A close up of a logo

Description generated with very high confidence

|  |
| --- |
| CapGemini Invent |
| Rapport de projet : Catégorisez automatiquement des questions |
| Open Classroom |

|  |
| --- |
| François Lemeille  27/09/2018 |

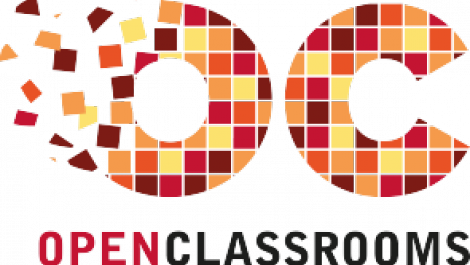


Table of Contents

[I. Introduction 2](#_Toc527384059)

[1. Contexte du projet 2](#_Toc527384060)

[2. Aperçu des données 2](#_Toc527384061)

[II. Nettoyage et premières analyses 3](#_Toc527384062)

[1. Corps et titre de la question 3](#_Toc527384063)

[a. Utilisation de sac de mots 3](#_Toc527384064)

[b. Utilisation d’expressions régulières 3](#_Toc527384065)

[c. Uniformisation de la casse 3](#_Toc527384066)

[d. Suppression des « stop words » 3](#_Toc527384067)

[e. Racinisation des mots 4](#_Toc527384068)

[2. Tags 4](#_Toc527384069)

[3. Premières analyses 4](#_Toc527384070)

[III. Méthode « Semi-supervisée » 5](#_Toc527384071)

[1. Approche 5](#_Toc527384072)

[2. Construction d’une métrique 6](#_Toc527384073)

[3. Résultats 7](#_Toc527384074)

[IV. Méthode supervisée 8](#_Toc527384075)

[1. Approche 8](#_Toc527384076)

[2. Résultats 10](#_Toc527384077)

[3. Approfondissements 10](#_Toc527384078)

[V. Conclusion 11](#_Toc527384079)

# Introduction

## Contexte du projet

Dans le cadre de ce cinquième projet, l’objectif est de créer un outil de proposition automatique de tags à partir de questions. Les questions considérées sont des questions provenant de StackOverflow, un célèbre forum d’échange dans le domaine de la programmation. Les questions portent donc sur des problèmes de développement informatique et c’est la communauté qui apportent les réponses. Un système de vote a également été mis en place afin d’identifier les questions et les réponses jugées les plus utiles, les plus claires et les mieux formulées.

StackOverflow, réelle référence du domaine, propose également d’associer des tags lorsque à la question posée par l’utilisateur pour une meilleure identification et classification des différentes questions. L’utilisation des tags nécessite une certaine connaissance du domaine de la programmation et de l’utilisation de StackOverflow. C’est dans ce but qu’a été développé l’outil de suggestion de tags : pour aider des utilisateurs novices à tagger leur question à partir des propositions de notre outil.

## Aperçu des données

Stack Overflow propose un outil d’export de données : "stackexchange explorer", qui recense un grand nombre de données authentiques de la plateforme d’entraide. Il est possible d’exécuter des requêtes sur cette API permettant, par exemple, de récupérer 50000 questions ayant un score (une note attribuée par le utilisateurs) supérieur à 50, ce qui permet de garantir que la question a été utile à la communauté et donc a bien été posée.

A screenshot of a cell phone

Description generated with very high confidenceNous obtenons la base de données suivante :

# Nettoyage et premières analyses

La visualisation des données extraites permet de se rendre compte que plusieurs traitements vont être nécessaires pour pouvoir manipuler celles-ci correctement.

## Corps et titre de la question

Dans un premier temps il s’agit de traiter la question en elle-même ainsi que le titre. Dans un souci de simplification et d’efficacité, le corps de la question et le titre ont été concaténés et seront traités dans la suite comme une seule et même entité de texte, nommée question par la suite.

L’observation de différents exemples montre la présence de nombreuses balises dans le corps de la question et des caractères spéciaux.

Les traitements qui sont appliqués à la question sont les suivants :

* Suppression des balises
* Suppressions des caractères spéciaux
* Uniformisation de la casse
* Suppression des mots courants, non différenciant du point de vue programmation
* Racinisation des mots

### Utilisation de sac de mots

La conversion de la question en sac de mots permet de considérer chaque mot indépendamment les uns des autres et ainsi de tous leur appliquer les mêmes traitements. Les mots sont isolés lorsqu’ils sont séparés par un espace, de telle sorte que chaque suite immédiate de caractère sera considérée comme un mot.

### Utilisation d’expressions régulières

Les expressions régulières sont représentées par des chaines de caractère qui permettent d’identifier et donc de sélectionner ou supprimer des types de caractères (texte, nombres, autres etc). Ici, une expression régulière est appliquée pour ne garder que les caractères alphabétiques. Celle-ci est couplée avec un filtre permettant de supprimer les « mots » de longueur 1 (i.e les lettres seules).

### Uniformisation de la casse

L’uniformisation de la casse (passage en minuscule) permet d’éviter que deux mêmes mots soient compris de manière différente lorsque l’un est écrit avec une majuscule.

### Suppression des « stop words »

Il existe de nombreux mots ou expressions « nécessaires » au language, ces mots récurrents sont employés dans les commentaires mais ne vont pas être différenciant selon les commentaires. On retrouve notamment dans les « stop words » les pronoms, les articles déterminants etc.

Une liste pré-définie est disponible dans de nombreux packages de nlp sous python, dans cette étude c’est le module « stop\_words » qui a été utilisé. Cette liste a été enrichie après une rapide étude sur les mots les plus fréquents, ce qui a permis de supprimer des mots récurrents dans notre corpus n’apportant pas de discrimination au niveau du sujet de la programmation.

### Racinisation des mots

La racinisation des mots correspond au fait de remplacer le mot considéré par sa racine. Ainsi un verbe conjuguer sera remplacé par son infinitif et un mot accordé (au féminin par exemple) sera remplacé par le singulier masculin. Cela permet de regrouper sous un même mot les mots ayant le même sens, mais qui n’aurait pas été considérés comme tels par la machine. Le module NLTK, très utilisé dans l’étude le traitement du langage naturel (NLP en anglais), possède une fonction implémentée réalisant cette opération : WordNetLemmatizer.

## Tags

Il est également nécessaire d’appliquer un traitement sur les tags. En effet ceux-ci sont récupérés au sein de < >. Ces balises sont donc supprimées, et les tags séparés par un espace.

L’objectif est ici d’apprendre à prédire les tags en fonction de la question. Il est donc nécessaire d’avoir de considérer des tags suffisamment présents pour que la machine puisse identifier quels éléments sont liés à ce tags. A cet effet l’étude a été limité aux 1500 tags les plus fréquents. Cela semble être un compromis raisonnable entre les 52000 tags présents dans notre jeu de données initial, une certaine exhaustivité et l’argument cité au-dessus. Avec cette sélection les tags considérés les moins fréquents sont tout de même tous présents sur plus de 4 questions.

## Premières analyses

A screenshot of a cell phone

Description generated with high confidencePour avoir une première idée de ce que contient les questions et de comment se comporte le corpus, il est utile d’afficher les mots les plus fréquents selon les différents champs de textes. Ici : Question (Corps + Titre), Titre et Tags.

A screenshot of a cell phone

Description generated with high confidence

A screenshot of a cell phone

Description generated with very high confidence

Ensuite, on peut s’attacher à vérifier si les les mots correspondant aux tags apparaissent dans la question (titre + corps) :

Dans 85.6% des cas, au moins 1 tags est présent dans la question, mais seulement dans 12.9% des cas tous les tags sont présents.

# Méthode « Semi-supervisée »

Un des modèles très utilisé et très efficace lorsqu’il s’agit d’identifier des sujets au sein d’un corpus est le Latent Dirichlet Allocation (LDA). Dans le cadre de cette étude ce modèle peut être adapté, grâce à quelques étapes supplémentaires, pour relier les questions des utilisateurs aux tags correspondants.

## Approche

Grossièrement, l’idée du modèle LDA est de construire, à partir du corpus et du nombre de « thèmes » (entre autres), une distribution d’appartenance de chaque mot présent dans le corpus à un thème. Chaque thème est représenté par une distribution des mots qui le composent et plus la valeur liée à ce mot est importante pour le mot donné, plus le mot est caractéristique de ce thème.

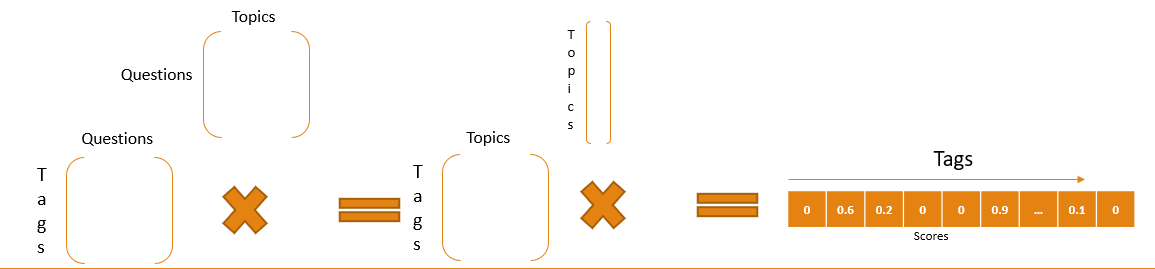
L’objectif est donc de lié notre corpus de question, nos thèmes identifiés par la LDA et notre liste de tags retenus.

Associer les questions et les tags est simple, le lien se fait par une matrice creuse (questions, tags) ou chaque question est flaggée des tags associés.

Associer les questions aux thèmes est justement le résultat de la LDA, qui fournit une matrice (questions, thèmes).

En multipliant la première matrice par la transposée de la seconde nous obtenons une matrice représentant le lien entre thème et tags. L’idée est d’ensuite appliquer notre modèle de LDA à la nouvelle question de l’utilisateur afin d’obtenir sa distribution de thèmes. Ensuite en multipliant ce vecteur à notre matrice thèmes/tags nous obtenons sa distribution de tags.

Voici un schéma permettant de résumer cette approche :



Reste alors à définir le nombre de tags que l’on souhaite prédire, ou le seuil d’acceptance à partir duquel le tags est prédit. Cette problématique est traitée dans la prochaine partie.

## Construction d’une métrique

Il est important dans le cadre d’une étude de ce type de définir une métrique, une règle de calcul de score, afin d’évaluer nos résultats de manière tangible et de pouvoir les comparer. Dans le cas présent, une classification multi-labels, plusieurs pistes sont possibles. Une des plus répandue est l’accuracy. Elle est définie comme le nombre de prédictions correctes divisé par le nombre d’individus à prédire. Dans le cas d’une classification multi-labels une prédiction est considérée comme positive si et seulement tous les tags prédits correspondent **exactement** aux tags réels.

Dans le cas présent cette métrique ne représente pas forcément le meilleur moyen d’évaluer notre modèle. En effet si le modèle prédit correctement 3 tags d’une question comportant 4 tags réels alors cette prédiction sera considérée simplement comme fausse, alors que cela semble être une prédiction respectable.

Une métrique plus spécifique doit être envisagée. La métrique qui a donc été proposée pour cette étude est la suivante :

Score\_implementé = MoyenneHarmonique(Rappel, Precision)

=)

Cette métrique permet de prendre en compte plusieurs choses :

* Elle affine le score selon le nombre de tags prédits correctement, le score peut prendre alors d’autres valeurs que 0 ou 1 pour une question
* Elle permet de trouver nombre de tags prédits et du nombre de tags réels.   
  En effet ne considérer que le nombre de tags réels incite à prédire un nombre important de tags pour augmenter les chances d’obtenir des prédictions correctes. Mais ne considérer que le nombre de tags prédits encourage à prédire un nombre très réduit de tags afin de limiter les chances de se tromper.

Cette métrique est ajustable par le nombre de tags prédits, qui est à la main de l’utilisateur du modèle. Il est alors possible de réaliser une optimisation de cette variable sur le jeu de test ou par validation croisée :

Un bon compromis ici semble de prédire 5 tags. La note est artificiellement plus élévée pour un nombre de tags prédits plus important, mais en prenant en compte l’aspect de l’étude, il apparait que prédire plus de 5 tags ferait apparaitre le modèle comme proposant de trop nombreux tags à l’utilisateur.

## Résultats

Pour pouvoir comparer les résultats du modèle, un score de base appelé ‘baseline’ a été calculé. La méthode derrière la baseline est très naïve : Le tag le plus fréquemment observé dans la base d’apprentissage est prédit comme résultat.

|  |  |  |  |
| --- | --- | --- | --- |
| Métrique | Précision | Rappel | Score implementé |
| Baseline | 0,042 | 0,097 | 0.7 |
| Modèle par LDA | 0,14 | 0,19 | 0.17 |

Le modèle s’appuyant sur la LDA permet donc d’obtenir des résultats intéressants. En effet il y a une nette amélioration par rapport à la baseline. Toutefois les scores obtenus ne sont pas encore totalement satisfaisant pour réaliser une suggestion fiable.

# Méthode supervisée

Le modèle LDA n’est pas nativement un modèle de classification, il existe des modèles réalisant cette tache de manière plus efficace. Ces algorithmes, comme la LDA finalement, s’entrainent sur des données numériques, et il faut donc transformer nos données textuelles.

## Approche

Les données considérées correspondent aux données obtenues après l’étape de nettoyage décrite précédemment dans ce rapport.

#### Vectorisation

L’idée est de créer un dictionnaire à partir de toutes les questions considérées. Tous les mots présents dans les questions (après traitement) seront présents dans le dictionnaire et correspondront à une colonne de notre dictionnaire. Chaque question sera alors représentée par rapport à ce dictionnaire. Il y a plusieurs moyens de représentation, de vectorisation :

* Par le nombre d’occurrences du mot (ex : CountVectorizer)
* Binairement, indication ou non de la présence ou non du mot
* Par le nombre d’occurrence pondéré. La pondération peut notamment se faire par la proportion de commentaires contenant ce mot. (Tf-idf : Term frequency inversed document frequency).

Les outils de vectorisation disponibles sur python acceptent d’autres paramètres, comme notamment le nombre minimum d’occurrences du mot pour le considérer, et la fréquence d’apparition maximale autorisée. Cela permet de filtrer les mots trop peu présents, et trop présents (trop généraux et donc non différenciants).

#### Choix des modèles

Plusieurs algorithmes sont populaires pour ce genre d’étude sur des données textes, notamment ceux reprenant les machines à vecteur de support et les arbres de décisions. Les algorithmes python les plus usités sont LinearSVC, RandomForest, NaiveBayes et SGD Classifier. Ce sont donc les modèles utilisés dans la suite de l’étude.

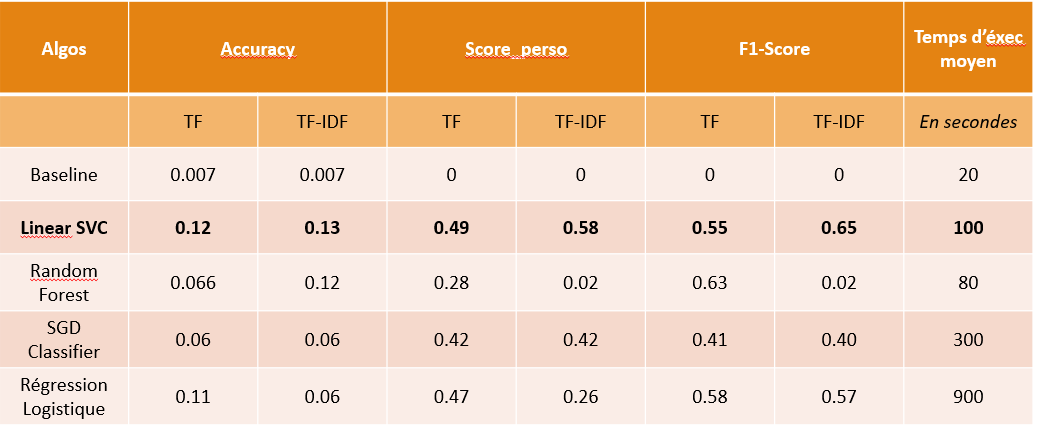
Dans le cadre d’une classification multi-labels comme ici, il est commun d’utiliser un nombre de modèle égal au nombre de classe. Chaque modèle permet de classifier une classe par rapport à l’union des autres classes (le reste de l’univers).

#### Métrique

La métrique considérée est la même que pour la méthode non supervisée, ce choix permet de pouvoir comparer les deux approches de manière similaire.

L’accuracy est également envisagée ici, car cela reste une des métriques les plus populaires pour ce genre d’étude de classification multi labels.

## Résultats



Le modèle LinearSVC est l’algorithme qui permet d’obtenir de très bonnes performances en respectant un tant d’exécution acceptable. C’est donc le modèle retenu (avec une vectorisation TF-IDF des données textes) et implémenté dans l’interface homme machine résultant du projet.

## Approfondissements

Afin d’aller un peu plus loin dans cette étude, et d’explorer de nouvelles choses il a été mené deux approfondissements :

* Utilisation d’un modèle probabiliste : Utilisation de SGD Classifier avec la métrique loss ‘modified\_huber’ et la fonction predict\_proba.   
  Les prédictions de ce modèle sont rendues sous forme de probabilités d’appartenance de chaque tag à la nouvelle question. Cela permet de pouvoir ajuster les choix de prédictions en fonction du seuil à partir duquel un tag est effectivement prédit par le modèle ou non. Car le nombre de tag effectivement prédit évolue avec le seuil : quand ce dernier augmente le nombre de tag prédit diminue.
* Optimisation des paramètres du modèle retenu par GridSearchCV. GridSearch CV est un outil permettant d’optimiser plusieurs paramètres d’un modèle sur une grille définie par différentes valeurs de ces paramètres. Cela permet notamment d’optimiser le nombre d’itérations maximal à chaque étape, ou encore la pénalisation de l’erreur (ce qui influe sur l’équilibre biais-variance de notre modèle).  
  L’optimisation de ces paramètres rend très souvent, et c’est le cas ici, le modèle plus performant.

# Conclusion

Les ressources, les résultats et les supports de cette étude se trouvent sur GitHub au lien suivant : <https://github.com/FrancoisLem/OC-Projet-5-Question-Tagging>

Une interface Homme-Machine a été développée et propose à partir d’une question et de son titre une sélection de tags à lier avec la question. Elle se trouve au lien suivant : <https://oc-automatic-tagging-fl.herokuapp.com/>

J’aimerais remercier mon mentor Amine Abdaoui pour sa précieuse aide tout au long de ce projet, qui m’a beaucoup apporté et sans qui cette étude n’aurait pas été aussi riche.