par exemple, dans les tâches de multi-classification, il est susceptible de bien fonctionner pour une certaine catégorie mais de ne pas bien fonctionner pour une certaine catégorie. Cependant, nous pouvons trouver des méthodes de traitement des caractéristiques et des classificateurs avec une précision et des taux de rappel relativement élevés pour les catégories qui nous intéressent le plus en fonction de nos besoins.

Les trois classificateurs *linear_svc*, *Logistic Regression* et *Perceptron* sont les trois classificateurs les plus performants dans tous les aspects testés dans ce projet. Parmi eux, la machine à vecteurs de support a le meilleur effet, car quel que soit le type de tâche de classification, quel ensemble de données et quel type de pré-traitement de texte, son taux de précision est le plus élevé et le temps d'exécution n'est pas long. Sans surprise, Random Forest et Decision Tree sont les résultats les moins satisfaisants dans chacun d'eux, et le temps d'exécution est encore long. Une fois la fonctionnalité trigramme adoptée, elle doit encore passer des dizaines de minutes d'attente. La performance de MultinomialNB et de Perceptron multicouche a été plus médiocre que prévu. Mais il convient de noter que MultinomialNB a une performance moyenne dans la précision de l'ensemble de test et de l'ensemble d'apprentissage, et il n'y a pratiquement pas de sur-ajustement. La précision de Perceptron multicouche n'est pas mauvaise, mais son temps d'exécution est beaucoup plus long que les autres classificateurs ; tandis que la précision de MultinomialNB n'est généralement pas aussi bonne que celle de Perceptron multicouche, mais son temps d'exécution est très rapide.

La segmentation des mots et le nettoyage du texte sont à la base de la formation d'un bon modèle, Pour une langue spécifique, nous devons utiliser une méthode de traitement spécifique, et nous devons également tenir compte des signes de ponctuation et des idiomes Internet, tels que 《troooooop》, 《tkt》. En plus des diverses fonctionnalités utilisées ci-dessus, afin d'obtenir un classificateur avec la plus grande précision et des performances uniformes dans tous les aspects, vous pouvez envisager d'ajouter du texte lors de la formation du modéliser les fonctionnalités POS des mots, ou lemmatiser sur les mots du texte (je l'ai essayé en TD dans cette leçon, ça marche bien). Selon les résultats de cette tâche, on peut en déduire que dans les tâches de multi-classification, pour la plupart des classificateurs, il suffit d'utiliser unigram ou Bi-gram pour les caractéristiques N-gram, de sorte que le volume vectoriel généré est relativement petit et que le temps d'exécution sera plus rapide.

De plus, plus les données doivent être divisées en catégories, plus il est difficile de former un modèle avec une grande précision. De plus, les données de classification de texte présentent souvent un déséquilibre de catégorie, c'est-à-dire que le nombre d'échantillons de certaines catégories est beaucoup plus important que celui d'autres catégories. Cela entraînera un biais du modèle en faveur

de la classe majoritaire, ce qui aggravera l'effet de classification de la classe minoritaire. Par conséquent, nous devrions explorer comment résoudre le problème d'asymétrie dans la classification des textes par l'échantillonnage, l'ajustement des poids et d'autres méthodes.

En plus des méthodes de classification de l'apprentissage automatique, je continuerai à essayer d'utiliser des méthodes d'apprentissage plus approfondies, telles que les méthodes de classification de texte utilisant des réseaux de neurones convolutifs (tels que TextCNN, FastText) et les méthodes de classification de texte de réseaux de neurones récurrents (tels que LSTM , GRU). Ces méthodes ont obtenu d'excellents résultats dans de nombreuses tâches de classification de texte.

En général, la classification de textes est une tâche de développement continu, et il existe de nombreuses directions de recherche intéressantes à explorer à l'avenir.

Bibliographie

Go-to Guide for Text Classification with Machine Learning. (2020, 2 mars). MonkeyLearn Blog. https://monkeylearn.com/blog/text-classification-machine-learning/

Bansal, S. (2022, 15 juin). *A Comprehensive Guide to Understand and Implement Text Classification in Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

Jason Brownlee. (2020, août 20). *4 Types of Classification Tasks in Machine Learning*. Python Machine Learning. https://machinelearningmastery.com/types-of-classification-in-machine-learning/

Gong, D. (2022, 12 juillet). *Top 6 Machine Learning Algorithms for Classification*. Medium. https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

ProjectPro. (2022, 27 juin). *Machine Learning NLP Text Classification Algorithms and Models*. https://www.projectpro.io/article/machine-learning-nlp-text-classification-algorithms-and-models/523

Shaikh, J. (2018, 21 juin). *Machine Learning, NLP : Text Classification using scikit-learn, python and NLTK.* Medium. https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

Annexes

Le dossier code contient trois fichiers .ipynb du code écrit pour chaque jeu de donnée, et le fichier json généré se trouve également dans ce dossier. Selon le fichier dont le nom correspond à l'ensemble de données, double-cliquez pour ouvrir le code et les résultats de l'ensemble de données que vous souhaitez voir. L'ordre des trois codes est d'abord le pré-traitement du texte ensuit former des vecteurs des différente caractéristiques, enfin, pour chaque vecteur généré, former les 7 modèles et obtenir les résultats de prédiction de l'ensemble de test et les scores d'évaluation associés. Il vous suffit de cliquer sur le bouton exécuter dans l'ordre de chaque cellule pour obtenir le résultat souhaité. Afin d'éviter un temps d'attente trop long, j'ai préalablement stocké les résultats dans les fichiers json.