Projet 5:
Stack Overflow - Prédiction de Tags

Edward Levavasseur

Ingénieur Machine Learning

Ce projet cherche a développer des méthodes de prédictions de tags sur des questions de Stack Overflow. Après avoir identifié les 500 mots principaux - ou features - des questions, j'utilise d'abord une méthode non-supervisée puis une méthode supervisée pour prédire les tags associés aux questions. La méthode non-supervisée consiste à appliquer un modèle de Latent Dirichlet Allocation (LDA), puis à attribuer les tags des sujets les plus pertinents à la question. La méthode supervisée consiste à appliquer une régression logistique puis un random forest classifier, paramétrées sur des donné training, pour prédire les tags des données test. Le score de accuracy sur la méthode supervisée dépasse 0.9 sur tous les tags des données test.

Keywords: Stack Overflow, Tags, Supervisé, Non-Supervisé

Tous les fichiers peuvent-être téléchargés sur:
https://github.com/EdwardLevavasseur/OpenClassrooms

1. Introduction

Générer des tags permet de regrouper des videos, des textes et autre contenus sur Internet selon la similarité de leurs thématiques. Lorsque le tagging est efficace, il permet de faciliter les recherches des internautes en les amenant vers des videos et textes proches ce qu'ils cherchent.

Puisque les tags ont pour but de regrouper les textes, les tags - pour être efficaces - doivent-être ni trop génériques ni trop précis. S'ils sont trop génériques, ils risquent de ne pas renvoyer l'internaute vers ce qu'il cherche. Si au contraire ils sont trop précis, ils risquent de ne pas suffisamment identifier les similarités entre les textes.

C'est une question qui s'est posé dans cet exerccice, où il s'agissait de proposer des tags de manière supervisée, et non-supervisée pour classer des questions du forum Stacks Overflow. De la même manière, le choix du nombre de features - ici les mots sur la base desquels sera établit la classification des textes - pose une problématique. Trop peu de features et la classification manquera de précision, un trop grand nombre de features et la classification sera affectés par les problèmes du fléau de la dimension.

En grande partie la question du nombre de tags à utiliser et le nombre de features sur lequel fonder l'analyse est aussi déterminé par la puissance de calcul du système d'exploitation. Après plusieurs tentatives, j'ai utilisé 500 features dans ce projet - pour identifier la singularité des textes suffisament précisément - et 200 tags pour ne pas suggérer des mots qui sont trop rares. En utilisant d'avantages de features et de tags, le temps de calcul était trop long.

Dans la partie suivante j'explique comment j'ai produit des bags of words pour chaque question, en identifiant les mots les plus pertinents. Dans la 3e partie j'explique comment j'utilise une approche non-supervisée pour générer des tags. Dans la 4e partie j'explique comment j'ai utilisé une approche supervisée pour générer des tags. Et dans la dernière partie je conclus.

2. Création de Bag of Words

Pour permettre de générer des tags associés à chaque question, j'ai choisi de créer un bag of words pour chaque question, ainsi qu'un Meta-bag-of-words obtenu en concatenant l'intégralité de toutes les questions.

Pour créer les bags of words, j'ai décidé d'utiliser les mots contenus dans les titres des questions, mais aussi d'utiliser les mots présents dans les textes des questions. Pour

cela, j'ai du commencer par filtrer tous les mots et symboles n'étant pas essentiel au contenu des questions. D'abord, j'ai choisi d'identifier tous les paragraphes (tout ce qui est compris entre ...) dans les textes. Ensuite j'ai supprimé les liens urls (<a href=...<a>) et le code (<code>...</code>) qui étaient inclus dans les paragrahes. Puis j'ai supprimé tous les symboles liés à la mise en forme du texte (..., ..., <s>...</s>). Enfin, pour chaque question j'opère une concatenation de tous les paragraphes de son texte, après avoir "cleané" ces paragraphes. J'obtiens donc un texte propre, purgé de mots et caractères parasites.

Ensuite j'ai créé une fonction qui élimine les charactères de ponctuation et qui n'ont pas leur place, ou n'ont aucune utilité dans un bag of words (_,-,+,0,...,9,>,<,[,],...). Puis j'ai créé une fonction qui met les mots d'une texte donné en Bag of Words, et qui consiste d'abord à "Tokenizer" le texte (le diviser en mots stockés dans une liste), puis à le "lematizer" (supprimés les accords sur les mots: pluriels, conjuguaison...).

J'applique les deux fonctions successivement au texte et au titre de chaque question, et je sauvegarde les bag of words résultant dans une variable attibuant un bag of words pour chaque texte.

	Clean_Body	Bags
0	I implemented an to a and i Added all the ne...	['i', 'implemented', 'an', 'to', 'a', 'and', '...
1	Is it possible to take the results from an arr...	['is', 'it', 'possible', 'to', 'take', 'the', ...
2	In , many methods of COM Interop interfaces ar...	['in', 'many', 'method', 'of', 'com', 'interop...
3	Here is my ajax Here is my conrollerI am getti...	['here', 'is', 'my', 'ajax', 'here', 'is', 'my...
4	I have made the following observations:The exe...	['i', 'have', 'made', 'the', 'following', 'obs...
...	...	...
49995	I want to remove my project name from URL.\ne....	['i', 'want', 'to', 'remove', 'my', 'project',...
49996	Let's say I have a CSV file which reads I'd li...	['let', 's', 'say', 'i', 'have', 'a', 'csv', '...
49997	I've been making an app using Play Framework 2...	['i', 've', 'been', 'making', 'an', 'app', 'us...
49998	All of these work of course, but which one is ...	['all', 'of', 'these', 'work', 'of', 'course',...
49999	I started to refactor my entire react native n...	['i', 'started', 'to', 'refactor', 'my', 'enti...

[50000 rows x 2 columns]

Figure 1. Génération de Bag of Words pour chaque Question

Ensuite j'applique ces mêmes deux fonctions au Méta-texte et au Méta-titre obtenu respectivement en concatenant les textes de toutes les questions, et en concatenant les titres de toutes les questions. Cela me permet d'établir une fréquence des mots dans tous les titres et toutes les questions de Stack Overflow. Ci-dessous, je montre la fréquence des mots les plus utilisés dans le texte des questions.

Dans figure 2 ci-dessous, on remarque que la majorité des mots sont des "stop words".

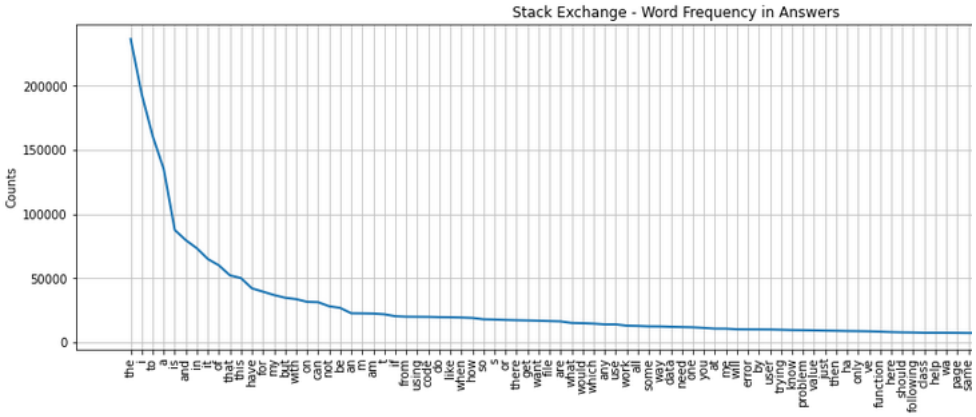


Figure 2. Fréquence des mots dans le texte des questions - avec Stop Words

Par conséquent, pour obtenir des bags of words plus pertinents, je supprime les stop words du meta texte, du meta titre, et je ne garde comme features que les 500 mots les plus fréquents dans les titres. Dans la figure 3, je montre les mots les plus fréquents dans les titres après avoir supprimé les stop words. Ces mots - ainsi que les autres mots parmis les 500 les plus fréquents - sont les features de l'analyse.

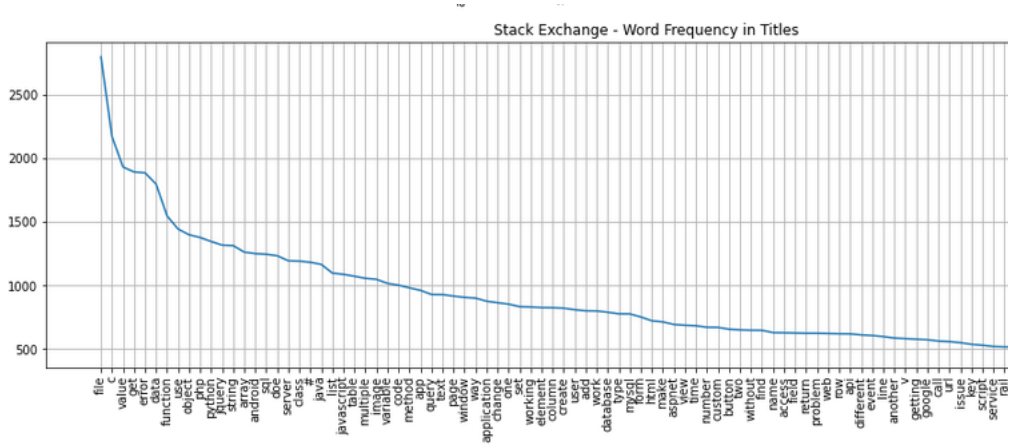


Figure 3. Fréquence des mots dans le texte des questions - sans Stop Words

Enfin je decide de ne garder dans les bags of words que les mots considérés commes étant les 500 features. Je montre dans la figure 4 je montre les bags of words de chaque question, contenant seulement les 500 features retenus pour l'analyse.

Une fois ce travail préliminaire de cleaning du texte, et de mise en bag of words

	Title	Clean_Bags
0	unresponsive UISearchBar	['make', 'problem', 'iphone', 'new', 'back', '...
1	Sorting array from db query by one column of t...	['possible', 'take', 'result', 'array', 'datab...
2	What does MethodImplAttribute(InternalCall, Ru...	['many', 'method', 'interface', 'value', 'call...
3	json data from jquery ajax to rails	['ajax', 'getting', 'error', 'json', 'data', '...
4	xclip does not terminate when tracing it	['copy', 'content', 'default', 'mean', 'button...
...	...	...
49995	Laravel deploy project and remove project name...	['want', 'remove', 'project', 'name', 'url', '...
49996	boolean indexing to store a column value as a ...	['csv', 'file', 'read', 'like', 'store', 'vari...
49997	No Persistence provider for EntityManager in p...	['making', 'app', 'framework', 'problem', 'run...
49998	JS quote best practices? ES6 / React -- single...	['work', 'one', 'best', 'practice', 'file', 't...
49999	React Navigation Hot reload not working	['react', 'native', 'navigation', 'working', '...

[50000 rows x 2 columns]

Figure 4. Génération de Bag of Words pour chaque Question

je peux passer à la prédiction de tags de manière supervisée et de manière non-supervisée. L'approche non-supervisée consiste à déterminer quels mots dans les bags of words sont pertinents comme tags, et repose sur une réduction des dimensionalités sous la forme d'un modèle de Latent Dirichlet Allocation (LDA), puis consiste à attribuer les mots les plus plus fréquents des topics associés à chaque texte. L'approche supervisée, consiste plus simplement à sélectionner un certains nombre de tags (ici 200) parmis les tags proposés par Stack Overflow pour chaque question, puis à prédire si chacun des tags est utilisé pour chaque texte, en utilisant les 500 features comme prédicteurs.

3. Approche Non-Supervisée

La première étape de l'approche Non-Supervisée consiste à conter le nombre de fois que chaque feature est utilisé dans les bags of words de chaque textes. Pour cela j'utilise la fonction "CountVectorizer()" de "sklearn", qui me produit une liste de listes, que je transforme en data frame pandas (montré dans la figure 5 ci-dessous).

	access	accessing	action	activity	add	...	write	writing	wrong	xcode	xml
10	0	0	0	0	0	...	1	0	0	0	0
11	0	0	0	0	0	...	0	0	0	0	0
12	3	2	0	0	0	...	0	0	0	0	2
13	0	0	0	0	0	...	0	0	0	0	0
14	0	0	0	0	1	...	0	0	1	0	0
15	0	0	0	0	0	...	0	0	1	0	0
16	0	0	0	0	0	...	0	0	0	0	0
17	0	0	0	0	0	...	0	0	0	0	0
18	0	0	0	0	0	...	1	0	0	0	0
19	0	0	0	0	0	...	0	0	0	0	0

Figure 5. Count Vectorizer

Ensuite pour opérer un clustering des textes les uns avec les autres, j'ai d'abord

cherché à réduire les dimensions, afin de limiter le temps de calcul des algorithme. J'ai d'abord essayé une PCA (Principal Component Analysis), mais je me suis rendu compte que l'on perdrait trop d'information en utilisant cet Algorithme. En effet sur le graphique **??** ci-dessous on voit qu'avec 100 dimensions par exemple on expliquait moins de 50% de la variance totale. Pour atteidre ne serait-ce 80% de la variance totale il faudrait utiliser plus de 250 dimensions. Par conséquence j'ai décidé d'utilisé un Latente Dirichlet Allocation model (LDA), qui de toute façon est bien plus pertinent pour la classification de textes.

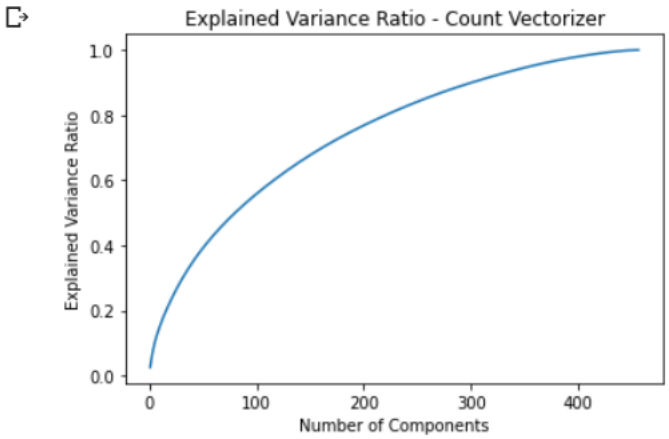


Figure 6. Réduction des dimensions: PCA

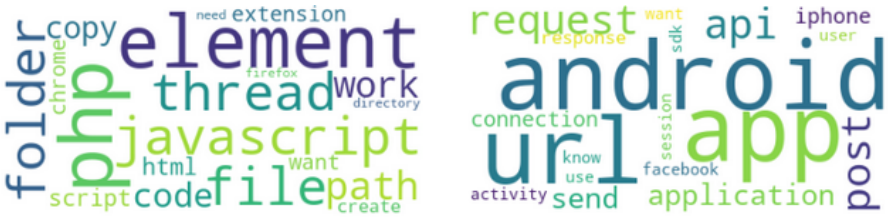
Aisni, ayant compté le nombre de features utilisés dans chaque question, j'applique à la base de données montrée en figure 5 un Latent Dirichlet Model (LDA) avec 20 composantes. Ce modèle LDA va chercher 20 sujets sous-jacents dans les textes en calculant la fréquence d'apparition des features les uns avec les autres. Ayant identifiés les 20 sujets sous-jacents (ou "topics"), le LDA va procéder à mapper chaque texte plus ou moins proche de chaque topic. Un texte qui serait composé exclusivement des features d'un seul "topic" et contenant toutes les features de ce topic aurait une proximité de 1 avec ce topic, et une proximité de 0 avec tous les autres topics. Un texte qui serait composé exclusivement des features de 2 topic différents, et contenant toutes les features de ces 2 topics aurait un score de proximité de 0.5 pour chacun de ces topics, et une proximité de 0 avec tous les autres topics. Dans les faits, chaque texte obtient un score de similarité compris entre 0 et 1 avec chacun des 20 topics. J'ai choisi 20 topics car avec un nombre trop faibles de topics je courrais le risque de mal distinguer les textes les uns des autres. Avec un nombre trop grands de topics je courrait le risque d'avoir des topics constitués de features peu pertinents,

simplement parce que dans mes données plusieurs textes par coincidence utiliseraient ces features ensemble dans leur texte. Cela revient à au problème d'overfitting, au sens d'une classification trop fortement attaché au textes utilisés en input dans la LDA, et pas vraiment révéléateur d'une réelle proximités entre les features. Je sauvegarde les résults obtenus dans un nouveau data frame pandas:

	0	1	2	...	17	18	19
10	0.002941	0.002941	0.002941	...	0.002941	0.002941	0.002941
11	0.001429	0.001429	0.001429	...	0.001429	0.001429	0.001429
12	0.000926	0.000926	0.000926	...	0.000926	0.000926	0.000926
13	0.003125	0.003125	0.003125	...	0.352815	0.003125	0.003125
14	0.099461	0.002381	0.002381	...	0.002381	0.213452	0.410126
15	0.001667	0.601120	0.001667	...	0.001667	0.001667	0.001667
16	0.005000	0.005000	0.005000	...	0.005000	0.005000	0.005000
17	0.243455	0.161859	0.002941	...	0.293837	0.002941	0.002941
18	0.002778	0.002778	0.002778	...	0.002778	0.002778	0.002778
19	0.002778	0.002778	0.002778	...	0.002778	0.002778	0.002778

Figure 7. Association des textes au 20 topics LDA

Dans la figure 7 ci-dessus, je montre l'association des questions aux 20 topics. On voit par exemple que la question 25 a un assez grande proximité avec le topic 1 (0.601120). Chaque topic est associé à un certain nombre de features particuliers, de manière plus ou moins forte. Je montre par exemple les mots les plus importants des topics 1 et des topics 0:



(a) Topic 0 (b) Topic 1

Figure 8. Mots importants des Topics 0 et 1 du LDA

Ensuite j'ai créé une fonction génératrice de tags, qui produit des tags pour un texte donné. Dans un premier temps cette fonction utilise le 'CountVectorizer()' qui a été paramétrisé sur les 500 features utilisés auparavant, pour compter le nombre de fois que chaque feature a été utilisé dans le texte. Ensuite cette fonction transforme cet espace dans l'espace du LDA qui a été paramétrisé au préalable sur l'ensemble des questions. Enfin j'ai créé une API permettant de générer des tags avec cette fonction, lorsqu'on insère un texte avec un titre (Figure 9).

Figure 9. API générant des Tags de manière non-supervisée

4. Approche Supervisée

Pour générer des tags de manière supervisée j'ai créé une base de données en concatenant les 500 features de chaque texte, à laquelle j'ajoute les 200 tags sous forme de variables binaires:

	0	1	2	3	4	5	6	...	winforms	wordpress	wpf	xaml	xcode	xml	xslt
10	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
12	3	2	0	0	0	0	0	...	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
14	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

[10 rows x 646 columns]

Figure 10. 500 features (0, 1, 2...) + 200 dummy pour les tags

Dans un premier temps je divise ces données en Training Set (70%) et en Test Set (30%)

Ayant construit ces bases de données j'applique des regressions logistique et random forest sur les données Training pour prédire si chaque texte est associé à chaque tag. Plus

précisément, j'ai créé une boucle avec 200 itérations (une pour chaque tag), où je prédis avec une regression logistique si chaque texte est associé à un tag donné. A chaque itération je stocke la fonction "fittée" dans un dictionaire, afin de pouvoir faire des prédictions plus rapidement par la suite, en faisant appel aux fonctions "fittées" du dictionnaire. Je fais ensuite de même avec des Random Forest Classifyier pour prédire à nouveau chaque tag.

Ensuite je créée un fonction qui prédit pour un texte donné, et son titre, si celui-ci doit-être associé à chacun des 200 tags. Dans cette fonction je fais appel à toutes les modèles logistiques qui sont "fittés" dans le dictionnaire des régressions logistiques, et je prédit si ce texte doit-être associé à chacun des tags. Dans un second temps je fais de même en utilisant les modèles random forest classifier qui sont fitées et stockées dans leur dictionnaire, pour prédire si chaque Tag doit-être associé au texte. Si la régression logistique ou le Random Forest Classifier prédisent que le texte doit-être associé à un tag donné, alors je considère que c'est un tag du texte.

Enfin, j'ai créé une API permettant de prédire si un texte et son titre doivent-être associés à un tag particulier en utilisant cette fonction.

Figure 11. API générant des Tags de manière non-supervisée

Enfin, Après avoir créé cette API, j'ai utilisé la fonction prédictive (regression logistique, puis random forest), pour prédire dans les données Test si chaque tag doit-être

associé à chaque texte. J'obtiens des Accuracy scores supérieurs à 0.9 pour tous les tags, que ce soit en utilisant les régressions logistiques ou Random Forest Classifier.

5. Conclusion

En somme, il nous était demandé de génerer API permettant de générer des tags de manière supervisée et non-supervisée pour des questions de Stack Overflow. Après avoir "cleané" le texte, en supprimant les caractères inutiles, j'ai procédé à mettre le texte et le titre de chaque question dans des bags of words. Après avoir identifié les 500 features principales, j'ai procédé à garder dans ces bags of words seulement les mots correspondant aux 500 features.

Pour générer des tags de manière non-supervisée, j'ai utilisé un Latent Dirichlet Allocation (LDA), ce qui m'a permis d'identifier 20 topics sous-jacents aux différentes questions, et qui m'a permis de mapper les questions selon leur proximité avec chacun des textes. J'ai ensuite créé un fonction qui mappe n'importe quel texte dans cet espace LDA, et qui lui attribue comme tags les mots principaux des textes dont il est le plus proche.

Pour générer des tags de manière supervisée, j'ai simplement procéder à créer 20 variables dummy que j'ai stocké dans une base de données avec les 500 features comptés dans chaque texte (par le biais du CountVectorizer()). Enfin j'ai lancé des régression logistiques et Random Forest Classifier pour prédire si les tags doivent-être associés aux textes. J'obtiens sur données Test - après avoir fitté sur données Training - des Accuracy Scores supérieurs à 0.9 pour chaque Tag. Cela peut-sembler élever, mais si un individu à utilisé dans son texte le mot du tag, la prédiction n'aura aucune difficulté à prédire ce tag.

Enfin j'ai créer des API pour la fonction génératrice de tags supervisée et non-supervisée.