Rapport du projet de classification des textes

Chengwanli YANG

Master 1

Méthodologie de la recherche en informatique

M1SOL23

Sorbonne Université —Faculté des Lettres

Décembre 2020

Table des matières

1. Introduction

De nos jours, l'internet est de plus en plus indispensable dans notre vie. Tous les jours, nous recevons plusieurs mails, parfois il y a des courriels indésirables que nous appelons aussi le spam. Selon la CNIL (Commission Nationale de l'Informatique et des Libertés), le spam est l'envoi massif de courriers électroniques non sollicités, à des personnes avec lesquelles l'expéditeur n'a jamais eu de contact et dont il a capté l'adresse électronique de façon irrégulière. Heureusement ces spams ne sont pas dans la réception de boîte, c'est parce qu'ils ont été filtrés par le serveur. Comment le serveur détecte automatiquement le spam ?

Dans la vie quotidienne, chacun s'exprime sur les réseaux sociaux (Twitter, Facebook), des tweets peuvent représenter le registre de langue, la stylistique de l'utilisateur, également son colère, sa joie ou sa tristesse. Comment traiter le sentiment derrière ces 140 caractères ?

Le point commun des réponses de ces 2 questions est la classification. Le serveur de fournisseur classifie le spam et le ham, des tweets sont classifiés en valeur négative et positive. Face aux grandes données variées, nous avons besoin de l'apprentissage automatique (machine learning) afin de rendre le travail plus efficace.

Pour la réalisation de ce projet, je travaillerai en 2 parties : détecter le SPAM et identifier le sentiment des tweets français, en traitant le jeu de donnée « Spam-Email-Classifier » de Balakishan Molankula[1] et l'autre « French Twitter Sentiment Analysis » de Hadry[2] contenant plus d'un million de tweets français. Pour cette tâche, j'utiliserai le langage python, et la bibliothèque Scikit-learn, réaliserai sur Jupyter notebook.

Dans ce rapport, je vais d'abord expliquer les 2 jeux de données, leurs tailles et des statistiques (nombre d'instances, phrases, mot). Et puis je présenterai les méthodes utilisées, des pré-traitements, des caractéristiques et des classifieurs différents. Enfin, j'analyserai les résultats et donnerai des perspectives.

---

[1] https://raw.githubusercontent.com/Balakishan77/Spam-Email-Classifier/master/spamham.csv

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

2. Jeux de données

2.1 Etat de l'art

La classification automatique est une partie essentielle de l'apprentissage automatique, elle appartient à l'apprentissage supervisé, c'est-à-dire l'apprentissage d'une fonction de prédiction à partir d'exemples annotés. Tout d'abord cette tâche consiste à attribuer une classe à chaque objet, en se basant sur des données statistiques. L'exemple le plus connu est le jeu de donnée des fleurs d'iris, avec 4 dimensions (largeur et longueur de sépales et pétales), la fleur d'iris est classée en 3 catégories.

De plus il existe des algorithmes différents à utiliser. Maintenant je présente les algorithmes que je vais utiliser. Premièrement le « Perceptron » (Frank Rosenblatt, 1975), c'est un algorithme d'apprentissage supervisé, il s'agit d'un neurone formel muni d'une règle d'apprentissage qui permet de déterminer automatiquement les poids synaptiques de manière à séparer un problème d'apprentissage supervisé[3]. Deuxièmement la « Naive Bayes » (Thomas Bayes, 1763), c'est un type de classification bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance des hypothèses. Un classifieur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques[4]. Troisièmement le « Logistic Regression » (Joseph Berkson), c'est un modèle de régression binomiale, il s'agit de modéliser au mieux un modèle mathématique simple à des observations réelles nombreuses[5]. Grâce à la bibliothèque Scikit-learn, il est facile de les utiliser.

En outre, il y a eu un autre résultat obtenu du jeu de données de « French Twitter Sentiment Analysis », le tutoriel « CamemBERT for French Tweets classification »[6] , le traitement avec le modèle CamemBERT. Ce modèle est spécial pour le français, à la base du modèle de BERT.

---

[3] https://fr.wikipedia.org/wiki/Perceptron#Règle_d'apprentissage_du_perceptron_(loi_de_Widrow-Hoff)

[4] https://fr.wikipedia.org/wiki/Classification_naïve_bayésienne#Classification_et_catégorisation_de_documents

[5] https://fr.wikipedia.org/wiki/Régression_logistique

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

2.2 Description des jeux de données

Les jeux de données « Spam-Email-Classifier » et « French Twitter Sentiment Analysis », sont en fichier csv, ci-après dénommé « spamham » et « tweets ». Le « spamham » contient 2 classes : « ham » et « spam », 5728 instances, au total 37303 mots. La valeur 1 est distribuée à la classe « spam » et la valeur 0 pour la classe « ham ». Il y a 1368 instances dans la classe « spam » et 4360 instances dans la classe « ham ». Dans ce travail, nous prenons 4009 instances à entraîner et 1719 instances à tester.

Un exemple typique de spam dans le corpus : *Subject: are you ready to get it ? hello ! **viagra** is the # 1 med to struggle with mens ' erectile dysfunction . like one jokes sais , it is stronq enouqh for a man , but made for a woman ; -) orderinq viagra online is a very convinient , fast and secure way ! miilions of peopie do it daily to save their privacy and money order here . . .* Dans ce message, le mot « Viagra » est un des mots augmentant le plus un spam score, évidemment il est distribué la valeur 1, autrement dit le spam.

Le corpus « tweets » contient aussi 2 classes : « positif » et « négatif », 1526724 instances, au total 252741 mots. La valeur 1 est distribuée à la classe « positif » et la valeur 0 pour la classe « négatif ». Il y a 771604 instances dans la classe « négatif » et 755120 instances dans la classe « positif ». Nous prenons 1068706 instances à entraîner et 458018 instances à tester.

Dans ce jeu de donnée, le sentiment est évident au tweet. Par exemple : *C'est le pire des pires. Je voudrais juste savoir où il se trouve. si triste. Je suis à la maison malade avec la grippe. Personne ne me parle sur le but.* Ce sont des tweets ayant le sentiment négatif, ce qui contient les mots péjoratifs « pire », « triste », « malade ». Et des tweets positifs disposent des mots mélioratifs : *Merci! Je suis **content** que vous ayez aimé ces photos. Lisez le livre. c'est vraiment **bien**. Je pense que la fin aurait pu être moquée un peu - mais cela vous laisse avec **la paix**.*

3. Méthode

3.1 Pré-traitement

Pour pré-traiter les jeux de donnée « spamham » et « tweets », 2 façons sont utilisées: l'enlèvement de stopwords et la conversion des caractères en minuscule. Les stopwords sont les mots non significatifs figurants dans un texte, ce sont souvent les prépositions, les articles, les pronoms. La conversion des caractères en minuscule permet de diminuer le nombre de mot à traiter, et d'économiser la mémoire.

3.2 Extraction des caractéristiques

Le sac de mots (Bag of words) est la méthode utilisée pour extraire les caractéristiques d'un texte. Elle permet de représenter numériquement nos données contextuelles aux modèles de l'apprentissage automatique, mais elle ne prend pas en compte l'ordre et la structure des mots dans le corpus. Tout d'abord, pour obtenir le sac de mots, il existe 2 moyens différents : « CountVectorizer » et « TfidfVectorizer ». Le « CountVectorizer » est la méthode plus utilisée pour la conversion d'une collection du texte en une matrice de nombres de token en gâchant l'ordre des mots . Et le « TfidfVectorizer » sert à la convertir en une matrice de TF-IDF, il donne une importance plus grande à certains mots, nous l'appelons aussi la méthode de pondération. De plus, le N-gramme de mot est une caractéristique importante. Les tokens dans le sac sans ordre, pour les traiter mieux avec le bon sens, nous avons besoin de N-gramme. Je vais utiliser l'unigramme, le bigramme et le trigramme afin de trouver la meilleure solution.

3.3 Classifieurs utilisés

Dans ce projet, j'ai utilisé 3 classifieurs : « Perceptron », « Naive Bayes » et « Logistic Regression ».

« Perceptron » est un classifieurs binaire, simplement une ligne sépare deux classes.

« Naive Bayes » est l'algorithme utilisé souvent au filtrage de spam.

« Logistic Regression » est plus efficace pour la classification binaire.

4. Résultats

4.1 Jeu de donnée— « spamham »

J’ai testé d’abord le N-gramme pour trouver le meilleur cas.

Ngram_range : (1, 1)
  Perceptron classifier : 0.9837
  naive_bayes classifier : 0.9878
  Logistic Regression classifier : 0.9884

Ngram_range : (1, 2)
  Perceptron classifier : 0.9802
  naive_bayes classifier : 0.9907
  Logistic Regression classifier : 0.9872

Ngram_range : (1, 3)
  Perceptron classifier : 0.9831
  naive_bayes classifier : 0.9895
  Logistic Regression classifier : 0.9855

Nous constatons qu’avec l’unigramme, le score est le plus meilleur, ensuit le test des stopwords et de mise en minuscule.

Stopwords False, Minuscules : False
  Perceptron classifier : 0.9855
  naive_bayes classifier : 0.9436
  Logistic Regression classifier : 0.9034

Stopwords False, Minuscules : True
  Perceptron classifier : 0.9488
  naive_bayes classifier : 0.8901
  Logistic Regression classifier : 0.9005

Stopwords True, Minuscules : False
  Perceptron classifier : 0.9511
  naive_bayes classifier : 0.9488
  Logistic Regression classifier : 0.9331

Stopwords True, Minuscules : True
  Perceptron classifier : 0.9506
  naive_bayes classifier : 0.9383
  Logistic Regression classifier : 0.9255

Nous trouvons que nous enlevons le stopsword et ne convertissons pas toutes les caractères en minuscule, les scores sont plus meilleurs que les autres, autrement dit « Stopwords True, Minuscules False ». Donc la caractéristique plus efficace est l'unigramme, avec le pré-traitement : l'enlèvement de stopword et non convertir caractère en minuscule.

De plus, avec les 3 classifieurs, nous recevons le résultat ci-dessous:

CountVectorizer						
		precision	recall	f1-score	confusion matrix	
					ham	spam
Perceptron	ham	0.99	0.98	0.99	1294	7
	spam	0.95	0.98	0.97	20	398
naive_bayes	han	1	0.99	0.99	1298	5
	spam	0.96	0.99	0.97	16	400
Logistic Regression	han	0.99	0.99	0.99	1304	11
	spam	0.98	0.97	0.97	10	394
TfidfVectorizer						
		precision	recall	f1-score	confusion matrix	
					ham	spam
Perceptron	ham	0.99	0.99	0.99	1301	19
	spam	0.97	0.95	0.96	13	386
naive_bayes	han	0.89	1	0.94	1313	165
	spam	1	0.59	0.74	1	240
Logistic Regression	han	0.97	1	0.98	1313	44
	spam	1	0.89	0.94	1	361

Tableau 1. Résultat de « spamham »

Nous voyons premièrement que les classifieurs marchent bien pour ce corpus, les résultats sont bons, avec la méthode « TfidfVectorizer », le classifieur « Naive Bayes » a bien classé le ham mais a fait plus d'erreurs sur le spam. D'après la matrice de confusion, il a considéré 165 messages indésirables comme le ham, par conséquent le rappel de spam est seulement 0.59. Deuxièmement, les résultats avec la méthode « CountVectorizer » sont plus stables, puisque les f1-socre de ham et spam sont plus proches que ceux de « TfidfVectorizer » pour tous les classifieurs. Troisièmement, il est difficile de dire définitivement quel classifieur le plus efficace selon le résultat, la frontière est ambiguë. Il semble que le classifieur « Logistic Regression » avec la méthode « CountVectorizer » donne le meilleur résultat, puisque ses scores (précision, rappel, f1) sont plus élevés. Cependant personnellement je préfère recevoir tous les hams malgré quelques spam, au lieu de trouver un ham

étant dans le message indésirable. Ainsi, de ma part, le classifieur « Naive Bayes », avec la méthode « CountVectorizer », présente le résultat satisfaisant, la précision de ham vers 1 (0.996), c'est-à-dire qu'il a classé correctement 1298 ham sur 1303 messages. La précision de spam est aussi bonne (0.96), les rappels de ham et spam sont plus élevés que les autres classifieurs.

4.2 Jeu de donnée— « tweets »

Tout d'abord le test de N-gramme et les stopwords et la mise en minuscule:

Ngram_range : (1, 1)
  Perceptron classifier : 0.7118
  Logistic Regression classifier : 0.7871
  naive_bayes classifier : 0.7692

Ngram_range : (1, 2)
  Perceptron classifier : 0.7587
  Logistic Regression classifier : 0.8025
  naive_bayes classifier : 0.7853

Ngram_range : (1, 3)
  Perceptron classifier : 0.7771
  Logistic Regression classifier : 0.8023
  naive_bayes classifier : 0.7883

Stopwords False, Minuscules : False
  Perceptron classifier : 0.6465
  Logistic Regression classifier : 0.5391
  naive_bayes classifier : 0.6462
Stopwords False, Minuscules : True
  Perceptron classifier : 0.6313
  Logistic Regression classifier : 0.5429
  naive_bayes classifier : 0.6588
Stopwords True, Minuscules : False
  Perceptron classifier : 0.6465
  Logistic Regression classifier : 0.5551
  naive_bayes classifier : 0.6222
Stopwords True, Minuscules : True
  Perceptron classifier : 0.5926
  Logistic Regression classifier : 0.5561
  naive_bayes classifier : 0.6335

Nous constatons qu'avec la caractéristique l'unigramme, le bigramme et le trigramme ainsi que le pré-traitement : « Stopwords False, Minuscules True », le score est le plus meilleur, donc la classification sera sous cette condition.

De plus, avec les 3 classifieurs, nous recevons le résultat ci-dessous:

CountVectorizer						
		precision	recall	f1-score	confusion matrix	
					negatif	positif
Perceptron	negatif	0.79	0.75	0.77	174112	44991
	positif	0.76	0.8	0.78	57102	181813
naive_bayes	negatif	0.77	0.83	0.8	191438	57187
	positif	0.81	0.75	0.78	39776	169617
Logistic Regression	negatif	0.81	0.8	0.8	184399	43716
	positif	0.8	0.81	0.8	46815.	183088
TfidfVectorizer						
		precision	recall	f1-score	confusion matrix	
					negatif	positif
Perceptron	negatif	0.8	0.76	0.78	174609	43455
	positif	0.76	0.81	0.79	56605	183349
naive_bayes	negatif	0.76	0.86	0.8	198635	64317
	positif	0.83	0.72	0.77	32579	162487
Logistic Regression	negatif	0.82	0.8	0.81	185041	40499
	positif	0.8	0.82	0.81	46173	186305

Tableau 2. Résultat de « tweets »

Nous constatons premièrement le résultat global est le moins bon que celui de jeu « spamham », l'intervalle est entre 0.7-0.8, c'est peut-être parce que le corpus a plus de bruits, précisément le registre des tweets est plus varié : le verlan, le symbole spécifique, même la faute d'orthographe, tout ce qui augmente la difficulté de la classification. Deuxièmement, les résultats de la méthode « TfidfVectorizer » sont légèrement meilleurs que ceux de « CountVectorizer ». La raison est le fait que la méthode « TfidfVectorizer » a donné l'importance à certains mots, ces mots apportaient des informations plus importantes que les autres, cela peut être le mot d'émotion, l'adverbe… En conséquence la classification est plus efficace. Troisièmement le classifieur « Logistic Regression » présente le meilleur résultat avec 2 méthodes de vectorisation. Selon la matrice de confusion, parmi les 231214 tweets négatifs, il a classé justement 184399 tweets avec « CountVectorizer », 185041 tweets avec « TfidfVectorizer ». Nous classons un tweet soit positif soit négatif, ce algorithme est adapté à cette classification, puisqu'il est un modèle de régression binomiale, à la fois son temps de calcul est plus vite que les autres classifieurs.

5. Conclusion et perspectives

Les 3 classifieurs « Perceptron », « Naive Bayes » et « Logistic Regression » marchent pas mal aux 2 jeux de données « spamham » et « tweets », ils donnent les résultats satisfaisants. Grosso modo, les résultats de l'algorithme « Logistic Regression » sont les meilleurs, il a bien classé le ham, le spam et le tweet négatif, le tweet positif. En pratique, ce classifieur est intéressant parce qu'il est intrinsèquement interprétable, en exemple de jeu « tweets » : lorsque l'on ajoute une instance à valeur positive, la prédiction augmente, et inversement pour une instance négative. Par contre le classifieur « Naive Bayes » répond à ma demande personnelle du filtrage de spam. Et puis, les caractéristiques jouent le rôle important. Le sac de mots et le N-gramme influence sur le résultat de la classification. J'ai traité le corpus « spamham » avec l'unigramme qui présente le meilleur score, et pour le jeu « tweets », l'unigramme, le bigramme, le trigramme ont été utilisés.

Ce travail pourrait quand même être amélioré. D'abord l'augment de la pureté du sac de mots. C'est-à-dire plus de pré-traitements : l'enlèvement maximum des éléments dont nous n'avons pas besoin. Notamment le corpus « tweets », il existe des symboles et des mots spécifiques, en exemple : *Ça va être un* ***looooooooooooooooooooooooooooooonnnnnnngggggggg*** *nuit au travail. A parié 5* ***£*** *que je ne pouvais pas rester tranquille pour tout un quart de travail au travail !! J'ai gagné ... et j'ai payé* ***£*** *5 mais ... iv j'ai perdu mon porte-monnaie, mon téléphone et ma puce!* Une fois de les enlever, le sac de mots serait plus pur. De plus cette méthode pourrait être appliqué à d'autres jeux de données (fleur d'iris, classification des fruits), le choix de classifieur est important. Par exemple l'algorithme « Decision tree » est une manière explicite aux décisions réalisées et aux processus. Il est plus efficace afin de classer 3, 4 ou plus de catégories. En outre, pour des jeux de données en autres langues, surtout les langues en lettre latine (italien, espagnol…), la méthode marcherait assez bien. Cependant les langues en caractère (chinois, coréen…), elle ne serait pas suffisante. Puisque les mots de ces langue ne sont pas séparés par l'espace, la méthode split ne fonctionne plus. Il est nécessaire d'importer la méthode spécifique pour découper le token, en exemple jieba pour le chinois. À la fois il faudrait enlever des ponctuations particulières (。 ， 【】 …).

Finalement l'apprentissage automatique rend la classification du texte plus efficace, d'après ce travail, des caractéristiques sont excessivement considérables, le nettoyage de corpus et le choix des classifieurs sont la clé de succès.

6. Bibliographie

(1) Rémy Kessler, Juan Manuel Torres-Moreno, Marc El-Bèze. (s. d.). Classification automatique de courriers électroniques par des méthodes mixtes d' apprentissage. juanmanuel.torres.free.fr. Consulté le 29 décembre 2020, à l'adresse http://juanmanuel.torres.free.fr/downloads/RSTI_ISI_kessler_torres_elbeze.pdf

(2) S. Seth and S. Biswas, "Multimodal Spam Classification Using Deep Learning Techniques," 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Jaipur, 2017, pp. 346-349, doi: 10.1109/SITIS.2017.91.

(3) M. Kanakaraj and R. M. R. Guddeti, "NLP based sentiment analysis on Twitter data using ensemble classifiers," 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, 2015, pp. 1-5, doi: 10.1109/ICSCN.2015.7219856.

(4) Benamara, Farah and Grouin, Cyril and Karoui, Jihen and Moriceau, Véronique and Robba, Isabelle Analyse d'opinion et langage figuratif dans des tweets : présentation et résultats du Défi Fouille de Textes DEFT2017. (2017) In: Atelier TALN 2017 : Défi Fouille de Textes (DEFT 2017), 26 June 2017 - 26 June 2017 (Orléans, France).

(5) Aurélien Massiot, Léa Naccache. (2020, 5 mai). NLP : une classification multilabels simple, efficace et interprétable. www.blog.octo.com. https://blog.octo.com/nlp-une-classification-multilabels-simple-efficace-et-interpretable/

7. Annexes

Pour chaque jeu de donnée, mon code est découpé en 2 fichiers. L'un est composé par le pré-traitement et l'extraction des caractéristiques, l'autre est les résultats des 3 classifieurs (matrice de confusion, précision, rappel et f1-score).