LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 1

Mise en place du système test

Département de génie informatique et de génie logiciel

École Polytechnique de Montréal

Automne 2020

1 Introduction

Dans ce premier travail pratique vous allez implémenter le système de filtrage du spam appelé RENEGE. Dans les prochains laboratoires, vous allez travailler sur le test et validation du système. Pour les tps vous allez utiliser Python. Si vous n'êtes pas à l'aise avec Python, voici un tutoriel : `https://www.programiz.com/python-programming`.

2 Objectifs

Les objectifs généraux de ce laboratoire sont :

1. Créer le système de filtrage de spam RENEGE.
2. Prendre connaissance des stratégies d'analyse du texte pour la classification spam/non-spam (ham).

3 Description générale

RENEGE (useR awarE bayesiaN filtErinG systEm) est un système de filtrage des spams qui tient compte l'utilisateur. Le système permet d'étiqueter les e-mails comme spam (non désirés) et ham (messages ok). Il s'agit essentiellement d'un classificateur bayésien associé à des règles heuristiques. Il est composé de plusieurs modules qui traitent, par utilisateur, la classification du spam et du ham.

À un niveau d'abstraction élevé, RENEGE lit un message électronique, en extrait le sujet, l'adresse de départ et, à l'aide d'un ensemble de règles et de probabilités, classe le message.

Dans RENEGE, les utilisateurs sont identifiés par leur adresse électronique. Chaque adresse électronique se voit attribuer un niveau de confiance, un nombre entier entre 0 et 100 :

— 0 signifie ne pas faire confiance du tout
— 100 signifie que cet utilisateur est sûr à 100 % et que tout message de sa part est sûr, quel qu'en soit le contenu

Le niveau de confiance de l'utilisateur est ajusté dynamiquement, mais peut également être forcé par le propriétaire de RENEGE. De cette façon, l'utilisateur peut forcer le message d'un ami, d'un membre de la famille ou d'une personne de confiance à circuler dans le système.

Le système prend également en charge les groupes : le courrier électronique peut être organisé en groupes, les étiquettes des groupes sont définies par l'utilisateur. Ainsi, ce dernier peut également attribuer un niveau de confiance par défaut à un groupe.

4 Principaux modules du système

Chaque module du système est implémenté dans le fichier python séparé. Les principaux modules du système sont les suivants :

- **Gestion des utilisateurs et groupes** : module pour réaliser la fonctionnalité CRUD (Create Read Update Delete) pour utilisateurs et groupes. Utilisateur est représenté par son adresse électronique et les informations qui lui sont associées. Groupe est une collection d'utilisateurs.
- **Traitement du texte** : module pour nettoyage et tokenisation du texte.
- **Création du vocabulaire** : module pour créer le vocabulaire avec la fréquence des mots trouvés dans les messages spam et ham.
- **Analyse des e-mails** : module pour le traitement des nouveaux e-mails. Réalise la logique d'ajout/mise à jour de l'information concernant les utilisateurs et groupes.
- **Calcule de probabilité de Bayes** : module pour calculer la probabilité que e-mail est spam ou ham.
- **Évaluation du système** : module principal pour gérer le système. Comprends une fonction d'évaluation de précision de détection du spam.

Gestion des utilisateurs et groupes

Ce module est réalisé dans le fichier crud.py. Les groupes et les e-mails peuvent être prédéfinis manuellement au format JSON et / ou mis à jour automatiquement. Le groupe par défaut est toujours présent et un nouvel e-mail sera toujours ajouté au groupe par défaut. Les groupes ainsi que les utilisateurs doivent être gérés dans le style CRUD : créer, lire, mettre à jour et supprimer des groupes et des utilisateurs.

Gestion utilisateurs

Un utilisateur est défini comme :

- Name : adresse e-mail.
- Trust : nombre entier entre 0 et 100.
- SpamN : nombre de messages de spam de cet utilisateur.
- HamN : nombre de messages ham de cet utilisateur.
- Date_of_first_seen_message : date du premier message vu (en unix timestamp)
- Date_of_last_seen_message :date du dernier message vu (en unix timestamp)
- Groups : liste des groupes auxquels il appartient.

Lorsqu'un nouvel e-mail est découvert, un nouvel utilisateur est ajouté. Il est aussi ajouté au groupe par défaut. Son niveau de confiance initial est fixé à 50.

Il devrait être possible de préremplir et de définir les valeurs initiales d'une liste d'utilisateurs. Le fichier doit être un fichier JSON. Dans ce laboratoire le fichier "users.json" est utilisé pour gérer d'utilisateurs. Le format de 'users.json' à respecter est montré dans la fig.1.

```
{
    "1": {
        "name": "admin@test.com",
        "Trust": 100,
        "SpamN": 0,
        "HamN": 10,
        "Date_of_first_seen_message": 1596848266.0,
        "Date_of_last_seen_message": 1596846550.0,
        "Groups": ["default"]
    },
    "2": {
        "name": "spam@test.com",
        "Trust": 0.0,
        "SpamN": 12,
        "HamN": 0,
        "Date_of_first_seen_message": 1596855166.0,
        "Date_of_last_seen_message": 1596860598.0,
        "Groups": ["default"]
    }
}
```


Figure 1 – Format de ’users.json’ à respecter

Gestion des groupes

Un groupe est défini comme :

— Name : 1 à 64 caractères.

— Trust : nombre entier entre 0 et 100.

— List of members : liste des membres du groupe.

Le groupe par défaut est prédéfini avec un niveau de confiance initiale de 50 %. Il devrait être possible de préremplir et de définir les valeurs initiales d’une liste de membres des groupes. Le fichier de gestion doit être un fichier JSON. Dans ce laboratoire le fichier "groups.json" est utilisé. Le format de ’groups.json’ à respecter est montré dans le fig.2.

Traitement du texte

Ce module est réalisé dans le fichier text_cleaner.py. Le traitement (ou nettoyage) du texte est nécessaire pour enlever tout le contenu inutile tel que ’stop-words’ (pronoms, conjonctions, etc), ponctuation et symboles. Ce contenu n’ajoute aucune valeur ou signification au vocabulaire global.

```
{
    "1": {
        "name": "default",
        "Trust": 50,
        "List_of_members": ["GP@paris.com", "farmer@paris.com", "bob@test.com", "mike@test.com"]
    },
        "2": {
        "name": "friends",
        "Trust": 100,
        "List_of_members": ["bob@test.com", "mike@test.com"]
    }
}
```


Figure 2 – Format de ’groups.json’ à respecter

Création du vocabulaire

Ce module est réalisé dans le fichier vocabulary_creator.py. Le vocabulaire représente les mots qui était trouvé dans les messages spam/ham et leur probabilité d’être trouvé dans ces messages. Vous devez créer le vocabulaire en basant sur les e-mails fournis dans le fichier "1000-mails.json".

Dans le système RENEGE le vocabulaire comprend 4 parties : mots dans le sujet des messages spam ("spam_sub"), mots dans le sujet des messages ham ("ham_sub"), mots dans le "body" des messages spam ("spam_body") et mots dans le "body" des messages ham ("ham_body"). Le format de ’vocabulary.json’ à respecter est montré dans le fig.3.

Prenons un exemple : la probabilité du mot dans la catégorie "spam_sub’ est calcule comme un ratio de nombre des fois le mot est apparu dans le sujet du message spam et le nombre total des mots trouvés dans le sujet des messages spam.

Analyse des e-mails

Ce module est réalisé dans le fichier renege.py. Les messages à analyser sont fournis dans le fichier "1000-mails.json". Pour chaque message retenu de "1000-mails.json", les fichiers ’users.json’ et ’groups.json’ sont modifié : soit un nouvel utilisateur est ajouté, soit les donnes d’utilisateur existant ou groupe à quel l’utilisateur appartient sont changées.

Calcule de probabilité de Bayes

Ce module est a réaliser dans le fichier email_analyzer.py. Le théorème de la probabilité de Bayes est exprimé mathématiquement comme :

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}, \tag{1}$$

ou P(A) - probabilité d'observer A, P(B) - probabilité d'observer B. P(A|B) - probabilité d'observer A si B est vrai et P(B|A) - probabilité d'observer B si A est vrai.

Dans le cas de classification des e-mails, la formule peut être exprimée comme ;

$$P(spam|w1 \cap w2 \cap ... \cap wn) = \frac{P(w1 \cap w2 \cap ... \cap wn|spam) * P(spam)}{P(w1 \cap w2 \cap ... \cap wn)}, \quad (2)$$

ou *w1, w2, wn* sont les mots. La probabilité :

$$P(w) = \frac{nombre\ de\ fois\ le\ mot\ a\ apparu\ dans\ le\ jeux\ de\ données}{nombre\ totale\ des\ mots\ dans\ le\ jeux\ de\ données} \quad (3)$$

va être le même pour les mots dans le spam et dans le ham. En raison de ça, pour classifier le message, on peut simplifier l'équation (2) à (4), ou on calcule les deux probabilités pour le message (d'être spam et mam) et choisit la maximale ;

$$max(P(w1 \cap w2 \cap ... \cap wn|spam) * P(spam), P(w1 \cap w2 \cap ... \cap wn|ham) * P(ham)), \quad (4)$$

ou

$$P(spam) = \frac{nombre\ des\ messages\ spam}{nombre\ totale\ des\ messages} \quad (5)$$

$$P(w|spam) = \frac{nombre\ de\ fois\ le\ mot\ a\ apparu\ dans\ les\ messages\ spams}{nombre\ des\ mots\ dans\ les\ messages\ spams}. \quad (6)$$


```
{
    "spam_sub": {
        "pill": 0.0031088082901554403,
        "new": 0.01037344398340249
    },
    "ham_sub": {
        "enrononlin": 0.0033594624860022394,
        "money": 0.0003502933706979595
    },
    "spam_body": {
        "recent": 0.004203520448375514,
        "survey": 0.0003502933706979595
    },
    "ham_body": {
        "member": 0.0020833333333333333,
        "call": 0.0031282586027111575
    }
}
```


Figure 3 – Format de 'vocabulary.json' à respecter

Dans le cas du système RENEGE, on va avoir la probabilité que le sujet et spam/-ham aussi que la probabilité du corps du message d'être spam/ham. La façon le plus simple de combiner les probabilités est :

$$P(spam|w1 \cap w2 \cap ... \cap wn) = \frac{1}{2} * P(spam\_sub|w1 \cap w2 \cap ... \cap wn) + \frac{1}{2} * P(spam\_body|w1 \cap w2 \cap ... \cap wn). \quad (7)$$

Dans ce tp, on vous demande de suggérer la façon plus optimale.

Exemple de calcule de probabilité que message est spam/ham

W	note	to	self	become	perfect
$P(W \mid Y = \text{spam})$	1/6	1/8	1/4	1/4	1/8
$P(W \mid Y = \text{ham})$	1/8	1/3	1/4	1/12	1/12

Figure 4 – Dictionnaire pour l'exemple

Cet exemple est tiré du document `https://edge.edx.org/c4x/BerkeleyX/CS188-FA14/asset/section11_sol.pdf`. Supposons, on a un dictionnaire montré sur la fig.4 . En totale on a analysé 100 messages. Parmi eux 30 était spam et 70 ham. On a donc conclu que P(spam) = 30/100 = 0.3 et P(ham) = 70/100 = 0.7. Maintenant on veut savoir si message "perfect note" est spam ou pas. Tout d'abord on va calculer la probabilité que ce message est spam :

$$P(spam|w1 = perfect \cap w2 = note) = P(w1 = perfect \cap w2 = note|spam) * P(spam) =$$

$$= P(w1 = perfect|spam) * P(w2 = note|spam) * P(spam) = \frac{1}{8} * \frac{1}{6} * 0.3 = 0.00625.$$

Après on calcule la probabilité que message est ham :

$$P(ham|w1 = perfect \cap w2 = note) = P(w1 = perfect \cap w2 = note|ham) * P(ham) =$$

$$= P(w1 = perfect|ham) * P(w2 = note|ham) * P(ham) = \frac{1}{12} * \frac{1}{8} * 0.7 = 0.00486.$$

En comparant les probabilités, on constate que message "perfect note" est spam, car :

$$P(spam|w1 = perfect \cap w2 = note) > P(ham|w1 = perfect \cap w2 = note)$$

0.00625 > 0.00486.

Évaluation du système

Ce module est réalisé dans le fichier main.py. Il lance le système RENEGE. Tout d'abord le vocabulaire est créé. Après, les e-mails sont classifiés et fichiers 'users.json' et 'groups.json' sont modifiés. Finalement, "accuracy", "precision" et "recall" du modèle sont évalués. Dans le cas optimale, vous devez avoir "accuracy" plus que haut que 70 %.

5 Travail à effectuer

1. Compléter toutes les fonctions du code et bien respecter les formats des fichiers JSON. Bien valider l'entrée des fonctions.
2. Commenter votre code.
3. Définir deux nouveaux groupes selon la valeur du trust et justifier le choix. Par exemple, les utilisateurs avec la valeur moins que 50 % peuvent être ajoutés dans le groupe "spam".
4. Proposer une façon de combinaison des probabilités de spam/ham pour sujet et corps du message électronique.
5. Évaluer le "accuracy", "precision" et "recall" du votre modèle bayésienne. Analyser le résultat.
6. Fournir des idées comment la performance du système peut être améliorée.

Remarques :

— Vous pouvez utiliser l'entrée des fonctions suggérées ou les changer. Vous pouvez ajouter des fonctions, mais tout les prototypes des fonctions données doivent être implémentés.
— Pour exécuter le système : python3 main.py
— On fournie le fichier "1000-mails.json" comme base de donnée de création du vocabulaire et de test. Vous pouvez séparer le contenu : 20% pour effectuer le test et 80% pour création du vocabulaire.

6 Livrables attendus

Les livrables suivants sont attendus :

— Un rapport pour le laboratoire : 5 points.
— Le dossier COMPLET contenant le projet (fichiers python et json) : 15 points.

Le tout à remettre dans une seule archive **zip** avec le titre matricule1_matricule2_matricule3_lab1.zip sur Moodle. Seulement un person d'équipe doit remettre le travail.

Le rapport doit contenir le titre et numéro du laboratoire, les noms et matricules des coéquipiers ainsi que le numéro du groupe.

7 Information important

1. Consultez le site Moodle du cours pour la date et l'heure limites de remise des fichiers (deux semaines après la première séance du laboratoire).
2. Un retard de [0,24h] sera pénalisé de 10%, de [24h, 48h] de 20% et de plus de 48h de 50%.

3. Aucun plagiat n’est toléré. Vous devez soumettre juste le code, qui était fait par les membres de votre équipe.