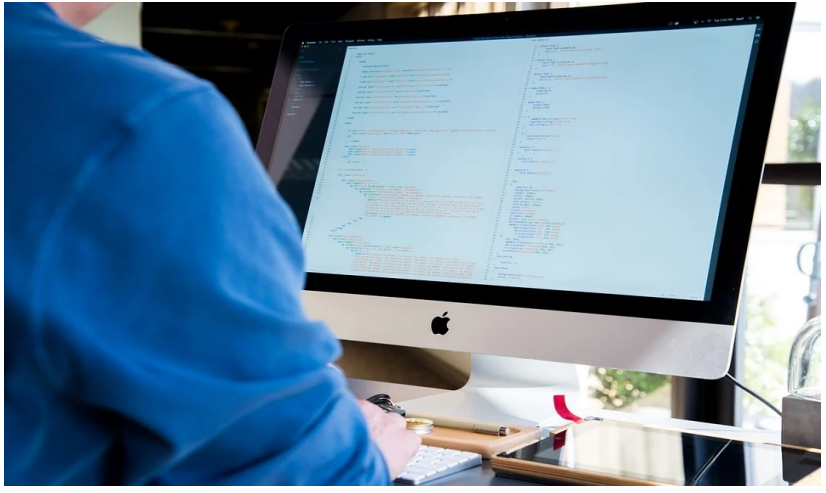
Projet de Data Science (Openclassrooms PJ6)
Catégoriser automatiquement des questions

12/08/2020

François BOYER

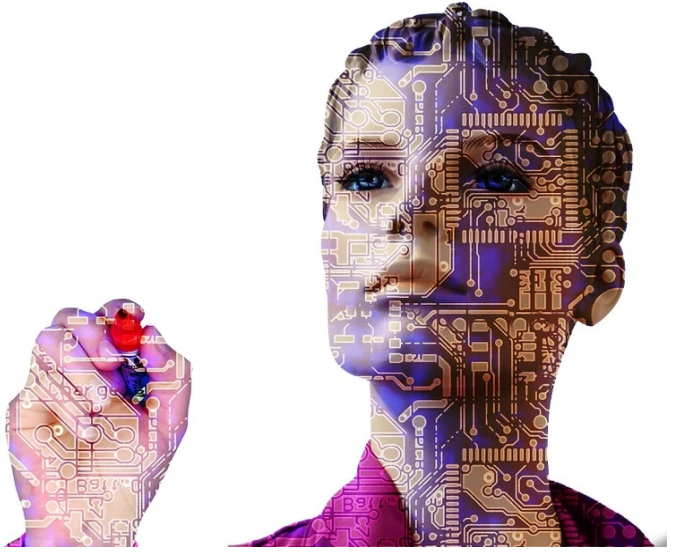
L’enjeu métier

Proposer au site StackOverflow (un forum d’aide pour les développeurs)
un modèle de suggestion automatique de tags



« Hi, please help me solve this problem on my code [….] »

⇨

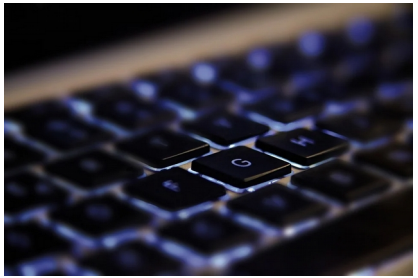


Suggested tags for your post :
#Python
#Data

Les problématiques Data Science

1/ L’analyse du texte en machine learning

⇒ Comment obtenir des features pertinentes ?



⇒ Et en plus, sur une problématique métier spécifique avec son propre vocabulaire ?

2/ Ensuite, choix de l’approche de suggestion de tags

⇒ Supervisé / Non supervisé ?

⇒ Classification en multi label

Démarche suivie

- Formulation de l’objectif
- Métriques de mesure
- Récupération des données
- Exploration / nettoyage des données
- Modélisation des documents (extraction des features : embedding)
- Test de différents modèles prédictifs
- Sélection du meilleur modèle et GridSearch
- Optim. precision / rappel
- Réalisation d’une API / Interface utilisateur

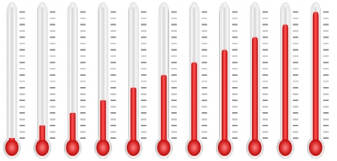
Formulation de l’objectif

Un problème de classification supervisée car nous allons utiliser les tags déjà assignés aux données d’apprentissage (labels)

⇒ Toutefois nous expérimenterons aussi une technique totalement non supervisée et en verrons la limite

Une classification multi label car chaque question peut être associée à 1 ou plusieurs tags

Métriques de mesure



Choix de privilégier la précision par rapport au rappel

⇒ Priorité à la qualité des suggestions vs la quantité des tags suggérés.

⇒ **Métrique :** precision micro

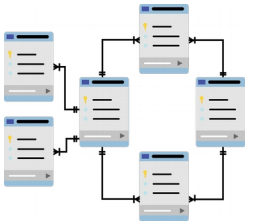
Choix de la métrique Precision micro plutôt que macro, car la precision macro accorde une importance égale à tous les labels (y compris les minoritaires ou les plus difficiles à prédire), alors que la precision micro accorde une importance égale à chaque prédiction quel que soit son label.

On souhaite que le modèle suggère au moins 1 tag par question dans 80 % des cas.

⇒ **Métrique :** % de documents ayant au moins 1 tag suggéré par le modèle

Récupération des données

Requêtes SQL sur le site : StackExchange Data Explorer



Choix réalisés :

⇒ Récupération de 300 000 posts

⇒ Récupération des données entre le 01/01/2019 et le 28/02/2019

On a été vigilants de façon à filtrer les requêtes pour récupérer uniquement les questions, et pas les réponses, conformément à l'objectif du projet

Test d’une approche totalement non supervisée avec LDA (Latent Dirichlet Allocation)

Sans utiliser les tags fournis dans le jeu de donnée, on a **inféré des topics pour les documents**

Topic #0: state let view title return

Topic #1: com google example https app

Topic #2: data 10 using like code

Topic #3: 2019 01 00 10 12

Topic #4: org java apache version core

Topic #5: input array form field type

Topic #6: http html content server application

Topic #7: lib py local file line

Topic #8: java exception method main run

Topic #9: log json console event result

Topic #10: php https url com link

etc.... (50 topics)

⇒ Vu la nature très technique du sujet des posts, **les tags obtenus via cette méthode ne sont pas très pertinents. Ils portent toutefois un certain sens**, mais moins précis que les vrais tags.

⇒ On ne retiendra pas cette approche totalement non supervisée

Nettoyage des données

Document (post d’origine)

⇩

Conservation du titre, corps du texte, tags

⇩

Suppression textes vides

⇩

Suppression tags HTML stackoverflow

⇩

Encodage des tags : 1 colonne / tag

Exploration des données

300 000 documents

1.6 Millions de tokens différents

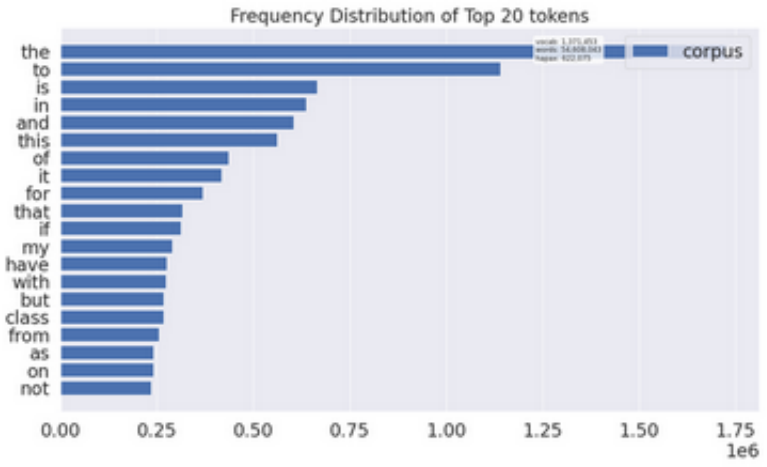
⇒ Le vocabulaire est très élevé si on le compare aux 170 000 mots couramment utilisés en langue anglaise

⇒ Cela s’explique par la nature technique du vocabulaire

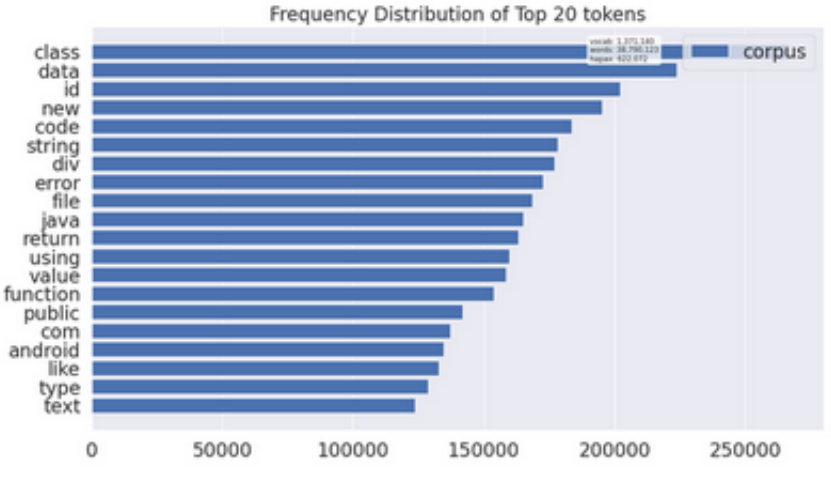
26 431 tags distincts

Choix des tokens à utiliser pour le texte

Avant supp. stop words

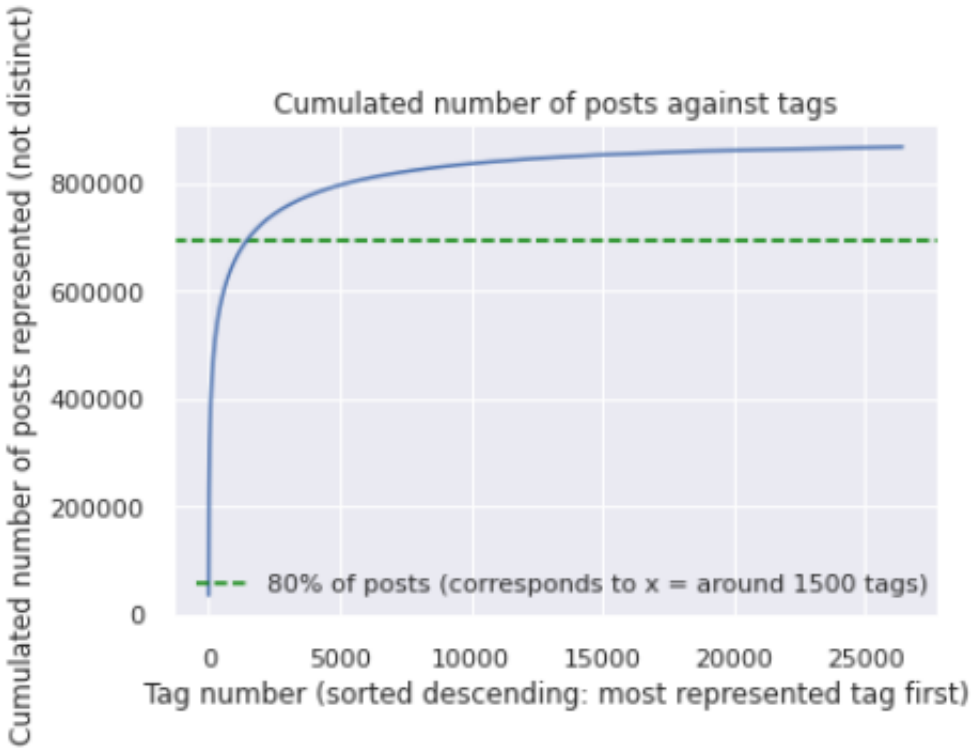


Après supp. stop words



Choix des tags à utiliser (Diapo. 1 / 2)

Choix de ne conserver que les tags les plus représentés dans le dataset.
⇒ Une première fourchette haute du nombre de tags à conserver :



⇒ 80% des occurrences de post (posts non distincts, un tag apparaissant dans plusieurs posts) sont couverts par 1500 tags

Choix des tags à utiliser (Diapo. 2 / 2)

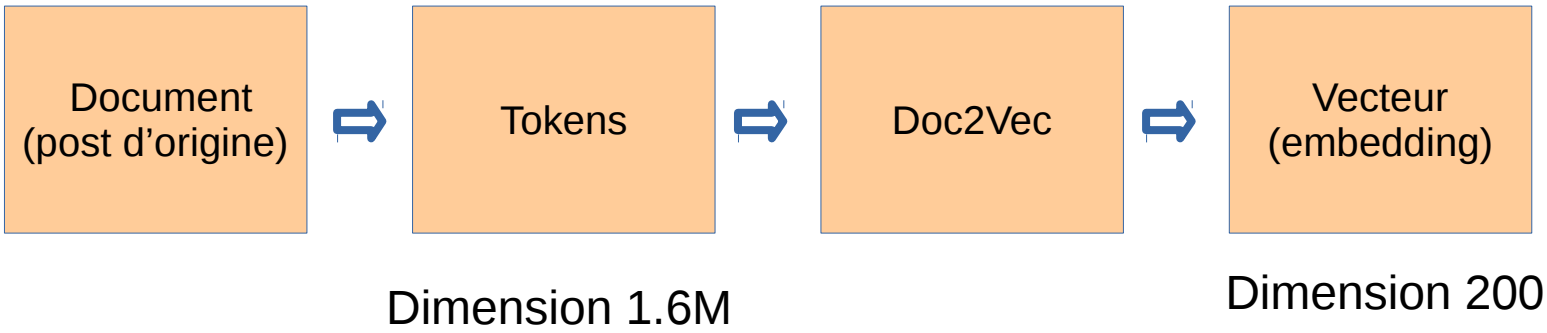
Pour des raisons de performance, ajout d'une limite supplémentaire:
⇒ Uniquement les tags présents dans plus de 0.1% des documents.

⇒ **373 tags** retenus pour la suite

⇒ Avec ces 373 tags, 92 % des posts ont au moins 1 tag à vrai

NB : le fait de monter à 1000 tags aurait augmenté la couverture à 95% des posts distincts.

Modélisation des documents avec doc2vec (embedding)



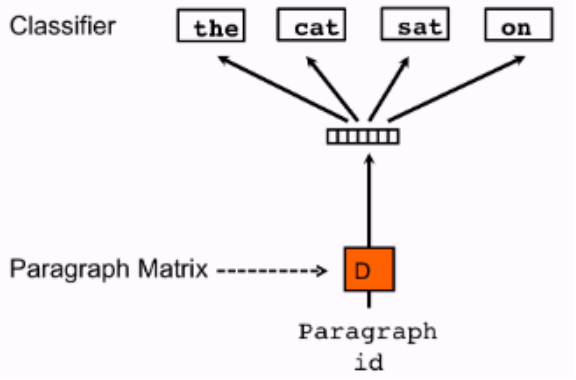
Implémentation de l'algorithme d'apprentissage non supervisé doc2vec pour convertir chaque vecteur de document en un nouveau vecteur de dimensions plus réduite

⇒ fondé sur le principe que les mots apparaissant dans des contextes similaires ont des significations apparentées

Mise en évidence par un exemple de la pertinence de l'embedding (voir le rapport)

Doc2Vec : explication synthétique du fonctionnement

Le modèle PV-DBOW est un réseau de neurones à 3 couches :



A chaque itération d'apprentissage, à partir d'un mot pris au hasard dans le document, on va prédire un ensemble de mots qui lui sont proches

Et c'est ensuite la couche du milieu qui est utilisée pour former l'embedding du modèle de notre projet décrit dans ce rapport .

Test de différents algorithmes avec l’embedding en input

Les étapes du modèle

Encodage « doc2vec »

Standard scaling (moyenne 0, écart type 1)

Prédicteur : KNN ou DecisionTree (CART) ou Perceptron

Niveau d’apprentissage vs quantité de données

Entraînement de plusieurs itérations d’un prédicteur KNN sur un nombre d’instances de plus en plus grand

Ci-dessous l’itération 0 (en abscisse) correspond à 20 000 documents dans le training set, et l’itération 4 correspond à 100 000 documents

⇒ Nous avons retenu 90 000 instances pour les tests de modèle qui ont suivi

Algorithme KNN (approche supervisée)

Entraînement d' un algorithme K Nearest Neighbours (avec k = 5)

	Precision micro	Exact match (accuracy)	% docs avec au moins 1 tag prédit	Recall
Training set	84%	16%	13 %	17%
Test set	62%	11%	13 %	10%

Le modèle prédit de bons résultats sur la métrique Precision micro qu'on avait retenue

En revanche : le modèle réalise trop peu de prédictions (13%)

Analyse des résultats du modèle

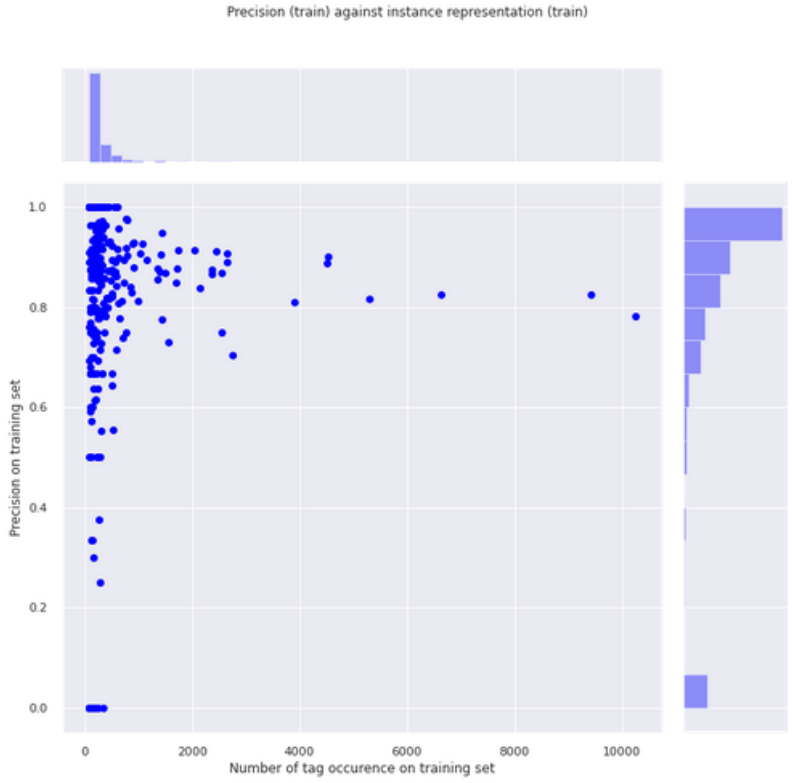
On a cherché à répondre aux questions suivantes :

⇒ Le modèle est-il plus performant pour prédire les tags qui sont les plus représentés dans le training set (en nombre d’occurences) ?

⇒ Est-ce que le modèle généralise bien ou est-il en overtfit ?

Comparaison :

precision vs
nombre d’occurence des tags dans les labels du training set



⇒ **Au delà d’environ 500 instances par tag, le precision score ne descend plus en dessous de 0.7** : il y a donc un lien entre les performances et le nombre de tags représentés dans le training set, mais ce lien est très faible, car la grande majorité des instances du training set sont déjà à moins de 500 instances et plus de 0.7 de precision.

Comparaison :

precision vs
nombre d’occurence des tags dans les labels du test set

⇒ **On constate un overfit** avec de nombreux tags ayant une précision à 0 : 213 sur le test set vs 33 sur le training set.

Cette precision à 0 signifie soit un precision score à 0 proprement dit, ou bien le fait qu’aucun de ces tags n’a été prédit à vrai

On peut faire l’hypothèse que l’augmentation du nombre de voisins de l’algorithme KNN, actuellement à 5, permettra d’atténuer ce problème.

Algorithme CART (approche supervisée / arbre de décision)

Entraînement d' un algorithme de type arbre de décision

	Precision micro	Exact match (accuracy)	% docs avec au moins 1 tag prédit	Recall
Training set	18%	12%	Non mesuré	19%
Test set	7%	Non mesuré	Non mesuré	2%

Les résultats obtenus avec l'approche arbre de décision sont nettement moins bons qu'avec l'approche KNN.

Ce point n'a pas été analysé davantage dans le cadre de ce projet, néanmoins, le rapport détaillé contient une tentative d'explication théorique de cette faible performance

Algorithme Perceptron (approche supervisée)

Entraînement d’ un algorithme de type réseau de neuronnes (1 couche)

	Precision micro	Exact match (accuracy)	% docs avec au moins 1 tag prédit	Recall
Training set	28%	Non mesuré	Non mesuré	Non mesuré
Test set	Non mesuré	Non mesuré	Non mesuré	Non mesuré

Le Perceptron 1 couche, dans son implémentation par défaut avec scikit learn, n’a pas fourni non plus de très bons résultats comparé à KNN.

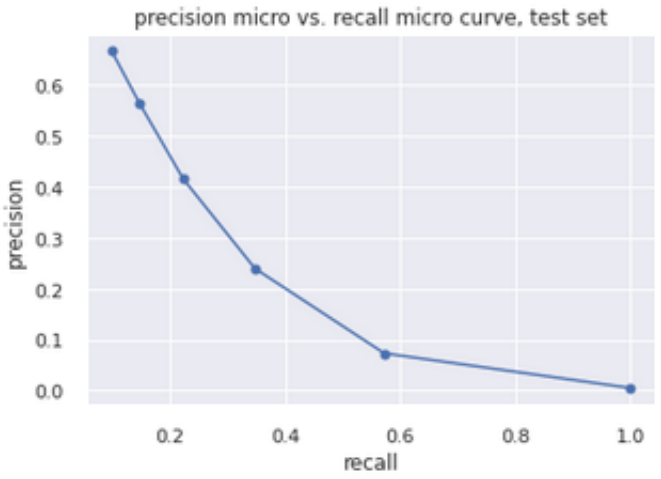
Sélection du meilleur modèle et optim. des paramètres

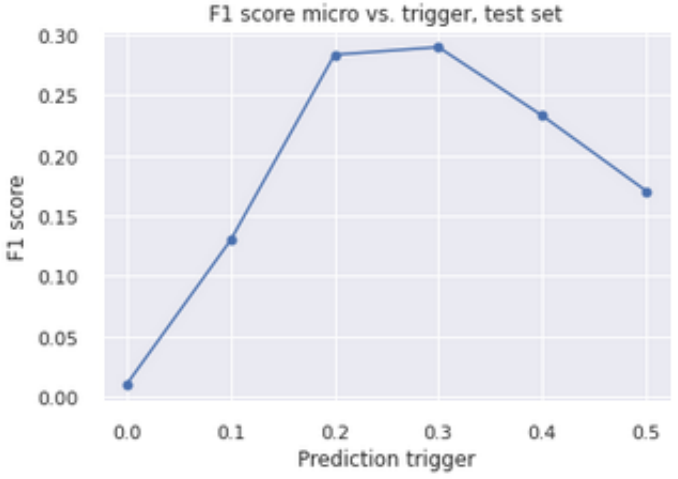
doc2vec : n dim	knn : n neighbors	mean test score
200	10	78%
200	5	62%
10	10	56%
10	5	40%

⇒ Résultats du meilleur modèle parmi les 4 ci-dessus, avec 200 dimensions doc2vec et 10 voisins :

	Precision micro	Exact match (accuracy)	**% docs avec au moins 1 tag prédit**	Recall
Training set	87%	14%	Non mesuré	12%
Test set	77%	11%	13%	6%

Compromis precision / rappel : optimisation du seuil de probabilité pour les prédictions (1/2)





⇒ L'objectif qu'on s'était fixés au départ étant de privilégier la précision au rappel :
Choix d'un seuil à 0.3 => precision ~ 0.4 et recall ~ 0.2

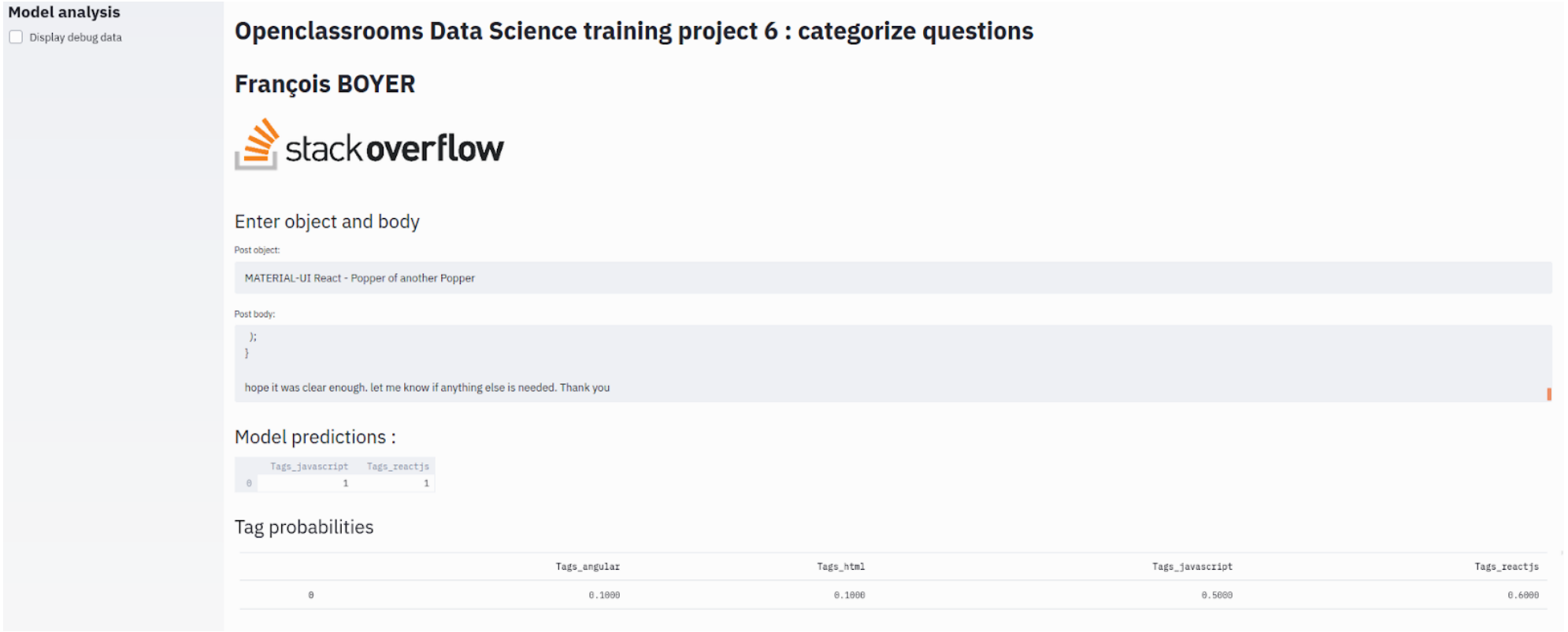
Compromis precision / rappel : optimisation du seuil de probabilité pour les prédictions (2/2)

	Precision micro	Exact match (accuracy)	**% docs avec au moins 1 tag prédit**	Recall
Training set	60%	13%	Non mesuré	34%
Test set	41%	11%	64%	22%

⇒ Le recall et le % de docs avec au moins 1 tag de prédit sont nettement améliorés
⇒ C’est donc ce modèle que nous retiendrons pour l’interface utilisateur finale

Réalisation d’une API avec interface utilisateur

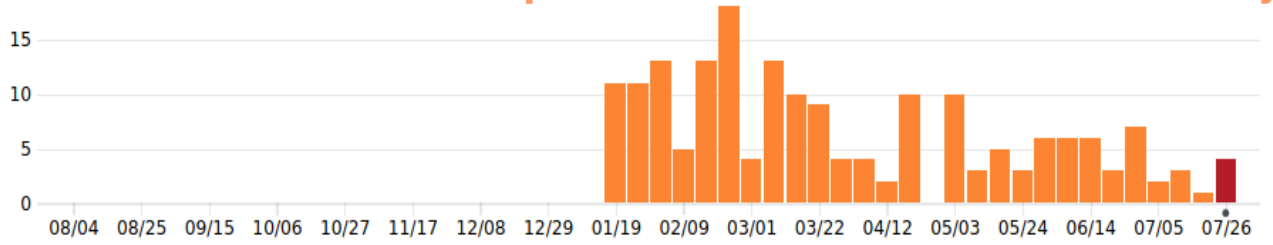
https://pj6.analysons.com/



Commits successifs du projet (et de l'API) sous GIT

Accès Github : https://github.com/Severac/openclassrooms

⇒ **Commits réalisés depuis le début de la formation fin janvier :**



⇒ **Commits réalisés sur l'API :**

Commits on Jul 18, 2020

PJ6 API : initialisation
Severac committed 14 days ago — `10c446c`

PJ6 GridSearch avancée : fonction scoring precision micro, et prepara...
Severac committed 15 days ago — `d600479`

Commits on Jul 28, 2020

PJ6: update mineure UI
Severac committed 4 days ago — `b0139f9`

Perspectives

⇒ Trouver une formalisation plus précise du choix du dernier modèle (avec le trigger de probabilité) : par exemple : utilisation du score F beta avec une pondération attribuée au score de precision qui soit plus importante que le recall. Et relancer une validation croisée sur cette base.

⇒ Analyser davantage les différents cas pour lesquels le modèle fait des erreurs : par exemple, analyser un échantillon de cas avec precision à 0 afin d'identifier des améliorations possibles

⇒ Analyser plus précisément les résultats d'autres modèles comme le Decision Tree

⇒ Pour la validation croisée : faire un stratified split fondé sur une répartition uniforme des différentes features : via un clustering fondé sur les features du doc2vec par exemple, en faisant le stratified split sur la base des clusters ainsi obtenus.

Fin de la présentation