



CentraleSupélec

&

OPENCLASSROOMS

# **Catégoriser automatiquement des questions**

Certificat « Ingénieur Machine Learning »

Présenté par : Xavier Barbier

**Résumé :** Stack Overflow est un site web proposant des questions et réponses sur un large choix de thèmes concernant la programmation informatique. Afin de retrouver facilement les questions le site permet la soumission de tags. L'objectif de ce projet est d'étudier les différentes possibilités d'automatisation de la soumission des tags. Plusieurs modèles ont été créés à partir des données textuelles des questions. Un point d'entrée d'une API est proposé sous la forme d'une web-app.

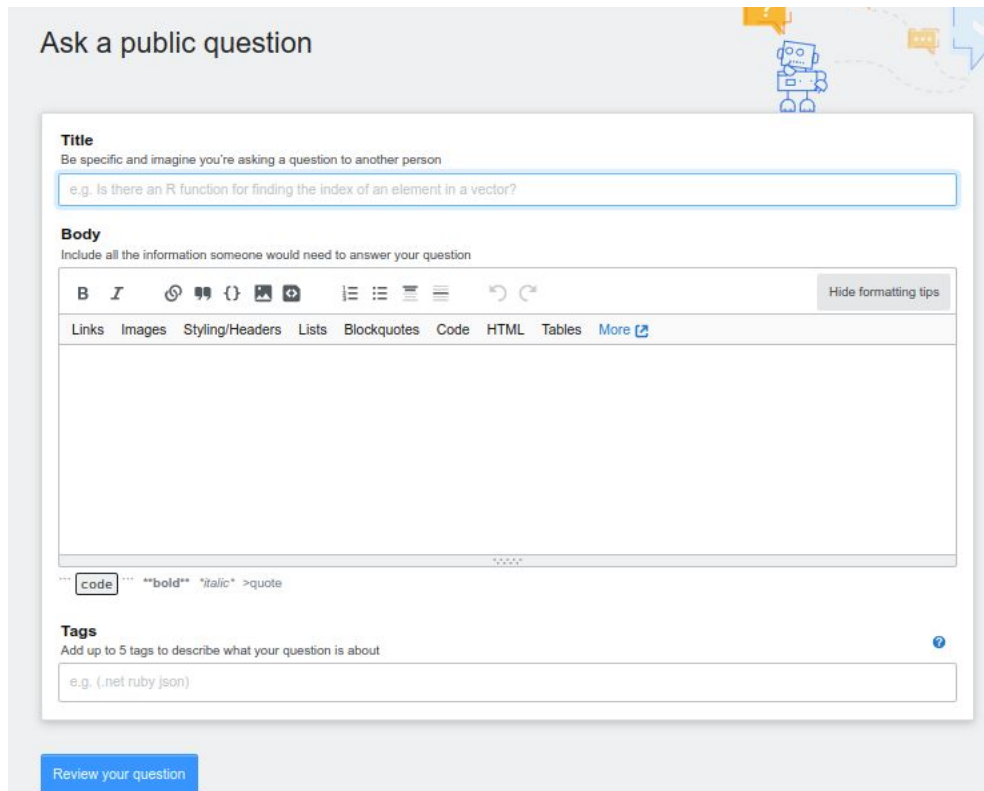
# Sommaire

<b>Sommaire</b>	<b>2</b>
<b>Contexte</b>	<b>2</b>
<b>Données</b>	<b>3</b>
<b>Preprocessing</b>	<b>5</b>
<b>Analyse exploratoire</b>	<b>6</b>
Title	6
Body	7
Tags	8
<b>Feature engineering</b>	<b>10</b>
Dictionnaire	10
Title	10
Body	10
Représentation numérique du texte	11
<b>Modèles supervisés</b>	<b>11</b>
Évaluations des modèles.	12
Performances	13
Gridsearch	13
Résultats	13
<b>Model non supervisés</b>	<b>14</b>
<b>API</b>	<b>15</b>
<b>Conclusion</b>	<b>16</b>

## Contexte

Stack Overflow est un site web proposant des questions et réponses sur un large choix de thèmes concernant la programmation informatique. Il a été lancé le 15 septembre 2008 par Jeff Atwood et Joël Spolsky. En août 2015, Stack Overflow revendique plus de 10 000 000 questions.

L'utilisation de "Tags" ("étiquettes") y tient une place importante. Il est en effet possible de soumettre plusieurs tags à la question posée.



The image shows a web form titled "Ask a public question". It has three main sections: "Title", "Body", and "Tags".

- Title:** A text input field with the instruction "Be specific and imagine you're asking a question to another person". The example text is "e.g. Is there an R function for finding the index of an element in a vector?".
- Body:** A rich text editor with the instruction "Include all the information someone would need to answer your question". It features a toolbar with icons for bold, italic, link, image, code, list, blockquote, and table. Below the toolbar are tabs for "Links", "Images", "Styling/Headers", "Lists", "Blockquotes", "Code", "HTML", "Tables", and "More". A "Hide formatting tips" button is also present.
- Tags:** A text input field with the instruction "Add up to 5 tags to describe what your question is about". The example text is "e.g. (.net ruby json)".

At the bottom of the form is a blue button labeled "Review your question".

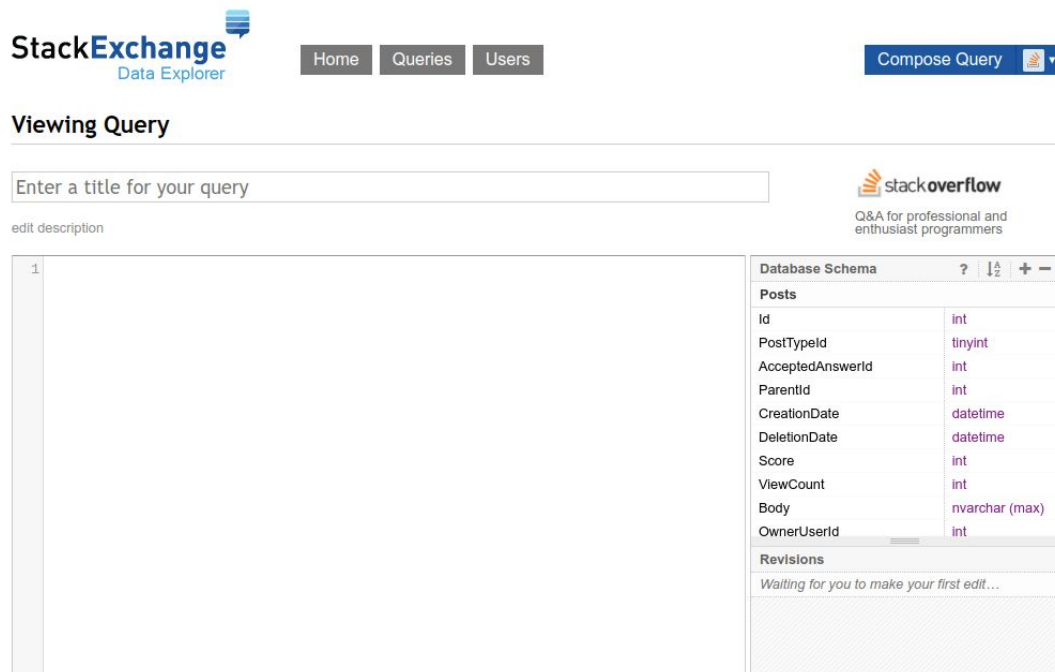
Cette pratique permet tout d'abord d'optimiser la réponse des membres de la communauté. Mais également de retrouver facilement les questions lors de recherches.

Cela ne pose pas de problème pour les utilisateurs expérimentés. Cependant, pour les nouveaux utilisateurs, il serait opportun de suggérer quelques tags relatifs à la question posée.

Pour répondre à cette problématique la solution pertinente est le traitement automatique du langage naturel (abr. TALN), ou traitement automatique de la langue naturelle, ou encore natural language processing (abr. NLP). Un domaine multidisciplinaire impliquant la création d'outils de traitement de la langue naturelle pour diverses applications.

## 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.



Cet outil "[stackexchange explorer](#)" permet de formuler des requêtes Structured Query Language (SQL) sur 29 tables. Le dispositif fonctionne cependant avec une limitation à 50 000 sorties par requête.

Ainsi plusieurs requêtes successives ont été effectuées à partir de la date du 01/01/2019, puis de la dernière date retournée par la requête précédente.

Nous avons sélectionné uniquement les publications de type questions (PostTypeId = 1) et présentant un vote positif de la communauté ( score >2).

Enfin, nous avons sélectionné les données relatives au titre (title) et au corps (Body) des questions.

Ci-dessous la première requête SLQ :

```
SELECT id, PostTypeId, Title, Body, Tags,CreationDate, Score
FROM Posts
WHERE CreationDate >= '2019/01/01'
AND PostTypeId = 1
AND Score > 2
ORDER BY CreationDate
```

# Preprocessing

La première étape du traitement des données texte est celle du nettoyage ou preprocessing. Celle-ci a pour objectif de les rendre utilisables ultérieurement par différents algorithmes.

Plusieurs possibilités :

- La tokenization, qui désigne le découpage en mots des différents documents qui constituent le corpus.
- La normalisation et la construction du dictionnaire qui permet de ne pas prendre en compte des détails importants au niveau local (ponctuation, majuscules)
- La suppression de ce qu'on appelle en anglais les *stopwords*. Ce sont les mots très courants dans la langue étudiée ("et", "à", "le"... en français) qui n'apportent pas de valeur informative.
- Le processus de « lemmatisation » consiste à représenter les mots sous leur forme canonique.

Dans notre cas, le prétraitement du texte comprend les étapes suivantes :

1. Retrait des balises HTML ou code brut.
2. Récupération uniquement du texte et suppression des nombres.
3. Tokenisation
4. Conversion des lettres en minuscules.
5. Lemmatisation
6. Retrait des stopwords

Exemple avant nettoyage et normalisation:

```
<p>This is actually for the thread on <a href="https://stackoverflow.com/questions/49162667/unknown-error-call-function-result-missing-value-for-selenium-send-keys-even">unknown error: call function result missing &#39;value&#39; for Selenium Send Keys even after chromedriver upgrade</a> but I guess my reputation isn't high enough to participate(lame).</p>
```

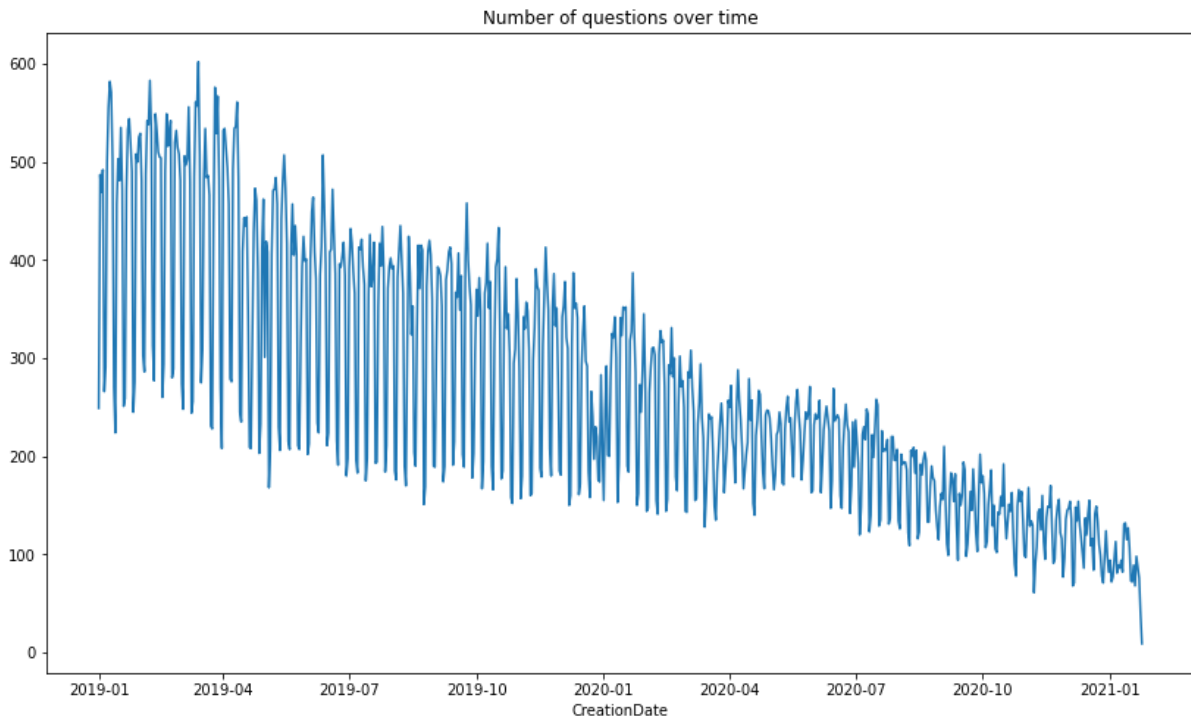
Exemple après nettoyage et normalisation:

```
'actually', 'thread', 'unknown', 'error', 'call', 'function', 'result', 'missing', 'value', 'selenium', 'send', 'key', 'even', 'chromedriver', 'upgrade', 'guess', 'reputation', 'high', 'enough', 'participate', 'lame'
```

A la fin de cette étape, nos variables Title et Body sont encore sous format de texte.

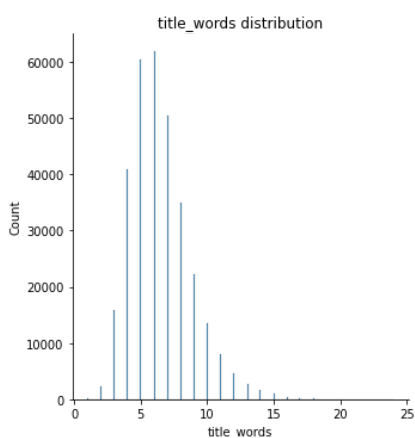
## Analyse exploratoire

Les données finales concernent 202 861 questions du 01/01/2019 au 24/01/2021.



Le nombre de questions disponibles s'explique par le filtre de score choisi lors de la récupération des questions. Une question récente a bénéficié de moins de temps pour recevoir des votes de la communauté, en comparaison avec les questions plus anciennes.

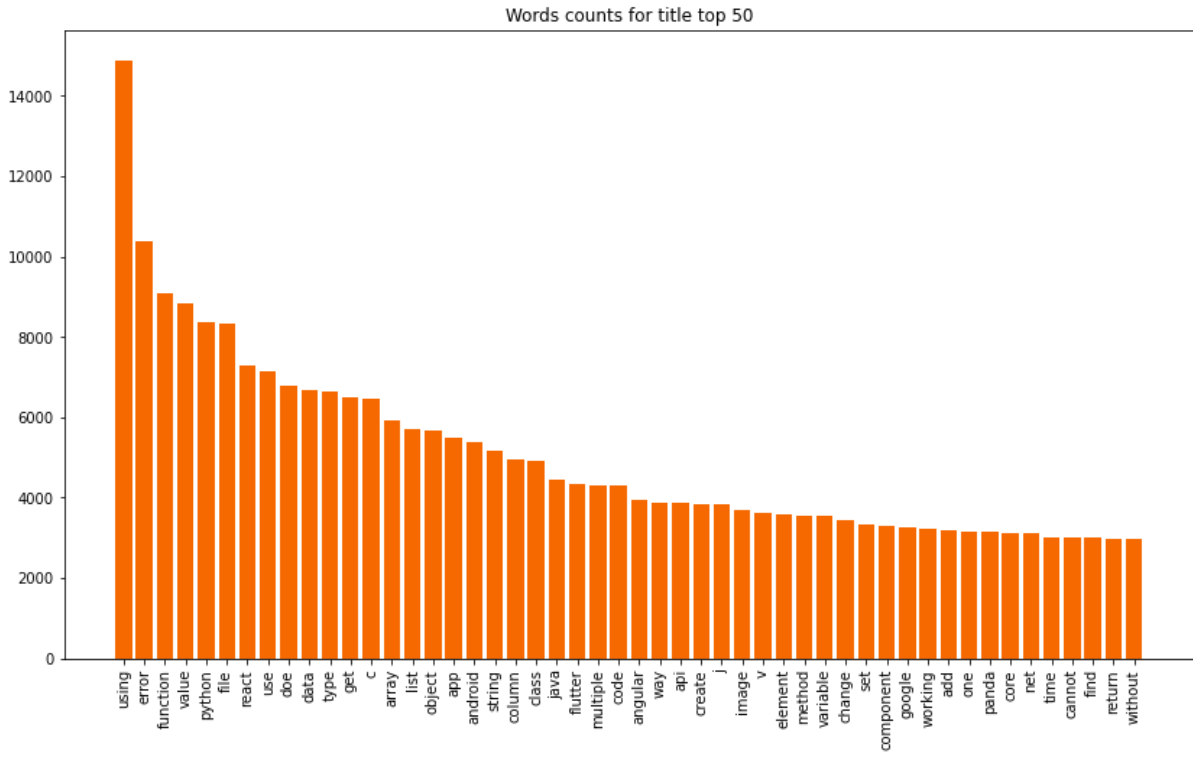
## Title



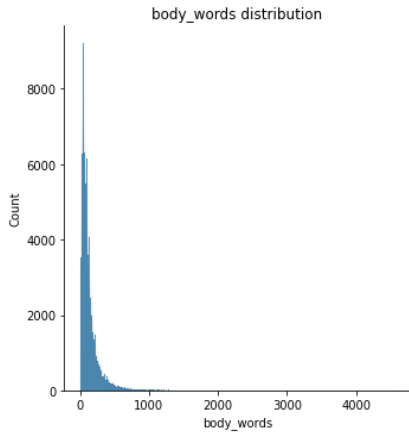
Il y a  $7 \pm 2$  mots par titre.

Il y a 4 114 mots uniques utilisés.

Les 5 mots les plus utilisés sont : using, python, error, fonction et value.



## Body

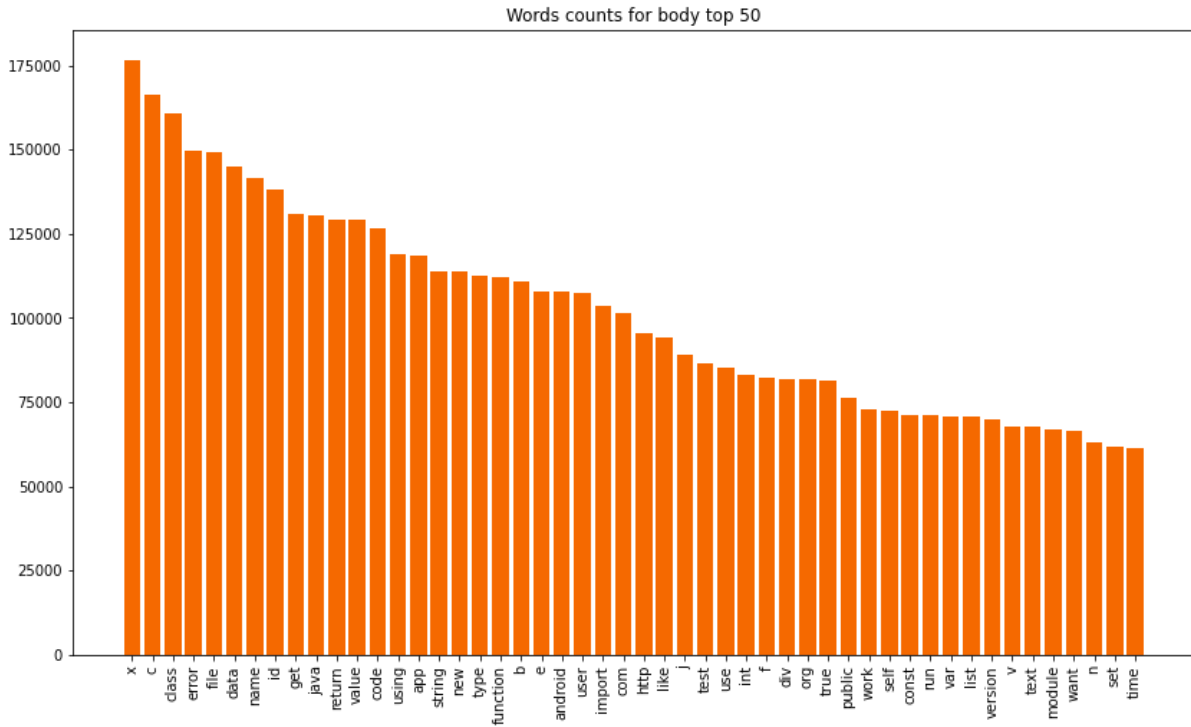


Il y a  $146 \pm 195$  mots par corps.

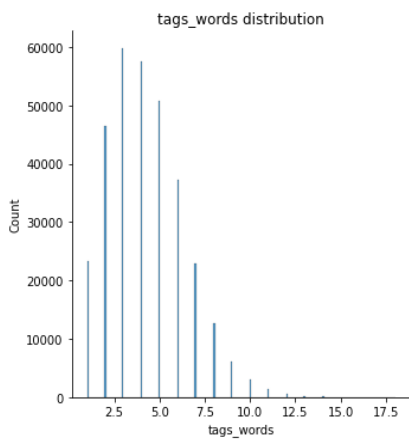
Il y a 541 371 mots uniques utilisés.

Les 5 mots les plus utilisés sont : x, class, c, error et file.





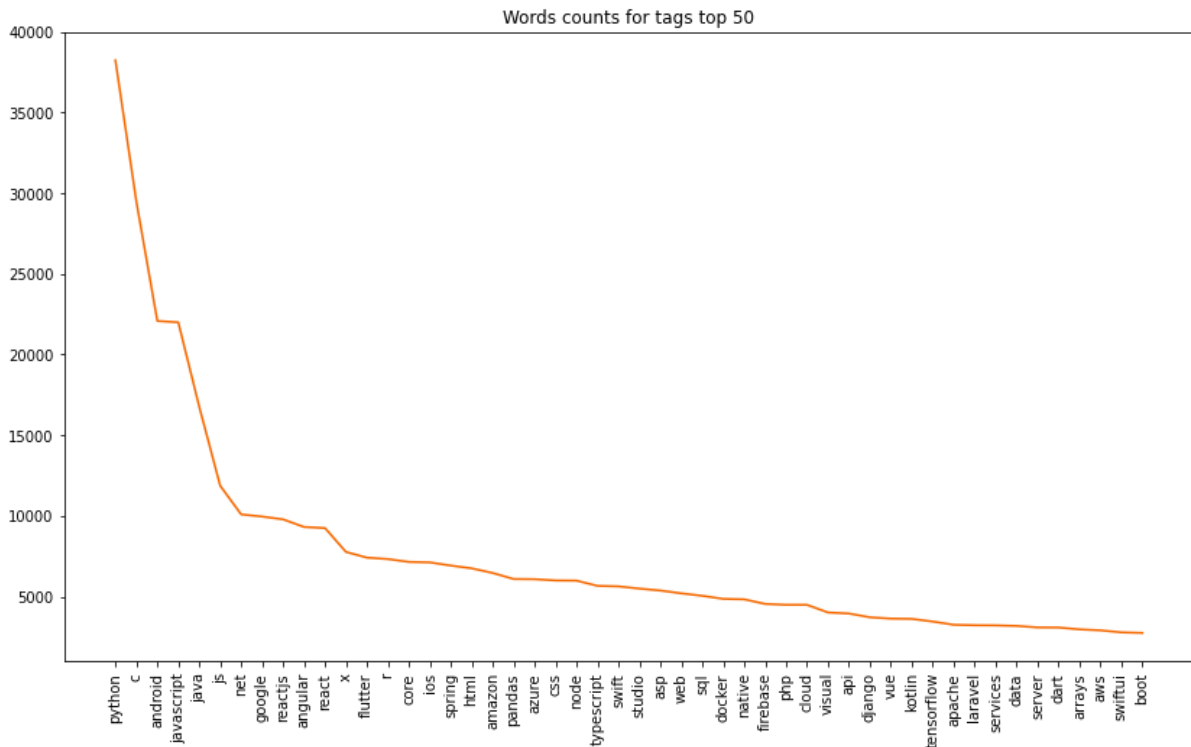
## Tags



Il y a  $4 \pm 2$  tags par questions.

Il y a 14 966 mots uniques utilisés.

Les 5 mots les plus utilisés sont : python, c , android, javascript et java.



Il est intéressant de constater que la diminution de la fréquence d'utilisation est progressive pour le titre et le corps des questions. A l'inverse, celle-ci est brutale pour les tags. Il semblerait que quelques mots concentrent une proportion importante des utilisations.

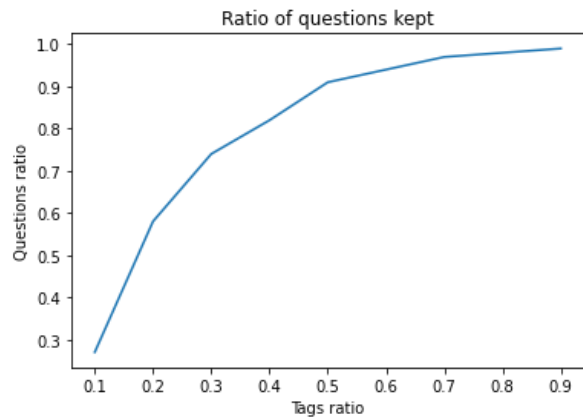
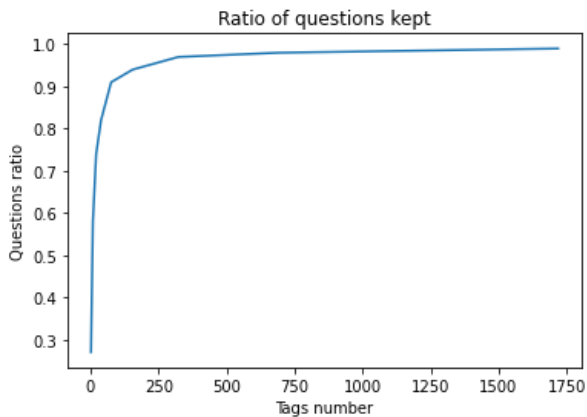
## Feature engineering

### Modèles supervisés

L'apprentissage supervisé est une tâche d'apprentissage automatique consistant à apprendre une fonction de prédiction à partir d'exemples annotés, au contraire de l'apprentissage non supervisé. Dans notre cas, la représentation numérique du texte (bag of words, tf-idf) servira à créer un modèle prédisant les tags.

### Target

Les 2 graphiques ci-dessous explorent le ratio de questions conservées (tags >= 1) selon la quantité de mots conservés dans le dictionnaire des Tags. A partir de 76 tags, nous sommes en mesure de couvrir 90% des questions.



Classiquement, un modèle de classification permet de prédire une classe, ou une étiquette. Cependant, dans le cas présent, il doit être possible de prédire plusieurs étiquettes. C'est donc une situation de classification multi-étiquettes.

La classification multi-étiquettes implique la prédiction de zéro ou plusieurs étiquettes de classe.

Contrairement aux tâches de classification normales où les étiquettes de classe sont mutuellement exclusives, la classification multi-étiquettes nécessite des algorithmes d'apprentissage automatique spécialisés qui prennent en charge la prédiction de plusieurs classes ou «étiquettes» mutuellement non exclusives.

Afin de résoudre cette problématique nous avons choisi une stratégie multiclasse One-vs-the-Rest (OvR). Aussi connue sous le nom de one-vs-all, cette stratégie consiste à ajuster un classificateur par classe.

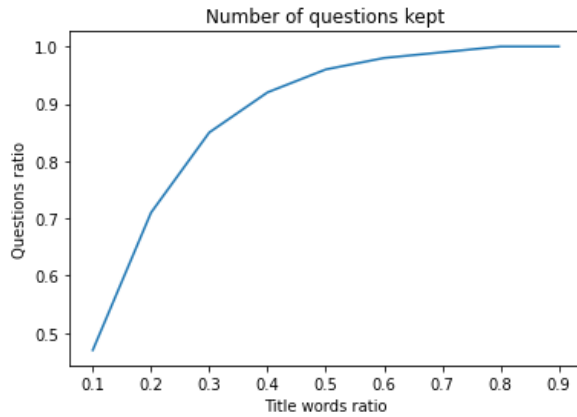
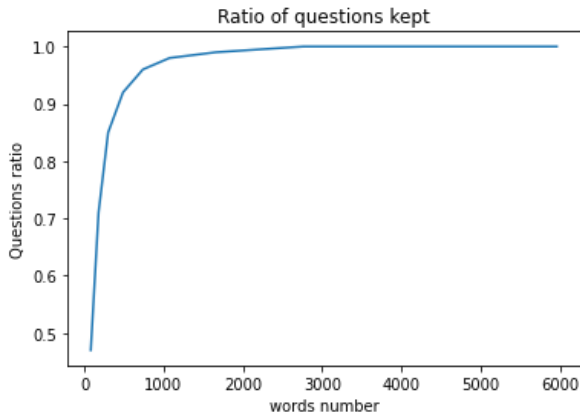
## Features

Les objectifs de cette étape sont multiples. Tout d'abord, apprécier la quantité de mots nécessaire pour, à la fois, extraire des informations pertinentes en conservant un maximum de questions non nulles, tout en prenant en compte les capacités machine. Ensuite, représenter notre texte sous forme numérique. Enfin, optimiser celles-ci en prévision de leurs utilisations ultérieures par des modèles de classification.

## Dictionnaire

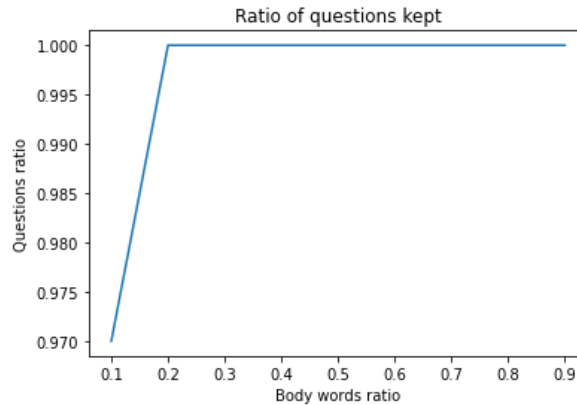
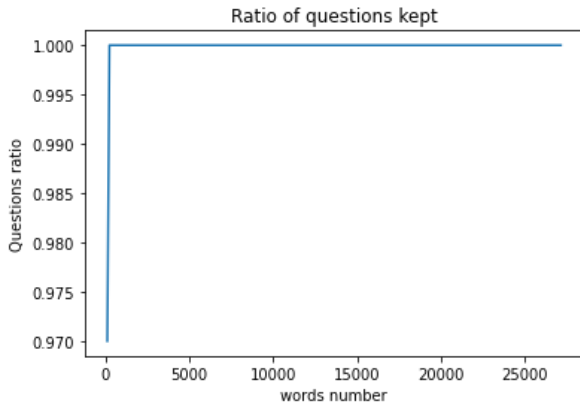
- **Title**

Les 2 graphiques ci-dessous explorent le ratio de questions conservées (tags  $\geq 1$ ) selon la quantité de mots conservés dans le dictionnaire des Titres. A partir de 750 mots environ, nous sommes en mesure de conserver 96% des questions.



- **Body**

Les 2 graphiques ci-dessous explorent le ratio de questions conservées (tags  $\geq 1$ ) selon la quantité de mots conservés dans le dictionnaire du corps. A partir de 250 mots environ, nous sommes en mesure de conserver 97% des questions.



Conclusion : Le choix se porte donc sur 76 tags, 750 mots pour le titre et 250 mots pour le corps.

## Représentation numérique du texte

Nous avons choisi d'utiliser deux représentations numériques du texte :

1. Représentation bag-of-words classique:

Le corpus peut être décrit au moyen d'un dictionnaire (de « mots »). Un document (Title et Body pour nous) est donc représenté par un vecteur de la même taille que le dictionnaire, dont la composante  $i$  indique le nombre d'occurrences du  $i$ -ème mot du dictionnaire dans le document.

2. Représentation tf-idf :

Le TF-IDF (de l'anglais *term frequency-inverse document frequency*) est une méthode de pondération. Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une

collection ou un corpus. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le document. Il varie également en fonction de la fréquence du mot dans le corpus.

## Modèles

Le classifieur retournant un résultat binaire pour chaque classe, notre attention s'est portée sur 2 modèles simples permettant de traiter les 1 000 features en entrée : multinomial Naive Bayes classifieur, et Linear Support Vector Classifier.

Multinomial Naïve Bayes utilise la fréquence des termes, c'est-à-dire le nombre de fois qu'un terme donné apparaît dans un document. Après normalisation, la fréquence du terme peut être utilisée pour calculer des estimations de maximum de vraisemblance basées sur les données d'apprentissage pour estimer la probabilité conditionnelle.

Issu de la famille des Support Vector Machine (SVM), l'idée du Linear Support Vector Classifier est simple: l'algorithme crée une ligne ou un hyperplan qui sépare les données en classes. Ce modèle est recommandé pour un grand nombre d'échantillons.

Le titre et le corps seront vectorisés et transformés séparément, puis empilés. Les deux représentations seront utilisées par des modèles supervisés.

## Évaluations des modèles.

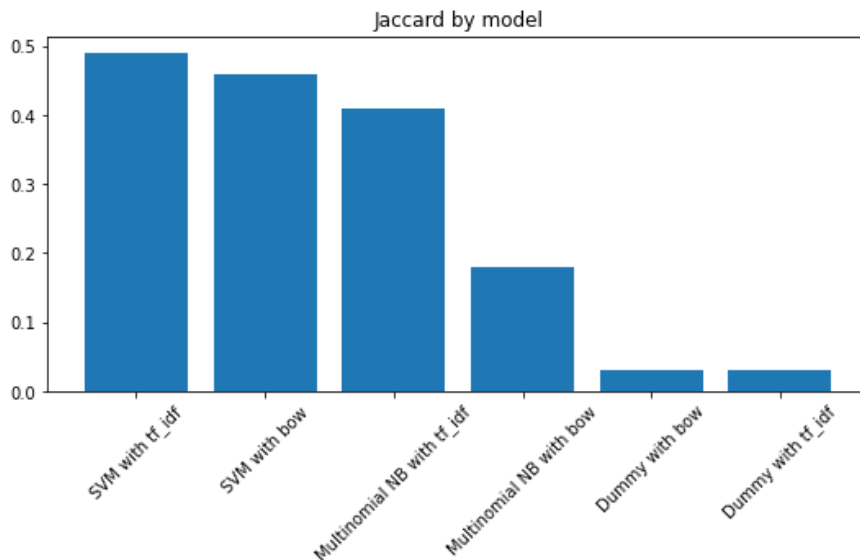
Dans le cadre d'une classification multi-étiquettes, nous utiliserons l'indice de Jaccard. L'indice de Jaccard est le rapport entre le cardinal (la taille) de l'intersection des ensembles considérés et le cardinal de l'union des ensembles.

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Il permet d'évaluer la similarité entre les ensembles réels et prédits.

## Performances

Les deux modèles ont été tout d'abord utilisés avec les paramètres par défaut.



Le SVM avec tf-idf en entrée est le plus performant avec un score de Jaccard de 0.49.

## Gridsearch

Après avoir identifié les modèles les plus performants avec les paramètres par défaut, nous avons effectué une recherche sur grille pour le SVM. Les paramètres d'optimisation ont été la régularisation (L1, L2) et le C (0.1,1,10). L'optimisation a été faite avec une validation croisée avec 5 fold et shuffle. Le score à optimiser a été le Jaccard. Les performances sur les jeux d'entraînement et de test ont été comparées pour apprécier un éventuel sur-apprentissage.

## Résultats

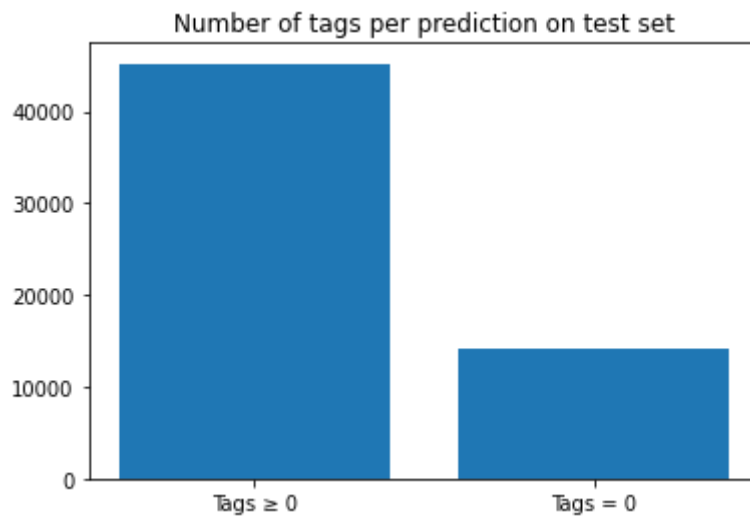
```
Gridsearch with Linear SVC and tf-idf
Gridsearch best params: {'estimator__C': 10, 'estimator__penalty': 'l2'}
Gridsearch Jaccard score on test split: 0.49
Gridsearch Jaccard score on train split: 0.51
```

La recherche sur grille ne permet pas d'amélioration significative de la performance du modèle. Toutefois, il est possible de constater qu'il n'y a pas de sur-apprentissage.

## Nombre de prédictions sans tags

Un défaut de la classification multi-étiquettes est qu'il est possible qu'aucune étiquette ne soit prédit. Pour apprécier cela nous avons utilisé les prédictions sur le

jeu de test et comptabilisé le nombre de prédictions avec un tag et celles-sans tags. Le graphique ci-dessous nous indique clairement que le modèle généralement prédit au moins un tag.



Ce constat ne permettant pas une utilisation systématique du modèle supervisé, un modèle non supervisé à été testé.

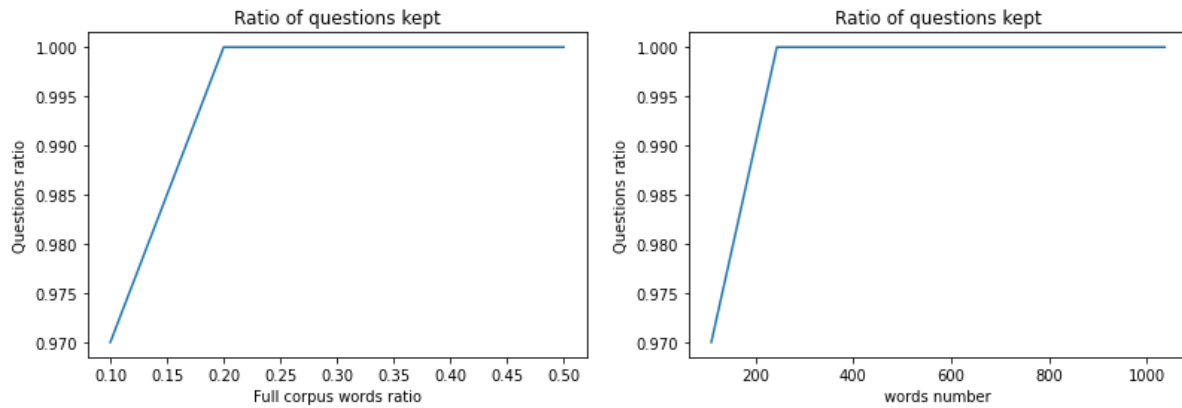
## Modèle non supervisé

L'apprentissage non supervisé désigne la situation d'apprentissage automatique où les données ne sont pas étiquetées. Il s'agit donc de découvrir les structures sous-jacentes à ces données non étiquetées.

Notre choix s'est porté sur le Latent Dirichlet Allocation (LDA). Le 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.

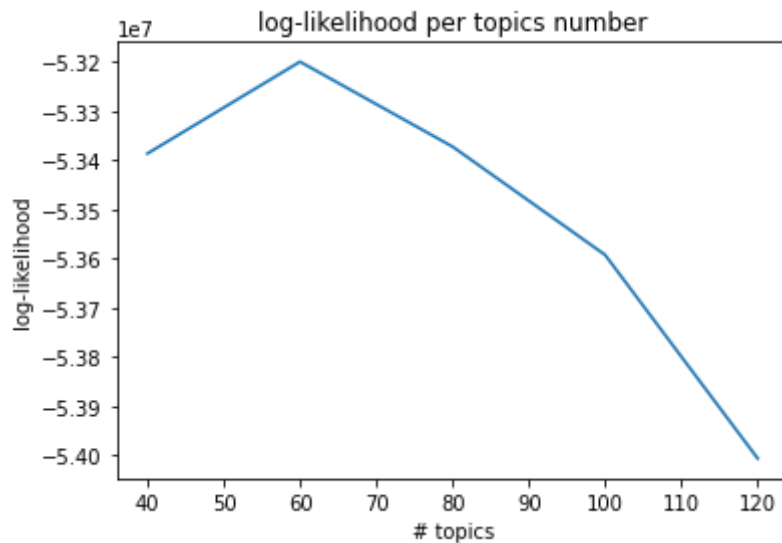
## Features

Comme précédemment pour les modèles supervisés, l'objectif est d'optimiser le nombre de features utilisées sans perdre de questions. Les 2 graphiques ci-dessous semblent indiquer que nous pouvons conserver l'ensemble des observations avec 250 mots dans le dictionnaire.



## Optimisation du modèle

Le nombre de sujets a été optimisé par la score de maximum de vraisemblance à partir d'une représentation en bag of words de 250 termes.



Le nombre optimal de topics semble être autour de 60. Une partie des sujets est disponible ci-dessous.



```
Topic 0:  
info task debug level  
Topic 1:  
size input output max  
Topic 2:  
true false state prop  
Topic 3:  
service net core token  
Topic 4:  
aws location resource permission  
Topic 5:  
end next vector double  
Topic 6:  
std template foo bar  
Topic 7:  
server client val stream  
Topic 8:  
option na select cell  
Topic 9:  
view io firebase fun  
Topic 10:  
map child flutter context  
Topic 11:  
db database sql spark  
Topic 12:  
java org internal gradle
```

---

Les mots clés associés aux différents sujets semblent cohérents. Cependant ils semblent moins pertinents que les tags complétés par les utilisateurs.

## API

Un point d'entrée API a pour objectif de permettre l'accès aux modèles créés précédemment par des utilisateurs.

Celui-ci a été conçu à partir de la librairie de tableau de bord Dash dans un notebook Jupyter. La démarche est de pouvoir proposer des suggestions à partir du modèle supervisé, SVM, tout d'abord. Dans le cas où celui-ci ne proposerait aucun tag, le modèle non supervisé, LDA, serait utilisé.

L'API a été déployé sur Heroku, depuis Github.

Celle-ci est disponible à l'adresse suivante :  
<https://stack-overflow-auto-tag.herokuapp.com/>

## Perspectives

- Nettoyage de termes spécifiques contexte ( ex : X, y) qui pourrait représenter un potentiel bruit.
- Accès direct Stack Exchange API pour suggestions plus fines des tendances récentes.
- Utilisation des réseaux de neurones qui ont démontré leurs performances dans le NLP.
- Exploration word embedding (word2vec).
- Augmenter le nombre de tags conservés.

## Conclusion

Ce projet avait pour objectif de créer un outil de suggestion automatique de tags pour le site Stack Overflow. Les données ont été récupérées depuis leur plateforme d'accès à une base de données mise à disposition.

Les données ont été nettoyées et pré-traitées. Deux représentations du texte ont été utilisées (Bag Of Words, tf-idf).

Dans le cadre d'un apprentissage supervisé tout d'abord, deux modèles de classification, Linear SVC et Multinomial Naïve Bayes, ont été comparés.

Une recherche sur grille a été effectuée pour le modèle le plus performant (Linear SVC). Celui-ci obtient un score de Jaccard de 0.51 et ne présente pas de sur-apprentissage.

Dans un second temps, un modèle d'apprentissage non supervisé a été utilisé. Le Latent Dirichlet Allocation obtient un nombre de topics optimal de 60. Les mots clés associés à ces topics semblent cohérents.

Enfin, un point d'entrée API a été développé à partir de la librairie Dash. Ce point a été déployé depuis Github sur la plateforme Heroku.