

From Bag of Words



Category	Percentage
Software	30%
Data Science	60%
Business	5%
Self Help	0%
Career	5%
Humor	0%
Travel	0%

OPENCLASSROOMS

Table des Matières

1. Introduction.....	3
2. Pré-traitement des données.....	4
3. Analyse exploratoire.....	6
4. Vectorisation.....	7
5. Modèles non-supervisés.....	8
6. Modèles supervisés.....	9

1. Introduction

Dans cette étude j'ai appliqué le traitement automatique du langage naturel, 'natural language processing' (NLP) en anglais, au texte html de 'Stackoverflow', le plus grand site de questions-réponses sur Internet consacré à la programmation et au développement. Le but du projet était de développer un application pour classer automatiquement chaque question posé sur le site et l'assigner des mot clés (tags). Pour cette tâche j'ai traité un jeux de données exporté par le site¹

Sur ce site j'ai téléchargé un tableau des questions et respectifs mots clé avec une requête SQL. J'ai sélectionné les 50000 questions plus vues, favorites par les utilisateurs (Favoritecount) et pertinentes (Score), avec plus d'une une réponse. Le code est le suivant:

```
SELECT
    TOP 50000 ViewCount,
    CreationDate,
    Title,
    Body,
    Tags,
    Score,
    CommentCount,
    AnswerCount,
    FavoriteCount

FROM Posts
WHERE
    FavoriteCount > 10
    AND AnswerCount > 1
    AND Score > 100
ORDER BY Score DESC
```

J'ai obtenu un fichier csv de 50MB, prêt pour être importé avec Python comme Pandas dataframe (Tableau 1).

```
[4]: print(df_raw.shape)
df_raw.head()
```

(37175, 9)

	ViewCount	CreationDate	Title	Body	Tags	Score	CommentCount	AnswerCount	FavoriteCount
0	1483128	2012-06-27 13:51:36	Why is processing a sorted array faster than p...	<p>Here is a piece of C++ code that shows some...	<java><c++><performance><optimization><branch-...	24320	22	26	10983
1	8547399	2009-05-29 18:09:14	How do I undo the most recent local commits in...	<p>I accidentally committed the wrong files to...	<git><version-control><git-commit><undo><pre-c...	20895	13	83	6776
2	8115583	2010-01-05 01:12:15	How do I delete a Git branch locally and remot...	<p>I want to delete a branch both locally and ...	<git><version-control><git-branch><git-push><q...	16826	6	40	5357
3	2782271	2008-11-15 09:51:09	What is the difference between 'git pull' and ...	<p>What are the differences between <code>git ...	<git><version-control><git-pull><git-fetch>	11833	9	35	2333
4	2783219	2009-01-25 15:25:19	What is the correct JSON content type?	<p>I've been messing around with <a href="http...	<json><http-headers><content-type>	10204	0	36	1446

Tableau 1. Données importées dans Pandas à partir du fichier obtenu par la requête SQL.

Ce document est structuré comme suit. Dans la section 2 on trouve le nettoyage et le pré-traitement des documents pour obtenir un corpus prêt à l'analyse et une liste des mots clé . La section 3 présente une analyse exploratoire des données. La section 4 traite la représentation vectorielle du texte. La modélisation non-supervisée est présentée dans la section 5 et la modélisation supervisée dans la section 6. La section 7 présente la choix du modèle pour le traitement automatique et les conclusions.

¹ 'Query Stack Overflow - Stack Exchange Data Explorer'
<<https://data.stackexchange.com/stackoverflow/query/new>> [accessed 5 June 2020].

2. Pré-traitement des données

L'objectif de cette étape est de standardiser le texte pour rendre son traitement facile par les algorithmes d'apprentissage automatique. Dans la colonne des mots clé, Tags, (Tableau 2) ça suffit d'enlever les parenthésés.

Title	Body	Tags
Why is processing a sorted array faster than p...	<p>Here is a piece of C++ code that shows some...	<java><c++><performance><optimization><branch-...
How do I undo the most recent local commits in...	<p>I accidentally committed the wrong files to...	<git><version-control><git-commit><undo><pre-c...
How do I delete a Git branch locally and remot...	<p>I want to delete a branch both locally and ...	<git><version-control><git-branch><git-push><g...
What is the difference between 'git pull' and ...	<p>What are the differences between <code>git ...	<git><version-control><git-pull><git-fetch>
What is the correct JSON content type?	<p>I've been messing around with <a href="http...	<json><http-headers><content-type>

Tableau 2. Les colonnes 'Body' et 'Tags' dans la forme originelle des fichiers html.

Par contre, le pré-traitement du corpus du texte (colonne 'Body') est plus complexe. Le texte de chaque question est dans la forme du langage html, par exemple dans la première ligne on lit:

```
<p>Here is a piece of C++ code that shows some very peculiar behavior. For some strange reason, sorting the data miraculously makes the code almost six times faster:</p>\n\n<pre class="lang-cpp prettyprint-override"><code>#include <algorithm>\n#include <ctime>\n#include <iostream>\nint main()\n{\n    // Generate data\n    const unsigned arraySize = 32768;\n    ,
```

Enlever les caractères html à la main serait très long, et pour ça j'ai utilisé le paquet "Beautiful soup". Le texte traité par "Beautiful soup" devient:

```
" Here is a piece of C++ code that shows some very peculiar behavior. For some strange reason, sorting the data miraculously makes the code almost six times faster:\n#include <algorithm>\n#include <ctime>\n#include <iostream>\n\nint main()\n{\n    // Generate data\n    const unsigned arraySize = 32768;\n    "
```

Mieux, mais pas encore prêt au traitement automatique. Après la concaténation avec les mots du titre, plusieurs étapes de nettoyage étaient nécessaires.

- traitement des contractions plus couramment utilisées en anglais, ex. changer "it's" en "it is".
- enlever les chiffres
- enlever les caractères spéciaux comme @, #, %, <
- enlever la ponctuation
- enlever les mots pas utiles (stopwords)

finalement j’ai séparé chaque phrase en mots ou “tokens” avec l’outil de segmentation “word-tokenizer” du paquet python “nltk” (natural language toolkit). À chaque on peut appliquer la racinisation (stemming) que donne une forme tronquée du mot, commune à toutes les variantes morphologiques

Ex : *cheval, chevaux, chevalier, chevalerie, chevaucher* ⇒ “cheva”

mais pour cet étude est plus intéressant de obtenir la forme canonique (le *lemme*) à partir du mot

- Pour un verbe: sa forme à l'infinitif
- Pour un nom, adjectif, article, ...: sa forme au masculin singulier

avec une fonction de lemmatisation. Pour ça le paquet “textblob” donne une meilleure performance que “nltk” .

Voici un exemple de question (post) avant:

move the most recent commit(s) to a new branch with git i would like to move the last several commits i have committed to master to a new branch and take master back to before those commits were made. unfortunately, my git-fu is not strong enough yet, any help? i.e. how can i go from this master a - b - c - d - e to this? newbranch c - d - e / master a - b

et après les dernières étapes de nettoyage (tokens, lemmatisation...)

```
[ 'recent', 'commit', 'new', 'branch', 'git', 'like', 'commits', 'commit',  
  'master', 'new', 'branch', 'master', 'commits', 'unfortunately', 'git-fu',  
  'strong', 'help', 'ie', 'master', 'c', 'd', 'e', 'newbranch', 'c', 'd', 'e',  
  'master' ]
```

3. Analyse exploratoire

Les mots plus fréquents dans le corpus nettoyé sont représentés dans le “nouage de mot” en Figure 1.



Figure 1. Les mots plus utilisés dans les corpus traité.

Ils sont des mots que on trouve suivant dans les post de Stackoverflow. Avec la fonction python “value_counts” appliquée à la colonne “tags” on retrouve les mots clés plus fréquentes (tableau 3).

tag	count	frequency (%)
javascript	3919	3.47796
python	3576	3.173561
java	3434	3.047541
c#	2742	2.433418
android	2450	2.17428

Tableau 3. Les 5 mots-clés (“tags”) plus fréquentes.

Avec un groupement des données textuelles par un mots clé spécifique on peut trouver les mots du corpus associés (exemple en Figure 2) et chercher des régularités distributionnelles que permettraient une prédiction du mot clé.

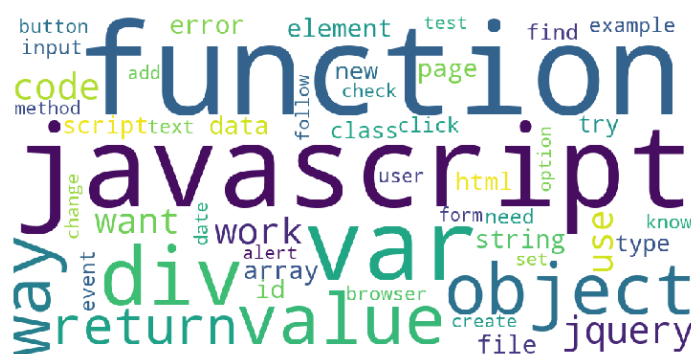


Figure 2. Mots du corpus plus fréquemment associés au mot clé “javascript”

Pour effectuer automatiquement cette tâche, il faut d'abord numériser les mots du corpus et les mots clés plus importants et représenter chaque mot comme un vecteur (vectorisation), puis appliquer aux données numériques des algorithmes d'apprentissage automatique.

4. Vectorisation

Dans la colonne des mot clés “Tags” on a suivant plusieurs valeurs dans chaque ligne. Pour les transformer en vecteurs, le paquet sklearn offre la fonction “MultiLabelBinarizer”. Cela donne une matrice avec plus de 8000 colonnes, avec chaque colonne correspondant à un mot-clé. Avant de l’utiliser comme valeur cible (y_true), j’ai appliqué la fonction “filter” et choisi que les 110 mot clés plus fréquents. Pour le transformer le corpus en une matrice des termes j’ai testé deux méthodes de vectorisation:

- Représentation vectorielle du sacs de mots

Le corpus est considéré comme un ensemble nonstructuré (sac-de-mots), sans information sur la séquentialité des mots dans le texte. À chaque mot présent dans le document est associée une valeur reflétant son importance comme descripteur du document. Pour cette méthode l'importance est calculée à partir de la fréquence du mots dans le corpus, term frequency (TF) en anglais. Finalement, le texte est donc décrit comme un vecteur d'un espace ayant pour dimensions les mots du vocabulaire, mais on peut réduire les nombre des colonnes en sélectionnant celles avec les

valeurs plus importants. Il suffit d'indiquer le numéro dans les options de la fonction "CountVectorizer" du paquet sklearn, et j'ai choisi 220 colonnes.

- Représentation vectorielle de la matrice des termes TF-IDF

Avec cette méthode on relativise l'importance d'un terme dans un document (TF) par son importance dans le corpus (InverseDocumentFrequency /IDF). Autrement dit, plus un terme apparaît dans un grand nombre de document, moins il est pertinent. Au lieu de "CountVectorizer", il j'ai utilisé la fonction "TfidfVectorizer" et obtenu une matrice avec les mêmes dimensions (220 colonnes) mais des variables (features) pondérées autrement (count vs. TF-IDF score en Tableau 3).

Mots avec la meilleure note TF-IDF				Mots plus fréquentes			
words	idf_weight	count	TF_IDF score	words	idf_weight	count	TF_IDF score
file	2.736351	17370	47530.417314	file	2.736351	17370	47530.417314
android	3.915541	10096	39531.297872	like	2.264955	14620	33113.642445
string	3.048507	12716	38764.813379	use	2.386402	14324	34182.821721
class	3.145947	11375	35785.151931	string	3.048507	12716	38764.813379
error	3.020527	11575	34962.603081	way	2.421332	11923	28869.540056
use	2.386402	14324	34182.821721	code	2.616212	11800	30871.307214
like	2.264955	14620	33113.642445	error	3.020527	11575	34962.603081
function	3.100097	10272	31844.193251	want	2.448669	11504	28169.487636
value	3.12546	9943	31076.444477	class	3.145947	11375	35785.151931
code	2.616212	11800	30871.307214	work	2.576914	11094	28588.279975

Tableau 3. Fréquence (count) vs. TF-IDF des mots dans le corpus.

La simple comparaison avec les tags plus fréquentes confirme les conclusions de cette étude². La représentation pondérée TF-IDF est meilleure que la fréquence non pondérée pour choisir les tokens associés aux mots clés.

5. Modèles non-supervisés

Le modèle non-supervisé classique pour l'extraction des thèmes ou "sujets latents" d'un texte, à partir d'une matrice terme – document, est l'allocation latente de Dirichlet, Latent Dirichlet Allocation (LDA) en anglais. Pour ça, il faut définir le nombre de sujets et le nombre de mots attribués à chaque sujet. Si on choisit 5 sujets pour notre corpus factorisé avec la méthode TF-IDF on obtient :

```
Topic #0: value data var table array key select like id date
Topic #1: android div text view page image class html button input
Topic #2: use difference line like find code python app version application
Topic #3: file error git run command try project work change script
Topic #4: string class return function method public new list int object
```

On voit que est raisonnable assigner au sujet 1 le mot clé 'android' et 'git' au sujet 3.

Une autre méthode pour l'extraction des thèmes des documents est la factorisation en matrices non négatives (NMF). Les mots associés aux 5 sujets du corpus avec l'NMF sont:

² Raffael Vogler, 'The Tf-Idf-Statistic For Keyword Extraction', *Joy of Data*, 2014, World
<<https://www.joyofdata.de/blog/tf-idf-statistic-keyword-extraction/>> [accessed 5 June 2020].

Topic #0: android id px text true view background button app item
Topic #1: string public new return int code function like use method
Topic #2: option value time select key php input type id name
Topic #3: file error git directory version run find http line build
Topic #4: div class id px text button type width input form

Pour choisir le nombre optimale j'ai partagé le documents du corpus entre une partie d'entraînement du modèle (data_train) et une de validation (data_test). La perplexité estimée sur un corpus de validation est inversement proportionnelle à la précision du modèle. La variation de la perplexité pour l'LDA en fonction des nombres des thèmes est représentée en Figure 3.

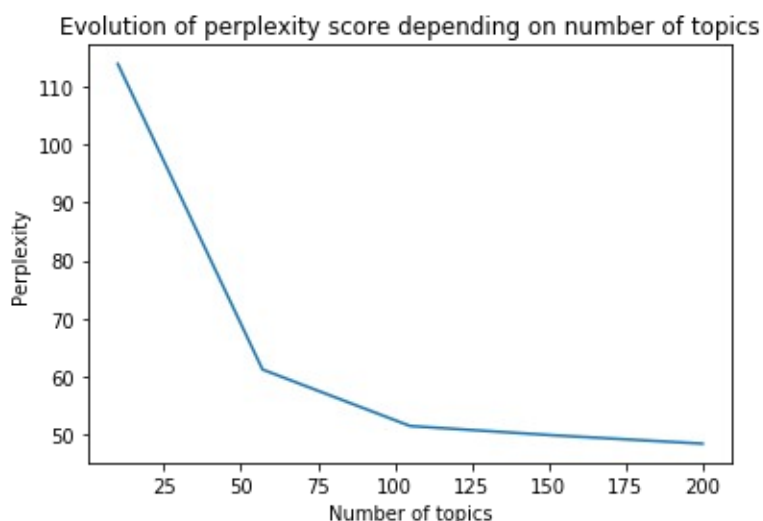


Figure 3. Perplexité de l'LDA en fonction du nombre des thèmes du corpus.

Le valeur optimale est celui du “coude”, le minimum des sujets latents avec une faible perplexité. Dans ce cas la il correspond à 55. En suite pour chaque thème j'ai codé une fonction (decorator) que assigne des mots-clés basés sur la note TF-IDF des mots contenus chaque document (post sur Stackoverflow). En suite , j'ai appliqué la même fonction en utilisant, à la place de l'LDA, l'extraction des sujets par factorisation en matrices non négatives (NMF).

Finalement j'ai testé le paquet d'extraction automatique des mots clés YAKE! Avec YAKE! Il suffit d'appliquer la fonction “extract_keywords” directement aux documents du corpus. Les principes de fonctionnement de YAKE! Sont décrits en détail ici³.

Pour évaluer les méthodes non-supervisées, j'ai vectorisé la colonne de prédictions de chaque modèle j'ai comparé la similarité du vecteur à celui des tags. La similarité textuelle a été mesurée avec la similitude cosinus des vecteurs de terme⁴. Les résultats montrent que la performance de YAKE! est la meilleure (Figure 4).

3 Ricardo Campos and others, 'YAKE! Collection-Independent Automatic Keyword Extractor', in *European Conference on Information Retrieval* (Springer, 2018), pp. 806–10.

4 Baoli Li and Liping Han, 'Distance Weighted Cosine Similarity Measure for Text Classification', in *International Conference on Intelligent Data Engineering and Automated Learning* (Springer, 2013), pp. 611–18.


```
y_tfidf_YAKE = tfidfVectorizer.transform(df_test['YAKE tags'])
print('Dummy score:', cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_dummy.reshape(1,-1)))
print('YAKE score:', cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_YAKE.reshape(1,-1)))
print('LDA score:', cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_LDA.reshape(1,-1)))
print('NMF score:', cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_NMF.reshape(1,-1)))
```

Dummy score: [[0.05342378]]
YAKE score: [[0.29256526]]
LDA score: [[0.18009483]]
NMF score: [[0.18110464]]

Figure 4. Résultats obtenus par les différents modèles non-supervisés.

6. Modèles supervisés

L'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets à partir des données dont le classement est connu. Pour cette étude les variables qualitatives sont les tokens du corpus et le classement est fait selon les mots-clés (tags). Comme pour l’LDA et l’NMF on dispose au départ d'un échantillon dit d'apprentissage, 80% du corpus (X_train) avec les tags correspondants à chaque question (y_train). Cet échantillon est utilisé pour l'apprentissage des règles de classement que sont en suite testées sur le restant 20% du corpus (échantillon de validation X_test). Les prédictions (y_pred) sont enfin comparées aux mots clés de chaque post (question) de l'échantillon (y_test) pour mesurer la exactitude des test (“accuracy” en anglais), pour chaque modèle⁵. Les résultats sont décrits en Figure 5.

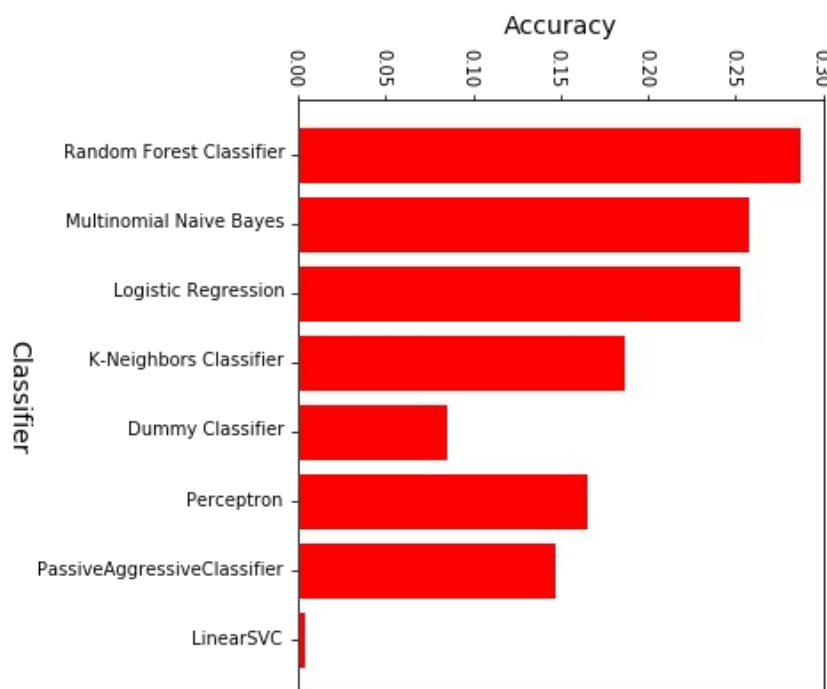


Figure 5. Exactitude (“accuracy”) de chaque modèle.

⁵ ‘Précision et rappel’, Wikipédia, 2020 <https://fr.wikipedia.org/w/index.php?title=Pr%C3%A9cision_et_rappel&oldid=169870260> [accessed 5 June 2020].

Le modèle supervisé plus performant est clairement celui des “forets aléatoires”, Random Forest en anglais, avec une optimisation des hyper-paramètres. Par contre il présente un inconvénient par rapport au calcul car il occupe presque 10 Gb de RAM si on modèle un corpus de 220 tokens et 110 mots clés. Si on réduit les dimensions du corpus avec une analyse des composants principaux (ACP) tous les modèles perdent fortement de la précision. En fonction de la puissance du système on pourrait donc choisir soit le Random Forest soit un autre algorithme un peu moins performant mais beaucoup moins lourd comme la Régression Logistique. Si on compare les deux avec la mesure de similitude cosinus des vecteurs de terme, on obtient des meilleurs résultats que pour les modèles non-supervisés. La similitude cosinus est de 0.616 pour le Random Forest, et de 0.564 pour la Régression Logistique.

7. Conclusions