Projet 5

CATEGORISEZ AUTOMATIQUEMENT DES QUESTIONS

Contents

Présentation de la problématique

Stack overflow est un des plus grands forums de questions – réponses pour les développeurs. Si nous avons besoin d'aide avec un code qui ne marche pas, il suffit souvent d'aller chercher une question qui existe déjà et qui était déjà résolue par les autres utilisateurs de la plateforme.

Aujourd'hui, la base de données de question – réponses devient immense et il est de plus en plus difficile de trouver des questions pertinentes. Afin de simplifier la recherche, les utilisateurs sont censés de rentrer jusqu'à 5 tags qui correspondent à la question. Les tags sont un champ de texte libre, même s'il existe un système d'aide qui nous propose des tags les plus fréquents quand nous commençons à éditer les tags, nous pouvons quand même renseigner tout et n'importe quoi.

Développer un système qui peut prédire un ou des tags à partir de notre question peut être utile à :

- Mieux classifier des questions
- Proposer des questions déjà existantes plus pertinentes
- Eviter des questions en doublons
- Fidéliser les utilisateurs en proposant un fonctionnement simple, rapide et efficace

Aujourd'hui, la base de données contient presque 47 millions de questions, dont 18,5 millions de questions complètes, c'est à dire qui contient et le titre, le corps et les tags. Le monde informatique est constamment en évolution, donc ce qui nous intéresse ce sont des questions les plus récentes. Seulement en 2019, les utilisateurs ont posté environ 4 millions de questions.

Le but de ce projet est de trouver un algorithme de machine learning approprié qui servira à proposer les tags automatiquement aux utilisateurs. Les prédictions de tags se feront à partir de titre et de corps de question saisi par l'utilisateur. L'algorithme est censé de proposer un ou plusieurs tags en temps réel, donc le calcul de la prédiction devrait être fait très rapidement.

Description de données

Notre projet sera basé sur un échantillon de questions complètes qui contient et les tags, le titre et le corps de question, postées en 2019. Nous avons téléchargé un jeu de données de 50 000 questions pour les analyses et la modélisation et un autre échantillon de 500 000 questions pour la création d'un modèle word2vect.

Cleaning

Features

Une feature peut être composée de deux variables : le titre et le corps de question. La particularité de corps est qu'il s'agit d'un texte qui contient des balises HTML et parfois aussi le code entouré de balises <code> et </code>. Hormis le traitement de texte standard, nous avons également enlevé le code, ainsi que les balises HTML. Le schéma suivant représente le workflow de cleaning:

Figure 1 - Cleaning workflow

Cible

La cible est composée de un ou plusieurs tags séparés par « <> ». Nous allons donc uniquement supprimer les chevrons et les remplacer par un espace.

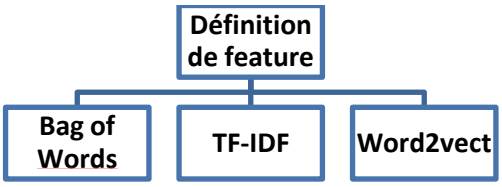
Feature engineering

Nous avons testé plusieurs façons de définition de features :

- Titre uniquement
- Titre + corps
- Titre + corps + bigrams titre + bigrams corps
- Titre pondéré + corps
- Titre pondéré + bigrams titre + corps

Ensuite, nous avons également appliqué plusieurs méthodes qui nous ont permis de créer une représentation numérique de données texte afin de pouvoir appliquer des algorithmes de modélisation.

Figure 2 - Feature engineering workflow



1. Bag of Words

   Il s'agit d'une méthode qui nous a permis de créer une matrice creuse à partir d'un corpus de texte. La matrice est de taille n, où n représente le nombre de question et de k colonnes, où k représente le nombre de n-grams de corpus. Les éléments de la matrice représentent le comptage de nombre de fois le mot apparaît dans la question. Au départ, nous nous sommes limités au nombre maximal de features = 5 000 pour obtenir une matrice de taille raisonnable.

   Nous avons également appliqué l'algorithme sur la cible, ce qui nous a permis de créer une matrice binaire, car chaque tag n'est pas présent plus qu'une fois pour chaque question. Cela nous arrange pour utiliser la matrice de cibles dans des modèles supervisés.

2. TF-IDF (Term Frequency – Inverse Document Frequency)

   Il s'agit d'une méthode équivalente à la méthode Bag of Words, sauf que les termes de la matrice sont calculés comme fréquences pondérées : nous prenons la fréquence de terme au sein de document et nous la divisons par une fréquence inversée de terme dans le corpus. Cela permet d'obtenir une certaine mesure d'exclusivité de mot clé par rapport aux autres documents.

3. Word2Vect

   Word2vect est une méthode de word embedding. Les mots sont représentés par vecteurs, nous avons donc la possibilité de calculer les distances, les similarités entre les mots, ainsi que d'effectuer des opérations arithmétiques. Il est possible d'utiliser des dictionnaires qui existent déjà et qui sont disponible en ligne, par exemple des vecteurs pré-entraînés sur Google News dataset (100 billion mots). Etant donné que notre problématique est spécifique dans le domaine d'informatique, nous avons créé notre propre modèle à partir de

500 000 questions publiées en 2019. Nous avons également testé 2 types de modèles : un qui contient des vecteurs de taille 300 et un autre de taille 400. L'image suivante représente les testes qui ont été effectué sur le modèle :

Figure 3 - Test de word2vect

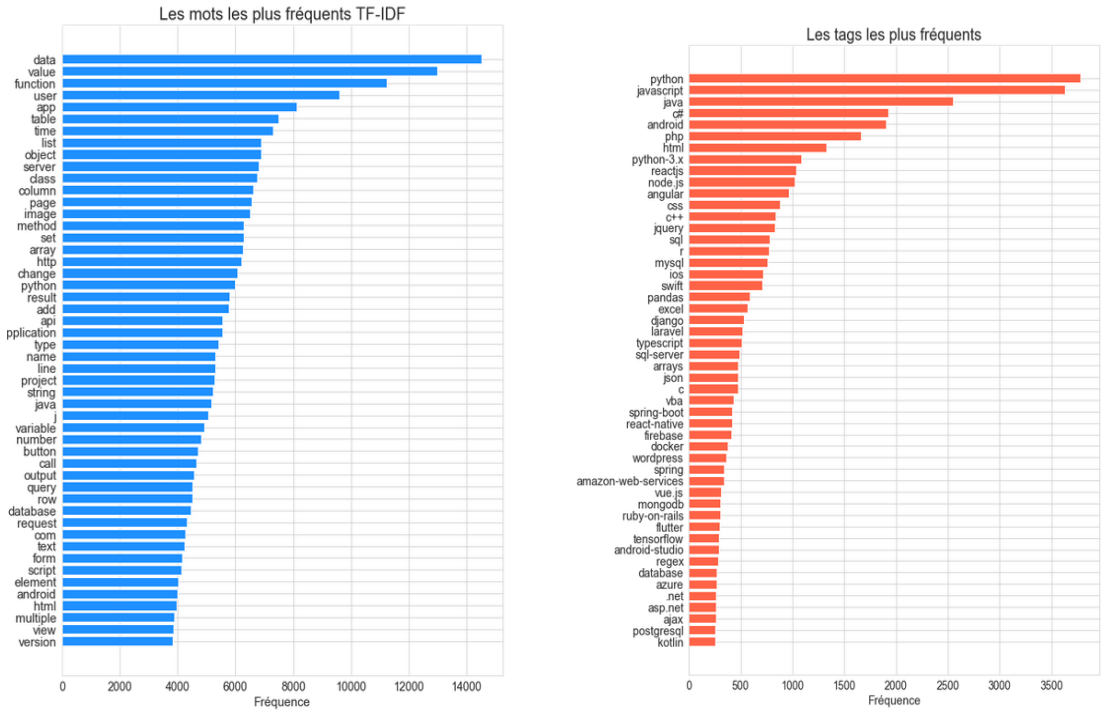
```
Entrée [327]: model1.doesnt_match("python javascript jupyter anaconda".split())

Out[327]: 'javascript'

Entrée [331]: model1.most_similar("python")

Out[331]: [('py', 0.5850811004638672),
           ('anaconda', 0.5812091827392578),
           ('miniconda', 0.5759146213531494),
           ('venv', 0.5366100072860718),
           ('pypy', 0.5239589214324951),
           ('pyenv', 0.49613383412361145),
           ('envs', 0.493344247341156),
           ('virtualenvs', 0.47054028511047363),
           ('pycharmprojects', 0.4705374836921692),
           ('pycharm', 0.4689590036869049)]
```

Analyse exploratoire

1. Les features

La Figure 4 représente les mots les plus fréquents TF-IDF après le nettoyage :

Figure 4 - Les features et cibles les plus fréquents

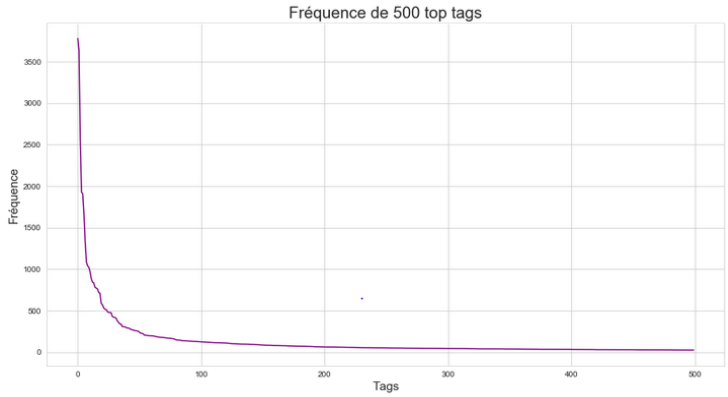


2. Cible

La cible contient de 1 jusqu'à 5 tags pour chaque question avec une moyenne de 3 tags par question. La Figure 4 représente les 50 tags les plus fréquents.

La distribution de tags est fortement aplatie vers la gauche : les tags les plus fréquents sont présents presque 4 000 fois, tandis que les tags les moins fréquents sont présent seulement 2 fois. Voici le zoom sur la distribution de 500 tags les plus fréquents :

Figure 5 - Distribution de fréquences de tags



Le défi de projet est donc de réduire le nombre de tags à un nombre raisonnable afin que nous puissions faire des prédictions bien ciblées et précises.

Modélisation non-supervisée

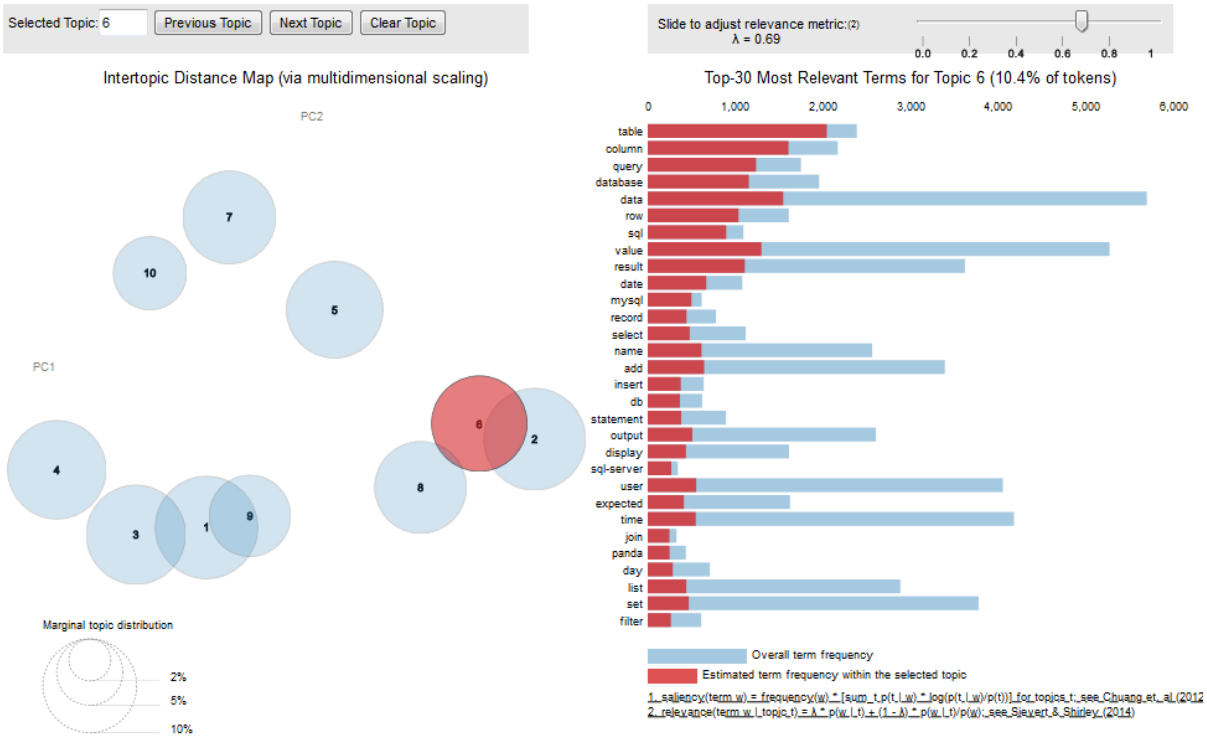
Latent Dirichlet Allocation

LDA est une méthode non-supervisée générative qui nous permets d'évaluer un nombre donné de sujets présents dans le corpus et de retourner les mots clés les plus pertinents pour chaque sujet à l'aide de distribution de mots associés à chaque thème.

L'objectif de cette méthode est de récupérer les catégories potentielles pour réduire de nombre de tags utilisé dans la modélisation supervisée. Après avoir testé plusieurs type de définition de features (titre, titre + corps, titre pondéré + corps, etc.) et après avoir testé plusieurs paramètres tels que nombre de thèmes et paramètres α et η, nous n'avons pas réussi à obtenir des thèmes précis, distincts et interprétables.

Dans l'exemple de visualisation dans la Figure 6, nous pouvons identifier le thème de document 6 qui concerne probablement le traitement d'informations dans les bases de données. Par contre, les autres sujets ne sont pas aussi précis.

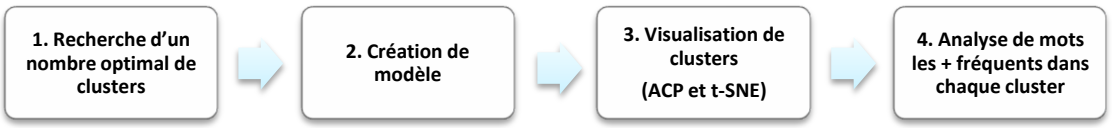
Figure 6 - Visualisation de thèmes avec LDAvis



Clustering

Deuxième méthode non-supervisée que nous avons testé afin de déterminer les sujets proches et ainsi réduire le nombre de tags est la méthode k-means. Le workflow est décrit dans la Figure 7.

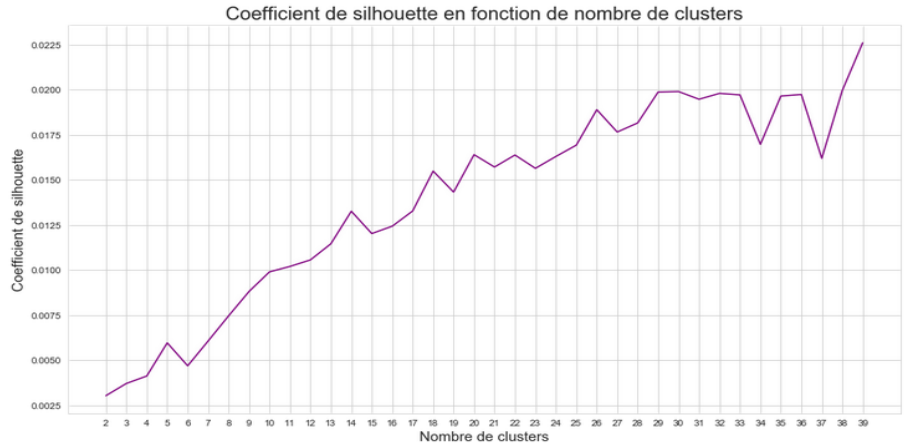
Figure 7 - Workflow clustering avec k-means



Nous avons appliqué la méthode sur plusieurs représentations de données : titre, titre + corps, titre pondéré + corps et tags. Afin d'optimiser le temps de calcul, nous avons appliqué la fonction scikit-learn qui performe l'algorithme en batch : MiniBatchKMeans.

Le nombre optimal de clusters était recherché avec le coefficient de silhouette, représenté dans la Figure 8.

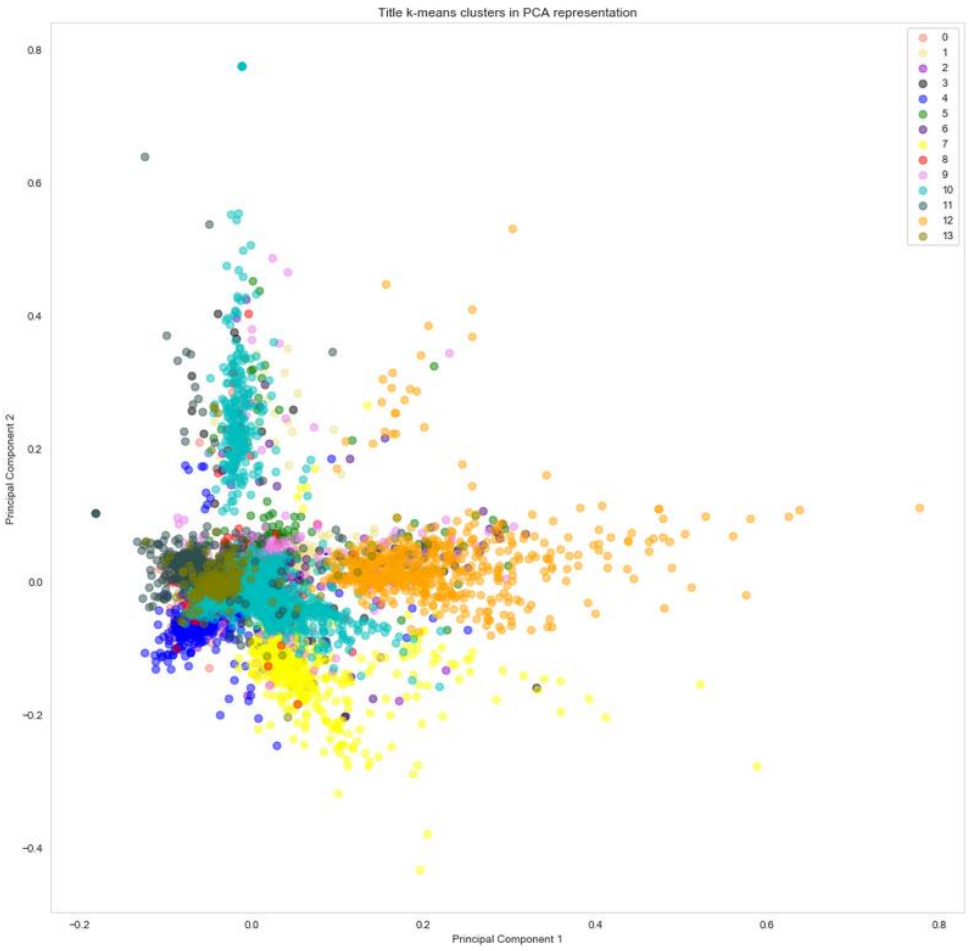
Figure 8 - Coefficient de silhouette en fonction de nombre de clusters



Nous constatons que le coefficient de silhouette est faible et augmente avec le nombre de clusters. Afin d'avoir un nombre de clusters raisonnable et simple à interpréter, nous avons décidé d'essayer de travailler avec 14 clusters (nombre représentant un pic sur le graphique).

Les clusters créés étaient visualisés avec APC et t-SNE. La Figure 9 représente le nuage de points avec les clusters en différents couleurs. Certains clusters sont homogènes, certains sont plus « éparpillés ». En tout cas, après avoir analysé les mots clés les plus fréquents pour chaque cluster, nous ne sommes pas capables d'identifier des clusters assez précis pour reformuler les sujets de questions pour la réduction de nombre de tags.

Figure 9 - Clusters visualisés avec ACP



Approche basé sur les fréquences

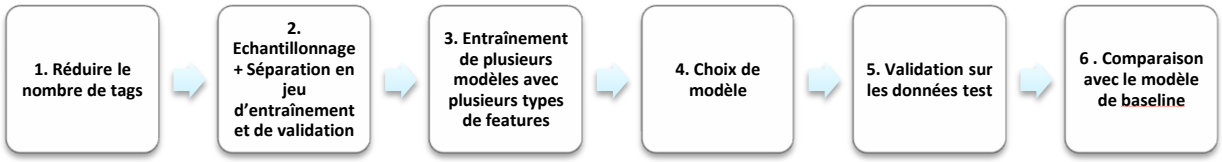
Une autre méthode non-supervisée appliquée aux données est une méthode inspirée par une approche non-supervisée proposée par Ha-Cohen-Kerner en 2003. Le principe de la méthode consiste en création d'un dictionnaire de tags et d'un dictionnaire d'expressions de question composée par titre, corps et leurs bigrams. Ensuite nous comparons les expressions communes aux dictionnaires. Les tags prédits sont les expressions communes avec les fréquences TF-IDF les plus élevées de la question. Nous avons ainsi déterminé 3 tags pour chaque question.

Les avantages de la méthode sont sa simplicité, rapidité de calcul et la capacité de prédire au moins un tag dans 99,99 % de cas. Le principal inconvénient est que la méthode ne permet pas d'estimer une expression qui ne figure pas parmi les mots édités par l'utilisateur.

Le modèle sera utilisé comme modèle de « baseline » pour comparer avec la modélisation supervisée.

Modélisation supervisée

Figure 10 - Workflow de modélisation supervisée



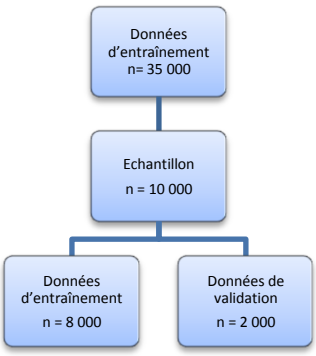
1. Réduction de nombre de tags

N'ayant pas trouvé une méthode appropriée parmi les méthodes de modélisation non supervisées, nous avons décidé de réduire le nombre de tags par suppression de ceux dont l'effectif est inférieur à 0.1 %. Cela nous a permis de garder seulement 130 tags les plus importants. En même temps, uniquement 15% de questions s'est retrouvé sans aucun tag après la suppression de tags peu fréquents.

2. Données utilisées pour choisir le modèle

Le dataflow pour choisir le modèle est représenté sur la Figure 11. Tout d'abord, afin de faciliter les calculs assez chronophages, nous avons créé un échantillon de 10 000 questions à partir de données d'entraînement. Ensuite, nous avons gardé 80% d'échantillon en tant que données d'entraînement et 20% en tant que données de validation.

Figure 11 - Data flow - choix de modèle



3. Entraînement de plusieurs modèles avec plusieurs types de features

Nous avons testé les modèles de classification multilabel suivants :

- Régression logistique
- KNN
- Arbres de décision
- Forêts aléatoires

La cible était une matrice binaire créée à l'aide de bag of words avec les 130 top tags et notre matrice de features était créée avec le texte qui représente un titre pondéré (3x) et le corps de texte. Nous avons essayé d'appliquer plusieurs transformations de features : bag of words, TF-IDF et deux modèles de word2vect, ce qui nous a permis de tester 16 modèles différents.

4. Choix de modèle

Afin de choisir le modèle le plus performant, nous avons utilisé le score F1 micro avg. Le calcul de ce score est basé sur le nombre de vrai positifs, faux négatifs et faux positifs global. Le F1 est une moyenne harmonique de précision et de recall. Il prend donc en considération et les tags prédits par erreur et les tags qui ne sont pas prédits par erreur.

Le meilleur modèle est régression logistique (OneVsRestClassifier avec linearSVC) avec un score F1 = 0.47, Précision = 0.75 et recall = 0.35.

5. Validation sur les données test

Nous avons validé le modèle sur le jeu de données de test que nous avons mis de côté dès le début de codage. Il s'agit de 15 000 question / 50 000. Afin de valider le modèle, il faut porter une attention particulière à l'ajustement de modèle TF-IDF. En effet, pour obtenir la même transformation de données d'entraînement et de données test, nous étions obligés de refaire la transformation en supprimant le nombre maximal de features. Nous avons donc obtenu une matrice de données d'entraînement avec 35 000 lignes et 54k colonnes.

Le résultat de validation sur le données test obtenu est le score de F1 = 0.56, Précision = 0.76 et recall = 0.44.

6. Comparaison avec le modèle de baseline

Afin de comparer les deux modèles, nous avons choisi des critères simples et pratiques en dehors de score habituel tels que F1 et précision qui nous donnent le score global, mais ne sont pas très parlant pour évaluer les prédictions question par question. Dans un premier temps, nous avons regardé le temps d'exécution d'algorithme à partir de moment où l'internaute pose la question et le moment où notre modèle prédit les tags. Ensuite nous avons calculé le nombre de questions sans aucun tag prédit, ainsi que nombre de questions avec au moins 1 prédiction correcte. Le Tableau 1 compare les deux modèles.

Tableau 1 - Comparaison modèle de baseline et modèle supervisé

	Critère	Modèle de baseline	Modèle supervisé
I.	Temps d'exécution	0.15 sec	0.06 sec
II.	Nombre de questions avec au moins 1 prédiction	99.99 %	59.46 %
III.	Nombre de questions avec au moins 1 prédiction correcte	36.89 %	50.41 %

7. Démonstration de fonctionnement

La Figure 12 représente les résultats de prédiction de deux modèles pour une question test aléatoire. Le modèle de baseline prédit 3 tags dans 0.15 seconds et le modèle supervisé 1 tag dans 0.06 seconds. Les deux résultats sont cohérents avec la question. La différence est que le modèle supervisé prédit un mot clé qui ne figure pas dans le texte de la question. Pourtant, il s'agit bien de sujet abordé par internaute.

Figure 12 - Démonstration fonctionnement modèle de baseline et modèle supervisé

Prédiction de tags pour une question test aléatoire :

Titre : IPV4 traffic not working with AWS egress only internet gateway

Corps : I have assigned an egress only internet gateway to my private subnet. Now I can connect with IPV6 websites bu
t not with IPV4 addresses.

Do I need NAT gateways to access IPV4 address from my EC2 machine? (Only outgoing traffic)

Tags prédits modèle de baseline : private
access
gateway
--- 0.15600037574768066 seconds ---

Tags prédits modèle supervisé : amazon-web-services
--- 0.06240034103393555 seconds ---

Conclusion

Les deux modèles sont assez performants au niveau de temps de calcul. Le modèle de baseline propose à utilisateur au moins un tag dans 99,99 % de cas, tandis que le modèle supervisé propose des tags seulement dans environ 60 % de cas. Par contre, le modèle supervisé est plus précis : dans 50 % de questions au moins un tag proposé est correct, tandis que le modèle de baseline donne une réponse correcte seulement dans environ 37 % de cas.

Le grand désavantage de notre modèle final est donc absence d'une prédiction positive dans 40% de cas, ce qui peut être gênant pour les utilisateurs qui s'attend à une aide automatique. Pour améliorer le modèle, nous pouvons envisager soit de baisser le seuil de prédiction positive, soit de créer un système de prédiction qui prend en compte une combinaison de tags sorties par les deux modèles. Le modèle supervisé va prédire les tags avec une meilleure précision, tandis que le modèle de baseline va « combler des trous » au cas où nous n'obtenons pas de prédiction de premier modèle.

Ressources

https://www.kaggle.com/c/word2vec-nlp-tutorial#part-1-for-beginners-bag-of-words
http://www.nltk.org/book/ch01.html
https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
https://towardsdatascience.com/using-word2vec-to-analyze-news-headlines-and-predict-article-success-cdeda5f14751
https://www.kaggle.com/jbencina/clustering-documents-with-tfidf-and-kmeans
https://www.kdnuggets.com/2019/09/overview-topics-extraction-python-latent-dirichlet-allocation.html
https://medium.com/datadriveninvestor/predicting-tags-for-the-questions-in-stack-overflow-29438367261e
https://towardsdatascience.com/improving-the-stack-overflow-search-algorithm-using-semantic-search-and-nlp-a23e20091d4c

Le code de projet, les documents, ainsi que le code final d'API est accessible ici :

https://github.com/Lenka-St/Projet5