Catégorisez automatiquement des questions

Francois Xavier WOCKMATCHIEU HAPPI

Projet 5
Parcours: **Ingénieur Machine learning**

Novembre 2021

Table des matières

Table des figures ii

Introduction 1

1 Contexte de l'étude 2

1.1 Problématique . . . 2

1.2 Interprétation . . . 2

1.3 Pistes de recherche . . . 2

2 Extraction et traitement des données 3

2.1 Extraction des données . . . 3

2.2 Traitement des données . . . 3

3 Exploration des données 4

3.1 Analyse des questions . . . 4

3.2 Analyse du corpus général . . . 4

3.3 Analyse des tags . . . 5

4 Modélisation 8

4.1 Approche Non supervisée : Latent Dirichlet Allocation . . . 8

4.2 Approches supervisées . . . 9

4.3 Comparaison des performances et stratégies d'exploitation des modèles . . . 11

5 Déploiement 13

5.1 Schéma fonctionnel de l'API . . . 13

5.2 Aperçu du résultat . . . 14

Conclusion 15

Références 16

Table des figures

2.1 Extraction de données 3
2.2 Traitement de données 3

3.1 Répartition des mots dans les questions 4
3.2 Quantification du corpus global 4
3.3 Terme plus fréquents-rares des mots dans le corpus 5
3.4 Répartition des tags 5
3.5 Distribution des occurrences des tags 6
3.6 Tags les plus fréquents et les moins fréquents 6
3.7 Nuage des tags 7

4.1 Recherche du nombre optimal de topics 8
4.2 Schéma fonctionnel du modèle LDA 9
4.3 Réduction des dimensions du TF-IDF 10
4.4 Multi-labélisation et échantillonnage 10
4.5 Modèles de classe supervisée 11
4.6 Quantification suivant F1 score 12
4.7 Quantification de la performance suivant Gaccard 12

5.1 Schéma fonctionnel de l'API 13
5.2 Illustration du déploiement 14

Introduction

Ce rapport constitue la synthèse de mes différents travaux relatifs au projet $N5$ de mon parcours Ingénieur Machine Learning. Le but du projet étant de cheminer vers la proposition d'une API de prédiction des tags correspondants aux diverses questions de programmation en exploitant les données du site Stack Overflow, véritable base de données de questions et réponses portant sur une diversité de thématiques de l'informatique.

Après un nettoyage et une transformation des textes, une analyse détaillée a été réalisée afin de mettre en lumière les points saillants des deux approches de la modélisation qui en ont suivi. L'analyse minutieuse de la performance des différents modèles nous a conduits à une stratégie combinatoire des différents modèles dans l'espoir d'accroître le pouvoir prédictif en déploiement. Les lignes suivantes présentent les différents axes de réflexions abordées et mis en pratiques, puis leurs différents résultats.

1. Contexte de l'étude

Cette section a pour but de présenter le contexte, son interprétation et les pistes de recherche envisagées.

1.1 Problématique

Aucune prétention ne peut se prévaloir un développeur ou utilisateur d'un langage de programmation d'ignorer le site Stack Overflow ; Tellement il est célèbre, dispose d'une communauté élargie et très efficace dans les solutions aux diverses erreurs de programmation. En pratique, le praticien de la programmation face à une difficulté quelconque pose sa question sur le site et la communauté lui apporte une ou des propositions de solutions. En suggérant quelques tags relatifs à sa question il est très probable de trouver une famille de solutions déjà proposées par la communauté aux différentes questions très proches à celles posées. A cet effet, associer des meilleurs tags à sa question permet alors d'optimiser non seulement le volume des réponses à sa question mais d'améliorer aussi la précision et la qualité des réponses attendues

1.2 Interprétation

Considérant d'une part qu'il est souvent difficile aux praticiens débutants de la programmation d'associer des tags pertinents à leurs questions et d'autre part que l'acquisition automatique des tags liées à la question peut optimiser le temps des praticiens mêmes les plus chevronnés de la programmation, il advient alors qu'un système de génération automatique des tags est d'une importance capitale pour toute la communauté d'un site comme celui de stack Overflow. C'est ainsi qu'est né le projet de conception d'un générateur automatique de tags pour de déverses questions relatives à la programmation, celles-ci allant du scientifique au web, en passant par le système et réseau.

1.3 Pistes de recherche

Le produit final étant un générateur automatique de tags correspondant à une question posée, il est donc intuitif de penser aux modèles d'apprentissages non supervisés reposant sur le traitement automatique du langage naturel (NLP), plus précisément le topic modèle. Cependant, cette approche non supervisée nous offrira pour chaque question des topics non intuitifs et la tache complémentaire consistera à trouver le ou les tags correspondant à ces topics.

Par ailleurs, nous mettrons en place des modèles d'apprentissage supervisé guidés par une phase d'analyse textuelle et enfin la mise en place d'une API de prédiction à partir de la combinaison des modèles les plus performants retenus constituera la charpente finale de notre projet.

2. Extraction et traitement des données

2.1 Extraction des données

Le site stackoverflow dont le lien est le suivant stackoverflow constitue l'interface de requête SQL des questions qui ont été traitées sur le site. Une donnée est une question accompagnée de ses caractéristiques. Pour souci d'optimisation, un filtre sur les caractéristiques des questions nécessaires à la modélisation accompagne la requête d'extraction. Puis une opération de recherche et de suppression des questions aux valeurs manquantes permet d'épurer la base de données d'apprentissage constituée.

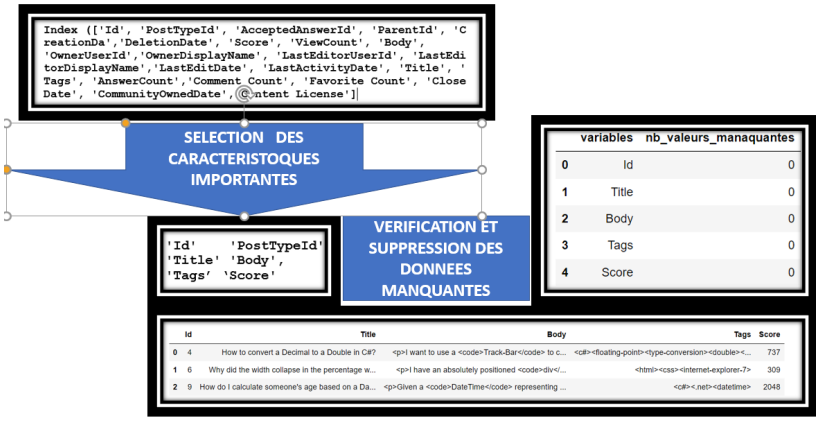


FIGURE 2.1 : Extraction de données.

2.2 Traitement des données

Les données étant de nature textuelle, il est impératif de s'en débarrasser des mots qui n'apportent aucune information (ponctuation, balise html, etc) puis de raciniser les plus informatifs. Le tableau ci-contre récapitule les opérations de traitements qui ont été réalisées.

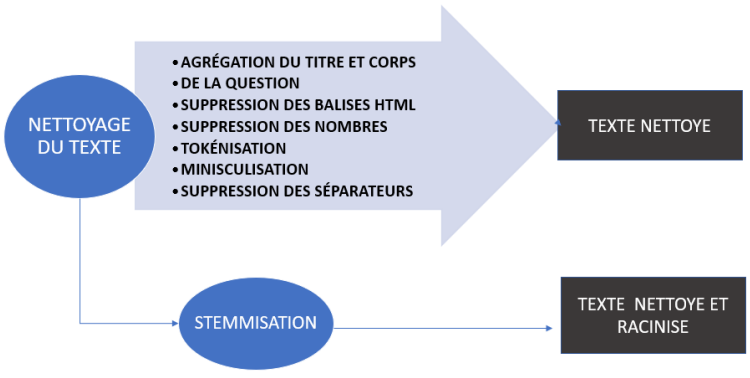


FIGURE 2.2 :Traitement de données.

3. Exploration des données

3.1 Analyse des questions

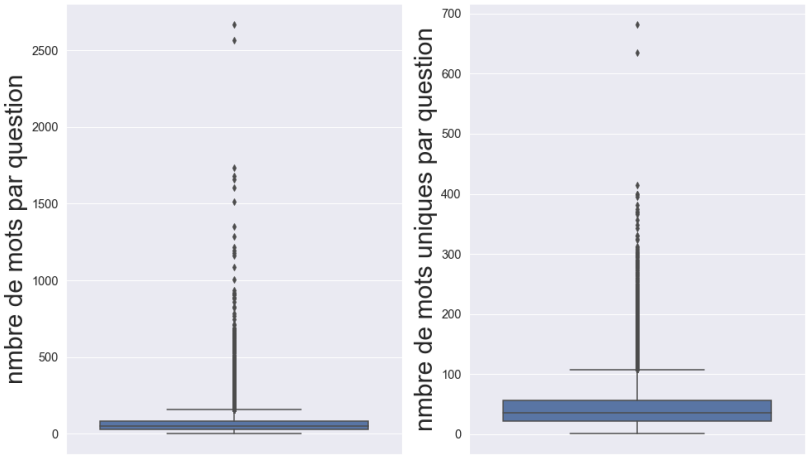


FIGURE 3.1 : Répartition des mots dans les questions.

Sur les 48883 questions nettoyées, nous avons en moyenne 66 mots par question avec un écart type de 65 mots. Par contre la racinisation (stemming) des mots nous permet d'éliminer les mots non informatifs pour passer en moyenne à 44 mots pour un écart type de 32 mots

3.2 Analyse du corpus général

Le corpus global étant l'un des éléments clés de la modélisation, il est important de comprendre la variation des occurrences des mots qui le composent afin d'appréhender les termes les plus fréquents et les plus rares.

CONTENUS DU CORPUS GLOBAL	
	Quantité
Mots	79995
Mots uniques	3242602

FIGURE 3.2 Quantification du corpus global.

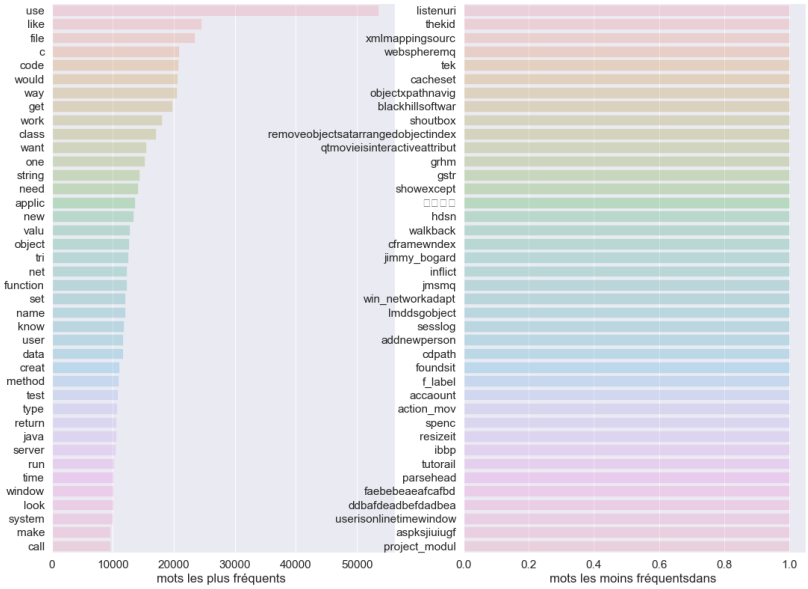


FIGURE 3.3 Terme plus fréquents-rares des mots dans le corpus.

3.3 Analyse des tags

Répartition des tags dans les questions

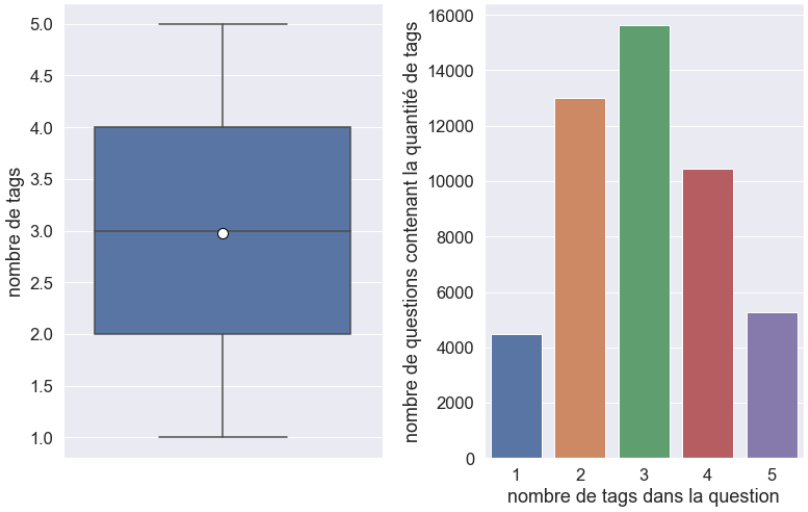


FIGURE 3.4 Répartition des tags.

Toutes les questions ont au moins un tag, en moyenne 3 tags et au maximum 5 tags. La médiane du nombre de tags par question est aussi de 3.

Répartition des occurrences des tags

Certains tags ont été utilisés plus de 1000 fois, mais on compte presque 3800 tags qui n'ont aucune occurrence.

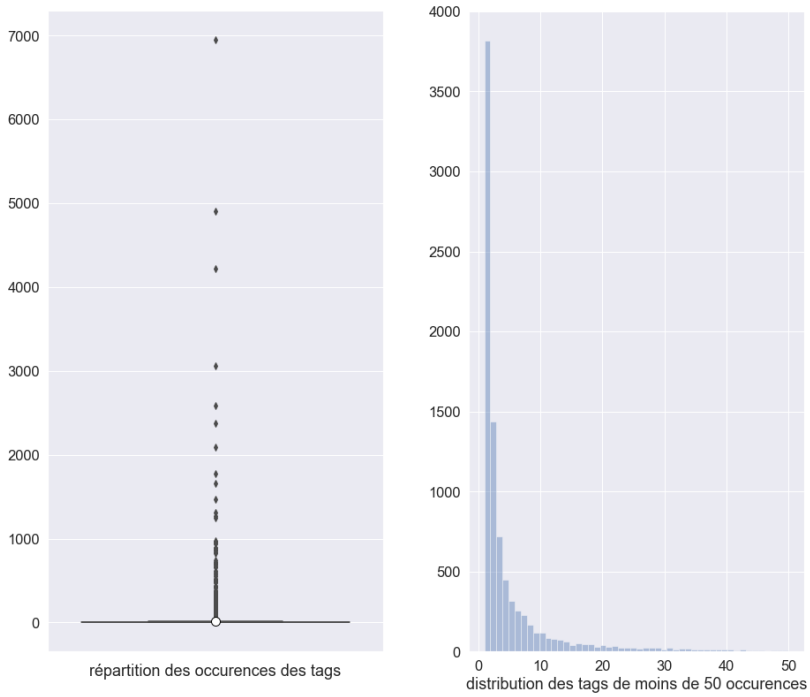


FIGURE 3.5 Distribution des occurrences des tags.

Les tags les plus et moins représentés

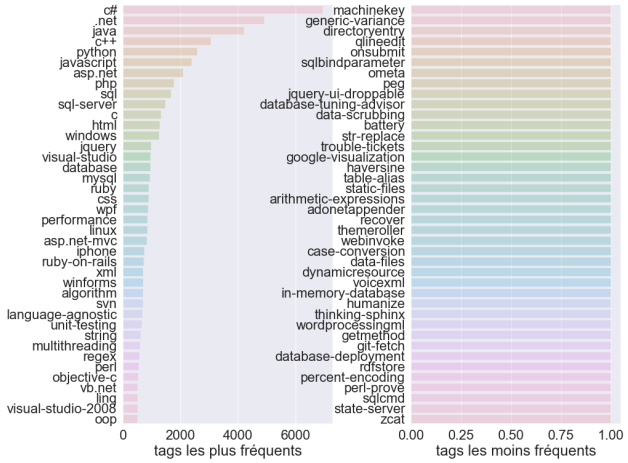


FIGURE 3.6 Tags les plus fréquents et les moins fréquents.

On constate bien que les tags C ,JAVA , C++,PHP, python, SQL, C sont les plus occurrents, ce qui témoignent la popularité et la haute utilité de ces langages.

Représentation graphique du Vocabulaire des tags

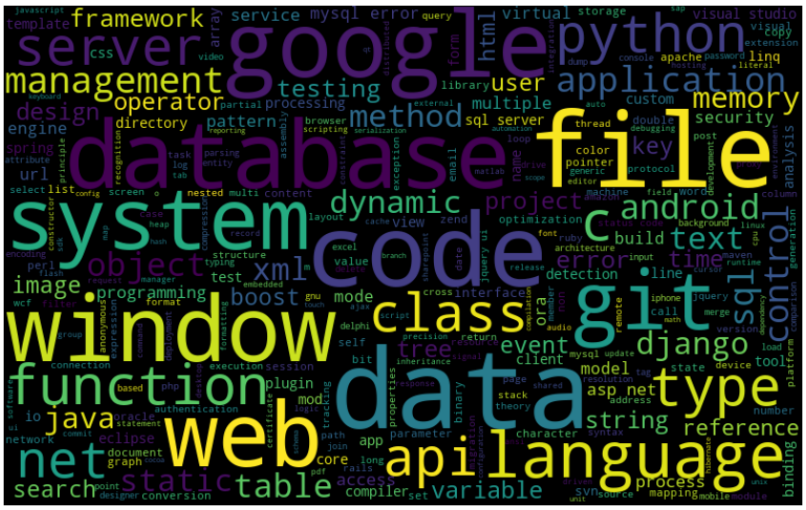


FIGURE 3.7 Nuage des tags.

4. Modélisation

4.1 Approche Non supervisée : Latent Dirichlet Allocation

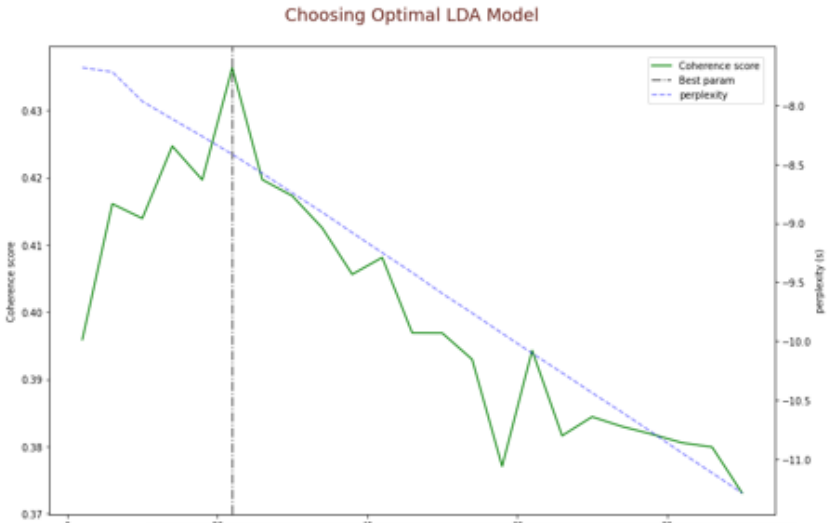
4.1.1 Méthode

L'allocation de **Dirichlet latente** (de l'anglais Latent Dirichlet Allocation) ou **LDA** est un modèle génératif probabiliste permettant d'expliquer des ensembles d'observations, par le moyen de groupes non observés, eux-mêmes définis par des similarités de données. En effet il s'agit d'un Topic Model qui pour un ensemble de documents donnés génère des sujets dont parlent ces documents avec une probabilité de liaison entre chaque couple (sujet - document).

Dans notre contexte de prédiction des tags, il nous restera impératif de retrouver la caractérisation des topics par des tags afin de savoir quels sont les tags qui correspondent à un topic précis généré par le modèle.

4.1.2 Optimisation des paramètres

L'un des paramètres les plus significatifs à optimiser reste le nombre de topics à générer par le modèle. En effet, ce dernier garde une influence sur la cohérence entre les topics générés et les documents. Pour cela, nous évaluons la cohérence et la perplexité du modèle en fonction du nombre de topics.




```
Entrée [99]: #best_nb_topics = coherence_test.loc[coherence_test.coherence.argmax(),"k"]
             nb_optimun = coherence_test.loc[coherence_test.coherence.argmax(),"k"]
             nb_optimun

Out[99]: 22
```

FIGURE 4.1 Recherche du nombre optimal de topics.

4.1.3 Prédiction des tags et performance

La prédiction des tags à travers les topics générés passe par l'établissement d'une correspondance entre ces topics et les tags suivant le schéma suivant :

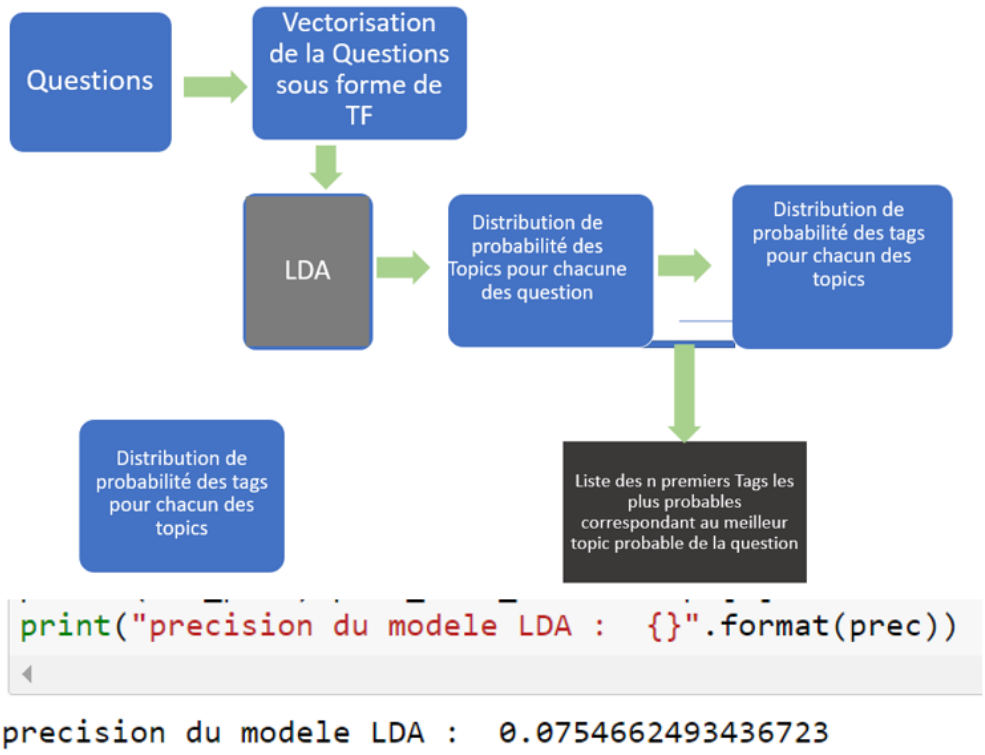


FIGURE 4.2 Schéma fonctionnel du modèle LDA.

Mais la précision du modèle étant très, et très faible, inexploitable, on est obligé de se retourner vers d'autres modèles.

4.2 Approches supervisées

4.2.1 Réduction des dimensions et échantillonnage des données

Pour tenir compte des mots rares pouvant caractériser un tag du fait de cette rareté, nous transformons notre corpus en TF-IDF.

Pour tenir compte des mots rares pouvant caractériser un tag du fait de cette rareté, nous transformons notre corpus en TF-IDF. Puis nous réduisons sa dimension en retenant exclusivement des questions correspondantes aux tags les plus fréquents.

Puis nous réduisons sa dimension en retenant exclusivement des questions correspondantes aux tags les plus fréquents.

Puis la cible de prédiction étant un vecteur de mots (un tag, deux ou plus) nous passons à la multi

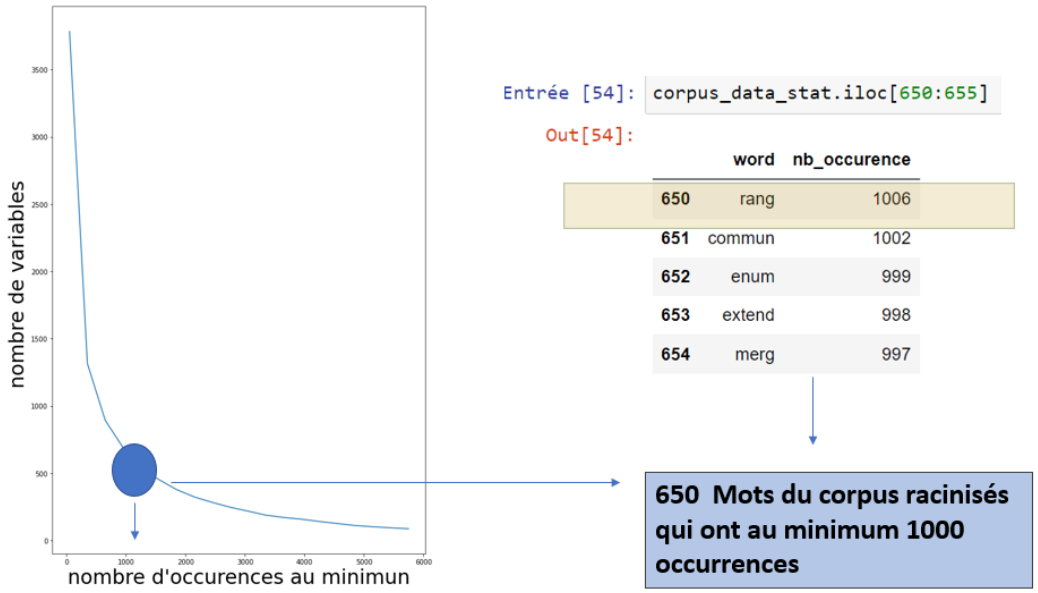


FIGURE 4.3 Réduction des dimensions du TF-IDF.

labélisation avant de procéder à l'échantillonnage

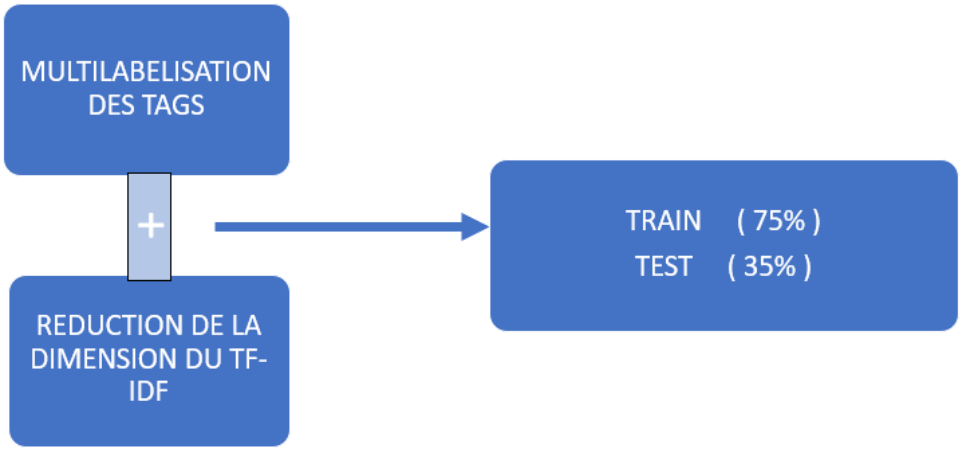


FIGURE 4.4 Multi-labélisation et échantillonnage.

4.2.2 Algorithmes entraînés

Le problème étant celui de la classification multi label, le choix des algorithmes a été fait à partir de leur capacité à prédire les sorties multiples. C'est ainsi que nous nous sommes retournés vers un panel

de 4 algorithmes résumés dans la figure 4.5.

ALGORIHMES ENTRAINES	PARAMETRES	STRATEGIE DE CROSS VALIDATION
KNN	grid_param = { 'leaf_size' : list(range(1,50)) , 'n_neighbors' : list(range(1,30)) , 'p': [1,2] }	
LOGISTIC REGRESSION	grid_param = { "estimator__penalty" : ["l1","l2"], "estimator__C" : [0.1,1.0,10,100], "estimator__solver": ["liblinear"]}	One-Vs-The- Rest classifier
RANDOM FOREST	Best param par gridsearch ({"estimator__min_samples_leaf": [1, 5, 10], "estimator__max_depth": [5, 15, 25,50], "estimator__class_weight": ["balanced"]})	One-Vs-The- Rest classifier
RANDOM FOREST CLASSIFIER CHAIN	====Best params Random Forest ===	

FIGURE 4.5 Modèles de classe supervisée.

4.3 Comparaison des performances et stratégies d'exploitation des modèles

En Précision, le modèle de la régression logistique est la plus meilleure Mais il cède sa première place au modèle de foret aléatoire sur le plan de la sensibilité. L'évaluation comparative au sens de la mesure de Jaccard vient confirmer le F1-score. C'est ainsi que les deux modelés (Foret Aléatoire et Régression Logistique) restent à privilégier dans la stratégie de déploiement

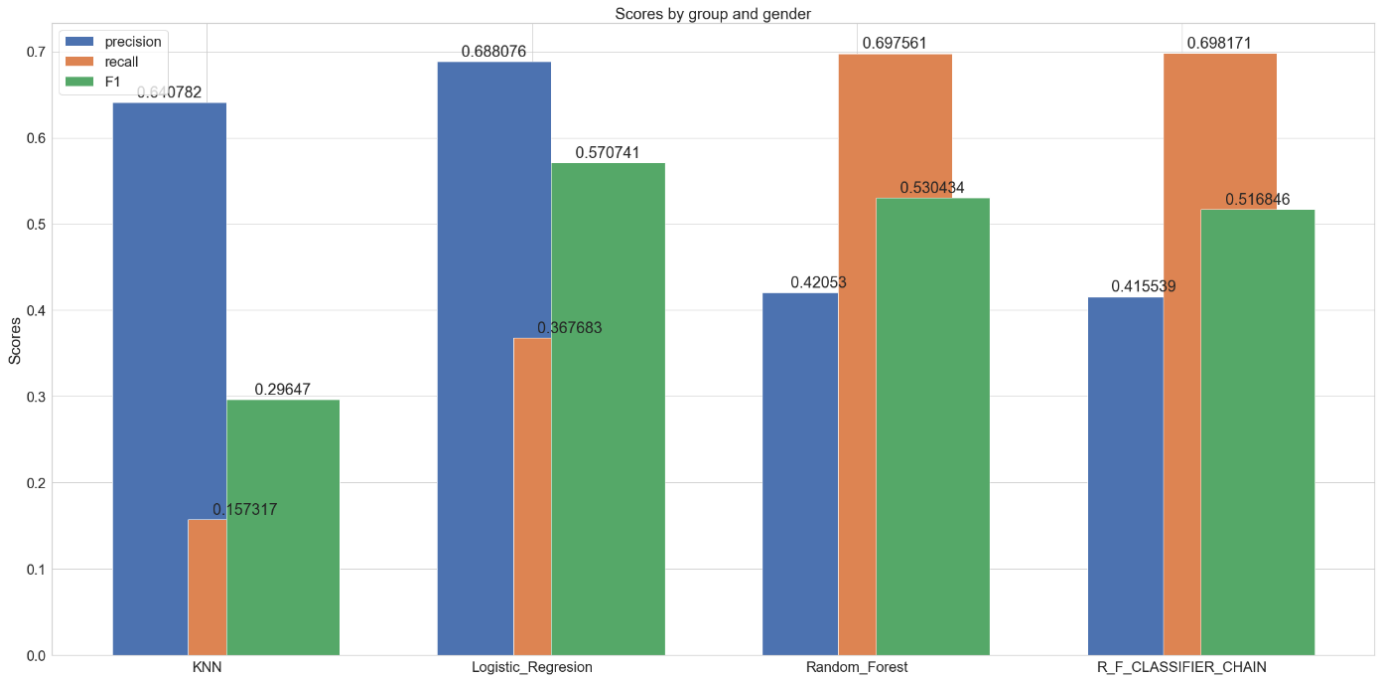


FIGURE 4.6 Quantification de la performance suivant F1 score.

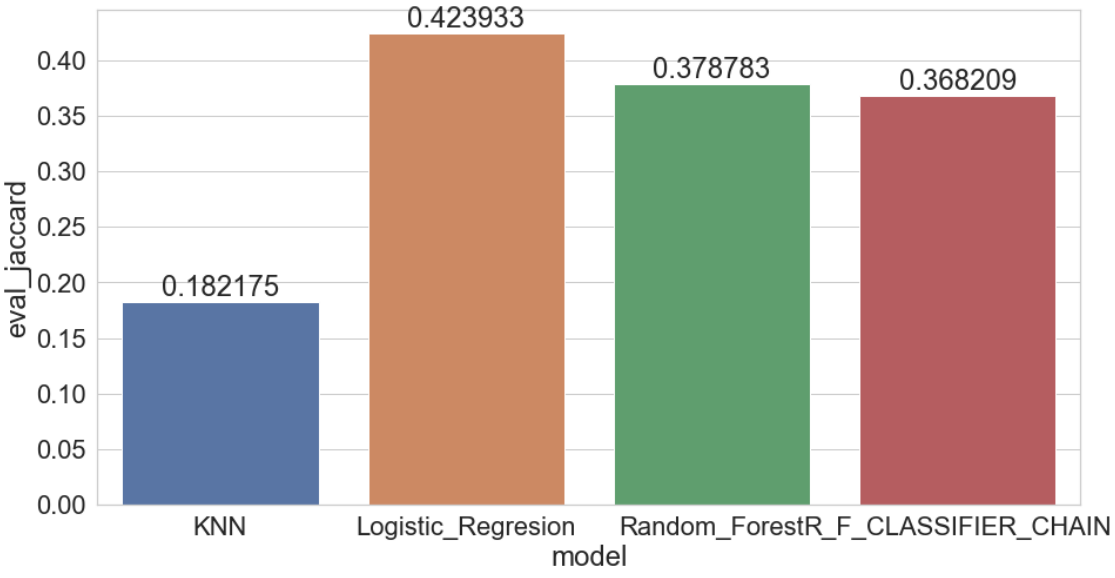


FIGURE 4.7 Quantification de la performance suivant Gaccard.

5. Déploiement

5.1 Schéma fonctionnel de l'API

La stratégie d'accroissement du pouvoir prédictif de l'application consiste en la combinaison des deux premiers modelés ayant une précision considérable. L'objectif étant de trouver des tags communs aux prédits de chacun des modèles, les proposer en tags les plus précis et garder la différence de cette intersection avec chacun des résultats comme autres tags de moindre précision, pouvant contribuer à la solution relative à la question posée . La figure 5.1 résume la structure fonctionnelle de l'API

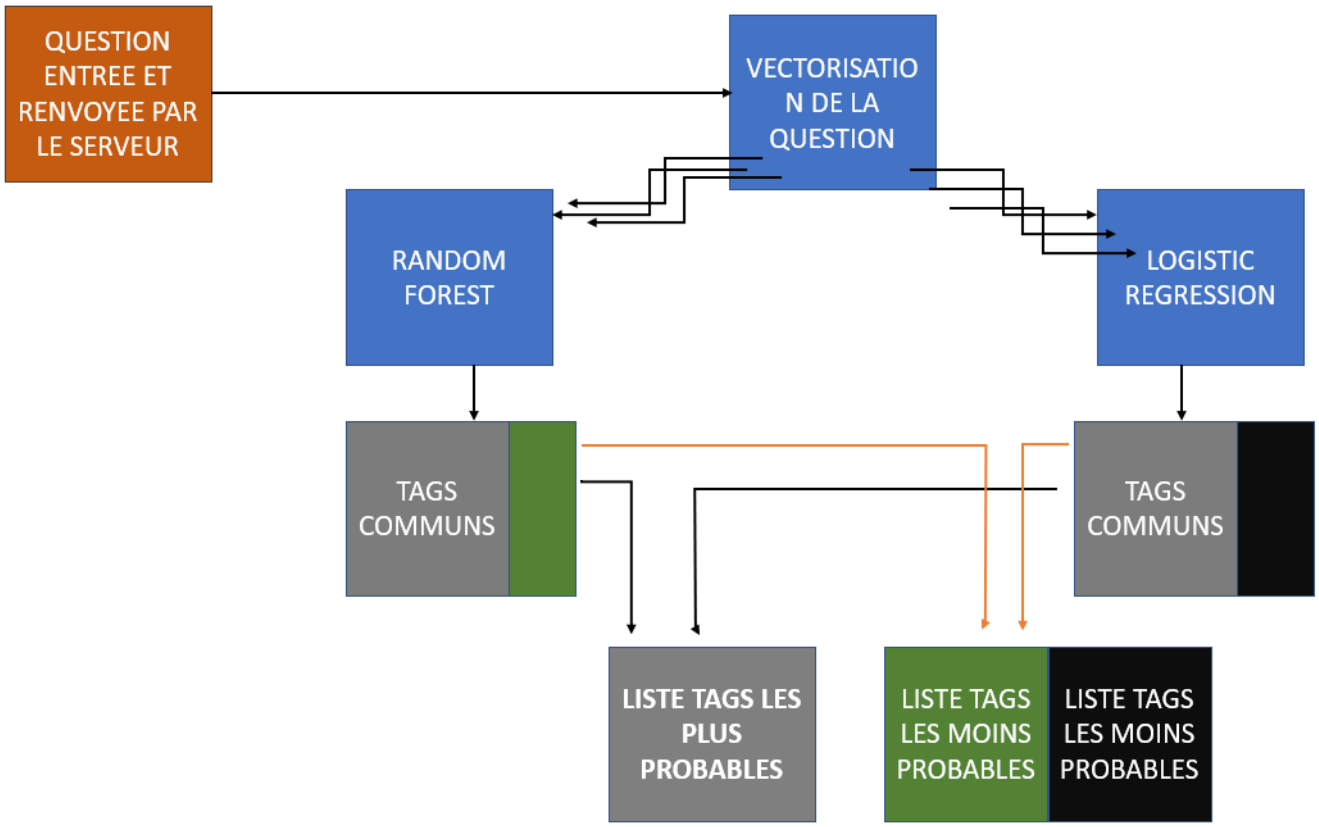


FIGURE 5.1 Schéma fonctionnel de l'API.

5.2 Aperçu du résultat

La figure 5.2 présente l'aperçu du résultat.

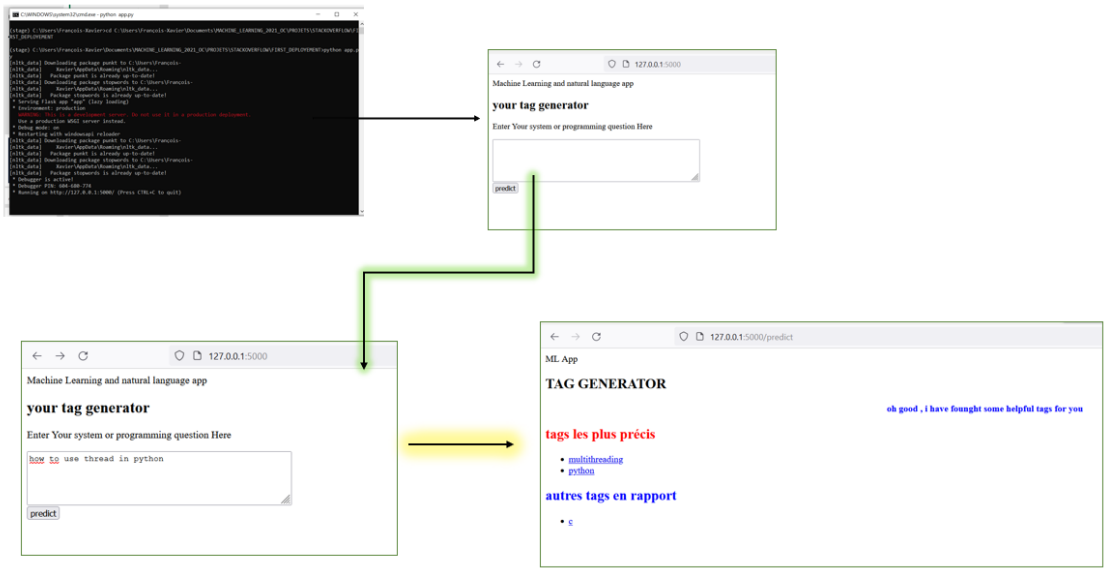


FIGURE 5.2 Illustration du déploiement.

Conclusion

Parmi les modèles entrainés, Certes le modèle Radom Forest meilleur en précision et le modèle de regression logistique meilleur en sensibilité, leurs performances attendues restent non suffisantes et ne permettent pas d'attendre quelques choses de plus sérieux en production. Le modèle LDA n'a même pus été convoqué en instance de comparaison avec d'autres modèles, tellement sa précision était presque nulle. C'est pourquoi il nous a semblé logique de procéder à une combinaison de modèles en lieu et place d'une prétendue sélection de modèle dans la conception de l'API, ceci afin d'augmenter légèrement la précision et la sensibilité des résultats. Cette insuffisance inattendue de la part du modèle LDA s'explique largement par le manque de volumétrie dans le corpus utilisé dans son entrainement, ceci à cause de sa gourmandise en mémoire et ressources de calculs, un impact aussi néfaste pour les modèles d'apprentissage supervisé abordés qui collent fortement aux données d'apprentissage et se prêtent par conséquent au surapprentissage. Une recherche d'optimisation de ces performances trouve sa solution idoine dans la massification des données d'entrainement en supervisé comme en non supervisé accompagnée des ressources adéquates pouvant supporter par ailleurs un nombre d'itérations considérables.

Références

[1] https://openclassrooms.com/fr/courses/4470541-analysez-vos-donnees-textuelles.
[2] https://en.wikipedia.org/wiki/Multi-label_classification.
[3] https://openclassrooms.com/fr/courses/4425066-concevez-un-site-avec-flask.
[4] https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/.
[5] https://towardsdatascience.com/understanding-random-forest-58381e0602d2.