

**Catégoriser automatiquement des questions**

Elise Andro – 06/08/2021

Table des matières

[Introduction 3](#_Toc78971029)

[Qui sont les clients ? 3](#_Toc78971030)

[Quelle est leur problématique ? 3](#_Toc78971031)

[Quelles données avons-nous à disposition ? 3](#_Toc78971032)

[Traitement des données 3](#_Toc78971033)

[Nettoyage 4](#_Toc78971034)

[Analyse Descriptive 4](#_Toc78971035)

[Analyse des tags 5](#_Toc78971036)

[Modèles non-supervisés 6](#_Toc78971037)

[Modèles supervisés 9](#_Toc78971038)

[Conclusion 12](#_Toc78971039)

# Introduction

## Qui sont les clients ?



Stack Overflow est un site web proposant des questions et réponses sur un large choix de thèmes concernant la programmation informatique. Il fait partie du réseau de sites Stack Exchange.

## Quelle est leur problématique ?

L’objectif de ce projet est de simplifier l’utilisation du site StackOverFlow et sa recherche par tag. Les utilisateurs entrent des tags pour rechercher une question et ses réponses.

Développer un système de suggestion de tag pour le site. Le système assignera automatiquement plusieurs tags pertinents à une question.

## Quelles données sont à disposition ?

Données : requête SQL depuis le site <https://data.stackexchange.com/stackoverflow/query/new>

Une image contenant texte

Description générée automatiquement

# Traitement des données

## Nettoyage

Les étapes de nettoyage effectués sont les suivantes :

* Nettoyage des caractères spéciaux (html)
* Fusion du titre et du contenu
* Tokenization
* Stop words
* Bigram et trigram
* Lemmatization

Résultats du nettoyage :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

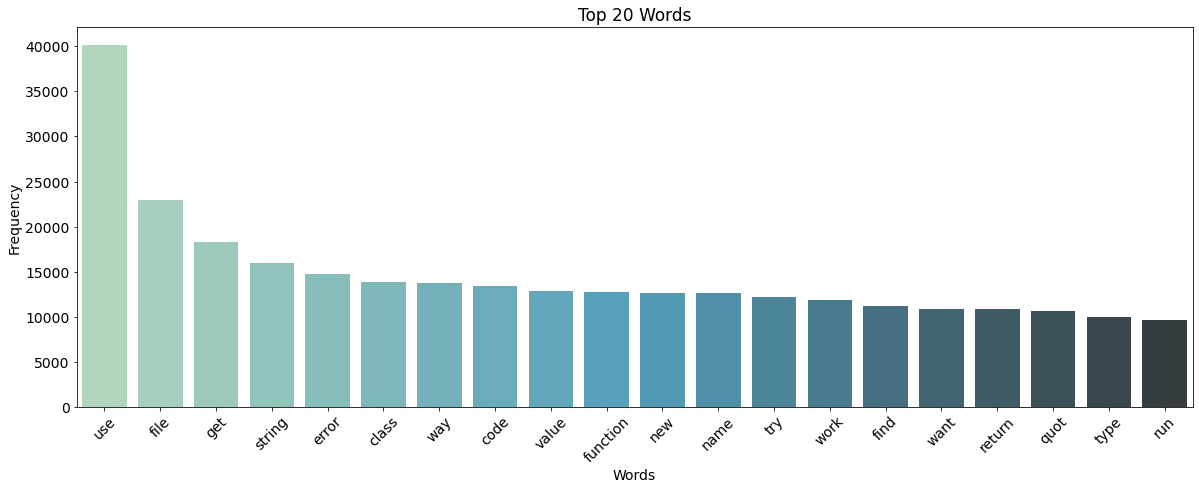
Description générée automatiquement

## 

## Analyse Descriptive

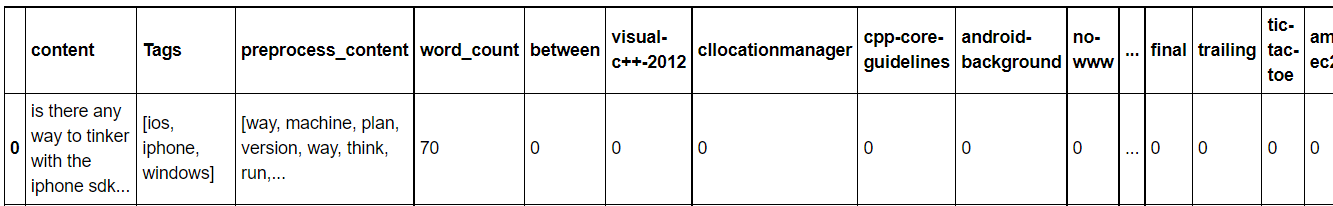
Analyse et exploration du contenu (titre+texte) pour prendre en main le corpus de document et avoir une idée de la diversité des données.



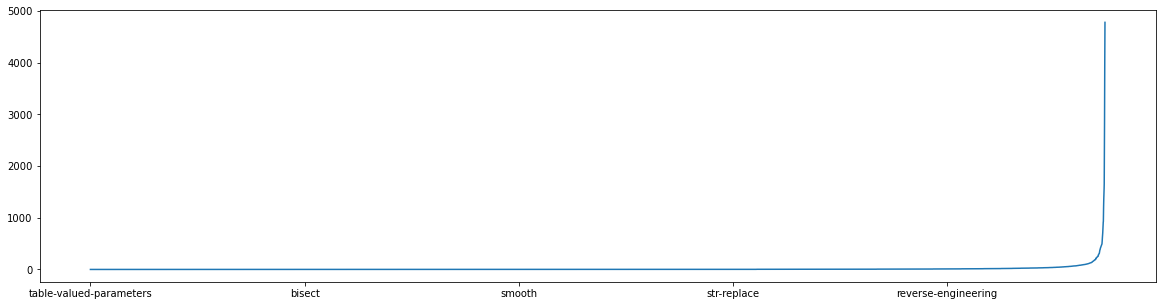


## Analyse des tags

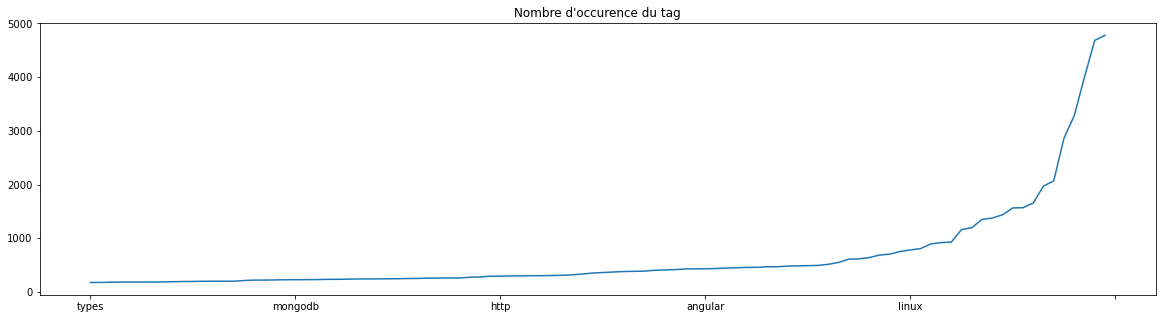
Pour analyser les tags, on crée un dataframe contenant une colonne par tags :



Ce dataframe permet d’effectuer l’analyse du nombre de tags et fréquence des tags. On peut constater que beaucoup de tags sont très peu utilisés.

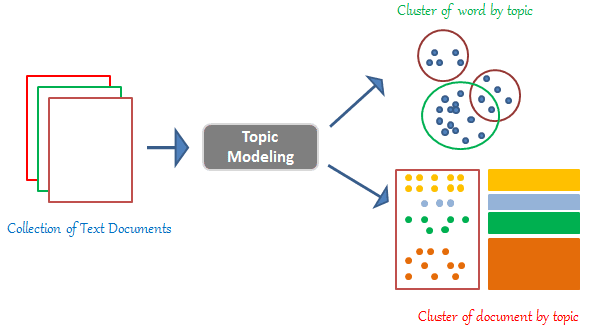


Pour faciliter l’analyse et les modèles, on ne conserve dans un premier temps que les 100 tags les plus utilisés :



# Modèles non-supervisés

Topic Modeling, le principe :



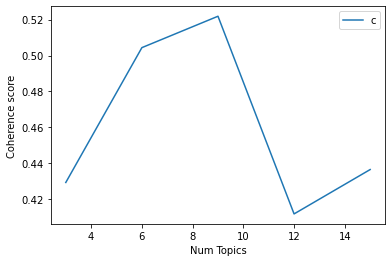
**Latent Dirichlet Allocation**

Il s’agit d’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. Le modèle LDA suppose que chaque document est un mélange d’un petit nombre de sujets ou thèmes, et que la génération de chaque occurrence d’un mot est attribuable (probabilité) à l’un des thèmes du document.

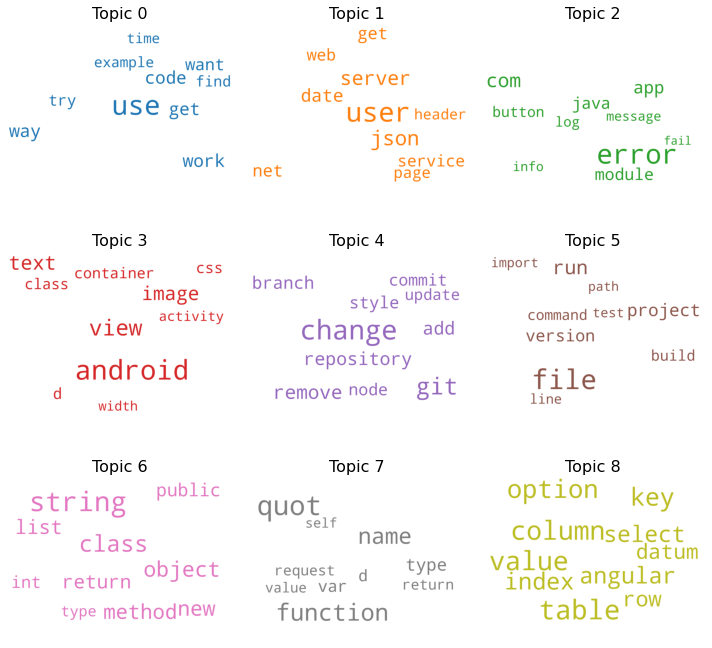
Application du modèle LDA aux données

En amont, il est nécessaire de créer un dictionnaire et un corpus pour entrainer le modèle.

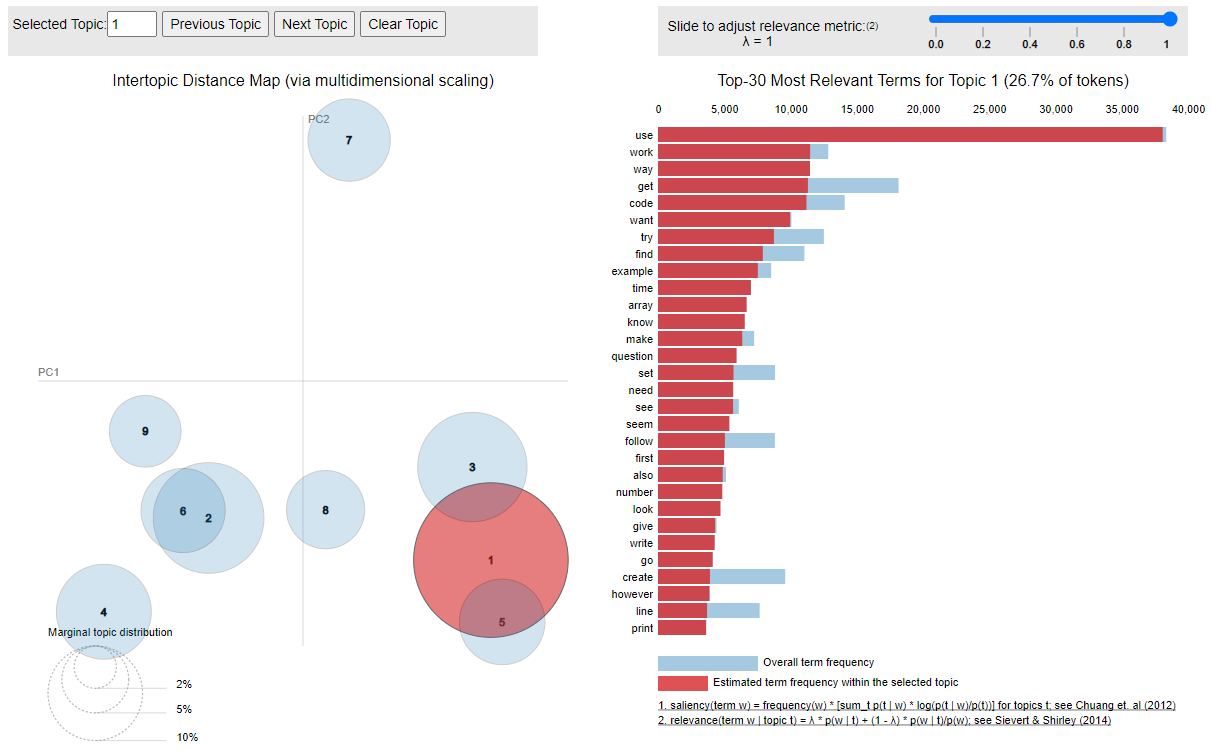
On entraine ensuite plusieurs modèles pour choisir le nombre optimal de topics en utilisant le score de cohérence comme métrique. Ici l’optimal est **9 topics.**



On visualise enfin les **top words** pour chaque topics :



Le package pyLDAvis permet de visualiser les résultats du modèle LDA sur les 2 axes principaux d’une ACP. On peut distinguer les distances entre les topics, leur importance dans le corpus et le top30 des mots pour chaque topics.

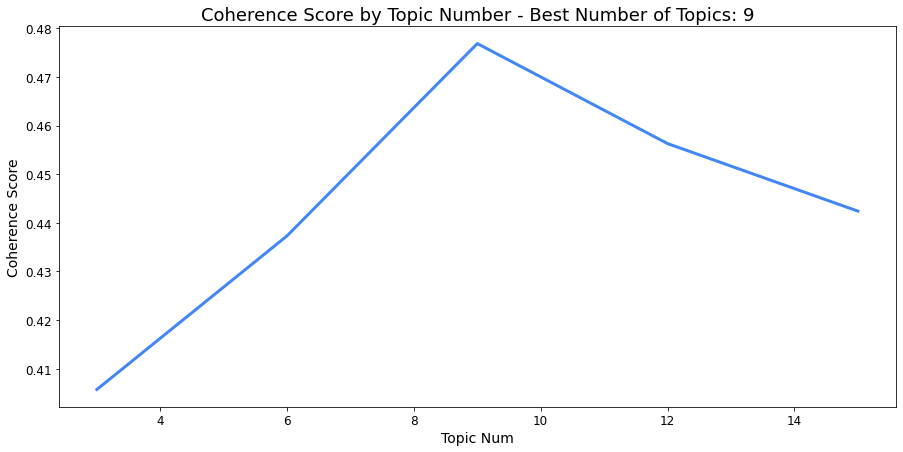


**Non-Negative Matrix Factorization**

La NMF est donc une technique de réduction de dimension adaptée aux matrices creuses contenant des données positives. Le modèle classe les documents sur la base d’un vocabulaire de mots. Chaque document se décompose (coefficients positifs) sur des “facteurs” ou thèmes, eux-mêmes définis chacun par un sous-ensemble de ces mots.

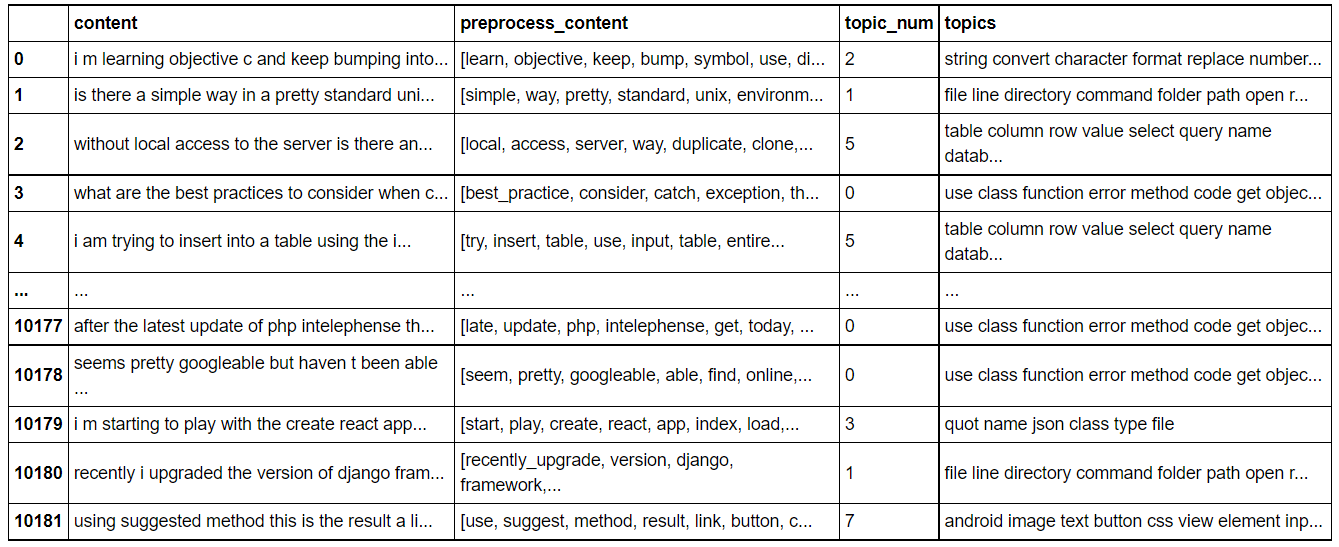
Application du modèle NMF aux données

Comme pour le LDA, on entraine plusieurs modèles pour choisir le nombre optimal de topics en mesurant le score de cohérence. L’idéal est également de **9 topics**.



On utilise le modèle optimal pour effectuer des prédictions. Pour cela, on entraine le modèle sur une part du dataset et on effectue des prédictions sur l’autre part.

On obtient pour chaque document du test set un topic ainsi qu’une liste de 10 mots (les top 10 du topics) qui peuvent être une prédiction de tags.



# Modèles supervisés

**Pre-process : TF-IDF**

Tf-Idf signifie fréquence de terme-fréquence de document inverse, et au lieu de calculer le nombre de mots dans chaque document de l'ensemble de données, il calcule le nombre normalisé où chaque nombre de mots est divisé par le nombre de documents dans lesquels ce mot apparaît.

**Test sur différents algorithmes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithme** | **Principe** | **Illustration** | **Résultats** |
| Baseline | Classifier qui prédit la classe la plus fréquente.  Permet de comparer les résultats avec les autres modèles |  | Accuracy = 0.12  F1\_score = 0.02 |
| Naive Bayes | Classifier qui suppose l’indépendance des variables entre elles |  | Accuracy = 0.45  F1\_score = 0.42 |
| Logistic Regression | Classifier permettant d’assigner une probabilité d’appartenance à une classe (0/1) grâce à la fonction sigmoid | Logistic Regression with Python. Logistic regression was once the most… |  by ODSC - Open Data Science | Medium | Accuracy = 0.47  F1\_score = 0.46 |
| Support Vector Machine | Classifier recherchant à maximiser les marges entre les échantillons les plus proches (vecteurs supports) | Support Vector Machine — Simply Explained | by Lilly Chen | Towards Data  Science | Accuracy = 0.46  F1\_score = 0.43 |
| Random Forest | La forêt aléatoire se repose sur plusieurs modèles dits « faibles » d’arbres de décision puis les combinent en prenant la solution de la majorité. | Random Forests®, Explained - KDnuggets | Accuracy = 0.456  F1\_score = 0.46 |

# Conclusion

Le modèle de Régression Logistic apporte les meilleurs résultats pour développer une fonctionnalité de génération automatique de tags.