École Polytechnique de Montréal

Baccalauréat en Génie Logiciel

Cours - LOG3430

TP1 spam/ham

Auteurs :

Maxime Tremblay-Rheault - 1946878
Maude Desrosiers-C. - 1946801
groupe labo B1

4 octobre 2020

Rapport de laboratoire LOG3430 Groupe B1

Introduction

Dans le cadre du cours log3430 un système de détection de courriels ‘spam’ a été implémenté. Celui-ci repose sur un modèle de prédiction probabiliste simple faisant intervenir le théorème de Bayes. Le programme est construit en plusieurs modules écrit en language python. Les prédictions effectuées à partir d’un échantillon donné ont montré un taux de succès dépassant 95% de succès.

Fonctionnement du système

Le programme est composé de quatre modules principaux.
D’abord, le module crud.py prend en charge toute la gestion de la lecture et de l’écriture dans les fichiers de données fournis en format .json. Ensuite, un répertoire de tout le vocabulaire fournir par le jeu de test est créé à l’aide du module vocabulary.py. Celui-ci s’avère utile afin de connaître un composant de la formule de Bayes, soit P(w1|spam) ou P(w1|ham). Le module qui réalise l’analyse mathématique se nomme email_analyser. Enfin, le module renege.py exécute des appels de fonctions d’autres modules, tel que crud et email_analyser, afin de traiter des emails entrants. Il attribue également des groupes aux expéditeurs de courriels. Le module renege peut également attribuer un niveau de confiance aux utilisateurs et aux groupes.

Afin de faciliter le travaille de comparaison de mots par le programme, un module d’épuration de texte nommé text_cleaner.py a été fourni. Sans celui-ci, l’analyse faite par email_analyser échoue. Il est à noter qu’un module dédié à la création de jeux de données a été conçu à des fins de test. Le module create_800-200.py crée deux fichiers .json à partir de celui fourni (1000-mails.json).

Calcul des probabilités

Tel que mentionné, le module email_analyser effectue le calcul de probabilité en se basant sur le théorème de Bayes. Voici la formule utilisée:

Théorème de Bayes : $P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}$

Adaptation pour le calcul du spam : $P(spam \mid message) = \frac{P(message \mid spam) * P(spam)}{P(message)}$

où $P(message)$ est $P(w1 \cap w2 \cap ... \cap wn) = P(w_1)P(w_2)...P(w_n)$

En supposant que les mots sont indépendants entre eux et en développant les message au numérateur on obtient:

$$P(spam \mid w1 \cap w2 \ldots \cap wn) = \frac{P(w1 \mid spam) * P(w2 \mid spam) * \ldots * P(wn \mid spam) * P(spam)}{P(message)}$$

$P(w1)$ est calculé par l'approximation suivante : $\frac{nombre\ d'occurences\ du\ mot\ dans\ le\ jeu\ de\ données}{nombre\ total\ de\ mots\ dans\ le\ jeu\ de\ données}$

Ceci est équivalent à la formule traditionnellement proposée en probabilité :
$P(w1 \mid spam) * P(spam) + P(w1 \mid ham) * P(ham)$

Lors de la conception du programme, la multiplication répétée de nombre décimaux a mené à une situation de *underflow* qui faussait les résultats.
Pour pallier cette situation problématique, nous avons eu recours à l'emploi de logarithmes.

En utilisant la propriété mathématique : $x = e^{ln(x)}$

Et en utilisant la propriété d'addition des exposants : $e^{x+y} = e^x\ e^y$

On peut passer de :

$$P(spam \mid message) = \frac{P(w1 \mid spam) * P(w2 \mid spam) * \ldots * P(wn \mid spam) * P(spam)}{P(message)}$$

À:

$$P(spam \mid message) = \frac{e^{ln(P(w1 \mid spam))} * e^{ln(P(w2 \mid spam))} * \ldots * e^{ln(P(wn \mid spam))} * e^{ln(P spam))}}{e^{ln(P(message))}}$$

Ce qui est équivalent à :

$$P(spam \mid message) = e^{ln(P(w1 \mid spam)) + ln(P(w2 \mid spam)) + \ldots + ln(P(wn \mid spam)) + ln(P(spam)) - ln(P(message))}$$

Ce qui est intéressant pour résoudre la situation de underflow, est la possibilité de comparer deux probabilités en leur appliquant un ln. Puisque ln est une fonction continue et croissante, on peut appliquer ln des deux côtés de l'équation et utiliser $ln(P(spam \mid w1 \cap w2 \ldots \cap wn))$ pour comparer les probabilités.

$$ln(P(spam \mid message) = ln(P(w1 \mid spam)) + ln(P(w2 \mid spam)) + \ldots + ln(P(wn \mid spam) + ln(P(spam)) - ln(P(message))$$

Un autre détail technique est à noter. Lorsque la probabilité d'un mot ($P(w \mid spam)$) n'apparaît pas dans le vocabulaire , une valeur de $1/(nombreMots_{contexte} + 1)$ et été mise. $Contexte$ fait référence à une combinaison de où se trouve le message (body ou sujet) et du type de message (spam ou ham).

Définition de nouveaux groupes

Les groupes 'friends' et 'junk' ont été ajoutés au groups.json. Initialement, seul le groupe 'default' faisait partie du fichier .json, alors il a semblé juste d'y inclure de nouveaux groupes basés sur le taux de confiance accordé par le programme renege.py. Un seuil de 'trust' de 0.7 a été établi pour distinguer les 'amis' du courrier indésirables.
Voici un exemple de pseudo code :
if trust_level $> 0.7 \rightarrow$ friend_list.append(user)
Else $\rightarrow$ junk_list.append(user)

Proposition d'une façon de combiner les probabilités Sujet et Corps

Deux variables, *alpha* et *beta*, ont été attribuées aux probabilités relatives au corps et au sujet du message respectivement. Afin de conclure si un message est spam ou ham, il est nécessaire d'évaluer les probabilités du sujet et du corps pour ensuite les combiner. Le poids associé au sujet et au corps, donné par *alpha* et *beta*, peut influencer le résultat final de la prédiction. Un choix simple serait de faire $\frac{1}{2}Proba_{body} + \frac{1}{2}Proba_{subject}$.
Pour améliorer la performance du programme, notre équipe a fait plusieurs simulations avec des coefficients *alpha* et *beta* différents. Notez que *beta* = 1 - *alpha*.
En voici quelques exemple:

alpha $= 1$, beta $= 0$ (Seulement Sujet qui compte)
Accuracy: 0.9849246231155779
Precision: 0.987012987012987
Recall: 0.9743589743589743

alpha $= 0$, beta $= 1$ (Seulement Corps qui compte)
Accuracy: 0.7487437185929648
Precision: 0.6296296296296297
Recall: 0.8717948717948718

alpha $= 0.15$, beta $= 0.85$
Accuracy: 0.9849246231155779
Precision: 0.9746835443037974
Recall: 0.9871794871794872

Nous avons conclus que les coefficients alpha = 0.15 et beta = 0.85 étaient les plus appropriés. Ce choix repose d'une part sur les résultats obtenus (Accuracy: 0.9849) et d'autre part sur le fait qu'au delà du coefficient alpha de 0.15, les valeurs ne semblaient pas changer significativement. Cela laisse supposer que cette valeur représente le seuil où la contribution de la probabilité du sujet n'était plus importante. En résumé, nous avons décidé de garder des coefficients qui donnaient de bons résultats et qui prenaient en compte les deux parties du message, soit le corps et le sujet.

Évalutation de accuracy, precision et recall

Faux positifs, faux négatifs, vrai positif et vrai négatif

Afin de bien saisir les concepts d'évaluation du système, il est primordial de comprendre les 4 possibilités quant aux résultats. Ainsi un résultat peut être :

- Faux positif (FP): le e-mail est un ham et le résultat obtenu est erroné
- Faux négatif (FN): le e-mail est un spam et le résultat obtenu est erroné
- Vrai positif (TP) : le e-mail est un spam et le résultat obtenu est juste
- Vrai négatif (TN) : le e-mail est un ham et le résultat obtenu est juste

	Résultat obtenu par le système		
Résultat attendu		e-mail est un spam	e-mail n'est pas un spam
	e-mail est un spam	Vrai positif (TP)	Faux négatif (FN)
	e-mail n'est pas un spam	Faux positif (FP)	Vrai négatif (TN)

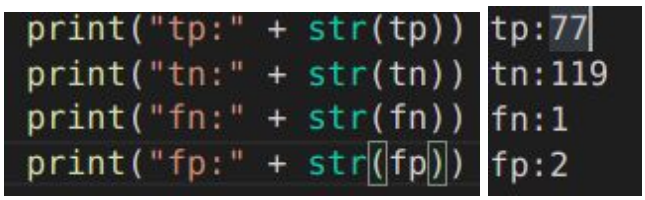
Il faut ainsi minimiser les faux positifs et les faux négatifs.

Accuracy (exactitude)

La valeur d'exactitude permet de déterminer à quel point les valeurs obtenues sont près des valeurs réelles. Mathématiquement, elle s'exprime par le nombre de résultats justes divisés par le nombre total de résultats.

$$(TP + TN) / (TP + TN + FN + FP)$$

Notre modèle a un très bon niveau d'exactitude se situant à 98%. Ceci signifie qu'il y a très peu de faux positifs et de faux négatifs. En effet, en faisant quelques appels à print() , il est possible de constater qu'il y a seulement 2 faux positifs et un faux négatif.



Precision (précision)

La précision est le ratio servant à mesurer l'impact des faux positifs. En effet, puisque celle-ci s'exprime comme suit, $(TP) / (TP + FP)$, elle s'approche de 100% si les faux positifs sont peu nombreux comparativement aux vrais positifs. Dans le cas de notre système, les faux positifs sont peu élevés et la précision a donc une valeur de 97%.

Recall

Cet indicateur tente de répondre à la question suivante: De tous les spams, quel pourcentage a été correctement identifié? Il fait donc appel aux faux négatifs qui sont des spams ayant étés mal identifiés. La valeur de *recall* s'approche de 100% si les faux négatifs sont peu nombreux comparativement aux vrais positifs.

$$(TP) / (TP + FN)$$

Dans le cas de notre système, les faux négatifs sont peu élevés et la précision a donc une valeur de 98.7%.

Améliorations possibles du système

Augmenter l'échantillon de classification

Afin d'améliorer le système, il serait possible d'augmenter le nombre de courriels servant à la classification. En effet, seulement 800 emails ont été utilisés à cette fin. Il serait envisageable d'entraîner notre système avec un nombre beaucoup plus élevé de courriel. Ceci permettrait d'avoir un vocabulaire plus garni et des probabilités (P(w|spam) et P(w|ham)) plus précises. De plus, un plus large échantillon de courriel pourrait éventuellement couvrir des mots dans plusieurs langues et ainsi créer un système plus versatile.

Prendre en compte la provenance du message

Il serait également possible de modifier la méthode is_spam() pour que la confiance en l'auteur du courriel influence le calcul des probabilités. Le calcul actuel prend uniquement compte des probabilités au niveau du corps du message et du sujet.

$$prob_{spam} = \alpha\, prob_{body} + \beta\, prob_{subject}$$

Il serait possible d'ajouter deux termes à cette formule pour prendre en compte la confiance à l'auteur du courriel et au groupe auquel fait partie cet expéditeur.

$$prob_{spam} = \alpha\, prob_{body} + \beta\, prob_{subject} + \gamma\, trust_{user} + \delta\, trust_{group}$$

Il faudrait alors ajuster les valeurs des coefficients $\alpha, \beta, \gamma \; et \; \delta$ pour arriver à éliminer complètement les faux positifs et les faux négatifs. Notez qu'une fois de plus, la somme des coefficients doit donner 1.