Extraire les données de mails pour déterminer si c'est un spam

Cours Ingénierie des langues

Projet, 1ère phase

Martial Goehry

Université Paris 8, LIASD
Licence d'informatique

29 avril 2022

Plan

1. Objectif
2. Corpus
3. Difficultés
4. Aspiration
5. Nettoyage
6. Stockage
7. Caractéristiques

Objectif

L'objectif est d'extraire des données d'e-mails pour déterminer s'ils sont des spams.

- langue concernée : Anglais
- type de corpus : corpus monolingue écrit
- type de données : e-mail

Corpus

Dataset de mail du projet SpamAssassin

- Anglais
- `https://spamassassin.apache.org/old/publiccorpus/`
- consulté le 27/01/2022
- 6065 fichiers email déjà trier en ham et spam

Dataset de mail de la compagnie Enron

- Anglais
- `https://www.kaggle.com/wcukierski/enron-email-dataset`
- consulté le 27/01/2022
- fichier CSV avec 33 834 245 lignes (un mail par ligne) non trier

Analyse des sites

Je n'ai pas réussi à trouver de dataset de mail en français, j'ai donc du me retourner vers les dataset en anglais.

Aspiration

J'ai recherché un outil simple me permettant d'extraire facilement le message d'un mail en faisant abstraction du nombre important de métadonnées associées. Mon choix s'est porté sur le module *email* de python. Je me suis donc orienté vers une solution tout python pour le traitement des données.
Déroulé :

1. Chargement des données brutes à l'aide de python.
2. Utilisation du module *email* pour importer et transformer les fichiers textes et csv en objet *email.message*.

Aspiration

Fichiers textes, SpamAssassin

Chaque mail est stocké dans un fichier texte. Ces fichiers sont déjà triés en spam et non spam (ham). Une fois le fichier ouvert j'utilise la fonction *email.message_from_string()*

Fichier CSV, compagnie Enron

Importation du fichier CSV avec le module *csv* de python.
Seulement 2 colonnes sont présentes : chemin, message.
Pour chaque ligne on applique la fonction *email.message_from_string()* à la partie correspondante au message.

A partir de ce point je ne traite plus que des objets *email.message*

Aspiration, difficultés

Encodage

Certains encodages ne sont pas pris en charge par le module *email*. J'ai du me limiter aux encodages suivants : ascii, Windows-1252, ISO-8859-1, utf-8.
J'utilise le module *chardet* pour faire la détection de l'encodage

Multiple charset

Les messages mails peuvent avoir plusieurs charset à l'intérieur du payload (*get_content_charset*). Les charset suivant m'ont posé des problèmes lors du traitement, j'ai du les écarter : unknown-8bit, default, default_charset, gb2312_charset, chinesebig5, big5.
Écarter ces charset me permet d'enlever les messages écrits en turc, chinois ou japonais.

Extraction des données

Une fois le message disponible, le module email se charge de convertir la chaîne de caractère en objet *email.Message*. De cet objet on va pouvoir extraire les données suivantes :

- objet du mail (Subject)
- expéditeur (From)
- corps du mail (payload)

Il est possible que le corps du mail soit séparé en plusieurs sous-partie (multi-part). Dans ce cas il faut concaténer les parties textuelles.

- text/plain
- text/html
- text/enriched

Les types html et enriched ont un pré-nettoyage pour retirer les balises.

Nettoyage

Les messages étant déjà chargés dans le programme en python j'ai trouvé préférable d'effectuer toutes les opérations de nettoyage directement après la phase d'importation.
Étapes du nettoyage du texte :

1. Suppression des balises html avec le module *BeautifulSoup* (bs4)
2. Suppression des balises eniched text avec le module *re*
3. Tout le texte en minuscule avec la méthode lower()
4. Suppression des réponses avec le module *re*
5. Substitutions des adresses mail, url et numéro de téléphone avec le module *re*
6. Substitutions des prix et des nombres avec le module *re*
7. Suppression des ponctuations superflues avec le module *re*

Expressions régulières

- Capture des balises enriched text : `<.*>`
- Capture des réponses : `^>.*$`
- Capture mail :
  `[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+`
- Capture url :
  `(http|ftp|https)?:\/\/([\w\-_]+(?:(?:\.[\w\-_]+)+))`
  `([\w\-\.,@?^=%&:/~\+#]*[\w\-\@?^=%&/~\+#])?`
- Capture téléphone1 : `\\(\\d{3}\\)\\d+-\\d+`
- Capture téléphone2 : `\\+\\d+([ .-]?\\d)+`
- Capture prix1 : `[$€£]( )?\\d+([.,]\\d+)?`
- Capture prix2 : `\\d+([.,]\\d+)?( )?[$€£]`
- Capture nombre : `\\d+`

Exemple de nettoyage texte brut

Brut

Message dedicated to be a sample to show how the process is clearing the text.
Begin reply : > He once said »> that it would be great End of reply. Substitutions : spamassassin-talk@example.sourceforge.net https ://www.inphonic.com/r.asp ?r=sourceforge1&refcode1=vs33 hello.foo.bar between $ 25 and 25,21 $
A number is : 2588,8 588 Phone type a : (359)1234-1000 Phone type b : +34 936 00 23 23 Ponctuation : —-## ..

Nettoyé

message dedicated to be a sample to show how the process is clearing the text. begin reply end of reply. substitutions MAIL URL URL between PRIX and PRIX a number is NOMBRE , NOMBRE NOMBRE phone type a TEL phone type b TEL ponctuation .

Nettoyage enriched text

Brut

<smaller>I'd like to swap with someone also using Simple DNS to take advantage of the trusted zone file transfer option.</smaller>

Nettoyé

I'd like to swap with someone also using Simple DNS to take advantage of the trusted zone file transfer option.

Nettoyage HTML brut


```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Foobar</title>
</head>
<body>
I actually thought of this kind of active chat at AOL
bringing up ads based on what was being discussed and
other features
  <pre wrap="">On 10/2/02 12:00 PM, "Mr. FoRK"
  <a class="moz-txt-link-rfc2396E"href="mailto:fork_
  list@hotmail.com">&lt;fork_list@hotmail.com&gt;</a>
  wrote: Hello There, General Kenobi !?
<br>
</body>
</html>
```

Nettoyage HTML nettoyé


```
Foobar


I actually thought of this kind of active chat at AOL
bringing up ads based on what was being discussed and
other features
  On 10/2/02 12:00 PM, "Mr. FoRK"

  wrote: Hello There, General Kenobi !?
```

Conclusion nettoyage

Le nettoyage du texte s'arrête avant l'étape de lemmatisation.
Initialement, je souhaitais réaliser cette étape avant le stockage, mais j'ai peur qu'une réduction trop importante n'altère le résultat de recherche de sentiment.
Je garde donc le texte avec une structure permettant d'en comprendre le sens.
Point à améliorer :

- Les mails que j'ai pu collecter contiennent des exemples de code en C que je n'ai pas été en mesure de retirer.

Moteur de base de données

- Base de données choisie : ElasticSearch
- Raisons de ce choix : J'avais envie de travailler avec une base de données NoSQL. L'utilisation du format JSON est assez souple et s'intègre bien avec python. L'interface graphique, Kibana, permet déjà une prévisualisation des données. ElasticSearch intègre nativement un moteur Lucene pour l'indexation des données textuelles. Il devrait être possible d'utiliser ce moteur pour faire l'analyse de sentiment, ou d'extraire les message pour les traiter avec un autre outil de traitement du langage.
- Résultat : J'ai pu récupérer et stocker le corps de 254 740 mails ainsi que quelques informations statistiques (nombre de mots, nombre de substitutions) et des informations de ciblage (adresses mail, sujet). L'indice de cette base occupe un espace de 448.53 Mb.

Avant d'utiliser sur le moteur ElasticSearch, j'avais choisis une base PostgreSQL avec une opération de tokenisation au préalable. J'ai abandonnée cette solution car :

- Temps de chargement des données dans la base : 2 jours contre 1 heure avec Elastic
- Taille de la base : 1.6G contre 448M avec Elastic
- Difficulté de reconstitution du message original

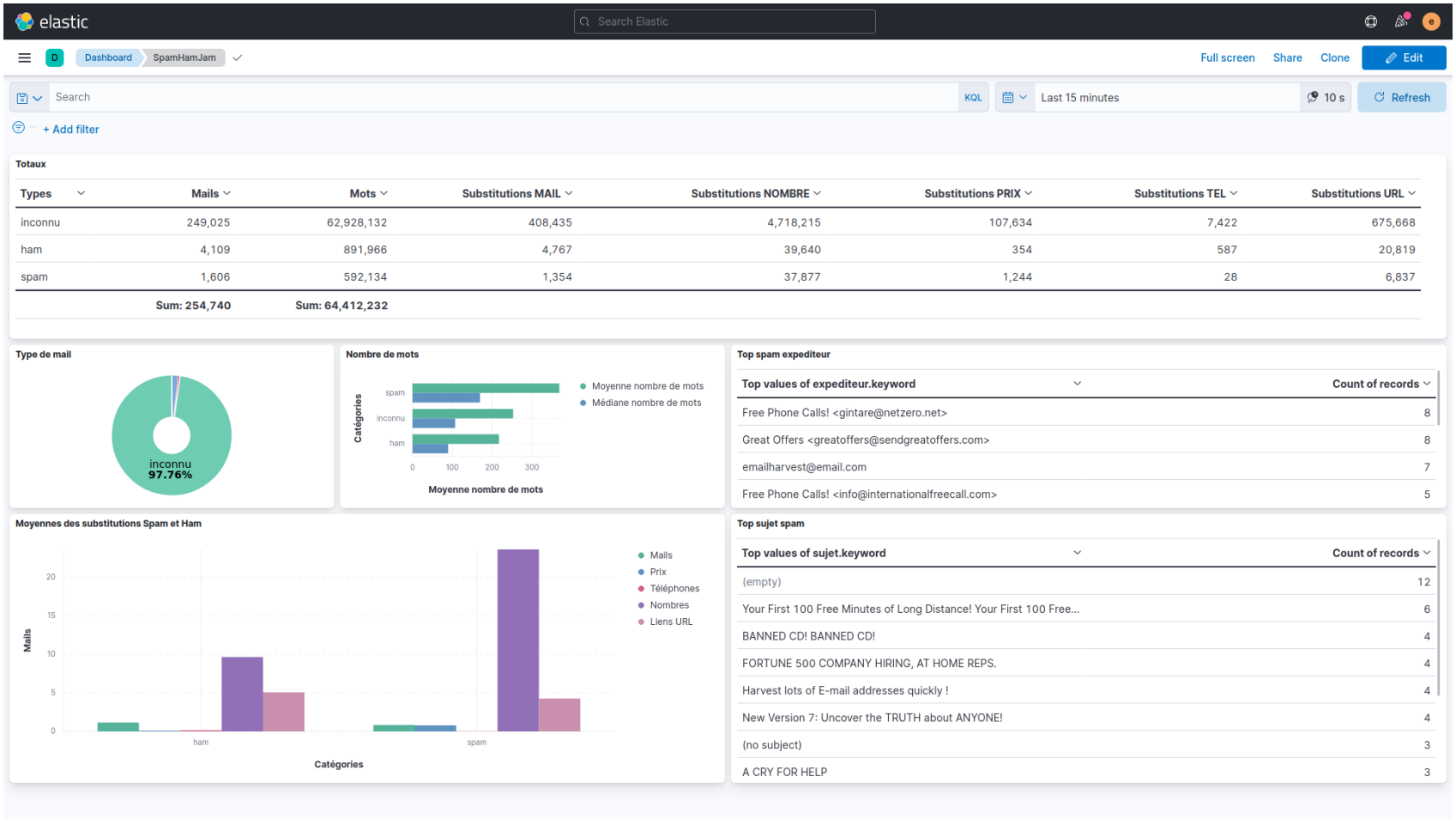
Il était cependant plus facile d'obtenir les statistiques suivantes :

- Nombre de mots uniques dans le corpus et dans chaque document
- Nombre de d'occurrences d'un mot dans le corpus ou dans chaque document

Données à stockées au format JSON dans une base ElasticSearch avec le mappage :

- document
  - hash : empreinte en md5
  - chemin : accès au fichier brut
  - categorie : ham, spam ou inconnu
  - sujet : objet du mail
  - expediteur : champ FROM
  - nombres_mots : nombre de mots dans le corps
  - substitutions : remplacement dans le corps
    - URL : liens URL
    - MAIL : adresses mails
    - TEL : numéros de téléphone
    - NOMBRE : chiffres sans indication monétaire
    - PRIX : chiffres avec des indications monétaire
  - message : corps du mail

Caractéristiques



Conclusion

Je pense avoir réussi à faire une chaîne d'opérations sur mails (récupération, nettoyage, stockage) qui devrait me permettre de maintenir un traitement uniforme dans la durée.
Je me suis directement orienté vers une solution complète en python, même si la majeure partie de mes traitement aurait pu être fait avec *sed* ou *awk*.

La suite

La prochaine étape va être de pouvoir faire l'analyse de sentiment à proprement parler. Le scoring que j'espère récupérer devrait pouvoir me permettre d'entraîner un modèle pour pouvoir affiner la détection des spams.

Références

- Documentation du paquet email de python, [en ligne], `https://docs.python.org/fr/3/library/email.html` (27/01/2022)
- Documentation du paquet regex de python, [en ligne], `https://docs.python.org/fr/3/library/re.html` (30/01/2022)
- Documentation du paquet beautifulsoup, extraire du texte de l'html, [en ligne], `https://beautiful-soup-4.readthedocs.io/en/latest/` (30/01/2022)
- Expression régulière URL, [en ligne], capturer les URL `https://www.i2tutorials.com/match-urls-using-regular-expressions-in-python/` (30/01/2022)

- Détecter l'encodage d'un texte, [en ligne], `https://chardet.readthedocs.io/en/latest/usage.html` (03/02/2022)
- Installer ElasticSearch avec docker, [en ligne], `https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html` (01/04/2022)
- Documentation python module ElasticSearch, [en ligne], `https://elasticsearch-py.readthedocs.io/en/v8.1.3/` (01/04/2022)
- Nettoyage de texte en python, [en ligne], `https://www.machinelearningplus.com/nlp/lemmatization-examples-python/` (28/01/2022)

Merci pour votre attention