



CATEGORISATION DES QUESTIONS

Projet 5: Parcours Data Science / OpenClassrooms



Ecrit par:

Marc Felix DEGNI

Apprentice Data Scientist Consultant OC/CC

NOVEMBER 1, 2018

Capgemini Invent



Contents

INTRODUCTION	1
I- Stack Overflow	2
II- Matériaux utilisés	2
II.1. Datas	2
II.2. Logiciels utilisés	2
II.2.1. Jupyter notebook.....	2
II.3. Flask & Heroku.....	3
II .3.1. Flask.....	3
II.3.2. Heroku	3
III- Méthodologie.....	3
III.1. <i>Data Preprocessing</i>	3
III.1.1. <i>Cleaning</i> des données	3
III.1.2. Tokenisation	4
III.1.3. Normalisation	4
III.1.4. Lemmatization ou le stemming	4
III.2. Analyse Exploratoire des données (EDA).....	4
III.3. Approche data.....	5
III.3.1. Approche semi-supervisée.....	5
III.3.2. Approche supervisée	8
CONCLUSION	9

Table des figures

Figure 1 : Data preprocessing.....	4
Figure 2 : Les 30 tags les plus fréquents dans le jeu de donnée	5
Figure 3 : Approche globale de la méthodologie	5
Figure 4 : Latent Dirichlet Allocation (LDA)	6
Figure 5 : Visualisation des topics extraits avec la méthode LDA sur notre jeu de donnée.....	6
Figure 6 : Les topics présents à forte probabilité dans les 4 premières questions	7
Figure 7: Matrix TF-IDF	8



INTRODUCTION

Le traitement de texte est de plus en plus utilisé de nos jours, pour répondre à des problématiques business. La majorité des entreprises françaises ont recours à des techniques de traitement de texte avancées pour réorienter leurs stratégies marketing et améliorer leurs prises de décisions.

En data science, un outil permet de manipuler à souhait tous types de textes : ce sont les données non structurées. Ces données peuvent provenir de diverses sources (base de données interne datawarehouse, internet via le scraping, pdf, etc). Avec le NLP (Natural Language Processing), il est possible de travailler sur des données non structurées, les traiter et en tirer de la valeur ajoutée (analyse de sentiment, classification de document, extraction de topic, etc).

Dans ce rapport, l'objectif recherché est d'apprendre à exploiter cet outil, mais aussi de maîtriser toutes les possibilités qu'il offre dans la résolution de problème business.

Ce rapport sera spécialement orienté vers les suggestions automatiques de tags du site web *Stack Overflow*.

La première partie de l'étude sera consacrée à la présentation du site web *Stack Overflow* et des données utilisées. La seconde partie, sera consacrée à l'explication de façon explicite des techniques de data science avec les résultats obtenus.



I- Stack Overflow

Stack Overflow est une plateforme qui a été créée par Jeff Atwood et Joël Spolsky en 2008 comme une alternative aux autres sites Q&A, dans le but d'aider les membres en leur proposant des questions et réponses sur un large choix de thèmes en matière de programmation informatique. C'est un site web dédié aux communautés de développeurs pour leur permettre d'apprendre, de partager leurs connaissances en programmation et même de construire leur carrière. Chaque membre a la possibilité de voter pour les questions et réponses postées, leur faisant ainsi gagner des points appelés réputation à leurs auteurs. Il est également possible de voter contre pour pénaliser l'auteur de la réponse et indiquer aux futurs lecteurs que cette réponse peut être incorrecte. Le but de ces votes est de mettre en avant les réponses de qualité, tout en récompensant leurs auteurs, leur donnant ainsi accès à des privilèges quand certains seuils de réputation sont atteints (ex : participer à des votes, limiter les publicités, pouvoir fermer les questions).

Les questions posées sont référencées par des tags, ce qui facilite la recherche des réponses sur un sujet et faire le lien avec d'autres questions parlant du même sujet.

Les utilisateurs de *Stack Overflow*, qui ont eu la chance de résoudre certaines difficultés grâce à ce site, pourront eux aussi, en retour, aider la communauté. Pour cela, chaque utilisateur est amené à développer un système de suggestion de tag pour le site. Celui-ci prendra la forme d'un algorithme de Machine Learning qui assigne automatiquement plusieurs tags pertinents à une question.

C'est donc pour contribuer à cette tâche que ce rapport est rédigé. Il traitera de l'utilisation des techniques d'intelligences artificielles pour proposer automatiquement des tags sur les questions des utilisateurs. Cela permettra d'indexer ces questions et de les référencer dans le but de faciliter la recherche au moyen des "tags".

Pour utiliser les arsenaux fournis par les algorithmes de Machine Learning, plusieurs outils sont indispensables. Ces outils seront présentés dans le chapitre suivant.

Pour poser une question sur ce site, l'utilisateur doit entrer plusieurs tags de manière à retrouver facilement la question par la suite. Pour les utilisateurs expérimentés cela ne pose pas de problème, mais pour les nouveaux utilisateurs, il serait judicieux de suggérer quelques tags relatifs à la question posée.

II- Matériaux utilisés

II.1. Datas

Stack Overflow propose un outil d'export de données nommé "*stackexchange explorer*". Cet outil permet de recenser un grand nombre de données authentiques de la plateforme d'entraide.

Ces données peuvent être obtenues via la requête SQL suivante :

```
SELECT* posts WHERE Id < 50000
```

Ce code permet de sélectionner toutes les colonnes sur lesquelles effectuer la data preparation.

II.2. Logiciels utilisés

II.2.1. Jupyter notebook



Jupyter Notebook est une application Web à source ouverte qui permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif. Les utilisations incluent le nettoyage et la transformation de données,



la simulation numérique, la modélisation statistique, la visualisation de données, l'apprentissage automatique, etc.

Cette application est utilisée comme un éditeur de texte pour la lecture des codes de conception de model de ce projet.

II.2.2. Langage de programmation



Le langage de programmation utilisé dans le cadre de notre recherche par le site *Stack Overflow* est Python.

Python est un langage de programmation objet interprété, multi-paradigme et multiplateformes, qui favorise la programmation impérative structurée, fonctionnelle et orientée objet.

La dernière version de Python, nommée Python 3 (version 3.7.1 du 20 octobre 2018) a été utilisée. C'est en effet, celle qui est compatibles aux packages et librairies utiles pour le machine Learning. Ce sont :

-  Scikit-learn
-  Gensim
-  Spacy
-  Pickle

II.3. Flask & Heroku

II .3.1. Flask



Flask est un framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de Template. L'avantage est qu'il permet au développeur de programmer en python, qu'il traduit immédiatement en *html*, *css* et *js*, pour l'afficher sur une page web. Ce Micro Framework a permis de créer l'API dans ce projet.

II.3.2. Heroku



Heroku est un service de cloud computing de type Plateforme en tant que service (*Platform as a Service* PaaS), qui permet de mettre en ligne (dans le Cloud), par des systèmes de « git-push » et « git-pull », le code API obtenu dans le cadre de cette étude.

III- Méthodologie

III.1. *Data Preprocessing*

La préparation de la donnée ou '*Data Preprocessing*', est un préalable à la manipulation des données sélectionnées. Il s'agit de procéder à un pré-traitement du corpus de texte récupéré. Il faut noter que le pré-traitement du texte constitue une étape très importante qui nécessite une observation minutieuse du contenu du texte, pour s'assurer de la cohérence avec les objectifs visés.

III.1.1. *Cleaning* des données

Les données requêtées sur le site *stackexchange explorer* sont en format html. Cette phase de *cleaning* consiste à récupérer uniquement les données textes du Body, c'est-à-dire extraire le texte présent dans les différentes balises html (Fig.1).

III.1.2. Tokenisation

La tokenization consiste à effectuer un découpage des différents mots présents dans les documents, de sorte à obtenir dans le corpus, des mots séparés qui seront étudiés individuellement. Cette séparation a aussi pour objectif de mettre en évidence certains mots de moindre importance. Ces mots sont généralement appelés ‘*stopwords*’. Ce sont des mots courants (adjectifs, déterminants) ou expressions inutiles qui n’apportent aucun élément important au travail effectué. Par exemple, la recherche d’information nécessite la suppression des mots vides, car vu leur caractère commun, il est inutile de les indexer ou de les utiliser dans une recherche (Fig.1).

III.1.3. Normalisation

La troisième étape consiste à normaliser la donnée obtenue. En effet, la normalisation est le processus d’harmonisation des ‘*tokens*’ : c’est le nettoyage de la donnée. C’est la suppression des éléments inutiles présents dans ce texte (les points, les apostrophes, les ponctuations), vu qu’ils n’apportent aucune information pertinente et ne permettent pas de résoudre la problématique (dans certains cas, ces ponctuations sont très utiles pour étudier le contexte ou la structure des documents : poème, etc).(Fig.1)

III.1.4. Lemmatization ou le stemming

L’étape de ‘*stemming*’ (racinalisation en français), permet de trouver sur la base de la racine de différents mots du corpus, la fréquence de leur apparition dans chaque document.

La *lemmatization* quant à elle, permet de garder la tige sur laquelle est construit un mot, par conséquent détermine la famille appartenant à un groupe de mot. L’objectif est cependant pareil que le *stemming*, c’est-à-dire éviter des doublons de mot dans notre corpus, et étudier la fréquence d’apparition d’un groupe de mot dans un document. Cela permet de regrouper les tags et donc, de faciliter les recherches.

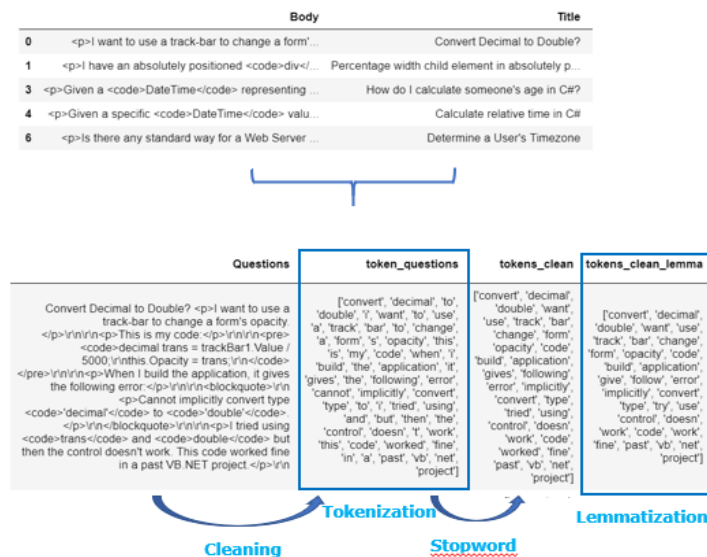


Figure 1 : Data preprocessing

III.2. Analyse Exploratoire des données (EDA)

La phase d’EDA consiste à repérer les tags les plus utilisées dans le jeu de donnée. Il ressort de façon très claire que les tags les plus fréquents sont des langages de programmation. Ce qui est évident car c’est la vocation première de ce site est de fournir des aides en programmation entre membres de la communauté (Fig.2).

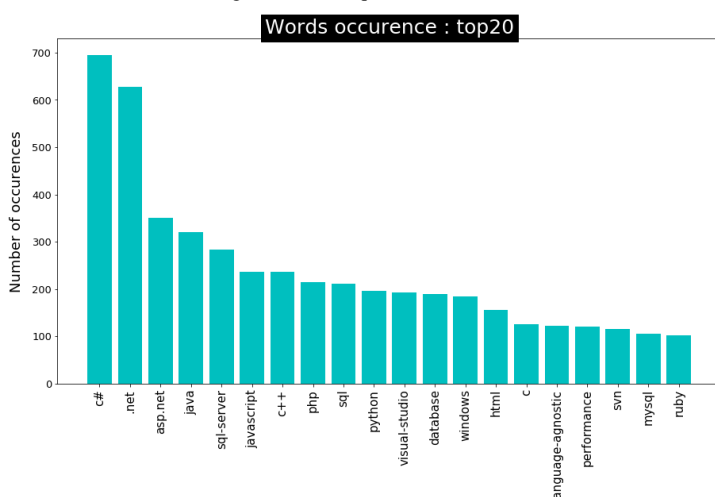


Figure 2 : Les 30 tags les plus fréquents dans le jeu de donnée

III.3. Approche data

Pour répondre à la problématique qui est de taguer les questions de façon automatique, deux (02) approches seront envisagées : l'approche semi-supervisée et l'approche purement supervisée.

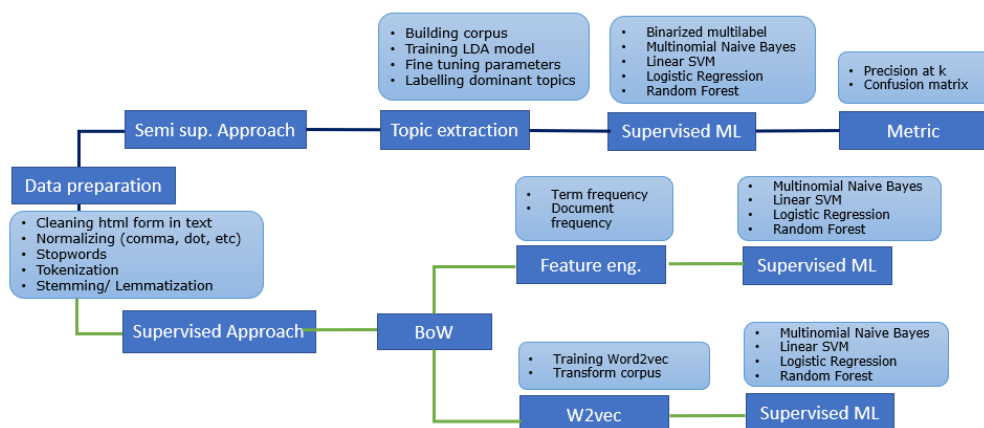


Figure 3 : Approche globale de la méthodologie

III.3.1. Approche semi-supervisée

III.3.1.1. Extraction de topic

L'approche semi-supervisée se compose d'une part de l'approche non supervisée, et de l'autre, de l'approche supervisée.

La partie non supervisée est utile pour l'extraction de topic. Pour ce faire, le travail s'effectuera uniquement sur un jeu de donnée (Body and Title), sans que l'on tienne compte des tags qui sont les labels à prédire.

Les traitements précédemment détaillés (III.1.) ayant été appliqués aux questions, l'on obtient donc un corpus de *tokens* lemmatisés sur lequel se fera l'entraînement d'un modèle de LDA (Fig.).

La méthode utilisée nommée Latent Dirichlet Allocation (LDA), est une méthode non-supervisée générative, qui se base sur les hypothèses suivantes :

- Chaque document du corpus est un ensemble de mots sans ordre (*bag-of-words*) ;
- Chaque document aborde un certain nombre de thèmes dans différentes proportions qui lui sont propres ;

- Chaque mot possède une distribution associée à chaque thème, ce qui permet de représenter chaque thème par une probabilité sur chaque mot ;
- Puisque l'on a accès uniquement aux documents, on doit déterminer quels sont les thèmes, les distributions de chaque mot sur les thèmes, la fréquence d'apparition de chaque thème sur le corpus (Fig.4).

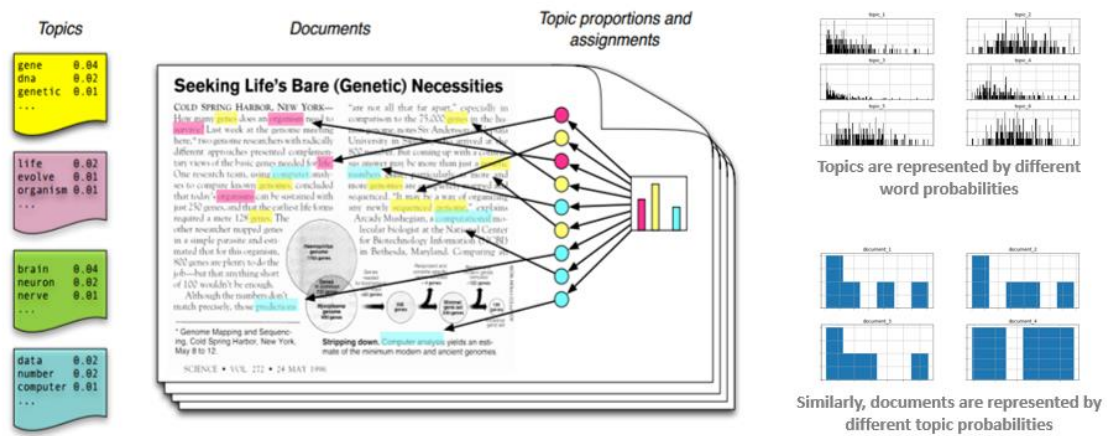


Figure 4 : Latent Dirichlet Allocation (LDA)

Pour cette étude, l'optimisation de la LDA a été effectuée à l'aide de 30 topics représentés par l'image ci-dessous :

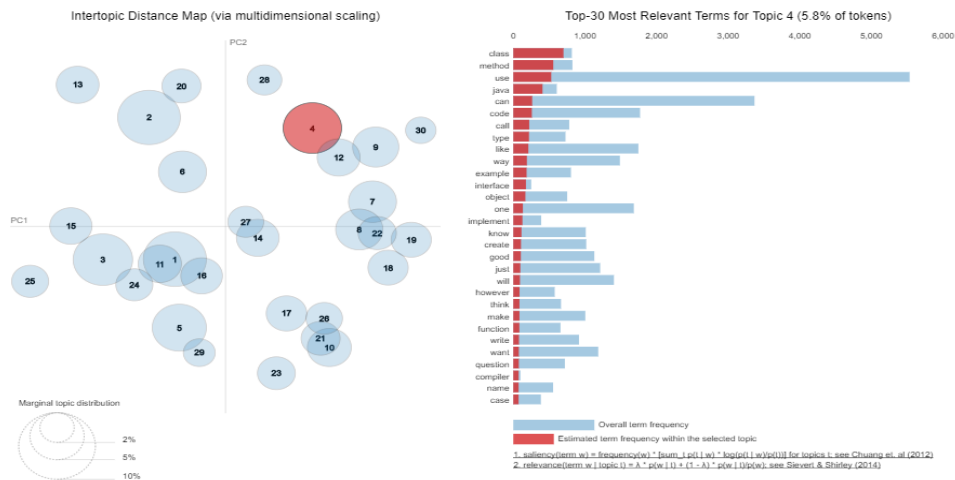


Figure 5 : Visualisation des topics extraits avec la méthode LDA sur notre jeu de donnée

Parmi ces 30 topics, les plus dominants de chaque question ont pu être déterminés (Fig.6).



Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
12.0	0.4467	array, difference, index, action, code, mapping, label, use, one, loop	[difference, union, union, difference]
23.0	0.2488	java, column, performance, add, number, limit, row, can, time, many	[open_source, alternative, matlab, fmincon, function, open_source, alternative, matlab, function, constrain, linear, optimization, rewrite, matlab, program, use, python, numpy, scipy, function, haven_find, equivalent, numpy, base, solution, ideal, language, will]
16.0	0.2592	multiple, sharepoint, can, server, trigger, one, account, insert, keyword, error_message	[come, bit, kernel, can, run, bit, binary, os, box, kernel, bit, binary, yet, can, run, bit, binary, work, cristi, diciu, file, mach, bit, executable, cristi, diciu, file, mach, kernel, mach, kernel, mach, universal, binary, architecture, mach, kernel, architecture, mach, executable, mach, kernel, architecture, ppc, mach, executable, ppc, cristi, diciu, cristi, diciu, echo]
42.0	0.2224	use, net, client, application, look, component, dynamic, service, web_service, good	[dynamic, code_analysis, dynamic, code_analysis, different, static_code_analysis, ie, can, catch, can, catch, static, ve_hear, bound, check, memory, analysis, thing, check, use, dynamic, analysis, adam]

Figure 6 : Les topics présents à forte probabilité dans les 4 premières questions

Passé cette étape, chaque topic sera par la suite labélisé. Ce processus consiste à faire correspondre chaque topic aux tags auquel il est associé. C'est ainsi que l'on aboutit à l'approche supervisée.

	Dominant_Topic	token_tag
0	4.0	[c#, floating-point, type-conversion, double, decimal]
1	1.0	[html, css, css3, internet-explorer-7]
2	2.0	[c#, .net, datetime]
3	27.0	[c#, datetime, time, datediff, relative-time-span]
4	20.0	[javascript, html, browser, timezone, timezoneoffset]

Dans cette partie, des algorithmes de classification seront entraînés sur le jeu de donnée précédent, pour essayer de labéliser les topics à plus forte probabilité d'apparition, extrais des questions (Body and Title) par la méthode LDA. Ce *dataset*, présente des multilabels à prédire. Pour cela, il est possible de binariser la *target*.

III.3.1.2. Binarisation de la target

La binarisation consiste à transformer des mots en vecteurs binaire 0 ou 1. A titre d'exemple, voici ci-dessous présenté un extrait de la *target* comprenant des multi-labels :

0	[c#, floating-point, type-conversion, double, decimal]
1	[html, css, css3, internet-explorer-7]
3	[c#, .net, datetime]
4	[c#, datetime, time, datediff, relative-time-span]
6	[javascript, html, browser, timezone, timezoneoffset]
7	[.net, math]

Après binarisation de ces multi labels par la méthode `MultiLabelBinarizer` de Scikit-Learn, une matrix de binaire sera obtenue :

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

III.3.1.3. Prédiction et les métriques de performance

Les deux algorithmes de classification utilisées, sont la régression logistique et la classification multinomial naïves bayes (le détail de ces méthodes ne sera pas donné dans ce document). La



performance est mesurée par le pourcentage moyen de *tags* juste prédits (moyen : la moyenne des probabilités sera faite pour chaque question).

Méthodes	Precision at K
Logistic regression	6,5 %
Multinomial Naives Bayes	6,18%



III.3.2. Approche supervisée

III.3.2.1. Feature extraction

Pour l'application de la méthode supervisée, le travail consistera à observer les questions (*title and body*) posées et les tags qui ont été attribué par le générateur de *Stack Overflow*. Ayant des données non structurées, il est important d'extraire des *features* (variables) sur lesquelles seront entraînées le model. Deux approches sont possibles pour la construction du Bag Of Word (BOW) : le TF-IDF et le Word2vec.

Bow: Term Frenquency (TF) – Inverse Document Frenquency (IDF)

La méthode TF-IDF, permet de décrire des mots en leur associant des poids. Ces poids sont obtenus de la façon suivante :

-  Pour un *token* donné, on calcul la fréquence d'apparition du mot dans le document (ou la question) : c'est le TF.
-  Puis l'on divise le résultat obtenu par le nombre de documents du corpus dans lequel ce mot apparait : C'est l'IDF.

Ainsi, plus le poids du mot est grand pour un document, plus il a de forte chance de caractériser le document. Et plus le poids du mot est faible, et plus il est commun à tous les documents.

Ainsi les mots à faible poids peuvent être supprimés pour améliorer la performance du modèle.

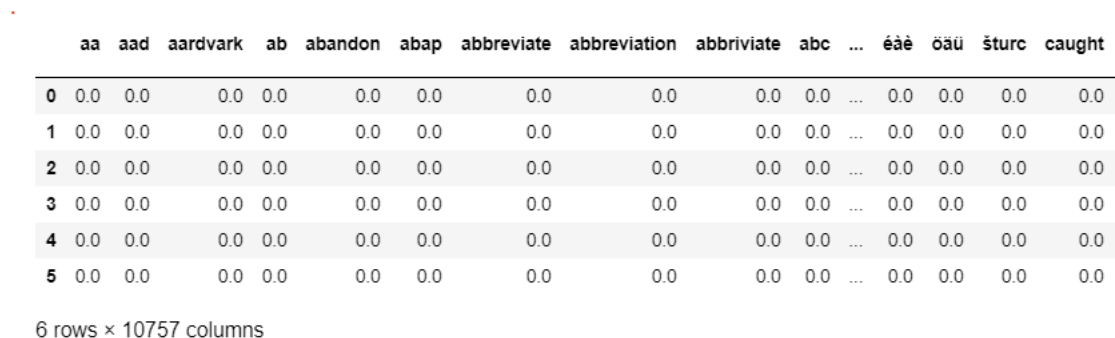


Figure 7: Matrix TF-IDF

L'inconvénient, est qu'il produit une matrice éparsée. Pour résoudre ce problème, l'on pourrait effectuer un plongement de mot (word embedding) dans un sous espace de dimension réduit de sorte à densifier la matrice : c'est le word2vec.

Bow : Word2Vec (Word Embedding)

La méthode *word2vec* permet de définir les mots dans un espace de n dimension, de sorte à les visualiser dans un graphique. Les mots les plus proches entre eux dans les graphs auront la même signification dans le contexte de la problématique et inversement.

Après avoir entraîné le *w2v* dans le contexte de la problématique, sur dix (10) dimensions, le résultat ci-dessous présente les mots les plus similaires à « *convert* » :



```
model.most_similar('convert')
```

```
executed in 16ms, finished 14:54:44 2018-09-17
```

```
[('hashcode', 0.9891046285629272),  
( 'hash', 0.9856103658676147),  
( 'iterate', 0.9854134321212769),  
( 'aq', 0.9842157959938049),  
( 'timezone', 0.9826910495758057),  
( 'chicago', 0.9822856783866882),  
( 'represent', 0.9810150861740112),  
( 'decrypt', 0.9808613657951355),  
( 'concatenate', 0.980566143989563),  
( 'representation', 0.9798986315727234)]
```

Ainsi, de nouvelles données textuelles seront définies dans ce nouveau dictionnaire entraîné. Dans ce dictionnaire, chaque mot de la question sera défini dans les coordonnées des autres mots, et pour chacun de ces mots, l'on prendra le poids moyen dans des coordonnées dans le sous espace inférieur.

III.3.2.2. Performance

En utilisant la métrique définie précédemment (ou précision at K), on obtient pour plusieurs algorithmes testés les résultats suivants :

Algorithme	Bow	Metrics
MNB	TF-IDF	27,12%
Linear SVC	TF-IDF	24,37%
MNB	Word2vec	8,90%
Linear SVC	Word2vec	2,62%

Effectuer l'inférence de ce modèle est relativement complexe techniquement. Il faut notamment passer

CONCLUSION

La méthode semi-supervisée présente des avantages dans l'extraction de topics. Cependant, cette méthode s'avère moins efficace au niveau de la labélisation supervisée des topics, documents de chaque question.

Il s'avère que la méthode purement supervisée produit des résultats assez cohérents et satisfaisants. Cependant plusieurs améliorations peuvent être menées :

- ✚ La réduction de la sparsité de la matrice TF-IDF par une analyse en composantes principales ou autre méthode de réduction de dimension. Les algorithmes utilisés auront alors moins de problèmes de calcul et pourront focaliser leur intelligence sur les mots ayant des poids dans au moins un document.
- ✚ Une autre piste d'amélioration serait la suppression des mots ayant un poids en dessous d'un seuil fixé, car ces *features* seront communes à chaque document et par conséquent n'y apporteront aucune valeur.
- ✚ Avec le *word2vec*, l'on pourrait à défaut d'utiliser un *MeanEmbedding* qui agrège les valeurs des poids des mots présents dans les questions, faire une *bag of word* basé sur une méthode TF-IDF.

Il semble évident que l'amélioration des résultats et la prédiction de tags justes, nécessite que l'utilisateur de l'application soit un utilisateur expérimenté. L'API déployé sur heroku peut être consulté via le lien

<https://autotag-suggestion-nlp.herokuapp.com/>