Mini projet 1 : You've got mail !

Fabien Tang
Valentin Colliard

25/02/18

1.1 Introduction

Lors de ce projet, nous avons étudié plusieurs problématiques d'apprentissage statistique (Machine Learning) en prenant comme application le traitement des mails et plus précisément la détection de mails spam/non spam.

1.2 Fonctions préliminaires

A l'aide des fichiers et fonctions fournis nous avons chargé et traité dans un premier temps les mails afin de n'obtenir dans des listes que leurs contenu.
Par la suite, nous avons réalisé une fonction split nous permettant de séparer ces listes en ensemble d'apprentissage et de test.

1.3 Classification d'emails et détection de spam

Afin de prédire si un mail est un spam ou non, nous avons d'abord réalisé un classifieur classant un mail X en fonction de sa longueur (nombre de mots le composant). Cette solution n'étant pas forcément adaptée à la détection de mail spam/non spam, elle nous servira notamment plus tard de comparaison pour la qualité de nos models.

Durant ce projet, nous utiliserons ainsi les notations suivantes :

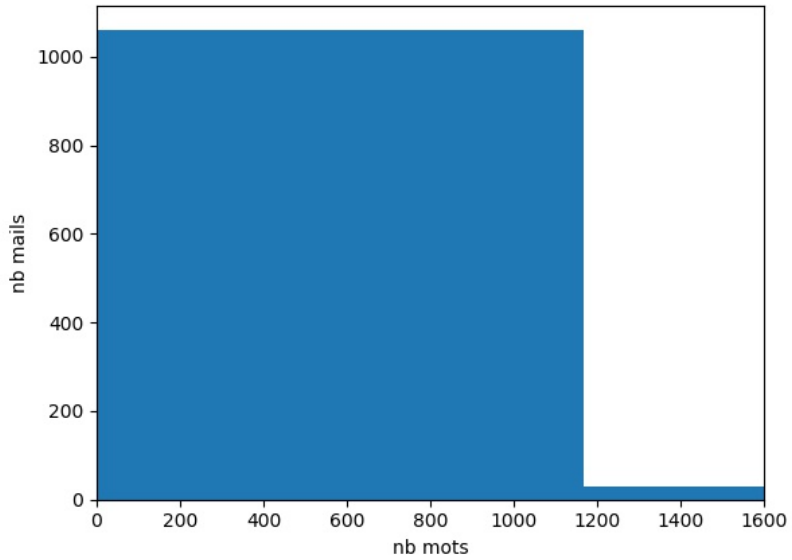
$P(Y = +1)$ = probabilité que le mail soit un spam.
$P(Y = -1)$ = probabilité que le mail ne soit pas un spam.
$P(X = x)$ = probabilité d'avoir un nombre de mots x.

L'objectif est donc de calculer la probabilité qu'un mail soit un spam sachant un nombre de mots x. D'après l'application de Bayes nous calculerons donc :

$P(Y = +1|X = x) = \frac{P(X=x|Y=+1)P(Y=+1)}{P(X=x)}$

$P(X = x|Y = +1)$ = probabilité d'avoir un nombre de mots x sachant que le mail est un spam.

Pour calculer ces probabilités, nous avons dû coder une fonction permettant de compter le nombre de mots d'un mail (soit le nombre d'espaces - 1).
Grâce à cette fonction, nous nous sommes rendu compte que la grande majorité de nos mails ne contenaient qu'un nombre de mots compris entre 0 et 1200.
Nous avons donc choisi d'ignorer les mails de plus de 1200 mots.



Afin de calculer la distribution $P(X = x|Y = +1)$, nous avons codé la fonction apprend_modele(spam,nonspam). 1200 valeurs différentes pour x nécessitant un espace mémoire conséquent, nous avons donc décidé de regrouper ces valeurs par intervalle de 50 lors de la composition de notre set d'apprentissage.

La fonction predit_email(emails,modele) se charge de compter le nombre de mots d'un mail en paramètre puis de comparer ce nombre avec les intervalles du modèle obtenu précédement pour en ressortir le label (spam/non spam) associé. Au final, nous avons obtenu les probabilités suivantes :

Predict/Result	Valeurs
spam/spam	164
non spam/non spam	619
spam/non spam	152
non spam/spam	367
nombre de mail test	1349
nombre de mail test ignoré (car nb mots>1200)	47
Accuracy	0.6013

(50% apprentissage et 50% test)

Malgrès que le score soit plutôt correcte, ces résultats ne sont pas forcément fiable car nous évaluons un mail en fonction du nombre de mots le composant et non sur sa sémantique.

1.4 Classification à partir du contenu d'un email

La méthode précédente ne prenant pas en compte la sémantique des mails, nous avons décidé dans cette seconde partie de réaliser un classifieur classant en fonction du contenu. Ainsi, chaque mail sera représenté par un vecteur binaire $(x1, x2, ..., xd) \in \{0, 1\}^d$, où chaque dimension i représentera un mot et la valeur xi indiquera la présence ou non du mot dans l'email.

Le principe du classifieur est le suivant :
- établir un set d'apprentissage contenant un maximum de mot associé à un label (spam/nonspam).
- déterminer pour un mail donné le nombre de mots spam/non spam le constituant.
==> S'il y a plus de mot de type spam, le mail sera alors considéré comme un spam et vice-versa.

Afin d'établir le set d'apprentissage, nous avons :
- définis la fonction compte_mot_coll(coll) permettant pour chaque mot de chaque mail d'une collection, de renvoyer le nombre de mails dans lequel ce dernier apparait.
- appliquer la fonction ci-dessus sur nos données de mail spam/non spam afin d'obtenir un set d'apprentissage de mot.

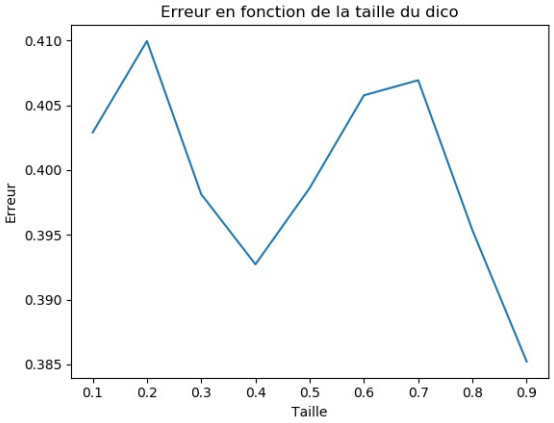
Ces opérations étant coûteuse en temps d'éxecution, nous avons utilisé des dictionnaires en prenant pour clé le mot en question et en valeur le nombre d'occurences. De plus, pour n'obtenir que des véritables mots et éviter les doublons, nous avons choisi de transformer les mots en minuscule et de ne considérer que ceux dont chaque caractère est compris dans l'alphabet.
Une fois notre set d'apprentissage établi, nous avons codé la fonction predict renvoyant "spam" ou "non spam" en fonction des mots composant un mail donné.
Score de précision obtenu :

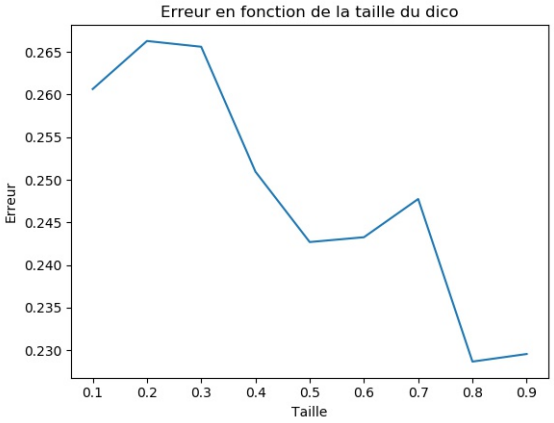
Predict/Result	Valeurs
spam/spam	214
non spam/non spam	771
spam/non spam	0
non spam/spam	317
nombre de mail test	1349
nombre de mail test ignoré (car nb mots>1500)	47
Accuracy	0.7565

(50% apprentissage et 50% test)

Ainsi que ce second classifieur est plus efficace que le premier du fait d'une meilleure accuracy.Nous en avons donc déduit qu'il était plus intéréssant de classer les mails en fonction de leur contenu que par rapport à leur taille.

D'autre part, nous avons remarqué dans les deux cas que plus la taille initial du dico servant à l'apprentissage est importante, plus l'accuracy augmente et donc plus le taux d'erreur (1-accuracy) diminue.





1.5 Visualisation

1.6 Conclusion

En conclusion, ce projet nous a permis d'étudier certain mécanisme d'apprentissage statistique. Les résultats des différents classfieurs nous ont permis d'affirmer que la classification à partir du contenu était beaucoup plus efficace que par la longueur. Ce projet a de plus mis en avant d'autres problématiques comme le pré-traitement des données (filtrage des mots avec accents, caractère spéciaux...) et l'optimisation de fonction (efficacité, coût, mémoire).