Catégoriser automatiquement des questions

Mestapha Oumouni

P5 OC parcours IML
Mentor: Samir Tanfous
Août 2022

OPENCLASSROOMS

stackoverflow

TABLE DES MATIÈRES

Contexte du projet

- Très grande plateforme de questions-réponses liée au développements informatiques
- Il faut entrer plusieurs étiquettes liées à la question.
- Tâche fastidieuse pour les utilisateurs moins expérimentés
- Les tags sont un texte libre

Objectifs: Développer un système automatique de suggestion de tags pour le site en assignant automatiquement plusieurs tags pertinents à une question

➔ Utilisons les algorithmes de machine learning

Source de données

Données importées depuis la plateforme "stackexchange explorer" proposé par Stack Overflow

```sql
SELECT TOP 500000 Title, Body, Tags, Id, Score, ViewCount, FavoriteCount, AnswerCount
FROM Posts
WHERE PostTypeId = 1 AND ViewCount > 10 AND FavoriteCount > 10
AND Score > 5 AND AnswerCount > 0 AND LEN(Tags) - LEN(REPLACE(Tags, '<',')) >= 5
```

Base de données de 27528 lignes et 8 colonnes (fichier .csv).
Comptent les titres, les corps et les tags associés à chaque question et les informations sur le score et le nombre de vues et de réponses.

Nettoyage et exploration des données

Particularité des questions et des Tags

Contient du code
Balise html
Tags avec des balise <>. seront remplacé par ‘ ’

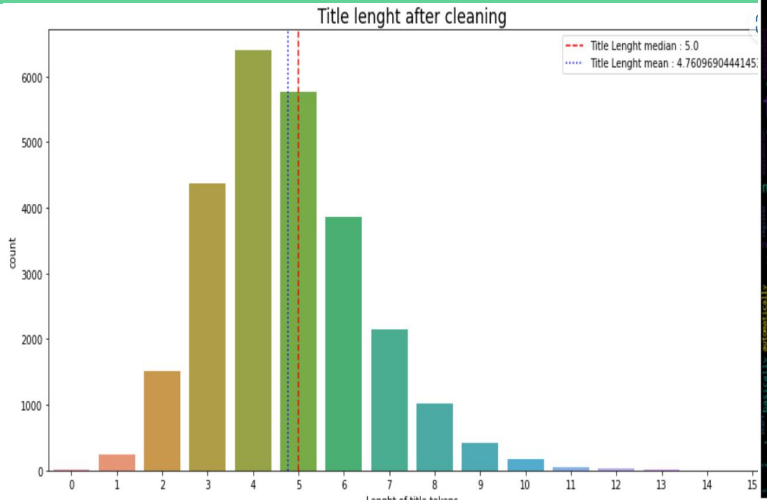
- ❏ Filtrer le corps suivant la langues anglaise
- ❏ Formatage des Tags
- ❏ Typage des questions
- ❏ Suppression des balises HTML avec BeautifulSoup
- ❏ Passage au minuscule
- ❏ Suppression des ponctuations, extra-espace, nombre, et d’autre caractères spéciales … sauf(c# et c++)
- ❏ Tokenisation
- ❏ Lemmatisation avec POS
- ❏ Suppression des stopwords
- ❏ Suppression les doublons de Tags

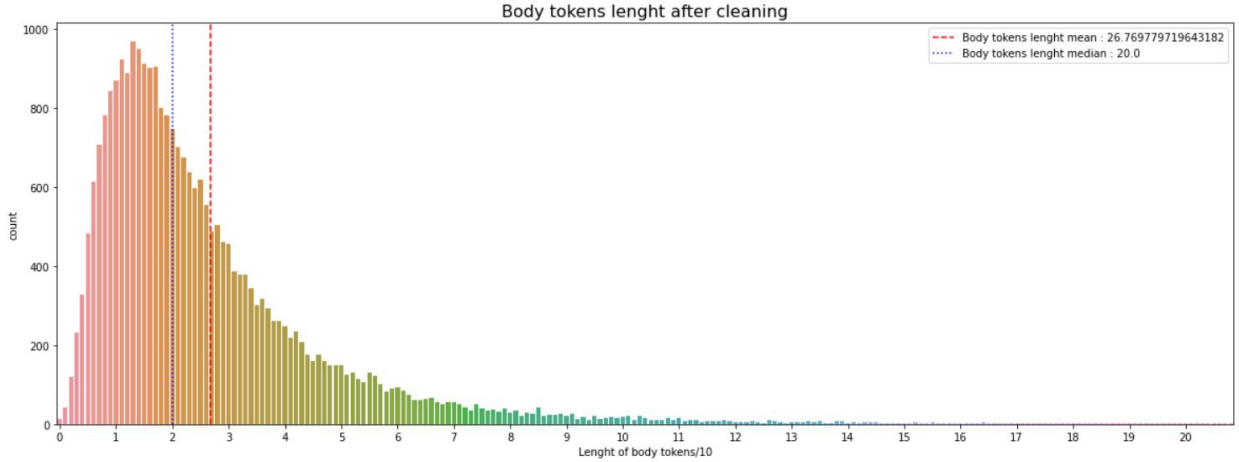
Nettoyage et exploration des données

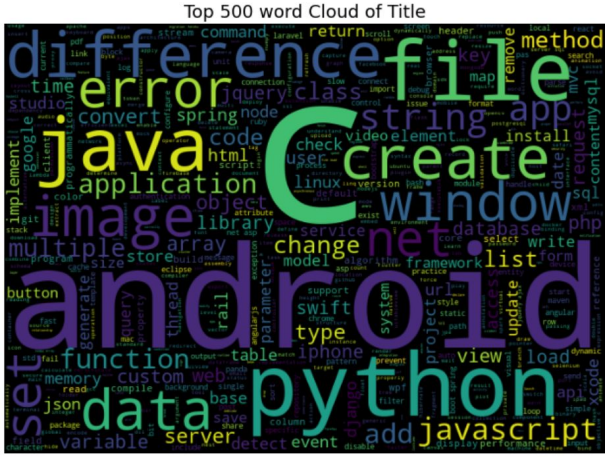
Nombre et fréquence des mots dans les titres, corps et tags.

Suppression des tags inexistants dans les listes de corps+title

filtrer les données suivant
len(tags)>0
1< len(title)< 10
5< len(body)< 120







Extraction des features

- Bag of Words: créer une matrice creuse (n,k)
  - n documents et k le nombre de n-grams
  - entrée ij compte le nombre de fois où le mot j apparaît dans le document i

- TF- idf = Term-Frequency – Inverse Document Frequency
  - matrice creuse (n,k)
  - entrées sont des fréquences des mots pondérées par les ordres des mots.
  - but: réduire l'importance relative des tokens très fréquents et moins significatifs

Word2vec: plongement des mots (avec des matrice dense)

- Les mots sont représentés par vecteurs => calculer les distances, similarités des mots opérations arithmétiques
- Dictionnaires pré-entraînés, exemple: Google News dataset (100 billion mots)
- Entrainer son propre modèle par un réseau de neurone

```
model_w2v = word2vec.Word2Vec(data.text_comb, size=164,
                              window=15, min_count=3, workers=7)
model_w2v.wv.most_similar(['python'])
```

```
[('ipython', 0.8268296718597412),
 ('print', 0.8149738311767578),
 ('variable', 0.8042786121368408),
 ('utf', 0.8016868829727173),
 ('binary', 0.8005759119987488),
 ('vba', 0.7923832535743713),
 ('indexerror', 0.7877026796340942),
 ('scipy', 0.7862895727157593),
 ('gensim', 0.7779965400695801),
 ('plot', 0.7752940058708191)]
```

Modélisation supervisée

Tf-idf features

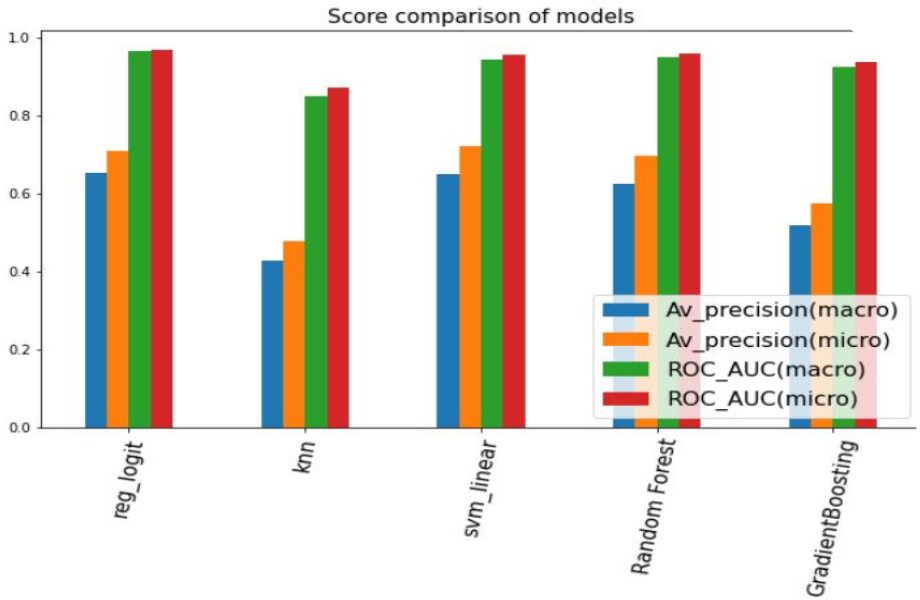
- Utilisation 50 top tags
- One-versus-rest(Classifier)
- Prédiction avec probabilité d'appartenance à chaque classe
- Seuil optimal de décision
- métrique Précision et ROC-AUC moyenne
- Comparaison des modèles via le temps de prédiction, le meilleur score et le nombre de tags vide

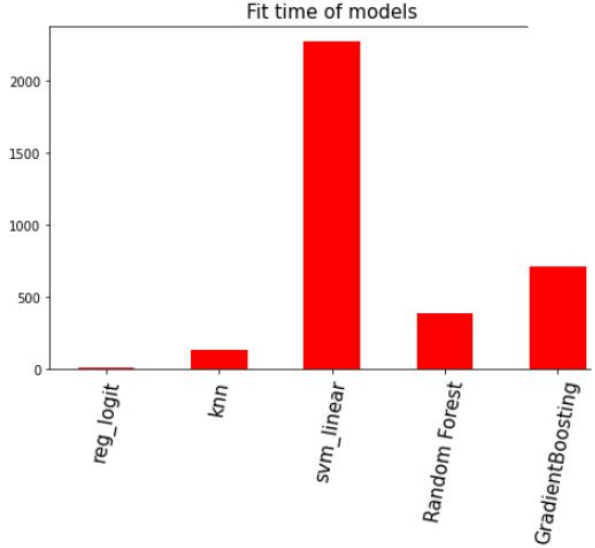
Avantage: prédiction directe des tags

Modèles

- Régression logistique
- KNN
- SVM (linéaire)
- Forêt aléatoire
- Gradient boosting

Modélisation supervisée (Comparaison)

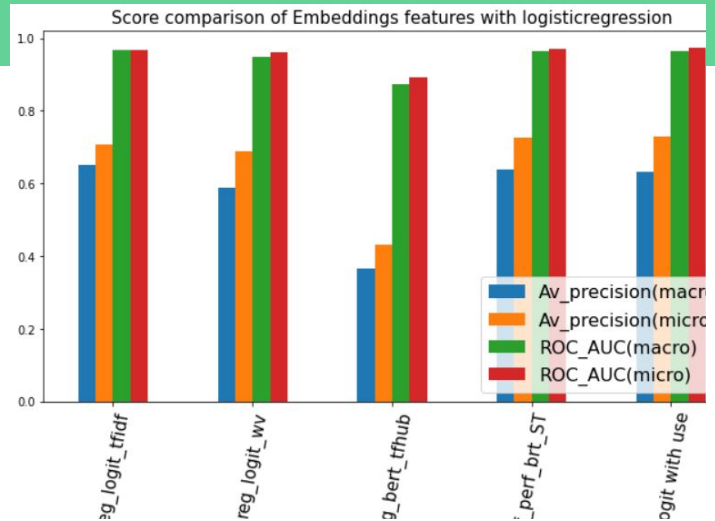


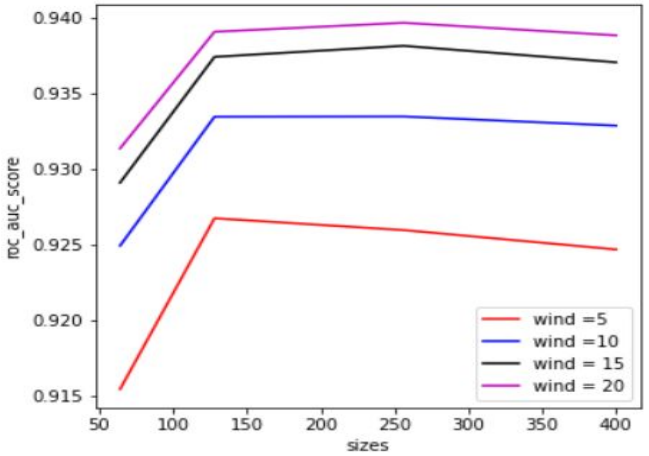


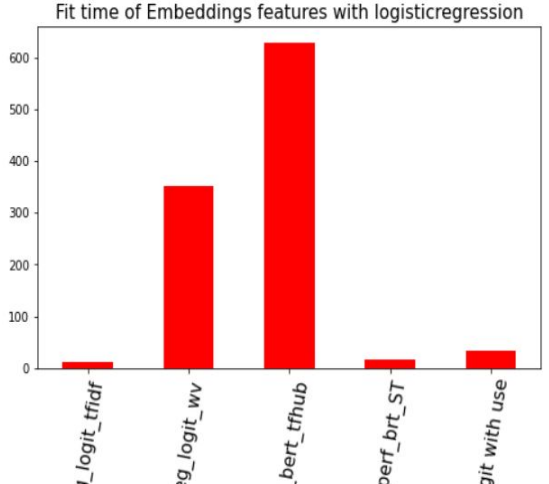

```
print('le nombre de prédiction vide par la Regression Logistic est: ', sum(count_lr==0))
print('le nombre de prédiction vide par le Random Forest est:       ', sum(count_rf==0))
print('le nombre de prédiction vide par le SVM:                     ', sum(count_svm==0))
```

```
le nombre de prédiction vide par la Regression Logistic est:  89
le nombre de prédiction vide par le Random Forest est:        192
le nombre de prédiction vide par le SVM:                      185
```

- Word2vec: paramètres à analyser pour améliorer les scores
- Les modèles près-entraînés USE et Bert-sentence améliore certains scores à un-deux point/100 .
- Le temps d'entraînement (matrice dense) est important/Tf-idf.
- Utilisation de GPU





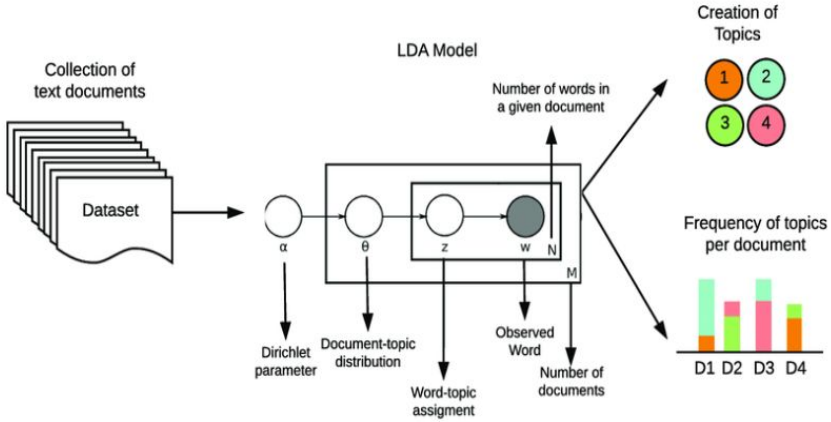


Modélisation non-supervisée (LDA modèle)

- ❏ LDA méthode non-supervisée générative des topics (cachés) dans le corpus
- ❏ Objectif: Trouvez les topics les plus pertinente pour représenter les documents

• Désavantage : Difficile de trouver des topics précis, distincts et interprétables.

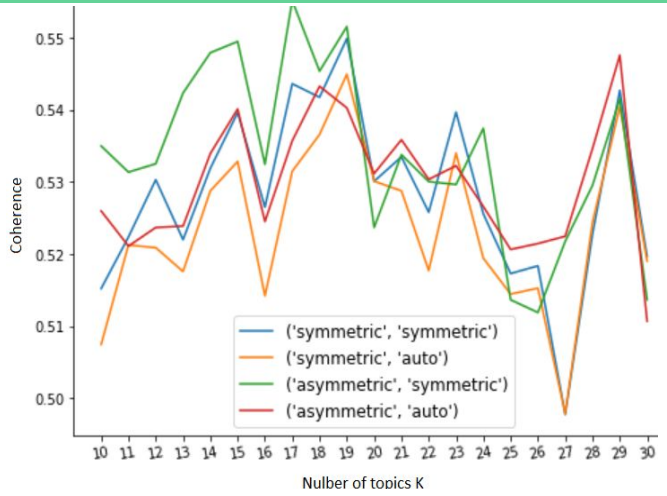
- ❏ NMF: algorithme alternative

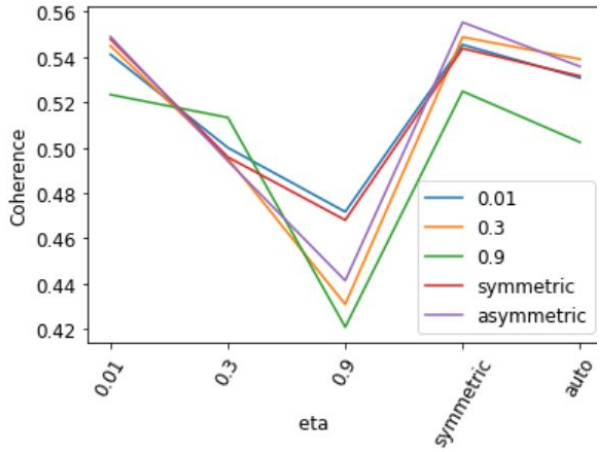


Preprocessing LDA avec Gensim:

- ❖ Bag of words: Dictionnaires des mots du corpus
- ❖ Analyse de sensibilité avec les paramètres:
  - ➢ alpha : paramètre de doc/topic distribution
  - ➢ eta : paramètre de topic-word distribution
  - ➢ nombre de topics K

Modélisation non-supervisé (LDA modèle)

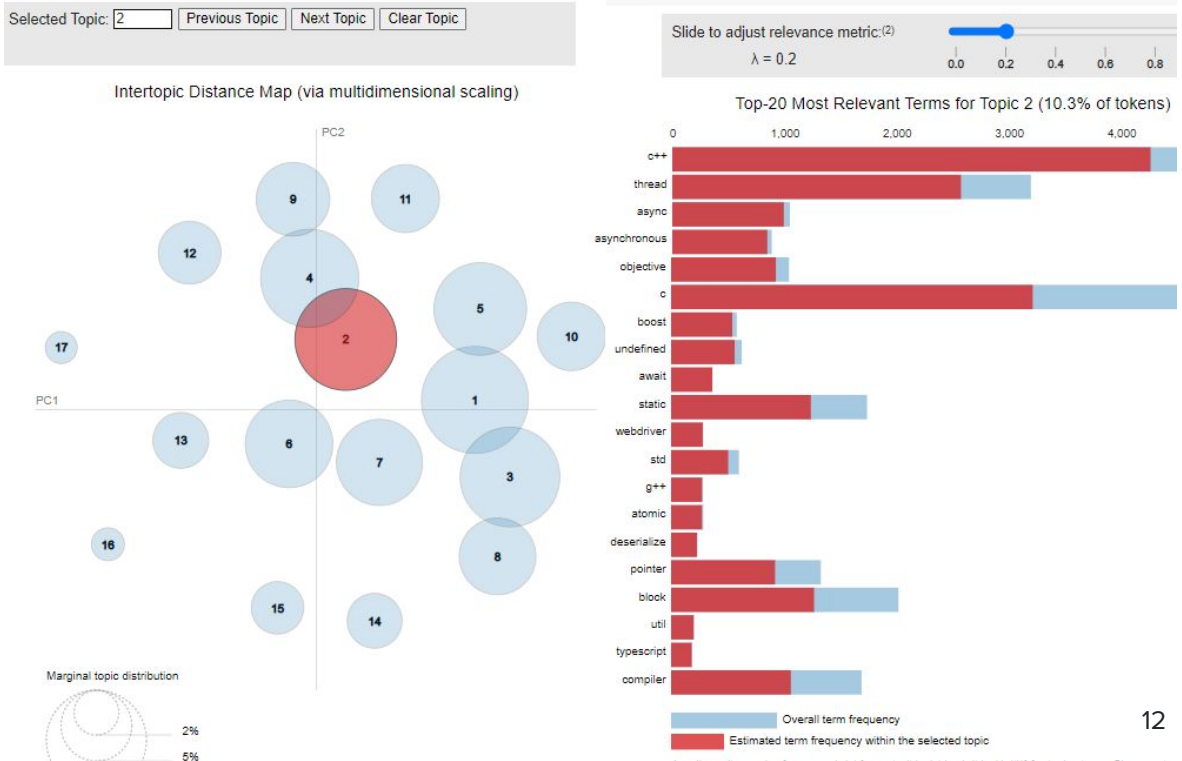




LDA entrainé avec: k =17, alpha =’asymmetric’, eta =symmetric

```
lda_model = gensim.models.LdaMulticore(corpus=corpus, id2word=id2word, num_topics=nb_opt, random_state=42,
                                       passes=10, alpha=alpha_opt, eta=eta_opt)

pyLDAvis.enable_notebook()
vis = gensimvis.prepare(lda_model, corpus, id2word, mds='mmds',R=20)
vis
```



Modélisation non-supervisé (LDA modèle)

deux approches de prédiction:

- ➢ Matrice des scores tags/topic: assigné à chaque doc
- ➢ 'n' tags : scores<= seuil
- ➢ Utiliser la matrice document/topic comme features dans un modèle de régression logistique

```
lda_model = gensim.models.LdaMulticore(corpus=corpus, id2word=id2word, num_topics=17, random_state=42,
                                       passes=10, alpha='asymmetric', eta='symmetric')
```

```
# obtain topic distributions for each document
topic_dist = pd.DataFrame(lda_model.get_document_topics(corpus, minimum_probability=0.0))
```

```
topic_tag = np.matmul(topic_dist.T, y_binarized)
# normalizing
topic_tag = topic_tag/np.sum(y_binarized,axis=0)
topic_tag.head(3)
```

	0	1	2	3	4	5	6	7	8	9
0	0.128682	0.069742	0.049993	0.162818	0.054312	0.290504	0.143000	0.457608	0.089989	0.093337
1	0.096367	0.053450	0.085711	0.307414	0.087008	0.073641	0.102542	0.052399	0.106214	0.071200
2	0.291931	0.096562	0.050193	0.081494	0.028274	0.124660	0.052230	0.065082	0.182549	0.111117

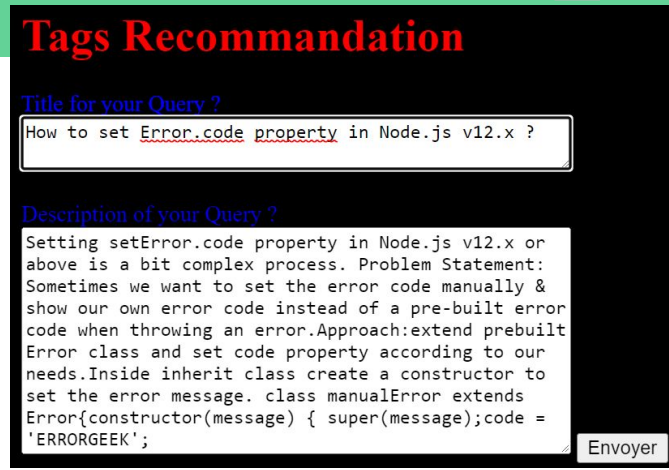
```
time1 =time.time()
x_train, x_test, y_train, y_test = train_test_split(topic_dist, y_binarized, test_size=
lr = OneVsRestClassifier(LogisticRegression()).fit(x_train, y_train)
y_pred_lr = lr.predict_proba(x_test)
fit_time = np.round(time.time() - time1,1)
#score of prediction
df_LDA_lr = metrics_report("LDA_lr", y_test, y_pred_lr,fit_time, df=None)
print(df_LDA_lr)
```

	Tags	best_topic	y_pred
0	[java, array, android]	1	(array, json, string)
1	[file, c]	0	(c++, linux, memory, multithreading)
2	[web, service, c#, net]	9	(database, mysql, server, sql)
3	[javascript]	1	(array, json, string)

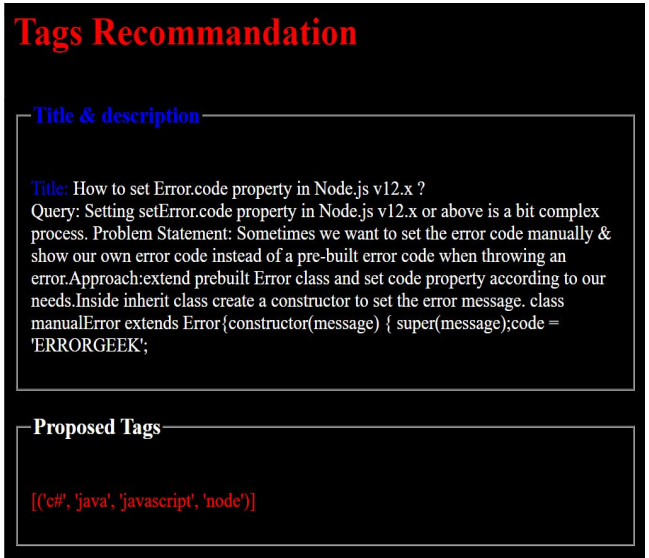
	LDA_lr
Av_precision(macro)	0.182246
Av_precision(micro)	0.252294
ROC_AUC(macro)	0.844211
ROC_AUC(micro)	0.866935
Fit_time	7.000000

Api et gestion de code

- Sérialisation du pipeline de modèle Tf-idf + ovr(LR)
- Module des fonctions de nettoyage et d'inversement
- Déploiement en local avec Flask
- L'api répond à l'adresse 127.0.0.1 :5000; methods = ['GET', 'POST']



- Dépôt disponible sur Github https://github.com/Bounkass/Proj5ml_oc.git
-
- Cloner : git clone https://github.com.....
-
- Instructions après modifications:
  - git status
  - git add nom_fichier
  - git commit -m "message"
  - git push -u origin master



Conclusion

- Application des approches de ML à un problème de NLP
- Développer un outil de suggestion de Tags (API) des questions des utilisateurs

- Meilleur modèle: Régression logistique
  - prédiction rapide
  - meilleure précision et moins de prédiction vide

- Pistes d'amélioration
  - Entraîner un modèle de Word embeddings adapté au sujet des questions
  - optimiser le seuil de ROC

Merci de votre attention!