Projet 5

Catégorisez automatiquement des questions

Novembre 2020 Maud Charbonneau

Sommaire

Présentation du projet et des données

Introduction

Stack Overflow est l'un des sites les plus connus de questions-réponses pour la programmation informatique. Il permet d'aider n'importe quel utilisateur à trouver une solution à son code qui ne marche pas ou à surmonter des blocages en accédant aux posts d'autres utilisateurs puisque beaucoup de problèmes ont déjà été reportés et/ou surmontés par d'autres. Toutes les questions posées, ainsi que les commentaires des réponses, sont reliées à plusieurs tags, permettant de savoir quel est le thème abordé. Ces tags sont choisis par les utilisateurs au moment de poster leur question.

Cependant, ayant un très grand nombre d'utilisateurs, et de plus en plus de nouveaux, il peut être compliqué pour ces derniers de sélectionner les tags qui correspondent au mieux. En effet, lorsque quelqu'un souhaite poser sa question, il doit entrer un titre, le corps de la question et jusqu'à 5 tags. Pour entrer les tags, des suggestions sont proposés au fur et à mesure que l'utilisateur écrit le nom du tag, mais cela n'aide pas à choisir les tags le plus pertinents.

Il pourrait donc être utile de proposer automatiquement des tags correspondants au titre et au contenu du texte écrit par la personne. Cela permettrait à tous les utilisateurs de trouver les questions qu'ils cherchent plus facilement, mais aussi d'éviter les questions répétitives. L'utilisation du site serait donc plus simple et agréable et les questions seraient mieux classées.

Le but du projet est donc d'utiliser les données pour créer un algorithme de Machine Learning qui assignera automatiquement un tag pertinent au contenu, soit au titre et au corps de la question.

Données

La base de données de Stack Overflow contient plus de 20 millions de questions. Pour ce projet, nous sélectionnerons dans un premier temps un échantillon de 250 000 questions, qui seront rapidement réduits à environ 60 000 questions. Ces questions ont été sélectionnées par une requête SQL sélectionnant les posts possédant un titre, un corps et un ou plusieurs tags. Les données proviennent également des posts les plus vus puisque ces deniers répondent à des problèmes récurrents.

```
Taille de la base train: (204432, 9)
Taille de la base test: (51109, 9)
```

La base comprend l'identifiant de l'utilisateur, le score de la question, le nombre de vues, le corps de la question, le titre, les tags, le nombre de réponses, le nombre de commentaires ainsi que le nombre de favoris.

	Id	Score	ViewCount	Body	Title	Tags	AnswerCount	CommentCount	FavoriteCount
0	26477388	23	5693.0	In SBT is the use of aggregate fol...	Is the use of 'aggregate' following by 'depend...	<build><sbt>	1.0	0	9.0
1	20580028	22	37747.0	I'm designed a <a href="http://en.wikipedia...	Flowchart "for each" loop loop without variabl...	<flowchart>	5.0	1	2.0
2	15096219	22	21065.0	I'm trying to register a new log \n\n<pr...	How to create a folder (if not present) with L...	<ruby-on-rails><ruby><logging>	3.0	0	4.0
3	16853747	87	11309.0	<pre><code>class Test{\n public static void...	Static block in Java not executed	<java><static><access-modifiers>	5.0	2	26.0
4	2036744	27	21402.0	I have to write some code in ML and it is m...	ML IDE and Compiler for Windows or Linux or Mac	<ide><compiler-construction><programming-langu...	6.0	0	17.0

Premières lignes du jeu de données

Cleaning

Cleaning de la question

Le nettoyage se fait en plusieurs étapes :

- Joindre titre et corps en une seule colonne
- Extraire les balises html
- Tokenisation et suppression des caractères spéciaux et majuscules (a)
- Suppression des stopwords (b)
- Lemmatisation (c)

Voici un exemple des trois dernières étapes sur une phrase comme :

“Hello these are 3 examples to show the different steps of the cleaning process.”

(a): [“hello”, “these”, “are”, “examples”, “to”, “show”, “the”, “different”, “steps”, “of”, “the”, “cleaning”, “process”]

(b): [“hello”, “examples”, “show”, “different”, “steps”, “cleaning”, “process”]

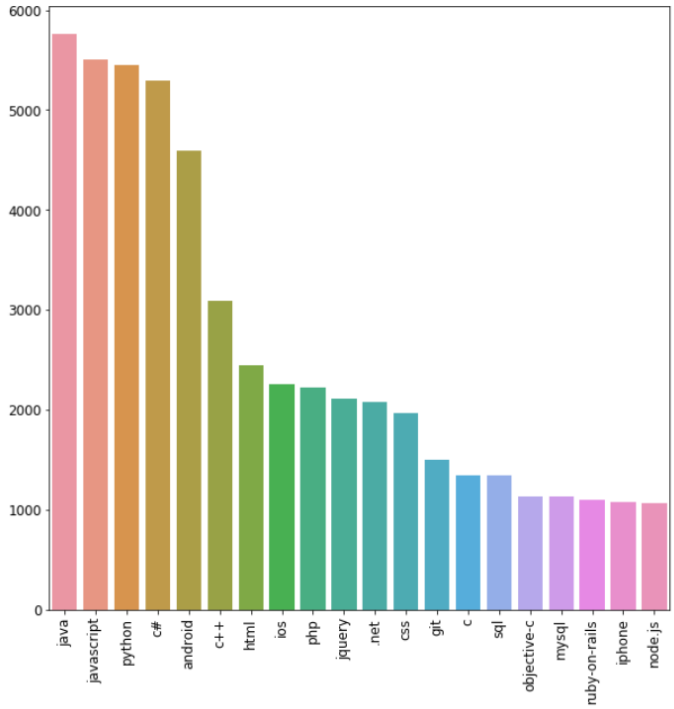
(c): [“hello”, “example”, “show”, “different”, “step”, “cleaning”, “process”]

Cleaning des tags

Le nettoyage des tags consiste en deux étapes :

- Suppression des caractères spéciaux
- Filtrage des filtres

Le filtrage consiste à sélectionner un nombre défini de tags dans les plus utilisés. Pour ce projet le nombre de tags choisi est 20 donc les tags proposés automatiquement seront les 20 tags les plus utilisés sur le site. Le résultat est affiché dans un graphique pour visualiser le nombre d’utilisation de ces tags.

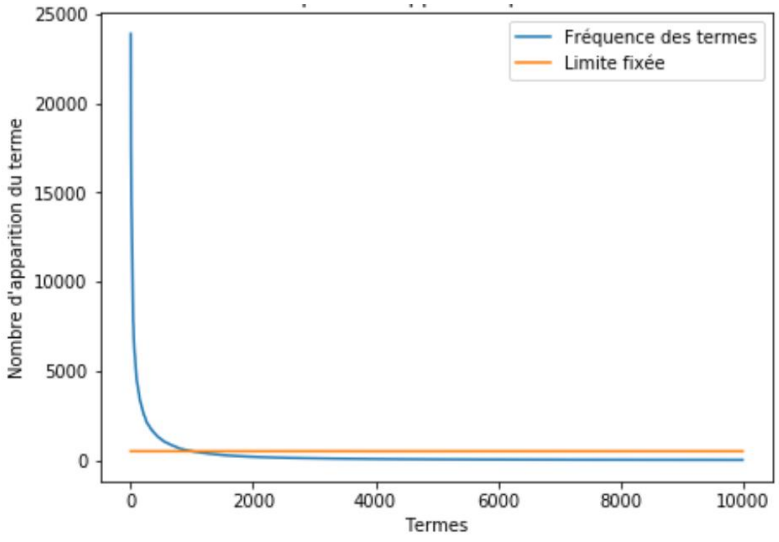


Fréquence d'utilisation des tags

Fréquence des termes

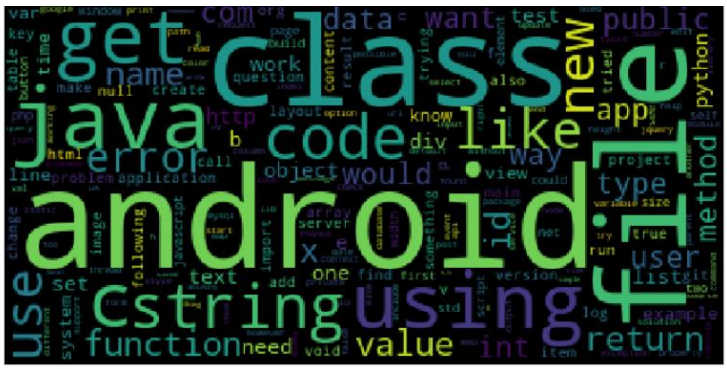
La prochaine étape du projet consiste à calculer la fréquence de tous les termes utilisés dans les questions. Le but est de pouvoir améliorer la liste des stopwords pour rendre l'analyse des mots plus efficace.

Beaucoup de termes sont utilisés très souvent et n'apporte pas beaucoup d'information utile mais prennent beaucoup de place. Pour cette raison, on fixe une limite, ici limite = 500, afin d'enlever tous les termes très peu utilisés. On peut voir sur le graphique ci-dessous que le nombre de termes utilisés très fréquemment est une très petite partie de tous les termes.



Fréquence d'apparition des termes

Par la même occasion, on visualise les termes les plus fréquents dans ce graphique. Plus le mot est grand, plus il est fréquent.



Termes les plus fréquents

Tf-Idf

Le tf-idf, ou Term-Frequency - Inverse Document Frequency, est une méthode d pondération permettant de créer une matrice contenant des mots ainsi que l'importance de chacun des mots par rapport à l'ensemble des documents. Dans le cas de ce projet, cette méthode permet d'évaluer l'importance d'un terme dans une question, relativement à toutes les autres questions. Le poids varie selon le nombre d'occurrence du terme dans le texte ainsi qu'en fonction de la fréquence du mot dans l'ensemble des questions.

La valeur, ou le poids de chaque terme, est défini par le tf-idf de cette façon :

$$poids = fréquence\ du\ terme \times idf(terme)$$

Idf(terme) correspond à l'inverse de la proportion de questions qui contient le terme.

Voici la matrice résultante de l'application du tf-idf sur la base de test.

	able	accept	access	according	account	achieve	across	action	active	activity	...	xcode	xml	xmlns	year	yes	yet	z	zero
0	0.010169	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.013161	0.0	0.000000
1	0.000000	0.0	0.0	0.0	0.0	0.015352	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.015575	0.0	0.000000
2	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000
3	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.012778
4	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000

Matrice mots/poids en sortie du tf-idf

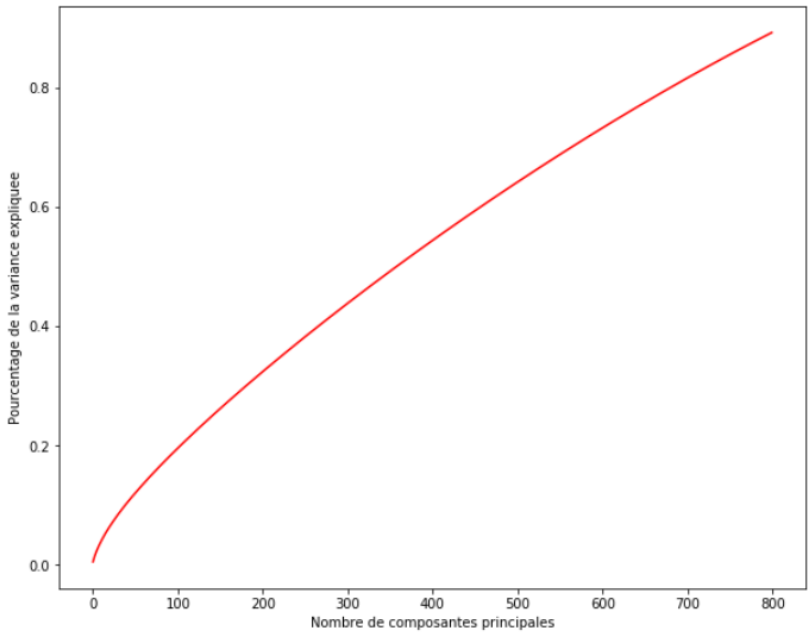
Analyse exploratoire

Analyse en Composantes Principales

L'ACP est une méthode qui permet de réduire la dimension des données en trouvant une nouvelle base dans laquelle les données sont représentées de telle sorte que la variance dans cette nouvelle base soit maximisée. L'objectif dans ce projet est de ne conserver qu'une partie

des variables pour alléger la base de données, simplifier la modélisation et garder les variables les plus importantes.

Pour sélectionner le nombre de composantes à garder, on affiche l'éboulis des valeurs propres. De cette façon on observe le pourcentage de variance expliquée en fonction du nombre de composantes principales. Dans le cadre du projet, le nombre de composantes a été fixé à 800 pour pouvoir représenter plus de 80% de la variance.



Eboulis des valeurs propres

Modélisation

Modélisation non supervisée

Latent Dirichlet Allocation

La méthode LDA se base l'hypothèse que chaque mot possède une certaine distribution par thème. Or dans l'ensemble des documents, certains thèmes sont abordés en différente proportion. Donc chaque document peut renvoyer des mots-clés pertinents au sujet abordé.

On applique donc cette méthode dans le cadre de notre projet pour visualiser les thèmes des questions et se rendre compte s'il est possible de comprendre le contenu de la question grâce aux mots clés obtenus.

```
Topics in LDA model:
Topic #0:
use javascript using would page like code way file j know html browser c jquery function window application one google need exa
mple time want image work question good java find script web chrome event used user library get net looking
Topic #1:
public thread class method exception static void task java catch private new system difference async throw test interface null
println try return string block queue object code main call e final instance wait console boolean writeline args run override i
nt
Topic #2:
java android org jar eclipse com gradle maven spring annotation dependency hibernate lang junit xml bean compile build support
google plugin apache sun class project util internal springframework version activitythread servlet jdk test http main error so
urce property v groupid
Topic #3:
android view layout button image activity app self color intent item text id io screen height swift fragment want width parent
r background set change textview action bar programmatically new dialog animation notification show drawable size application k
eyboard like menu
Topic #4:
date input datetime form jquery button value text event click day function option time td javascript div label element format s
elect month type id field html get mm submit name checkbox want var hour using alert child box year tr
Topic #5:
file python error project version app run install studio android c path application build window directory module package py us
ing command library get xcode code running test lib installed use line ruby system visual folder import debug php program proce
ss
```

Mots clés résultant de la LDA pour chaque document

Voici une partie du résultat obtenu. Chaque document renvoie des mots-clés différents qui permettent d'avoir une idée des tags qui pourraient y être associés. Cependant, certains sont trop similaires pour pouvoir les différencier. Tous les sujets ne sont donc pas assez précis.

Modélisation supervisée

Choix des données

Pour tester différents modèles et trouver le meilleur, on crée une nouvelle base provenant de X_train et Y_train, uniquement utilisée pour choisir le modèle, appelé base 2. Le jeu de données étant très important, pour vérifier que les modèles fonctionnent, on fait un premier test sur une autre base, la base E, provenant des 10 000 premiers éléments de X_train et Y_train.

Modèles testés

Après avoir utilisé l'ACP pour réduire la dimension des données, on peut tester différents modèles de modélisation. Les modèles choisis pour être testés sont :

- Arbre de décision
- Régression logistique
- SVM

Choix du modèle

Pour définir le meilleur modèle, il est possible d'évaluer plusieurs types de scores. La précision, le rappel ou le f-score.

- La précision mesure la capacité du système à refuser les solutions non-pertinentes

- Le rappel mesure la capacité du système à donner toutes les solutions pertinentes.
- Le F-score, ou F-mesure, mesure la capacité du système à donner toutes les solutions pertinentes et à refuser les autres.

Le F-score est un compromis entre la précision et le rappel qui donne la performance du système et dans le cadre du projet c'est le score le plus pertinent.

Après s'être assuré du bon fonctionnement des différents modèles, on passe à la modélisation sur la base 2. Les scores obtenus sont les suivants :

- Arbre de décision : 0.39
- Régression Logistique : 0.66
- SVM : 0.68

Le modèle avec le meilleur score est donc la SVM. Cependant, ayant eu des problèmes à faire tourner les modèles, j'ai voulu comparer également le temps d'exécution. On obtient les temps suivants :

- Arbre de décision : 101.86959886550903
- Régression Logistique : 24.22258186340332
- SVM : 1478.547325372696

On remarque que le temps d'exécution de la SVM est plus de 60 fois supérieur à celui de la régression logistique et n'est donc pas exploitable p. Bien que le score soit légèrement inférieur à celui de la SVM, on choisit donc comme modèle la régression linéaire.

Tuning des paramètres et modèle final

Une fois le modèle sélectionné, il faut tenter de l'optimiser. Pour cela, on utilise GridSearchCV afin de tester différentes combinaisons de paramètres. Les meilleurs paramètres ainsi que le meilleur score correspondant est obtenu ainsi et définissent le modèle final.

Les trois paramètres variables dans le cadre de notre projet ainsi que les meilleurs paramètres sont les suivants :

- Penalty : Utilisé pour spécifier la norme utilisée dans la pénalisation.
    - Meilleur paramètre : l2
- C : Inverse de la force de régularisation, plus la valeur est petite et plus la régularisation est forte.
    - Meilleur paramètre : 0,01
- Max_iter: Nombre maximum d'itérations nécessaires pour que les solveurs convergent.
    - Meilleur paramètre : 100

Validation du modèle

Pour valider le modèle, on définit le modèle avec ses meilleurs paramètres avant de le tester sur les données test de la première base. On obtient un score final de 0,71.

model_final = LogisticRegression(penalty='l2', C=0.01, max_iter=100)

Conclusion