

# Projet Errol - Catégorisation des mails en HAM et SPAM

Cours Ingénierie des langues  
**Projet, 1ère phase**

MARTIAL GOEHRY

Université Paris 8, LIASD  
Licence d'informatique

15 septembre 2025

# Plan

2/33

- 1 Général
- 2 Fouille
- 3 Caractéristiques
- 4 Traitement du langage
- 5 Vectorisation
- 6 Conclusion

# Objectif

3/33

L'objectif est d'extraire des données d'e-mails pour déterminer s'ils sont des courriers légitimes ou de sollicitations indésirées.

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

# Présentation de l'infrastructure

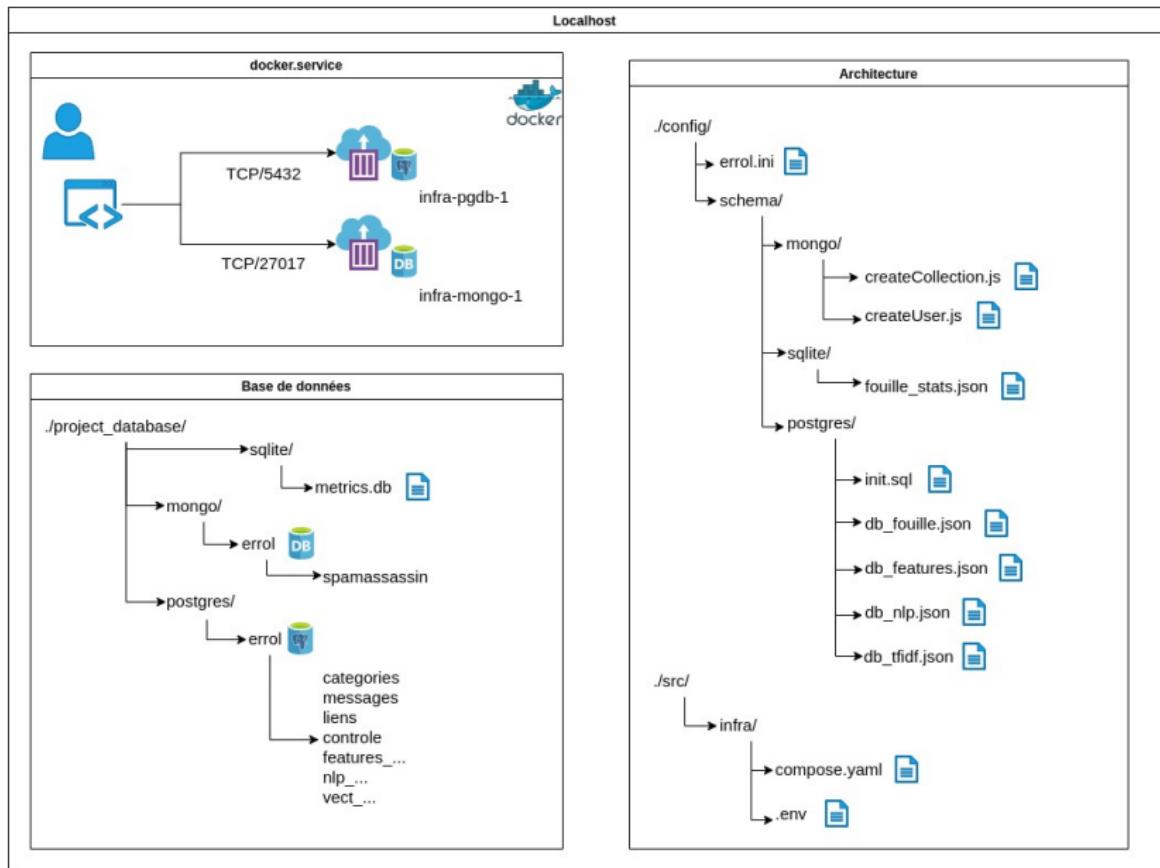
4/33

Les traitements sont basés sur un ensemble de d'instruction écrites en Python.

L'infrastructure de données utilise 1 base NoSQL (mongoDB) et 2 bases SQL (Postgres et SQLite).

Les moteurs et services sont conteneurisés pour limiter l'impact sur la machine hôtes.

L'initialisation des schémas des bases de données est gérée directement par le programme Python.



# Déroulé

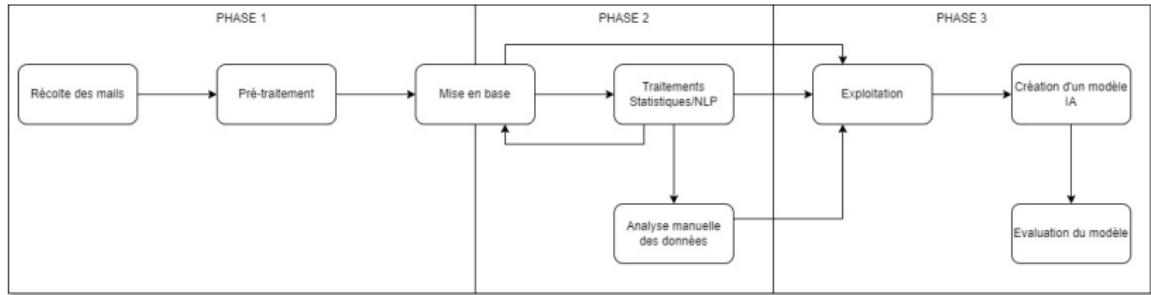
6/33

L'exécution des traitements se déroule en plusieurs phases :

- Importation des données et prétraitement (fouille.py)
- Recherche de caractéristiques (features.py)
- Traitement du langage naturel (nlp.py)
- Vectorisation (vecteurs.py)

## Manquant

La partie sur l'utilisation des vecteurs dans un modèle d'apprentissage n'est pas traitée



# Récupération des mails

8/33

Faute de sources, il n'a pas été possible de mettre en place une automatisation pour la récupération des mails. Il a été nécessaire de les récupérer manuellement dans d'autres projets du même type

## Alternative

Une autre possibilité est de créer un site web dans lequel les utilisateurs auraient pu ajouter des nouveaux documents. Il aurait fallu prendre en compte la mise en place de cette architecture et les règlementations RGPD.

# Corpus

9/33

## Dataset de mail du projet SpamAssassin

- Anglais
- <https://spamassassin.apache.org/old/publiccorpus/>
- consulté le 27/01/2022
- environ 6000 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 millions de lignes (un mail par ligne) non trié
- Non utilisé ici

# Etapes

10/33

Afin d'accélérer cette phase, chaque mail est traité dans un processus à par grâce au *multiprocessing.Pool()*.

## Importation

Les mails sont chargé en mémoire depuis les fichiers en utilisant la fonction *email.message\_from\_binary\_file()*

## Extraction

Un mail peut être multipart. On récupère récursivement uniquement les types text/plain, text/html, text/enriched. Les Headers et les autres types de MIME sont ignorés (audio, video, image...)

# Etapes : Nettoyage

11/33

Cette étape permettre de nettoyer le texte de certains éléments non souhaités :

- ① Suppression des balises html avec le module *BeautifulSoup* (bs4)
- ② Suppression des balises enriched text avec le module *re*
- ③ Suppression des réponses avec le module *re*
- ④ Substitutions des adresses mail, url et numéro de téléphone avec le module *re*
- ⑤ Substitutions des prix et des nombres avec le module *re*
- ⑥ Suppression des ponctuations superflues avec le module *re*

# Expressions régulières du nettoyage

12/33

- 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+

# Fouille - difficultés

13/33

## Encodage

Certains encodages ne sont pas pris en charge par le module *email*. J'ai dû 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 écarté : unknown-8bit, default, default\_charset, gb2312\_charset, chinesebig5, big5.  
Écarter ces charset permet d'enlever les messages écrits en turc, chinois ou japonais.

# Fouille - difficultés

14/33

## Langues

Le corpus contenant des mails dans d'autres langues. J'ai fait appel au module langdetect pour exclure les mails qui ne sont pas majoritairement en anglais.

# Exemple de nettoyage texte brut

15/33

## 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

16/33

## 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

17/33

```
<!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

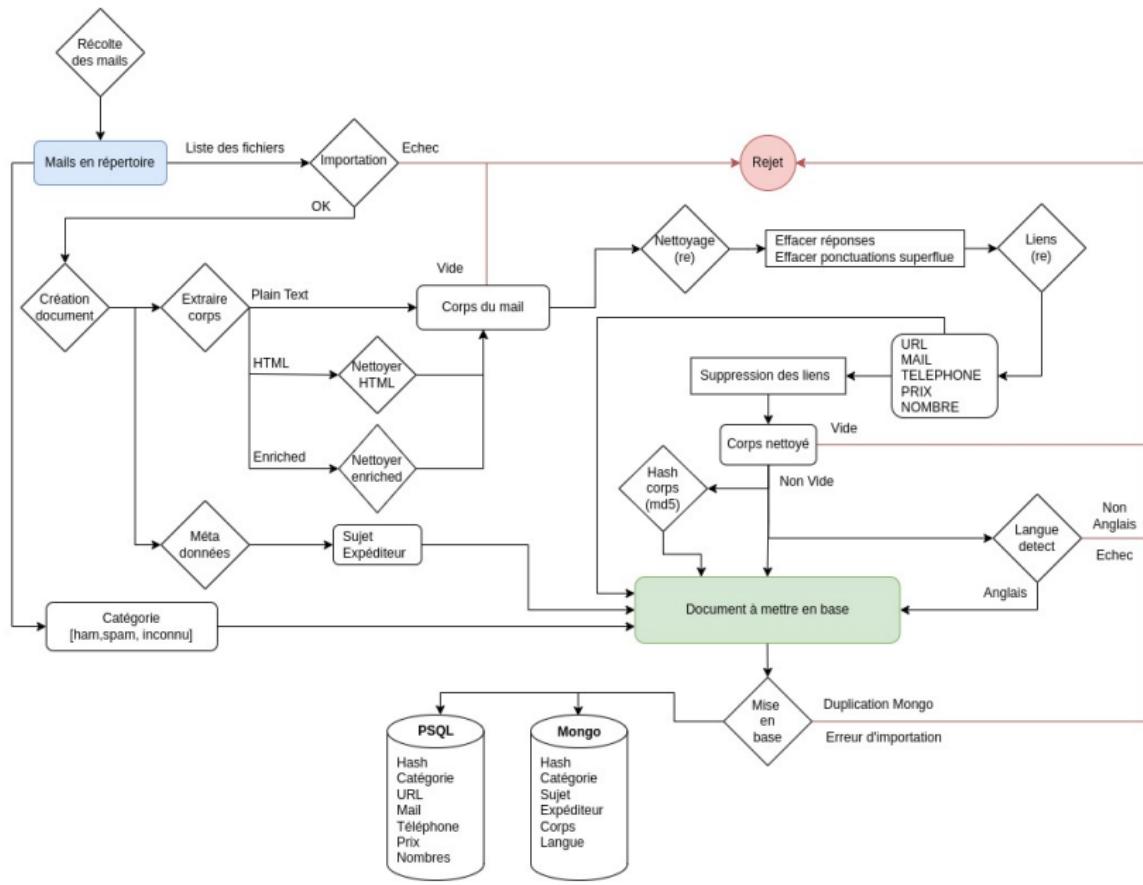
18/33

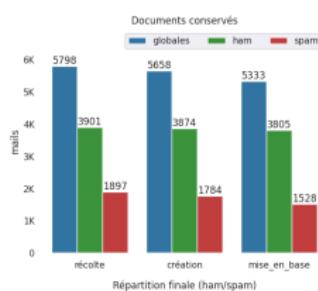
Foobar

I actually thought of this kind of active chat at AOL bringing up ads based on what was being discussed and other features

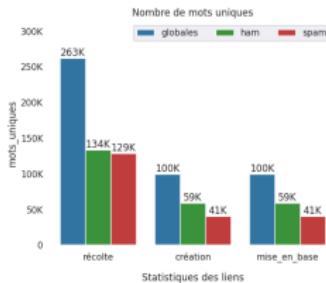
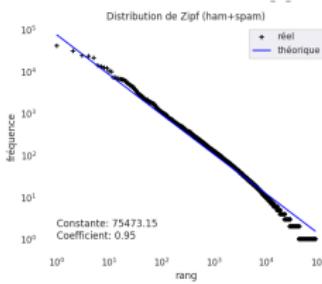
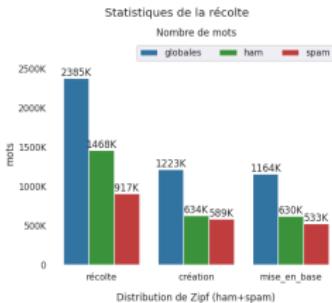
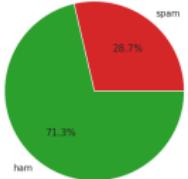
On 10/2/02 12:00 PM, "Mr. FoRK"

wrote: Hello There, General Kenobi !?





Répartition finale (ham/spam)



Statistiques des liens

Adresses mail:

- Mean - ham: 1.13 | spam: 1.02
- Q50 - ham: 1.00 | spam: 0.00
- Q90 - ham: 3.00 | spam: 2.00
- Max - ham: 20.00 | spam: 69.00

url:

- Mean - ham: 3.81 | spam: 5.26
- Q50 - ham: 2.00 | spam: 3.00
- Q90 - ham: 6.00 | spam: 9.00
- Max - ham: 470.00 | spam: 295.00

Nombres:

- Mean - ham: 7.16 | spam: 30.35
- Q50 - ham: 1.20 | spam: 7.00
- Q90 - ham: 14.00 | spam: 34.30
- Max - ham: 731.00 | spam: 15801.00

Prix:

- Mean - ham: 0.04 | spam: 0.81
- Q50 - ham: 0.00 | spam: 0.00
- Q90 - ham: 0.60 | spam: 2.00
- Max - ham: 17.00 | spam: 29.00

Les substitutions des adresses mails, des liens et des nombres, n'est pas très concluante.

# Recherche de caractéristiques

21/33

Cette étape va se concentrer sur la forme plutôt que sur le fond. L'idée est de comprendre si certains éléments non textuels sont plus présent dans un type que dans un autre

## Eléments recherchés

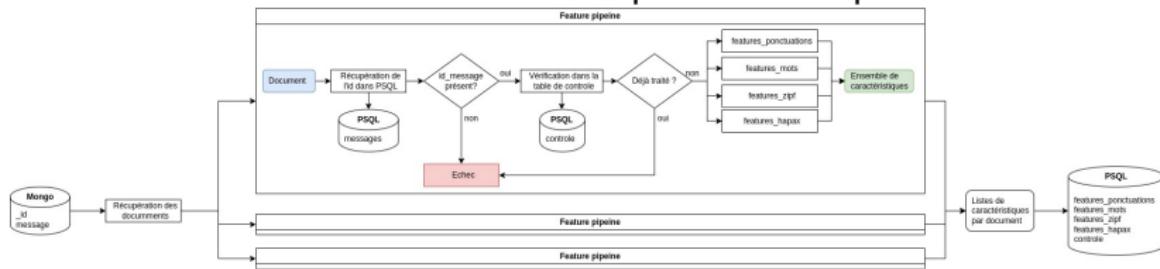
- Nombre de chars en majuscule et minuscule
- Nombre de mots en majuscule et capitalisé
- Nombre de ponctuations simple (point, virgule, exclamation, interrogation)
- Nombre d'espaces (espaces, tabulation, ligne vide)
- Nombre de lignes
- Données de la distribution de Zipf (constante, coefficient)
- Données de la recherche d'Hapax (nombre, ratio dans le texte et dans la liste des mots uniques)

# Exécution

22/33

Plusieurs fonctions de comptage sont exécutées itérativement sur chaque message. Toujours dans un souci de performance, le module multiprocessing est utilisé.

Le schéma si dessous montre les étapes de cette phase.



## Méthodes python utilisée

- str.count(...)
- len(re.findall(...))

# Résultats

23/33

Sans représenter les détails du rapport, on peut lister les éléments suivants :

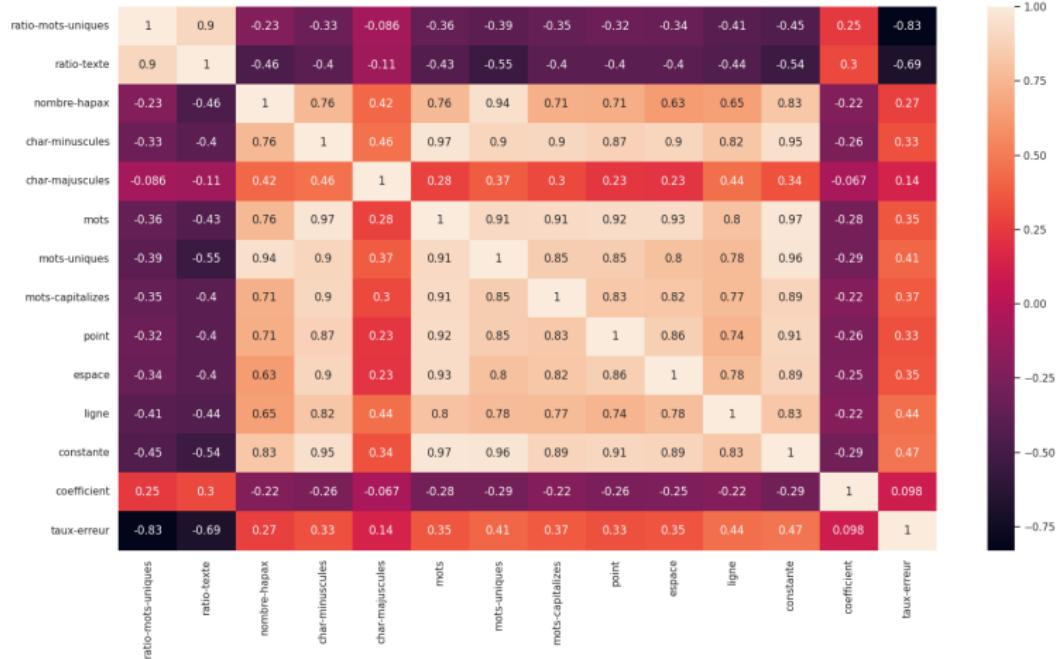
- Les spams utilisent plus de mots et de mots uniques
- Les mots complètement en majuscules sont très peu utilisés
- L'utilisation des ponctuations et des espaces est plus prononcé dans les spam

## Caractéristiques retenues

- ratio-mots-uniques
- nombre hapax
- char-majuscule
- espaces

# Covariance des caractéristiques

24/33



# Traitement du langage

25/33

La technique choisie est la lemmatisation en utilisant le modèle neuronal de StandfordNLP (Stanza).

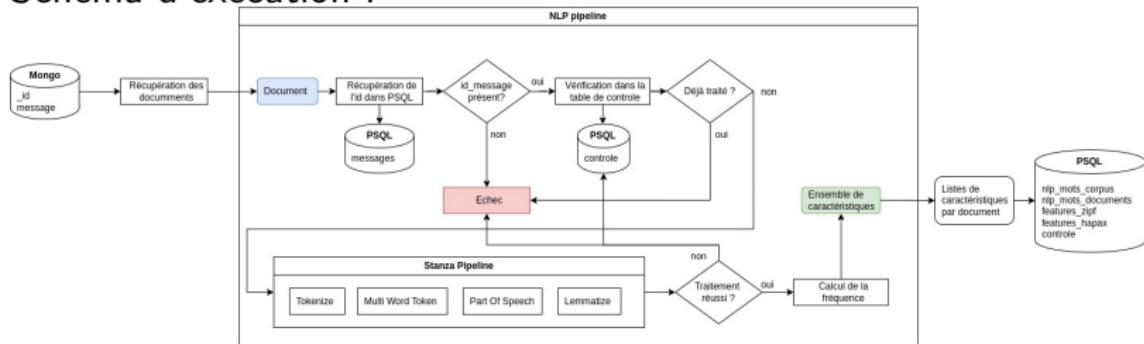
## Etapes de la pipeline Stanza

- ① Tokenisation
- ② Multi-Word Token Expansion
- ③ Part Of Speech Tagging
- ④ Lemmatisation

Le comptage des mots est effectué par une fonction annexe (`zipf.freq_mot()`) L'utilisation du multiprocessing n'est pas possible ici du fait de l'utilisation de CUDA et la carte graphique.

# Traitement du langage

## Schéma d'exécution :



## Tableau des résultats :

catégorie	mots	mots uniques
global	658524	39947
ham	355595	27787
spam	302929	19852

# Vectorisation

27/33

En se basant sur une liste des mots les plus fréquents (ici 2940), il a été possible de transformer ces 5333 textes en données numériques. La méthode de vectorisation est celle du Term Frequency-Inverse Document Frequency (TFIDF).

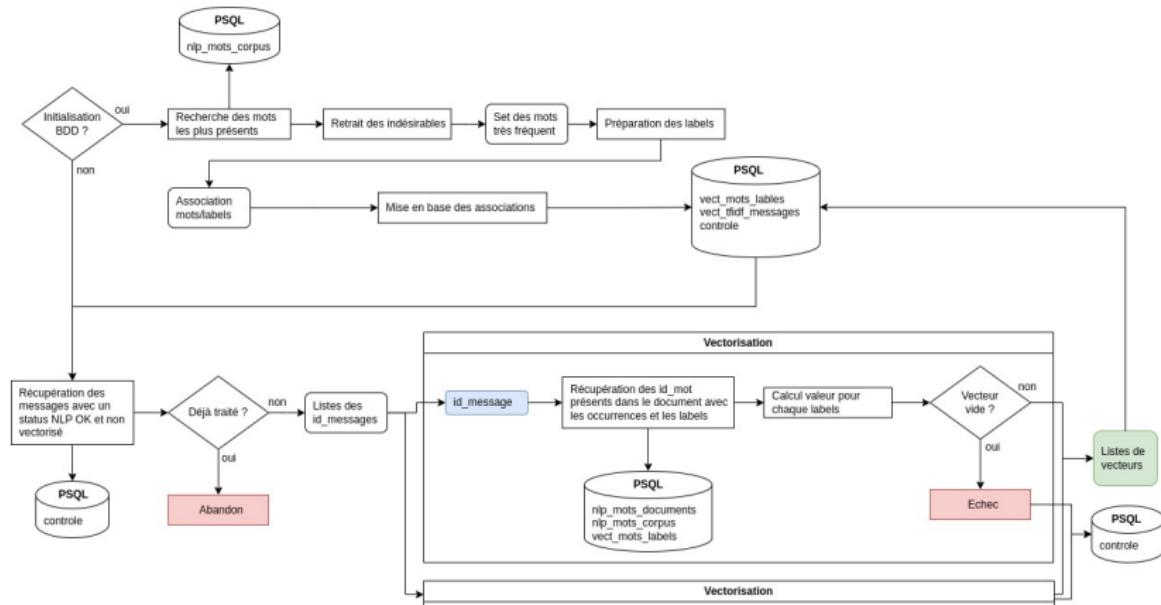
## Echantillons de mots du vecteur

list, get, use, email, one, make, free, time, person, send, new, information, would, work, say, receive, good, like, write, message, go, business, mailing, address, please, click, order, money, year, want, report, name, need, find, know, also, see, take, company, day, e-mail, remove, group, file, site, change, system, mail, program, look, ...

Les mots sans connotations de sens ont été retiré du vecteur.

# Schéma de la vectorisation

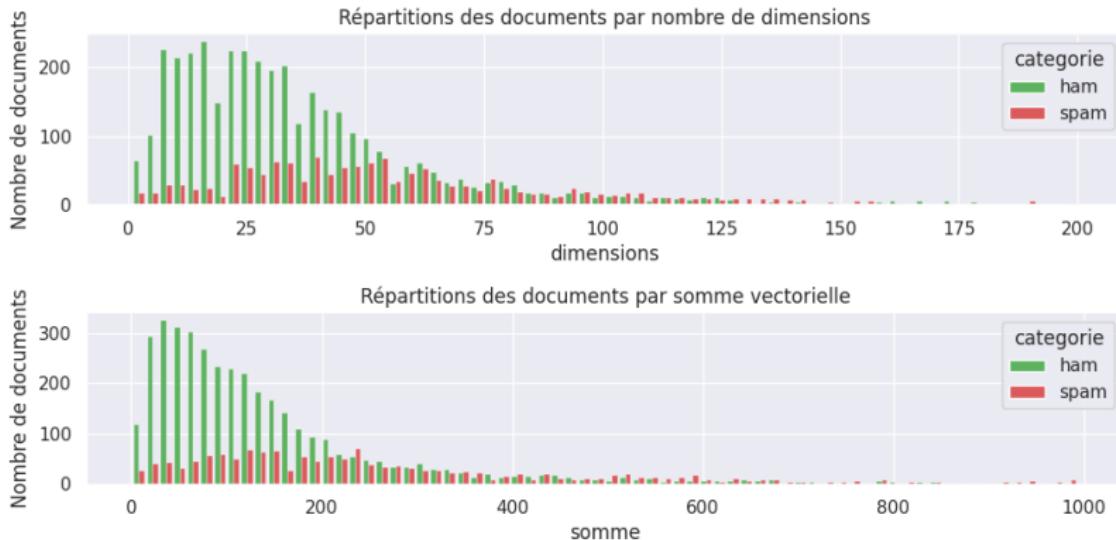
28/33



Il a été à nouveau possible d'utiliser le multiprocessing sur cette action.

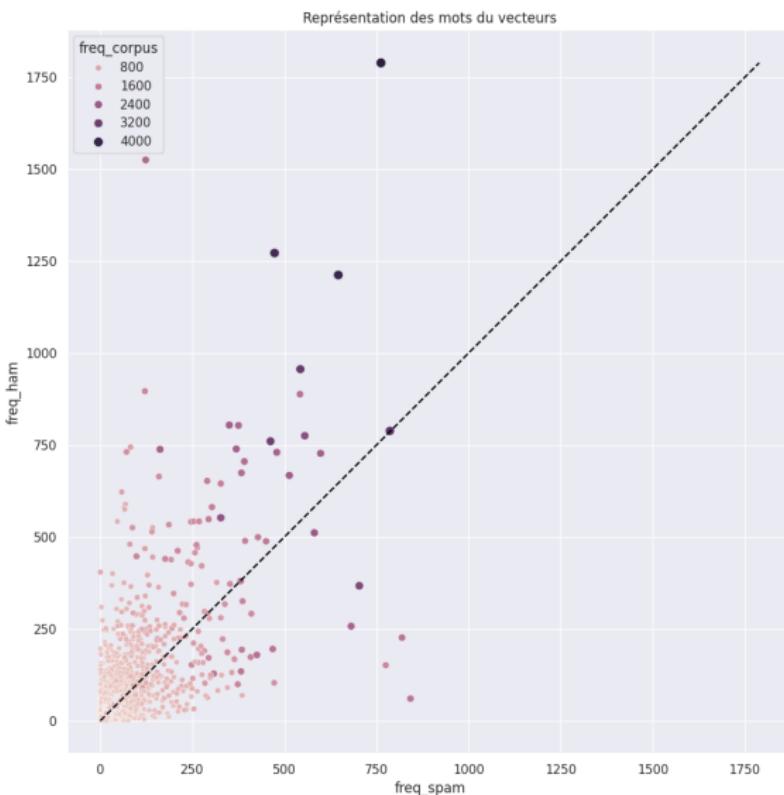
# Résultats

29/33



# Résultats

30/33



# Conclusion

31/33

Il a été possible d'effectuer tout un ensemble de traitement pour convertir des éléments textuels en données numériques. Ces opérations successives ont considérablement réduit le volume de données à traiter pour les modèles d'apprentissage.

Phase/Etape	documents	mots	mots uniques
Récolte	5798	2385120	262614
Transformation	5658	1222793	99772
Sauvegarde	5333	1163550	99772
Traitement du langage	5333	658524	39947
Vectorisation	5333	453068	2942

# Et ensuite

32/33

## Modélisation

Les étapes de modélisation et d'apprentissage ne sont pas présentes ici. Lors de la version précédente du projet, j'avais obtenu des résultats encourageant avec Random Tree Forest ou Support Vector Machine.

## Amélioration

L'analyse actuelle n'est pas très poussée pour la recherche de données abérantes qui pourrait corrompre les modèles futurs.

Merci pour votre attention