

Catégorisation automatique de questions stackoverflow



introduction

- StackOverflow est un site célèbre de question-réponses liées au développement informatique.
- Plusieurs tags sont associés à chaque question afin de pouvoir retrouver facilement la question par la suite.
- développer un système de suggestion de tag en utilisant un algorithme de machine learning
- approches testées: supervisée et non-supervisée.

Goals

- 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.

1) 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.

2) La deuxième approche est l'utilisation d'un modèle supervisée multi-label. Cela implique d'avoir un nombre fini de tags prévisibles.

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
```

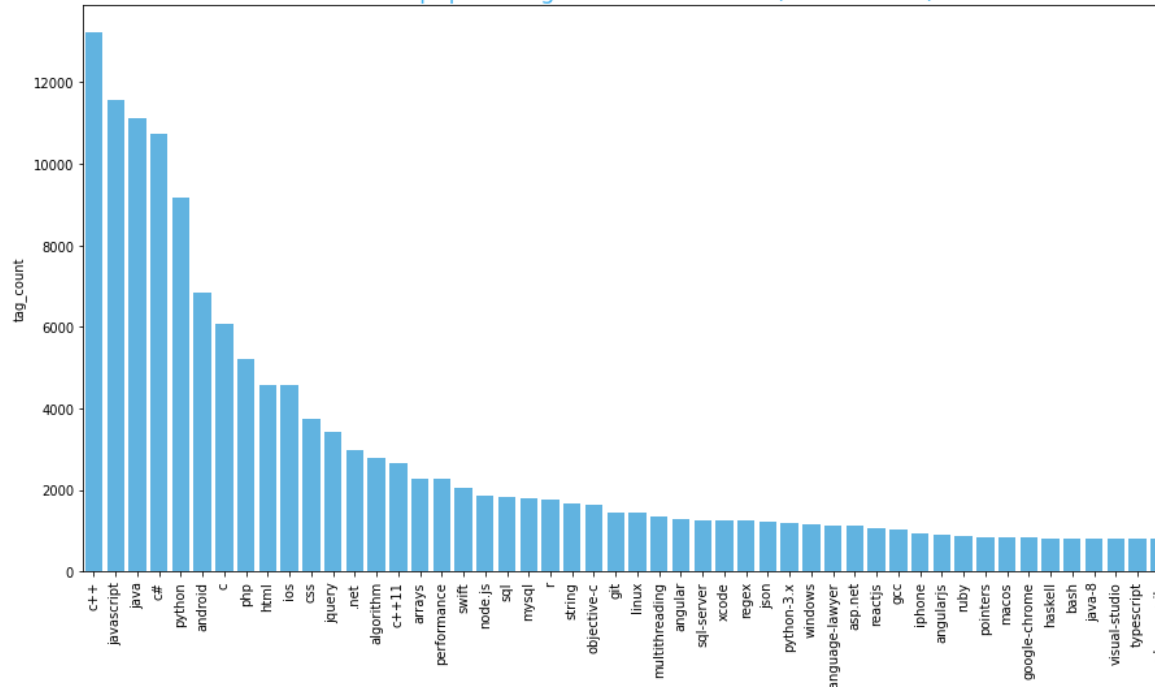
Elles portent sur la période 2011 / 2022
et uniquement sur les posts "de qualité"
ayant au minimum

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

116569 lignes récupérées

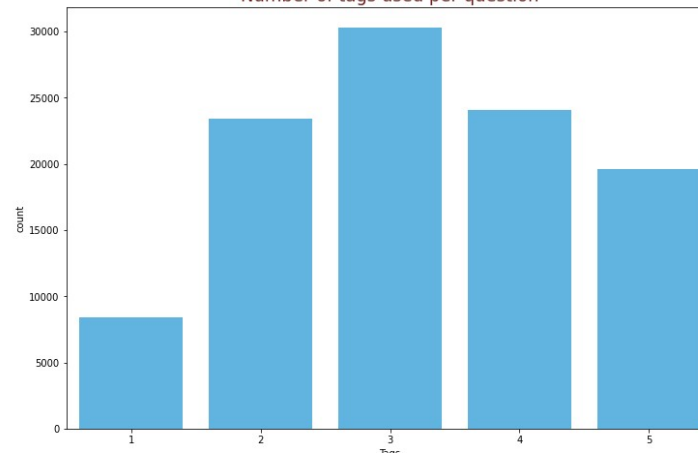
Exploration des tags

50 most popular tags in Stackoverflow (2011 - 2021)



Sélection des 50 top tags

Number of tags used per question



Prédiction multi classe
Car 3 tags en moyenne

Pre traitement

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[^\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
-

Approche non supervisées

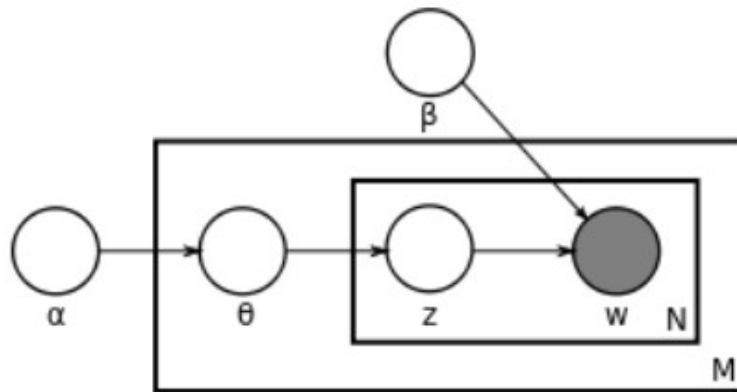
LDA (Latent Dirichlet Allocation).

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

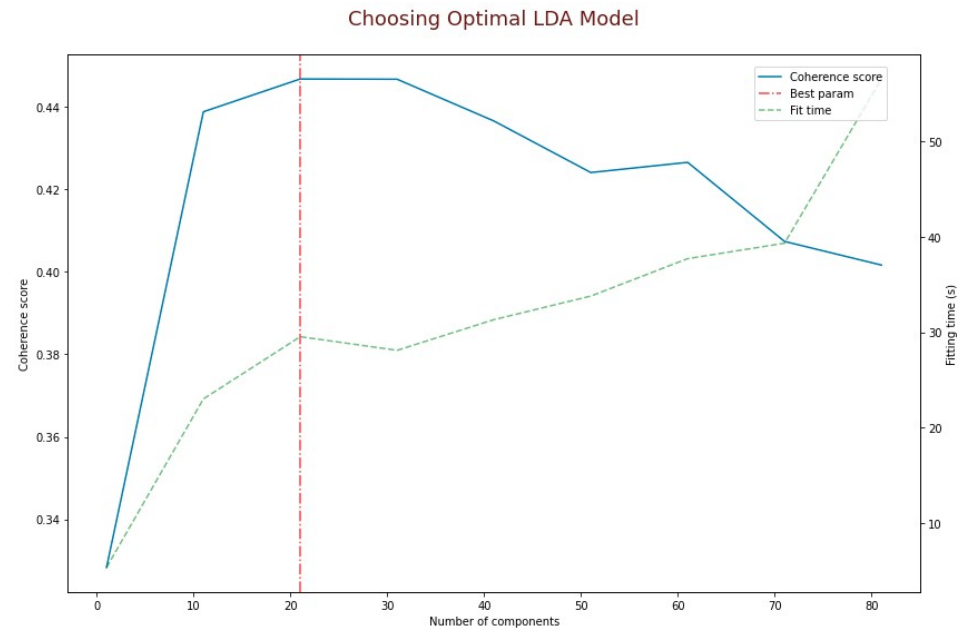
N : Ensemble des variables liées à un mot

z : Distribution d'un topic pour un mot

w : Mot

Resultat LDA

Topic 0:
number memory question time bit way size solution numbers example
Topic 1:
project new element elements remove build add operator ve solution
Topic 2:
com https source http google service xml github org play
Topic 3:
app android user ios application react index device date using
Topic 4:
string way like convert use using strings want best format
Topic 5:
server json api request default git web client response using
Topic 6:
js javascript value form input key node field jquery item
Topic 7:
php functions thread language use generate random course feature like
Topic 8:
code python like use work using just know don doesn
Topic 9:
java data exception map structure arrays gradle arguments spring firebase
Topic 10:
list data table database query sql column like using want
Topic 11:
error code following studio getting message visual errors compile tried
Topic 12:
class type object method like objects use methods classes reference
Topic 13:
function array value null pointer standard type undefined return int
Topic 14:
code test net time run output core using does difference
Topic 15:
button chrome browser component click event console window log website
Topic 16:
file files line page script using swift load folder read
Topic 17:
image view html text angular like using want css page
Topic 18:
run windows running version command 10 using application package installed
Topic 19:
variable does called code value loop returns variables return compiler



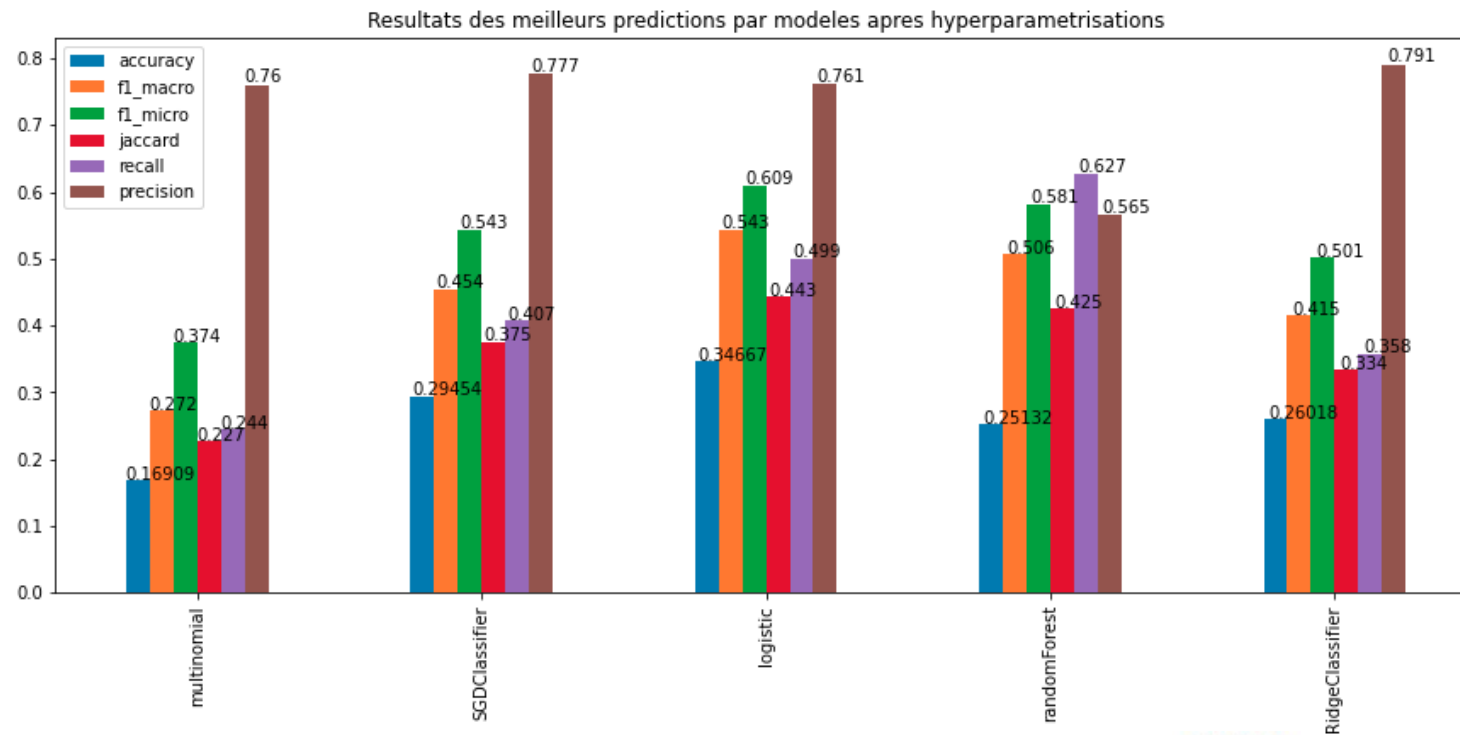
Perplexity: -7.03284251881742

Coherence Score: 0.4388432848486331

Approche supervisée

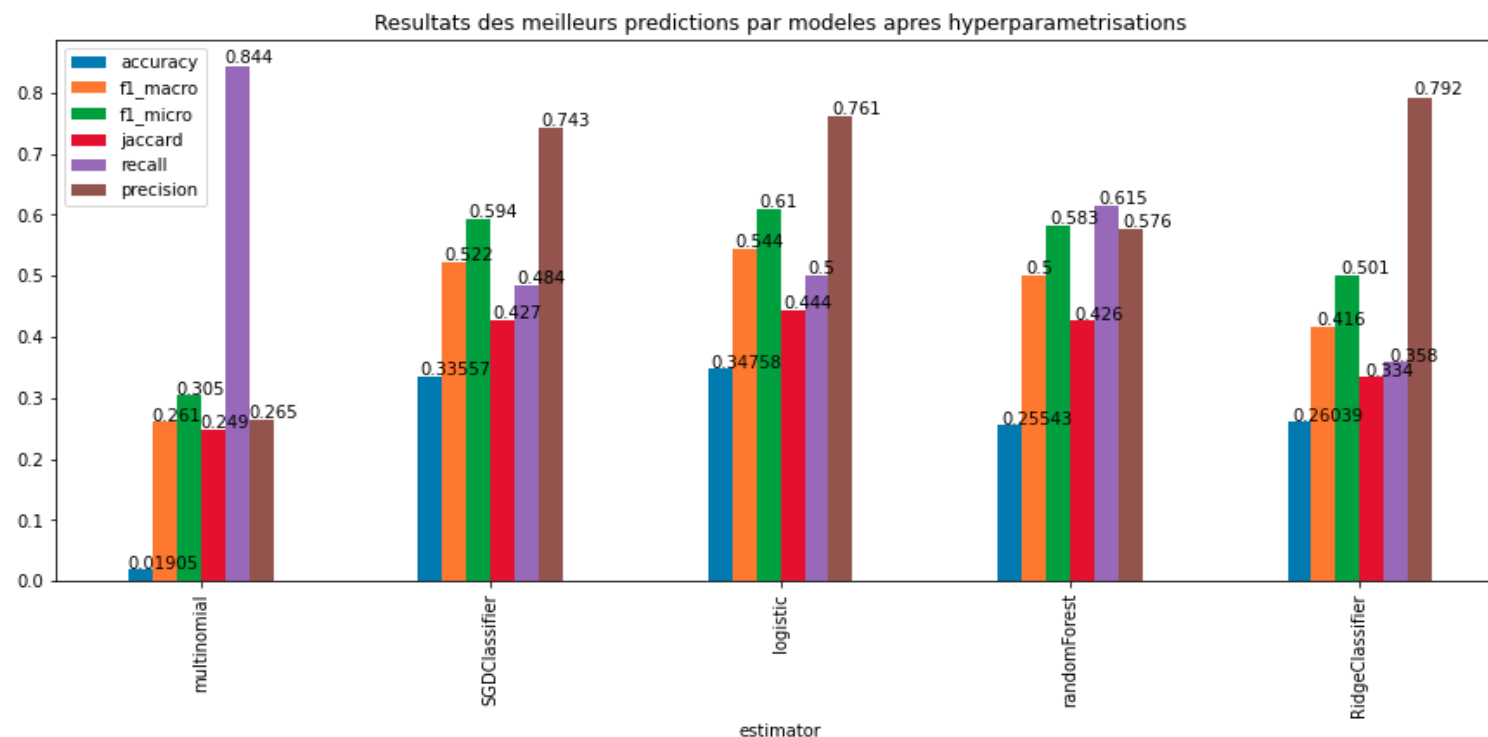
- Cette fois-ci le modèle va prédire des multi-label. Les labels sont encodés pour l'apprentissage avec MultiLabelBinarizer de scikit-learn. A l'inverse, on les décode pour prédire.
- Comme la mémoire est limitée et les données aussi, on ne peut prédire qu'un nombre maximum de Tags différents. Cela implique une réduction du jeu d'entraînement aux questions qui contiennent au moins un tag parmi les « n plus utilisés » (ici 50)
- Nous allons dans un premier temps créer des features pour nos algorithmes (avec TF-IDF, wor2vec et doc2vec)
- Nous testerons ensuite chaque features sur différents modèles avec gridSearchCV , avec optimisation des hyper-paramètres
- Nous utiliserons One Vs Rest pour la classification multi-classe

Approche supervisée tf-idf gridsearch oneVsRest



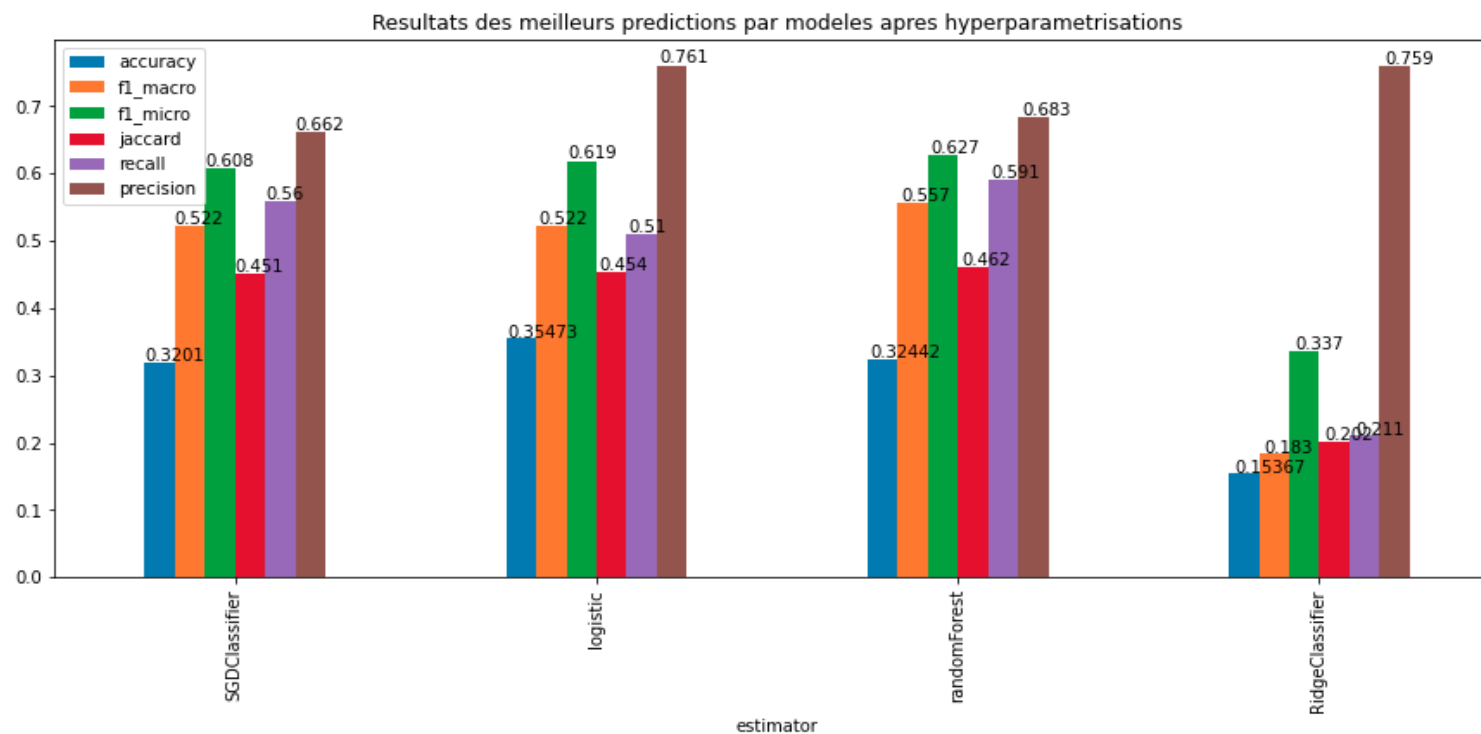
estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
multinomial	0.16909	0.272	0.374	0.227	0.244	0.760	63.66%
SGDClassifier	0.29454	0.454	0.543	0.375	0.407	0.777	40.29%
logistic	0.34667	0.543	0.609	0.443	0.499	0.761	28.45%
randomForest	0.25132	0.506	0.581	0.425	0.627	0.565	12.36%
RidgeClassifier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

Approche supervisée tfidf avec score gridsearch



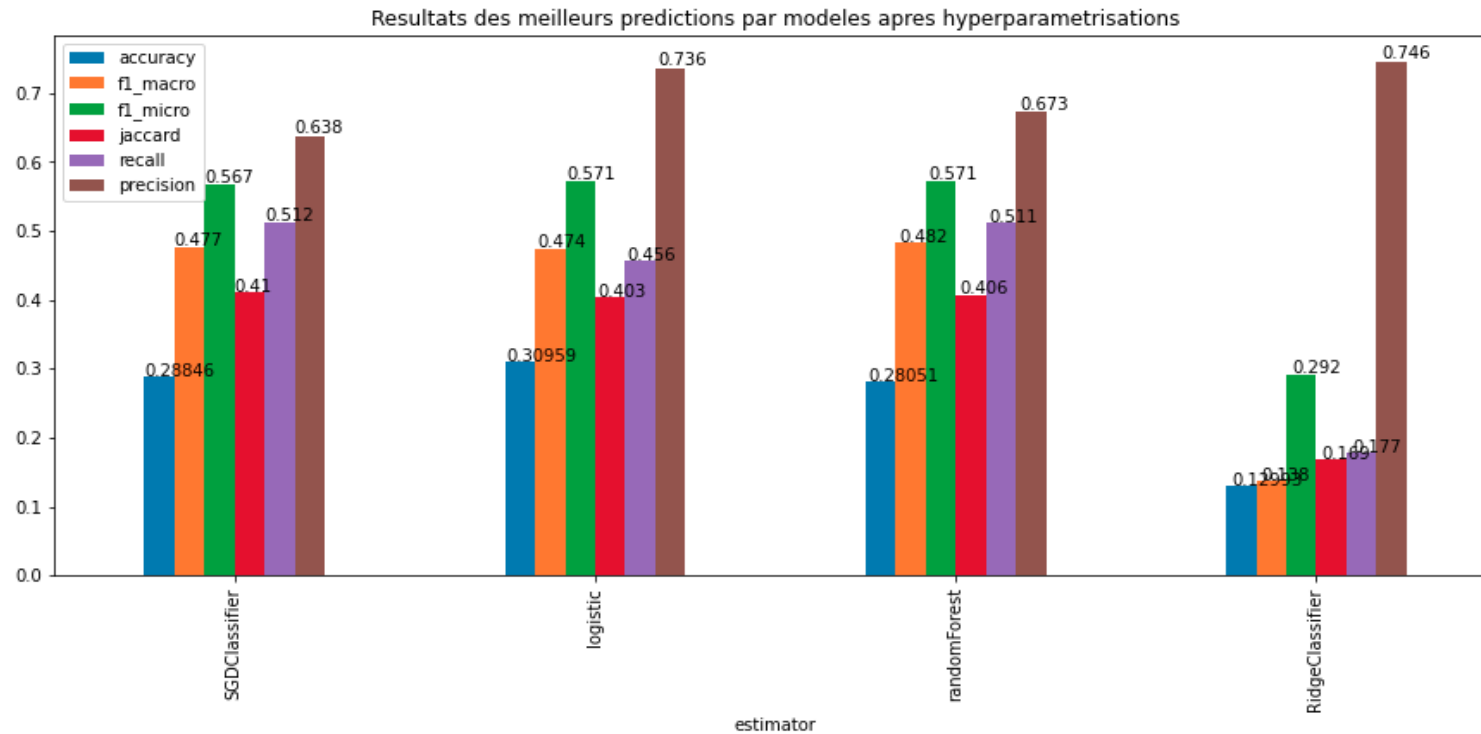
estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
multinomial	0.01905	0.261	0.305	0.249	0.844	0.265	0.21%
SGDClassifier	0.33557	0.522	0.594	0.427	0.484	0.743	28.58%
logistic	0.34758	0.544	0.610	0.444	0.500	0.761	28.33%
randomForest	0.25543	0.500	0.583	0.426	0.615	0.576	13.32%
RidgeClassifier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

Approche supervisée word2vec gridsearch



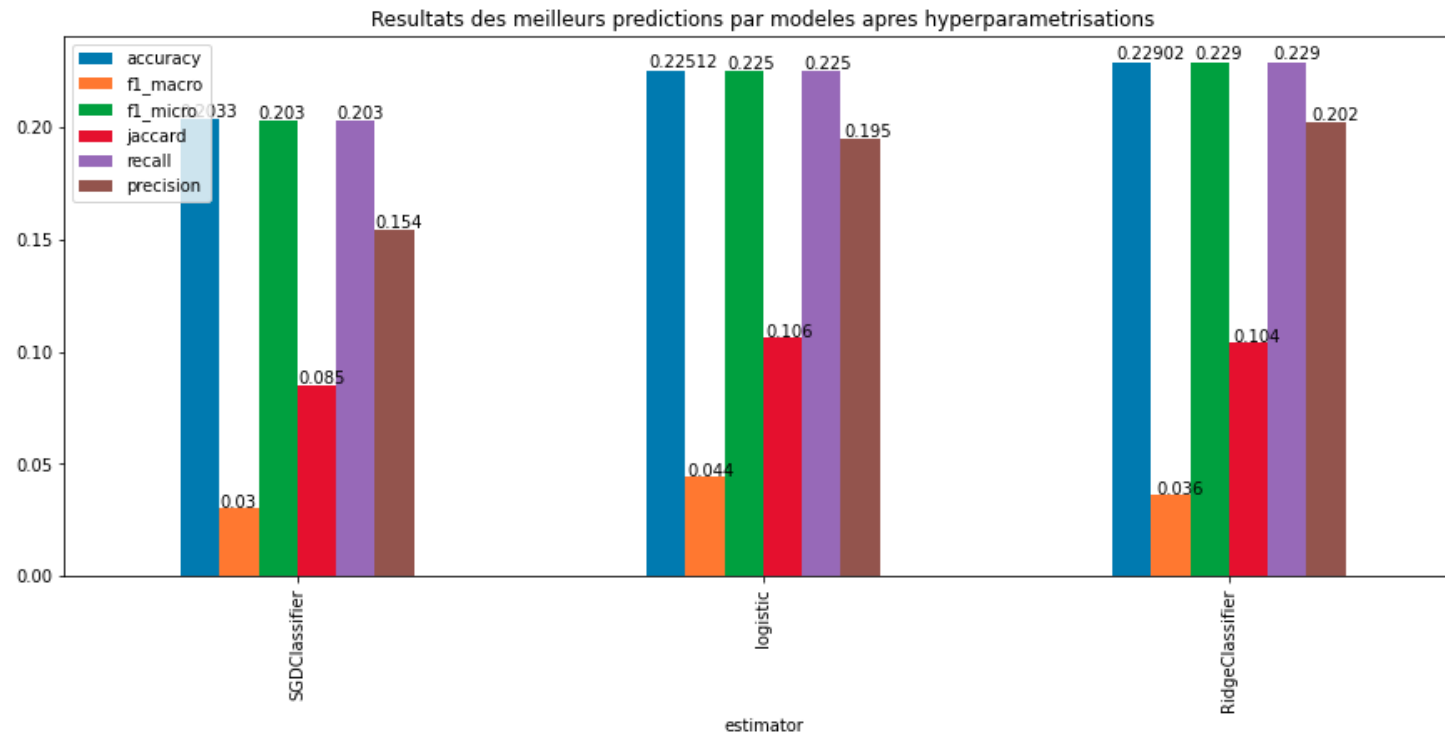
estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
SGDClassifier	0.32010	0.522	0.608	0.451	0.560	0.662	20.71%
logistic	0.35473	0.522	0.619	0.454	0.510	0.761	29.13%
randomForest	0.32442	0.557	0.627	0.462	0.591	0.683	21.96%
RidgeClassifier	0.15367	0.183	0.337	0.202	0.211	0.759	68.42%

Approche supervisée word2vec wikipedia



estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision	untaged
SGDClassifier	0.28846	0.477	0.567	0.410	0.512	0.638	24.1%
logistic	0.30959	0.474	0.571	0.403	0.456	0.736	34.09%
randomForest	0.28051	0.482	0.571	0.406	0.511	0.673	28.68%
RidgeClassifier	0.12993	0.138	0.292	0.169	0.177	0.746	72.61%

Approche supervisée doc2vec



estimator	accuracy	f1_macro	f1_micro	jaccard	recall	precision
SGDClassifier	0.20330	0.030	0.203	0.085	0.203	0.154
logistic	0.22512	0.044	0.225	0.106	0.225	0.195
RidgeClassifier	0.22902	0.036	0.229	0.104	0.229	0.202

Sélection de nos modèles

Au vu des différents tests effectués, nous allons sélectionner :

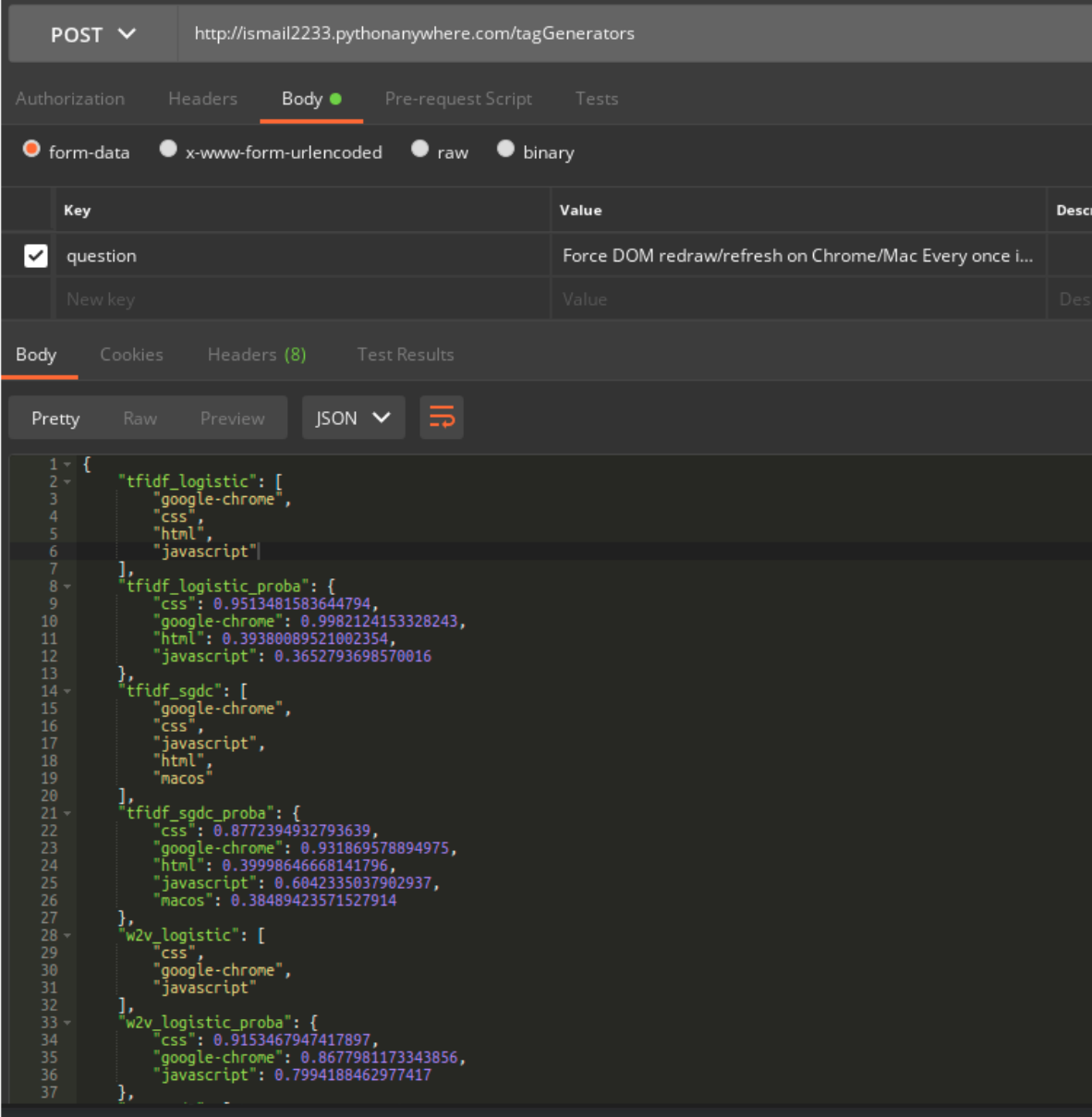
Modèles : LogisticRegression et SGDClassifier

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": 0.39380889521802354,
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 prédit 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%

Comparaisons

Avantages

inconvénients

Approche supervisée

Modelés et métriques connus

Bonnes performances

Rapide une fois que le modèle est entraîné

Peu de faux positifs

Besoin de pré-traitements
Supplémentaires

Temps de calcul long (surtout
word2vec)

Plus d'éléments à maintenir

Approche non supervisée

Mise en place rapide

Pas de pre traitement

Un modèle facile à maintenir

Bonne vision des sujets abordés dans un
corpus de documents

Peut faire ressortir des nouveaux sujets pas
encore couverts par les tags actuels

Difficile d'évaluer les performances
du modèle

Conclusion/ amélioration

- Les modèles non-supervisés ont une meilleure sensibilité
- 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

Thank you

Questions ?