

2021

# PROJET STACKOVERFLOW CATÉGORISATION AUTOMATIQUE



**Duleme Méyi, oc/Centrale Supélec**

# SOMMAIRE

**03** Introduction

**04** Les données:  
Chargement et exploration

**0?** LDA

**0?** Régression Logistique

**0?** Implémentation de l'API

**0?** Conclusion

# INTRODUCTION

Stackoverflow est la plus importante communauté en ligne de développeurs. Grâce à l'interface de Questions/Réponses, ils apprennent et partagent leurs connaissances en programmation. En pratique, il suffit d'entrer des mots clés pour trouver la réponse à votre problématique, si quelqu'un l'a déjà rencontrée. Ce système permet de diminuer les duplicitas de questions et de trouver rapidement des discussions sur une diversité de sujets très précis.

Dans ce projet, notre mission est de développer un système de suggestion de tags pour une question donnée.

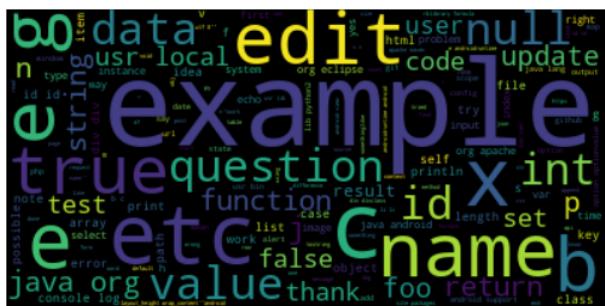
Pour ce faire, nous récupérons les données sur le site StackOverflow à l'aide de requêtes SQL, puis nous mettons en forme les données. Ces données ont ensuite été explorées de façon univariée et multivariées [...]

Pour finir, le lien vers une API est mis en place pour tester l'algorithme sur des questions à tagguer.

# **LES DONNÉES: CHARGEMENT ET EXPLORATION**

Stack Overflow propose un outil d'export de données - "stackexchange explorer", qui recense un grand nombre de données authentiques de la plateforme d'entraide. Sur cette interface, nous lançons le code SQL ci-contre pour sélectionner les questions favorites du site. L'intérêt étant que les tags y sont bien choisis. D'une part, ils le sont selon les utilisateurs de la plateforme. De plus, étant donné que le nombre de votes est important, il est raisonnable de supposer que la question est plus facile à trouver que d'autres sans votes.

```
SELECT
  Id, Title , Body, Tags
FROM
  Posts
WHERE
  FavoriteCount > 50
```

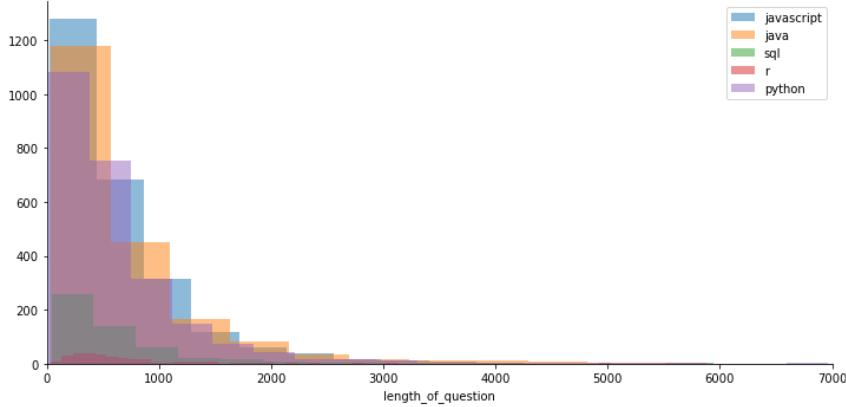


Une fois les données chargées, j'ai sélectionné les 50 tags les plus utilisés pour les sorties, après avoir éliminé les balises HTML, pour ensuite les encoder en colonnes binaires.

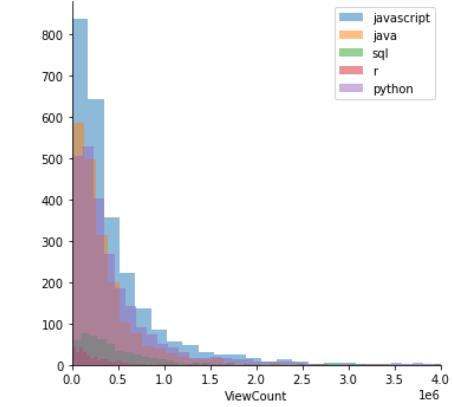
J'ai également réalisé un préprocessing des documents (ici les posts du forum) selon les étapes suivantes:

- Elimination des balises ht
  - Tokenisation
  - Elimination des stopwords et des urls

### Distribution of question lengths for each programming languages



### Distribution of question View count for each programming language



# MISE EN LIGNE DES NOTEBOOKS

La mise à disposition des notebooks en relation avec ce projet a été réalisée via l'interface GitHub. J'ai donc installé Git et ai couplé mon repository local avec le repository github en entrant mes identifiants.

Une fois cela réalisé, j'ai régulièrement mis à jour les fichiers grâce à un push du repository github. Ces lignes ont été lancées sur l'interface Git Bash.



## ***Etape 1 - Création d'un repository NLP\_Stackoverflow***

```
$cd NLP_Stackoverflow  
$git remote add origin https://github.com/meyiduleme/NLP_Stackoverflow.git
```



## ***Etape 2 - Couplage des remote repository***

```
$ git remote add origin https://github.com/user/repo.git  
# Set a new remote  
  
$ git remote -v  
# Verify new remote
```



## ***Etape 3 - Push régulier***

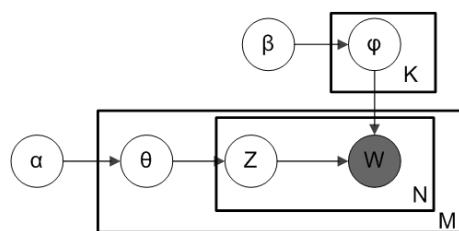
```
$ git add *  
$ git commit -m "Message"  
$ git branch -M main  
$ git push -u origin main
```

# LDA

La première méthode de classification proposée est une méthode d'apprentissage non-supervisée : le LDA.

La première méthode vraiment efficace est nommée LDA (Latent Dirichlet Allocation). C'est une méthode non-supervisée générative qui se base sur les hypothèses suivantes :

- Chaque document du corpus est un ensemble de mots sans ordre (bag-of-words) ;
- Chaque document  $m$  aborde un certain nombre de thèmes dans différentes proportions qui lui sont propres  $p(\theta_m)$
- Chaque mot possède une distribution associée à chaque thème  $p(\phi_k)$
- On peut ainsi représenter chaque thème par une probabilité sur chaque mot.
- $z_n$  représente le thème du mot  $w_n$



## Résultats

Top 1      python27 normal jpg nsnumber mongoose p4 insecure errordocument li normally

Top 2      anim i686 viewing additionally integerlist isreachable xargs layout\_margintop  
layout\_width layoutparams

Top 3      setcolor htdocs font colors requestcode url php paths generic weakly

# REGRESSION LOGISTIQUE MULTIPLE

Le second modèle implémenté est un modèle supervisé de régression logistique multiple avec la méthode de One vs Rest.

Metric	Score
--------	-------

Completer Coompléter

Completer Completer

Completer Completer

# API

Le modèle choisi est  
L'API

L'API a été implémentée en enregistrant le modèle sous format cPickle puis en le mettant en ligne via Streamlit. Le résultat est présenté ci-contre.

## CAPTURE DECRAIN API

Observations

# DISCUSSION

Constats, blocages et pistes d'améliorations