

Catégorisation automatique de questions stackoverflow Rapport



Sommaire

Cleaning effectué
Exploration des tags
Préparation du texte
Modèle non supervisé
Modèle supervisé
Evaluation globale
API
Conclusions
Axes d'améliorations

résumé

Résumé : Projet 5 IML

Ce rapport présente mon travail effectué pour le projet 5 Openclassrooms du parcours Ingénieur Machine Learning.

Ce projet consiste à proposer une API de prédiction de tags (mots-clés) en relation avec les questions posées par les utilisateurs sur le forum web StackOverFlow.

Ce document présente les différentes approches supervisées et non-supervisées, les traitements du texte tel que le «stemming», les modèles choisis et leur optimisation.

Aussi, une évaluation finale sera faite pour choisir le meilleur modèle.

Enfin, un modèle adapté est proposé pour tester une API de prédiction.

Cette API est opérationnelle pour le test : <http://ismail2233.pythonanywhere.com/>

Rappel du contexte

Besoin identifié

Stack Overflow est un site communautaire de questions-réponses. Le support adresse une cible de profils techniques du secteur informatique (développeurs, devops, datascientists...). Le site, lancé en 2008, revendique une audience de 100 millions de visiteurs par mois et en moyenne 4 questions posées toutes les minutes (source : Stack Overflow advertising).

Un utilisateur souhaitant poser une question passe par un formulaire où il doit renseigner :

- Un titre
- Un corps de texte
- Entre un et cinq tags servant à catégoriser la question

Le choix des tags peut s'avérer délicat pour un non initié à la plateforme. Afin d'y remédier, nous souhaitons proposer un outil de suggestion de tags à destination des utilisateurs.

Solution préconisée

L'outil se basera sur le corps de texte fourni afin de suggérer un ensemble de tags. Il sera propulsé par d'un modèle de machine learning. Deux approches, supervisée et non supervisée, seront étudiées au cours des travaux. La solution sera mise à disposition sous forme d'API mise en production.

Interprétation

Au lieu que l'utilisateur trouve seul des tags, on cherche à l'aider en lui proposant automatiquement des tags en relation avec sa question.

Le but est donc de trouver un moyen de détecter, à partir du texte saisi, les mots clés qui pourraient s'y rattacher.

Pistes envisagées

Deux principales approches sont possibles.

La première approche est celle qui utilise un modèle non-supervisé, qui cherche les principaux sujets « topics », inconnus à l'avance, que peuvent traiter des sous-ensembles de questions déjà posées.

Ensuite, grâce aux questions déjà « taguées », on peut associer leurs tags avec les topics dominants des questions.

La deuxième approche est l'utilisation d'un modèle supervisée multi-class.

Cela implique d'avoir un nombre fini de tags prévisibles.

Ensuite, l'évaluation des deux approches permettra de faire un choix pour implémenter une API de test qui pourrait prédire des tags à partir d'un texte de question quelconque.

Récupération des données

```
1 DECLARE @start_date DATE
2 DECLARE @end_date DATE
3 SET @start_date = '2012-03-22'
4 SET @end_date = DATEADD(m, 12, @start_date)
5
6 select
7     p.Id,
8     p.CreationDate,
9     p.Title,
10    p.Body,
11    p.Tags,
12    p.ViewCount,
13    p.CommentCount,
14    p.AnswerCount,
15    p.Score,
16    sum(case when VoteTypeId = 2 then 1 else 0 end) as [up] ,
17    sum(case when VoteTypeId = 3 then 1 else 0 end) as [down]
18
19 from Votes v join Posts p on v.PostId = p.Id
20 LEFT JOIN PostTypes as t ON p.PostTypeId = t.id
21 group by
22     p.Id,
23     p.CreationDate,
24     p.Title,
25     p.Body,
26     p.Tags,
27     p.ViewCount,
28     p.CommentCount,
29     p.AnswerCount,
30     p.Score,
31     v.VoteTypeId,
32     t.Name
33 HAVING v.VoteTypeId in (2,3)
34 AND p.CreationDate between @start_date and @end_date
35 AND t.Name = 'Question'
36 AND p.ViewCount > 20
37 AND p.CommentCount > 5
38 AND p.AnswerCount > 1
39 AND p.Score > 5
40 AND len(p.Tags) > 0
```

Elles portent sur la période 2011 / 2022
et uniquement sur les posts "de qualité"
ayant au minimum

1 réponse,
5 commentaires,
20 vues
et un score supérieur à 5.

116569 lignes récupérées

Nettoyage du texte

Nettoyage du texte

Ce traitement passe les textes en minuscule, ne conserve que les caractères alphabétiques et ne garde que les termes de plus de trois lettres. Le filtrage sur la taille des termes permet de retirer ceux qui sont générique liés au code (if, for ...).

Tokenisation

La tokenisation permet de transformer les textes passés en entrée en liste de termes distincts (token). Pendant le traitement les termes génériques (stop words) ne sont pas conservés dans la liste des tokens. NLTK propose des listes de stop words génériques. Nous attirons l'attention sur le fait que le corpus utilisé pour les travaux est de nature spécifique. En effet les posts contiennent souvent du code, des messages d'erreur de compilateurs / interpréteurs ou des logs alors qu'aucune liste de stop words mis à disposition ne porte sur le thème. Le risque est de laisser passer des termes génériques vis-à-vis du contexte étudié. Sans allouer un important temps de travail il s'avère difficile de construire une liste de stop words spécifique exhaustive.

Filtrage à l'aide d'un modèle de POS (Part of Speech tagging)

Lors de pré-traitement des posts nous cherchons avant tout à identifier à termes liés à des technologies utilisées. Ces dernières sont généralement des noms. Afin de mettre en œuvre le filtrage nous utilisons un modèle de POS. Ce dernier se base sur des chaînes de Markov. Leur principe est d'identifier la probabilité la plus forte de la fonction grammaticale d'un terme par rapport :

- A la fonction grammaticale du terme précédent
- A la probabilité la plus forte de l'association entre le terme et une fonction grammaticale particulière

Ce type de modèle est entraîné de manière supervisée. NLTK propose un modèle pré-entraîné. Nous attirons une nouvelle fois l'attention sur le fait que le corpus utilisé pour les travaux est de nature spécifique. Aucun corpus mis à disposition par NLTK ne couvre le domaine étudié. Le risque est que certaines technologies, dont la dénomination ne découle pas d'un nom commun, ne soient pas retenues par le filtre.

Cleanage

Les entraînements de modèles de machine learning ne peuvent être réalisés sur les textes brutes. D'une part ils contiennent des éléments qui n'ont aucune valeur sémantique (balises html, caractères non alphabétiques, mots génériques ...) et un nombre potentiel de déclinaisons élevé de certains termes. D'autre part, les modèles de machines learning nécessitent des prédicteurs et des cibles de nature numérique

```
def text_cleaner(txt, nlp):
    # Remove specific type of word
    txt = remove_specific_typeOfwords(txt, nlp)
    # Case normalization
    txt = txt.lower()
    # Remove unicode characters
    txt = txt.encode("ascii", "ignore").decode()
    # Remove English contractions
    txt = re.sub("\w+", " ", txt)
    # Remove accent
    txt = remove_accented_chars(txt)
    # Remove punctuation but not # or ++
    # txt = re.sub('[^\w\s#]', " ", txt)
    txt = re.sub('[^\w\s(#|++)]', " ", txt)
    # Remove links
    txt = re.sub(r'http*\S+', " ", txt)
    # Remove numbers
    txt = re.sub(r'\w*\d+\w*', " ", txt)
    # Remove extra spaces
    txt = re.sub("\s+", ' ', txt)
    # Tokenization with exception for C# and c++
    txt = regexp_tokenize(txt, pattern=r"\s[!.,;]", gaps=True)
    # remove # character alone after tokenization
    txt = [element for element in txt if element != '#']
    # txt = [element for element in txt if len(element) != 1]
    # # List of stop words in select language from NLTK
    # # Remove stop words
    stop_words = stopwords.words("english")
    # # Remove stop words
    txt = [word for word in txt if word not in stop_words ]
    # # Lemmatizer
    wn = nltk.WordNetLemmatizer()
    txt = [wn.lemmatize(word) for word in txt]

    return txt
```

Suppression des balises et contenu
du code

Nettoyage du texte

Filtrage POS (part of speech)

Tokenisation/lemmatisation

Stop word english

Creation des features

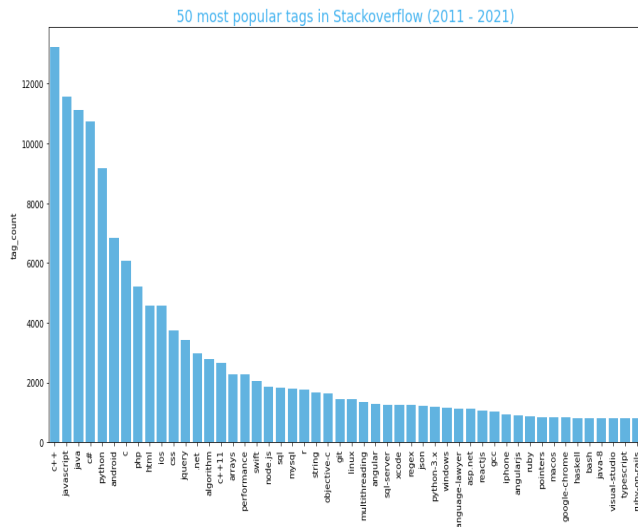
Librairies :

- BeautifulSoup
- NLTK
- Scikit learn

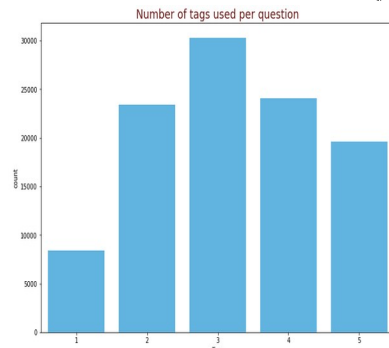
- Agrégation du « Title » et du « Body »
- Mise en minuscule
- Suppression caractères unicode (esp)
- Suppression tag de codes
- Suppression des nombres, ponctuation
-

Analyse des tags

Il y a 15698 tags différents pour les 116569 questions.
Chaque question possède de 1 à 6 tags.
Le plus souvent, les questions ont entre 2 et 4 Tags.



Sélection des 50 top tags



Prédiction multi classe
Car 3 tags en moyenne

Modèle non supervisée

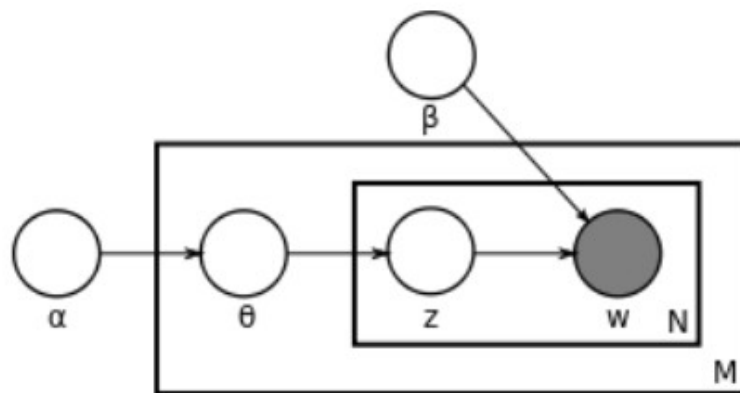
LDA (Latent Dirichlet Allocation).

Il s'agit d'un modèle génératif probabiliste. L'algorithme modélise une approche proche d'un clustering.

Il permet de regrouper les documents d'un ensemble par k topics (thèmes) et d'associer chaque mot β de chaque document à un des topics.

Chaque document est alors composé d'un mélange θ d'un petit nombre de topics.

Lors de son initialisation, chaque mot de chaque document est aléatoirement associé à un topic. L'apprentissage consiste à optimiser la probabilité qu'un topic t génère le mot w dans un document d .



α : Ensemble de tous les topics

β : Ensemble des mots de tous les documents

M : Ensemble des variables liée à un document

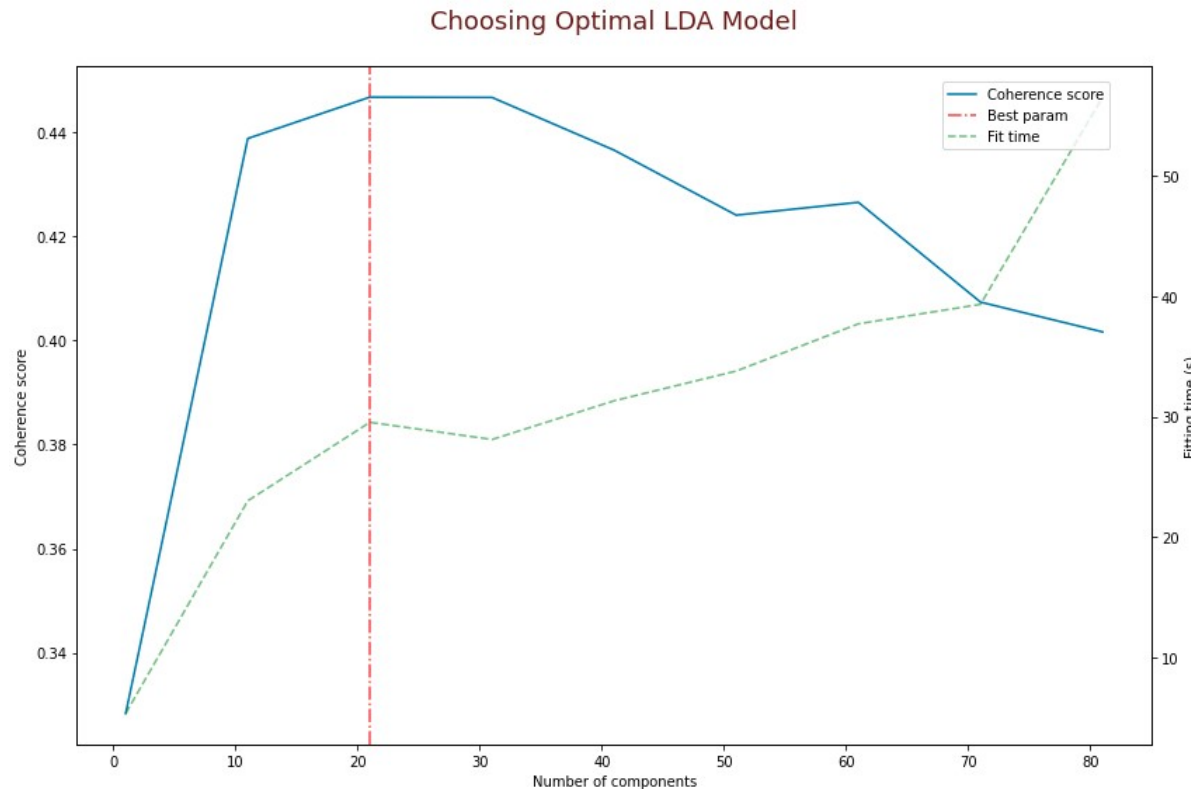
θ : Distribution d'un topic pour un document

N : Ensemble des variables liées à un mot

z : Distribution d'un topic pour un mot

w : Mot

Modèle non supervisée



Nombre de topics optimal 21

Metriques :

- Perplexité
- Coherence

les topics générés restent très généraux et ne permettent pas une catégorisation cohérente pour notre problème d'auto-tagging. Nous allons donc tester des modèles supervisés.

Vectorisation de notre corpus

Afin de pouvoir entraîner les modèles nous avons besoin de transformer les listes de tokens lemmatisés en vecteurs.

- **TF*IDF** (pour Term Frequency * Inverse Document Frequency) est le résultat d'un calcul, permettant d'obtenir un poids, une évaluation de la pertinence d'un document par rapport à un terme, en tenant compte de deux facteurs :

- 1) la fréquence de ce mot dans le document (TF)
- 2) le nombre de documents contenant ce mot (IDF) dans le corpus étudié.

- **Word2Vec** est un groupe de modèles utilisé pour le plongement lexical
Word2vec est une sorte d'ACP non linéaire en ce sens qu'il réduit les dimensions
Ce sont des réseaux de neurones artificiels à deux couches entraînés pour reconstruire le contexte linguistique des mots.

Deux architectures ont été initialement proposées pour apprendre les Word2vec,
Le modèle de sacs de mots continus (CBOW: continuous bag of words) et le modèle skip-gram.

- 1) Le CBOW vise à prédire un mot étant donné son contexte, c'est-à-dire étant donné les mots qui en sont proches dans le texte.

- 2) Le skip-gram a une architecture symétrique visant à prédire les mots du contexte étant donné un mot en entrée.

- **Doc2vec** est une extension des approches Word2Vec dans lesquelles on ajoute un "token" associé à chaque document

Métriques utilisées

L'accuracy indique le pourcentage de bonnes prédictions. C'est un très bon indicateur parce qu'il est très simple à comprendre. **L'accuracy** permet de connaître la proportion de bonnes prédictions par rapport à toutes les prédictions.

Le **F1-score** est une métrique pour évaluer la performance des modèles de classification à 2 classes ou plus. Il est particulièrement utilisé pour les problèmes utilisant des données déséquilibrées comme la détection de fraudes ou la prédiction d'incidents graves.

Le F1-score permet de résumer les valeurs de la precision et du recall en une seule métrique. Mathématiquement, le F1-score est défini comme étant la moyenne harmonique de la precision et du recall.

Le F1-Score combine subtilement la précision et le rappel. Il est plus intéressant que l'accuracy car le nombre de vrais négatifs (tn) n'est pas pris en compte

Indice de Jaccard

l'indice de Jaccard ou coefficient de Jaccard est le rapport entre la taille des termes communs de 2 documents sur la taille de l'union des 2 documents :

le rapport entre la cardinalité (la taille) de l'intersection des ensembles considérés et la cardinalité de l'union des ensembles. Il **permet d'évaluer la similarité entre les ensembles**.

Le **recall** permet de savoir le pourcentage de positifs bien prédit par notre modèle. En d'autres termes c'est le nombre de positifs bien prédit (Vrai Positif) divisé par l'ensemble des positifs (Vrai Positif + Faux Négatif).

La précision correspond au nombre de documents correctement attribués à la classe i par rapport au nombre total de documents prédits comme appartenant à la classe i (total predicted positive)

Le rappel correspond au nombre de documents correctement attribués à la classe i par rapport au nombre total de documents appartenant à la classe i (total true positive).

Modèle supervisée

Approche supervisée

Cette approche a consisté à entraîner différents modèles supervisés avec des couples de textes vectorisés / tags associés. Chaque document pouvant être associé à un à plusieurs tags, Nous avons donc à faire à un problème de classification multi-classes et multi-labels.

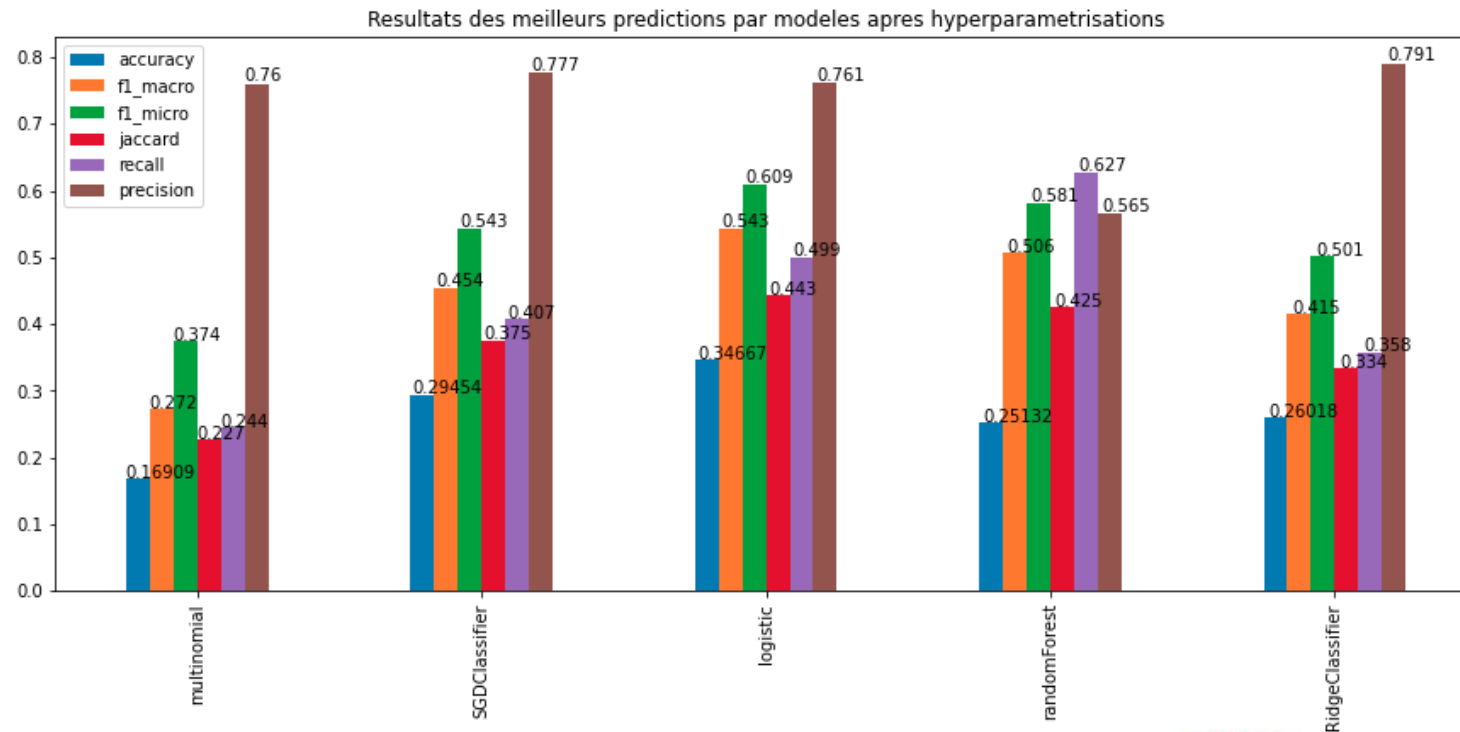
Partitionnement des données

Lors du partitionnement des données nous avons conservé 80% des données pour le jeu D'entraînement et 20% pour le jeu de test.

Nous avons testé cinq modèles supervisés :

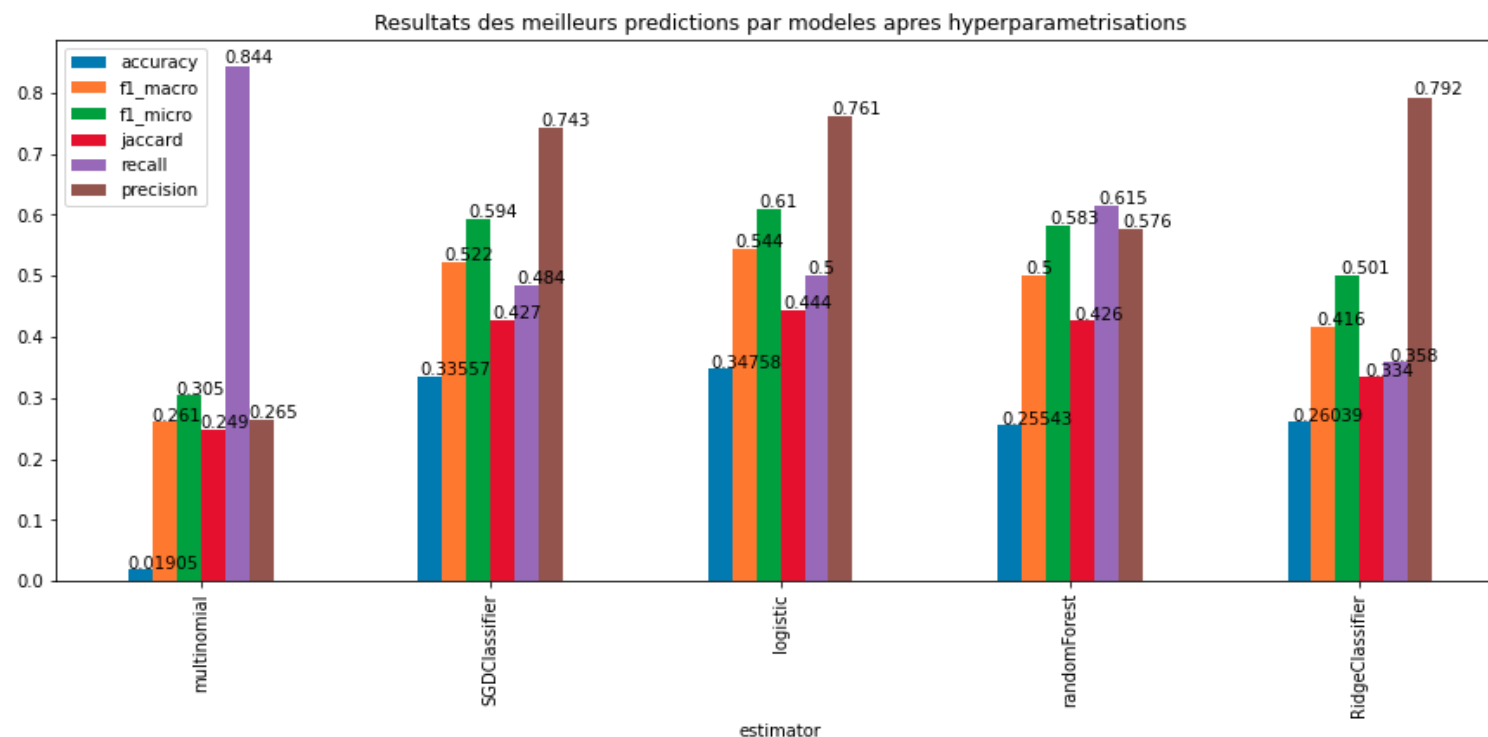
- multinomialNB
- SGDClassifier
- LogisticRegression
- randomForest
- RidgeClassifier

Approche supervisée tf-idf gridsearch oneVsRest



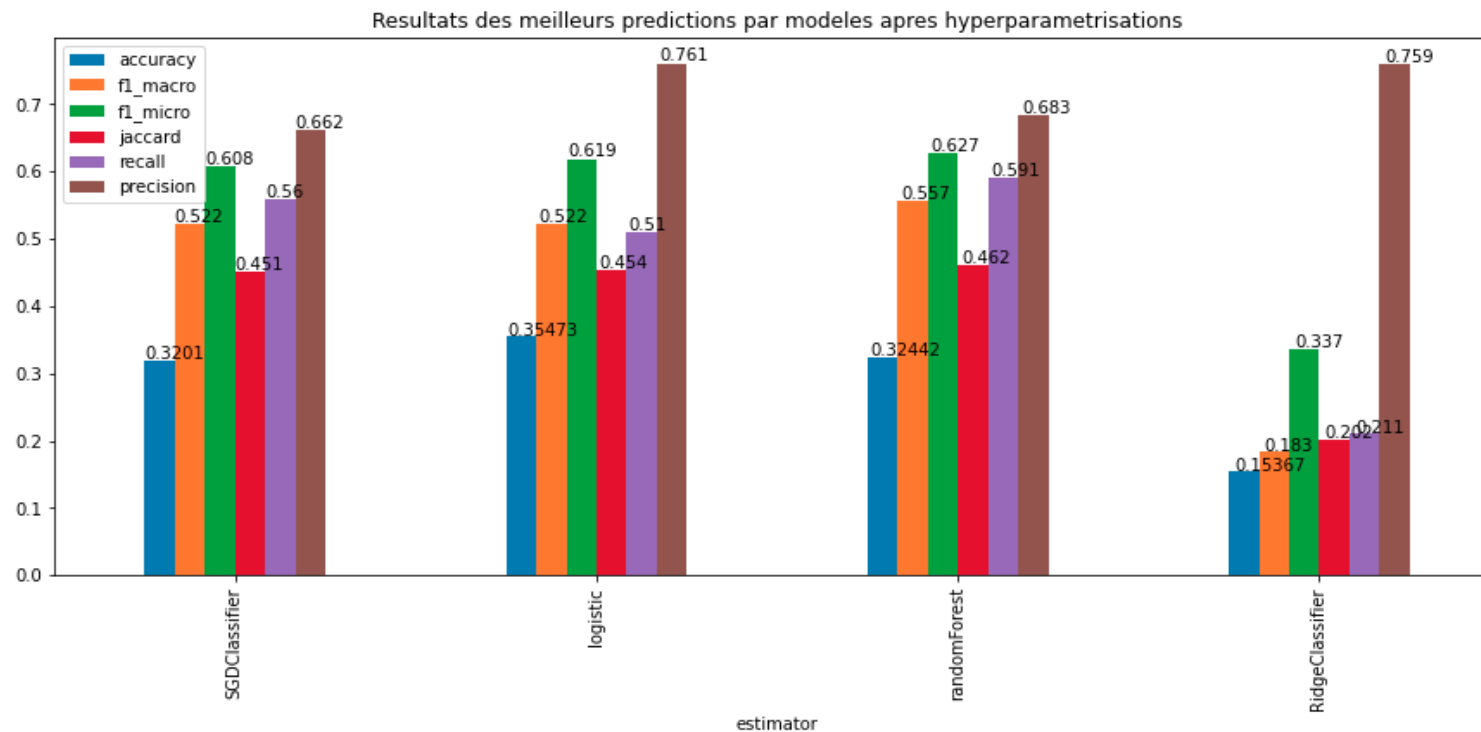
estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
multinomial	0.16909	0.272	0.374	0.227	0.244	0.760	63.66%
SGDClassifier	0.29454	0.454	0.543	0.375	0.407	0.777	40.29%
logistic	0.34667	0.543	0.609	0.443	0.499	0.761	28.45%
randomForest	0.25132	0.506	0.581	0.425	0.627	0.565	12.36%
RidgeClassifier	0.26018	0.415	0.501	0.334	0.358	0.791	47.0%

Approche supervisée tfidf avec score gridsearch



estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
multinomial	0.01905	0.261	0.305	0.249	0.844	0.265	0.21%
SGDClassifier	0.33557	0.522	0.594	0.427	0.484	0.743	28.58%
logistic	0.34758	0.544	0.610	0.444	0.500	0.761	28.33%
randomForest	0.25543	0.500	0.583	0.426	0.615	0.576	13.32%
RidgeClassifier	0.26039	0.416	0.501	0.334	0.358	0.792	46.97%

Approche supervisée word2vec gridsearch



estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
SGDClassifier	0.32010	0.522	0.608	0.451	0.560	0.662	20.71%
logistic	0.35473	0.522	0.619	0.454	0.510	0.761	29.13%
randomForest	0.32442	0.557	0.627	0.462	0.591	0.683	21.96%
RidgeClassifier	0.15367	0.183	0.337	0.202	0.211	0.759	68.42%

Sélection de nos modèles

Au vu des différents tests effectués, nous allons sélectionner :

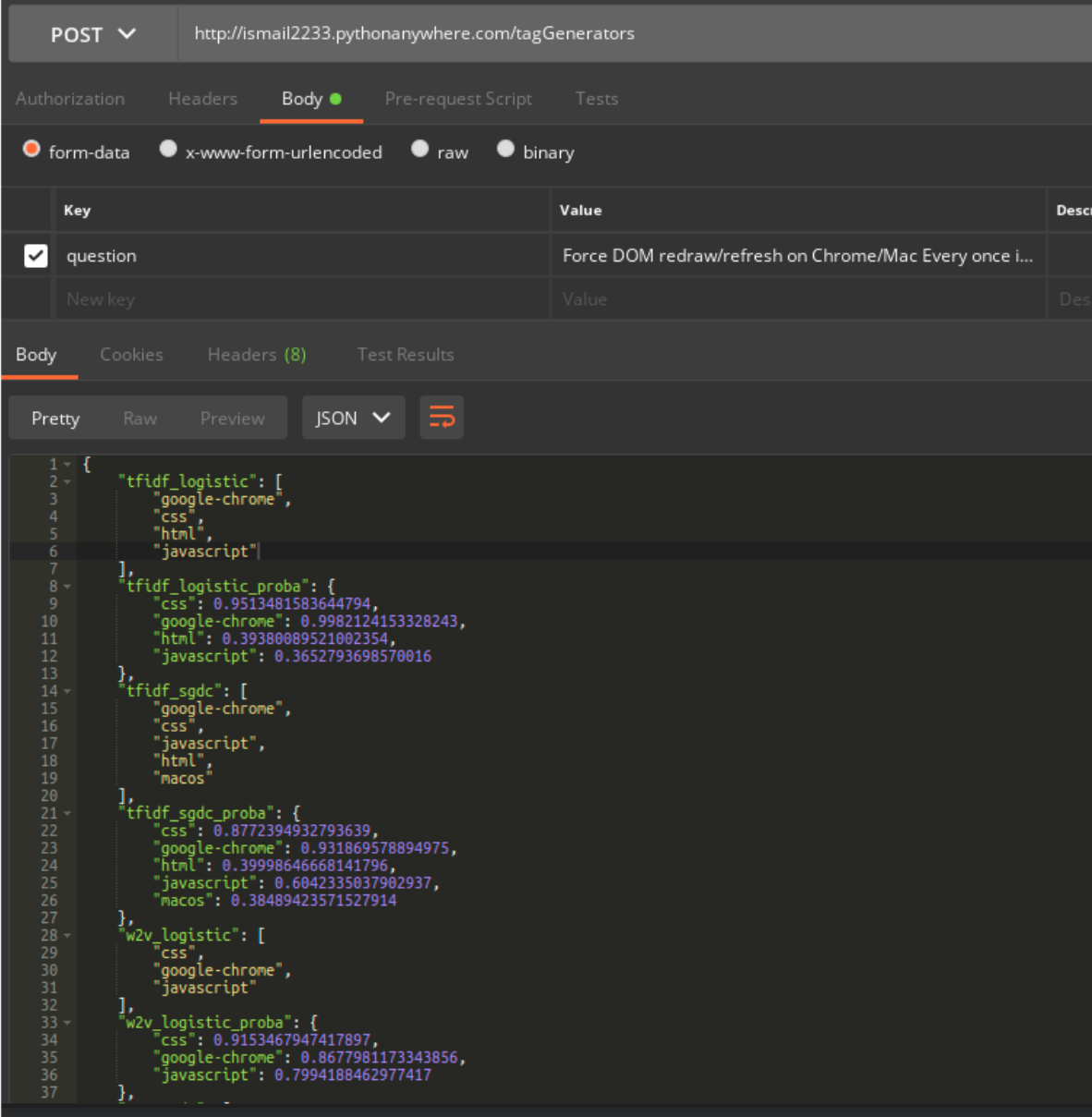
Modèles : LogisticRegression et SGDClassifier

features: TF-IDF et word2vec

API

Code Github :
<https://github.com/ismailazdad/stackoverflowTags/>

Url :
<http://ismail2233.pythonanywhere.com/>




POST ▼ http://ismail2233.pythonanywhere.com/tagGenerators

Authorization Headers **Body** Pre-request Script Tests

☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

	Key	Value	Description
<input checked="" type="checkbox"/>	question	Force DOM redraw/refresh on Chrome/Mac Every once i...	
	New key	Value	Description

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON ▼ 

```
1 {
2   "tfidf_logistic": [
3     "google-chrome",
4     "css",
5     "html",
6     "javascript"
7   ],
8   "tfidf_logistic_proba": {
9     "css": 0.9513481583644794,
10    "google-chrome": 0.9982124153328243,
11    "html": 0.39380889521002354,
12    "javascript": 0.3652793698570016
13  },
14  "tfidf_sgdc": [
15    "google-chrome",
16    "css",
17    "javascript",
18    "html",
19    "macos"
20  ],
21  "tfidf_sgdc_proba": {
22    "css": 0.8772394932793639,
23    "google-chrome": 0.931869578894975,
24    "html": 0.39998646668141796,
25    "javascript": 0.6042335037902937,
26    "macos": 0.38489423571527914
27  },
28  "w2v_logistic": [
29    "css",
30    "google-chrome",
31    "javascript"
32  ],
33  "w2v_logistic_proba": {
34    "css": 0.9153467947417897,
35    "google-chrome": 0.8677981173343856,
36    "javascript": 0.7994188462977417
37  },
38  }
```

API/Site web

Application développer avec le framework Flask,
Respectant le standard MVC (modèle, vue, contrôler)
La partie Vue est la partie visuel la page web (html,css,javascript)
La partie contrôler récupère la requête du formulaire et la transmet au
Service qui elle contient les modèles et predit les tags de la question sur les modèles et
renvoie les résultats

Stackoverflow questions tagging generator

Question Sample:

I need to write method that return power of only integer numbers I need to write a method in java to return the power of only integer number and i want this method to return -1 or fire expection if the number exceeds the Integer.MAX_VALUE:\nI tried the first and easv

Predict tags

LOGISTIC model with tf-idf #java conf : 94%	SGDC model with tf-idf #java conf : 87%
LOGISTIC model with word2vec #java conf : 91%	SGDC model with word2vec #performance conf : 100% #java conf : 95%

Conclusion/ amélioration

- Les modèles supervisés sont précis mais peu sensibles. Il arrive qu'ils ne prédisent pas de Tags du tout. Nos meilleurs scores sont obtenu en supervisé avec le modèle LogisticRegression.
- Afin d améliorer nos modèles , nous aurions pu utiliser un Voting Classifier afin d agréger nos modèles les plus performants, et donc proposer des tags plus pertinents par le vote de nos modèles
- Le NLP est un domaine riche, en y consacrant plus de temps, comme la sélection de « sentences » plus judicieux et d autres algorithmes de détection, nous aurions pu améliorer nos performances
- Un serveur plus performant , nous permettrait d accélérer nos temps de calcul
- Utiliser des modèles plus performants du type réseaux de Neurones pré-entraînés et BERT