

Projet Stackoverflow

Efkan TUREDI

Intro



Grand utilisateur de Stackoverflow, nous décidons de donner notre contribution à la communauté d'utilisateurs de la plateforme. Ainsi, nous devons développer un système de tags automatiques pour chaque nouveau post soumis par la communauté.

- Nous devons déployer une API permettant de réaliser la tâche ci-dessus
- Un rapport et un notebook accompagneront cet exercice

Récupération des données

Stackexchange Explorer (1/2)

- ❑ Nous souhaitons extraire une quantité suffisante de données sans faire de compromis sur la qualité de celles-ci. C'est pourquoi nous allons uniquement extraire les 50,000 questions les plus "vues" sur la plateforme
- ❑ Pour les 4 derniers semestres, nous faisons des requêtes SQL pour extraire les questions postées durant ces périodes.

Stackexchange Explorer (2/2)

```
1 SELECT TOP(50000) Id, CreationDate, Score, ViewCount, AnswerCount, CommentCount
2 FROM Posts
3 WHERE CreationDate BETWEEN CONVERT(datetime, '2020-01-01') AND CONVERT(datetime, '2020-01-01')
4 AND Score IS NOT NULL
5 AND ViewCount IS NOT NULL
6 AND AnswerCount IS NOT NULL
7 AND CommentCount IS NOT NULL
8 AND FavoriteCount IS NOT NULL
9 ORDER BY ViewCount DESC
```

Le preprocessing

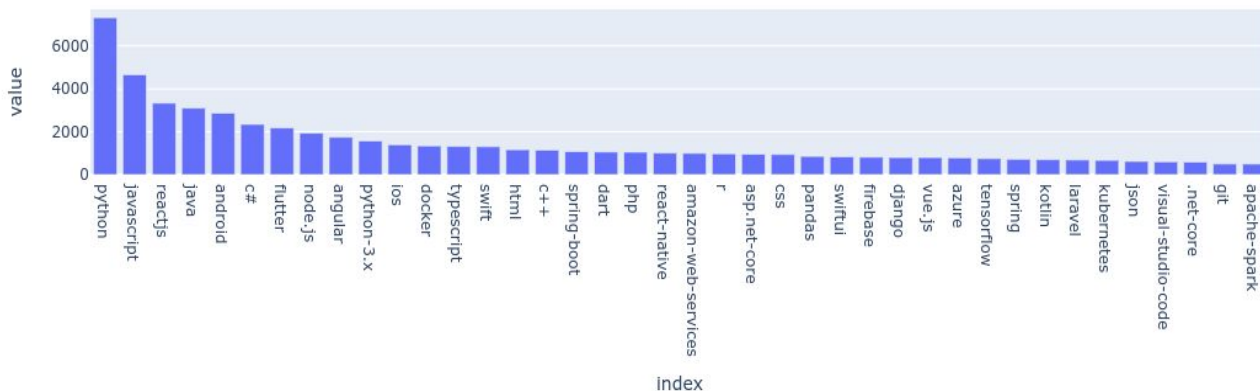
Nos données ont quelques particularités à maîtriser

- ❑ 3 Colonnes nous intéressent en particulier: Title, Body (features), et Tags (target).
- ❑ Chacunes de ces colonnes possèdent des particularités avant d'appliquer du preprocessing classiques de traitement de languages: tokenization, lemmatization, stop_words, ponctuation, etc...

Title	Body	Tags
Error message "DevTools failed to load SourceM...	<p>I'm trying to display an image selected fro...	<javascript><html>
When adding a JavaScript library, Chrome compl...	<h3>My code</h3>\n<pre class="lang-html pretty...	<google-chrome-devtools>
Could not load dynamic library 'cudart64_101.d...	<p>I just installed the latest version of Tens...	<python><python-3.x> <tensorflow><keras> <tensor...
SessionNotCreatedException: Message: session n...	<p>I am currently new to robot framework.I am ...	<selenium><google-chrome><selenium-webdriver><...
error NG6002: Appears in the NgModule.imports ...	<p>First time using firestore and I'm getting ...	<angular><google-cloud-firestore><angularfire>

Colonne Tags (Target)

- La colonne tags possède ~23,000 tags. Nous voulons conserver uniquement les 500 plus populaires pour réduire le bruit. D'autant plus que les tags les plus communs, ont un poids important dans cette donnée.



Colonnes Title et Body (Features)

- ❑ Peu de particularités pour la colonne Title qui constitue simplement une suite de mots à manipuler correctement
- ❑ La colonne Body devra être pré-traité avec Beautiful Soup. On extrait les tags <p> uniquement de chaque post pour extraire la partie qui nous sera le plus utile

```
[35]: from bs4 import BeautifulSoup

[36]: db['Body'] = db['Body'].apply(lambda x: BeautifulSoup(x).get_text())

[37]: db['Body'] = db['Body'].apply(lambda x: clean_text(x))
```

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\ve", " have ", text)
    text = re.sub(r"can't", "can not ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\re", " are ", text)
    text = re.sub(r"\d", " would ", text)
    text = re.sub(r"\ll", " will ", text)
    text = re.sub(r'\scuse', " excuse ", text)
    text = re.sub(r'\n', " ", text)
    text = re.sub(r'\xa0', " ", text)
    text = re.sub('\s+', ' ', text)
    text = text.strip(' ')
    return text
```

Traitement de languages naturels

- ❑ Nous allons appliquer les transformation suivantes: tokenization, transformations en minuscules, verification de stop_words, et lemmatization
- ❑ La fonction “apply” de pandas nous seras très utile dans ces transformations

```
[36]: db['Body'] = db['Body'].apply(lambda x: BeautifulSoup(x).get_text())

[37]: db['Body'] = db['Body'].apply(lambda x: clean_text(x))

[38]: db['Body'][14]

[38]: "using angular and during ng serve, i am getting the error: error from chokidar (c:\\): error: ebus
tack.log.tmp error from chokidar (c:\\): error: ebusy: resource busy or locked, lstat 'c:\\\\hiberfil
y: resource busy or locked, lstat 'c:\\\\pagefile.sys error from chokidar (c:\\): error: ebusy: resou

[39]: punct = '!"#$%&\\'()*+,-./:;<=>?@[\\]^_`{|}~'

[40]: db['Body'] = db['Body'].apply(lambda x: clean_punct(x, tags_top_500))

Now we have to remove the stopwords, tokenize and lemmatize the Body column

[41]: db['Body'] = db['Body'].apply(lambda x: tokenizer(x))

[42]: db['Body'] = db['Body'].apply(lambda x: stop_words_check(x, stop_words))

[43]: db['Body'] = db['Body'].apply(lambda x: list_lemmatizer(x))

[44]: db['Body'] = db['Body'].apply(lambda x: " ".join(x))
```

Création de la colonne "Post"

- ❑ Pour nous simplifier la tâche lorsqu'il faudra séparer les données en train et test set, et accélérer les transformations que nous ferons, nous décidons de créer une colonne Post qui sera la fusion entre Title et Body.
- ❑ Nous donnons un poids supplémentaire à la colonne Title par rapport à Body pour compenser la différence de nombre de caractères entre les deux colonnes. Aussi, la colonne Title est possiblement plus indicatrice du contenu d'un post de l'utilisateur par sa nature même: elle est censée expliquer de manière concise et efficace la question de l'utilisateur, afin que celle-ci trouve son public

```
[45]: #Ici, on a met plus de poids sur les titres en mettant Title plusieurs fois dans la nouvelle colonne que l'on créer  
db['Post'] = db['Title']+" "+db['Body']+db['Title']+" "+db['Title']+" "+db['Title']
```

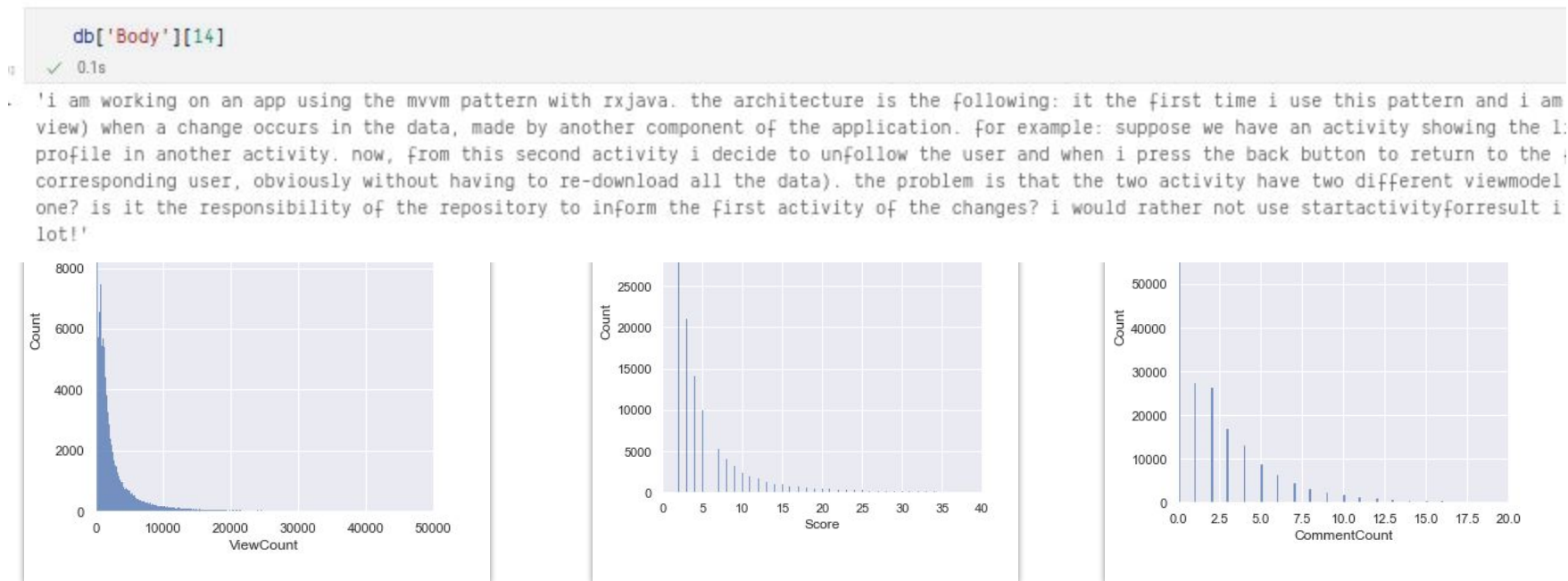
BDD en sortie de preprocessing

[46]:

	Id	CreationDate	Score	ViewCount	AnswerCount	CommentCount	FavoriteCount	Title	Body	Tags	Post
0	61339968	2020-04-21 09:16:35	147	387745	12	5	29	error message devtools fail load sourcemap cou...	try display image select local machine need lo...	[javascript, html]	error message devtools fail load sourcemap cou...
1	59823283	2020-01-20 12:26:53	114	266124	15	0	47	could load dynamic library cudart dll tensorfl...	instal latest version tensorflow via pip insta...	[python, python-3.x, tensorflow, keras, tensor...	could load dynamic library cudart dll tensorfl...
2	60296873	2020-02-19 09:24:42	109	253667	32	7	29	sessionnotcreatedexception message session cre...	currently new robot framework currently use la...	[selenium, google-chrome, selenium-webdriver, ...]	sessionnotcreatedexception message session cre...
3	60290309	2020-02-18 22:24:39	176	247270	30	3	18	error appear ngmodule import appmodule could r...	first time use firestore get error seem proble...	[angular, google-cloud-firestore]	error appear ngmodule import appmodule could r...
4	59601077	2020-01-05 14:54:15	189	244810	20	2	34	intellij error java error release version support	use intellij idea ultimate whenever try start ...	[java, intellij-idea]	intellij error java error release version supp...

Quelques graphiques sur nos données secondaires

- ❑ Les données numériques qui ont permis le filtrage de nos données initiales ont des courbes similaires et attendues



Exemple de nettoyage (1/3)

```
db['Body']][14]
```

✓ 0.2s

"I'm working on an app using the MVVM pattern with RxJava. The architecture is the following:\n\nIt's the first time i use this pattern and i'm (View) when a change occurs in the data, made by another component of the application.\n\nFor example: suppose we have an Activity showing the list of users in another Activity. Now, from this second Activity i decide to unfollow the user and when i press the back button to return to the first Activity (showing the list of users corresponding user, obviously without having to re-download all the data). \n\nThe problem is that the two Activity have two different ViewModels. How to handle this? Is it the responsibility of the Repository to inform the first Activity of the changes?\n\nI'd rather not use startActivityForResult if it's not the best solution.\n\nThanks a lot!\n"

```
db['Body'] := db['Body'].apply(lambda x: clean_text(x))
```

✓ 2.3s

```
db['Body']][14]
```

✓ 0.1s

.. 'i am working on an app using the mvvm pattern with rxjava. the architecture is the following: it the first time i use this pattern and i am in the first Activity (showing the list of users) when a change occurs in the data, made by another component of the application. for example: suppose we have an activity showing the list of users in another activity. now, from this second activity i decide to unfollow the user and when i press the back button to return to the first Activity (showing the list of users corresponding user, obviously without having to re-download all the data). the problem is that the two activity have two different viewmodels. how to handle this? is it the responsibility of the repository to inform the first activity of the changes? i would rather not use startActivityForResult if it's not the best solution.\n\nlot!'

Exemple de nettoyage (2/3)

```
db['Body'] = db['Body'].apply(lambda x: clean_punct(x, tags_top_500))
```

✓ 8.6s

```
db['Body'][14]
```

✓ 0.2s

'i am working on an app using the mvvm pattern with rxjava the architecture is the following it the first time i use th when a change occurs in the data made by another component of the application for example suppose we have an activity s another activity now from this second activity i decide to unfollow the user and when i press the back button to return obviously without having to re download all the data the problem is that the two activity have two different viewmodel responsibility of the repository to inform the first activity of the changes i would rather not use startactivityforres

Exemple de nettoyage (3/3)

```
db['Body'] = db['Body'].apply(lambda x: tokenizer(x))
```

✓ 1.1s

```
db['Body'] = db['Body'].apply(lambda x: stop_words_check(x, stop_words))
```

✓ 1.5s

```
db['Body'] = db['Body'].apply(lambda x: list_lemmatizer(x))
```

✓ 12.2s

```
db['Body'] = db['Body'].apply(lambda x: "-".join(x))
```

✓ 0.2s

```
db['Body'][14]
```

✓ 0.1s

'work app use mvvm pattern rxjava architecture follow first time use pattern sure best way update viewmodel consequently correspond vi follow like social app list select user open profile another activity second activity decide unfollow user press back button return fi download data problem two activity two different viewmodel make change make second activity affect viewmodel first one responsibility inject viewmodel first activity second one thank lot'

Approche Supervisée

Split train / test

- ❑ Nous faisons le split train / test set avec les modules de sklearn. A noter que nous faisons d'ores et déjà deux dataset pour nos algorithmes: un dataset basé sur BoW et un sur TF-IDF.
- ❑ Nous utiliserons TF-IDF dans le cadre des analyses supervisées. Notre database BoW est réservé à notre approche non supervisée
- ❑ Nous encodons les Tags via le module MultiLabelBinarizer

```
[60]: vectorizer_X_BoW = CountVectorizer()

      X_bow = vectorizer_X_BoW.fit_transform(X)

[61]: X_train_bow, X_test_bow, y_train_bow, y_test_bow = train_test_split(X_bow, y_encoded, test_size = 0.20, random_state = 0) # Do 80/20 split

[63]: X_tfidf = vectorizer_X.fit_transform(X)

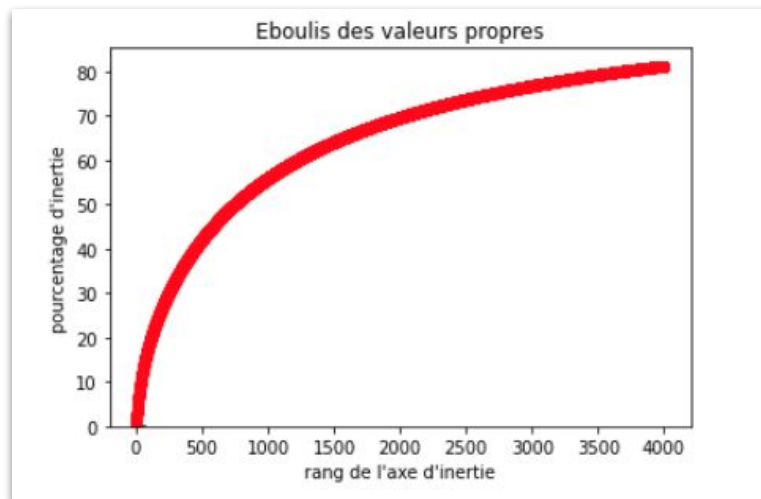
[64]: X_tfidf.shape

[64]: (47824, 30000)

[65]: X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y_encoded, test_size = 0.20, random_state = 0) # Do 80/20 split
```

Reduction de dimension de BoW via Truncated_SVD

- ❑ Nous utiliserons la méthode Truncated_SVD, pour réaliser une réduction de dimension de nos données, et ainsi éliminer du bruit de notre base de données, et accélérer le processus d'entraînement de nos algorithmes dans le cadre de BoW.
- ❑ Nous mettons ci-dessous le graphique permettant d'optimiser le nombre de composantes à choisir pour avoir 70% de la variance.
- ❑ Ainsi nous choisirons donc 2500 composantes pour notre réduction de dimension



Approche MultiOutputClassifier

- ❑ Nous utiliserons l'approche MultiOutputClassifier au dépend de OneVsRest. Toutefois notre approche multilabel aurait aussi pu être mis en place avec OneVsRest avec le bon paramétrage
- ❑ Nous utiliserons l'accuracy, F1 Score, Jaccard, Recall et la Précision pour comparer les résultats, et donnerons une attention particulière à Jaccard, Recall et la Précision

```
dummy = DummyClassifier()
svc = SGDClassifier(n_jobs=-1) #Hinge Loss / Reg = L2
lin_svc = LinearSVC()
lr = LogisticRegression(solver="liblinear")

metrics_results = pd.DataFrame(index=["Accuracy", "F1", "Jaccard", "Recall", "Precision"])

for classifier in [dummy, lr, lin_svc, svc]:
    clf = MultiOutputClassifier(classifier, n_jobs=-1)
    clf.fit(X_train_svd, y_train)
    y_pred = clf.predict(X_test_svd)
    metrics_score(classifier.__class__.__name__, df=metrics_results, y_true=y_test, y_pred=y_pred)
metrics_results
```

SGDClassifier avec GridSearchCV

- ❑ Un SVC lineaire (Hingeloss + Regularisation L2) via SGDClassifier est la méthode la plus efficace, car non seulement elle obtient les meilleurs résultats, mais c'est aussi la plus rapide à entraîner sur nos données. Nous faisons tourner un GridSearchCV sur cet algorithme mais l'excédent de performance est négligeable et ne vaut pas la peine du temps de calcul longs pour l'optimisation de hyperparamètres.

	DummyClassifier	SGDClassifier	Grid_opt_clf
Accuracy	0.0	0.163617	0.165708
F1	0.0	0.687760	0.688079
Jaccard	0.0	0.342946	0.344391
Recall	0.0	0.371426	0.372840
Precision	0.0	0.661086	0.662871

	Test labels	Supervised predicted labels
0	(pdf, python, user-interface)	()
1	(deep-learning, google-colaboratory)	(google-colaboratory, python)
2	(docker, gradle)	(docker,)
3	(flutter, flutter-layout)	(android, flutter)
4	(http, ionic-framework)	(ionic-framework,)
...
9560	(deep-learning, keras, python)	(keras, python, tensorflow)
9561	(javascript, reactjs, redux, typescript)	()
9562	(c++,)	()
9563	(graphql,)	()
9564	(java, spring, spring-boot)	()

9565 rows × 2 columns

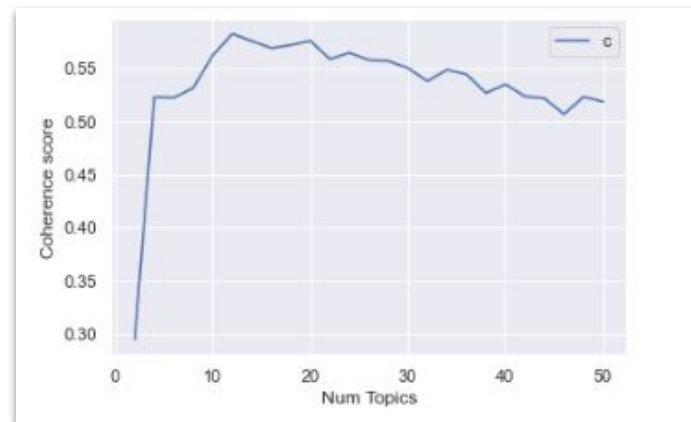
Approche Non-Supervisée

Gensim LDAMulticore

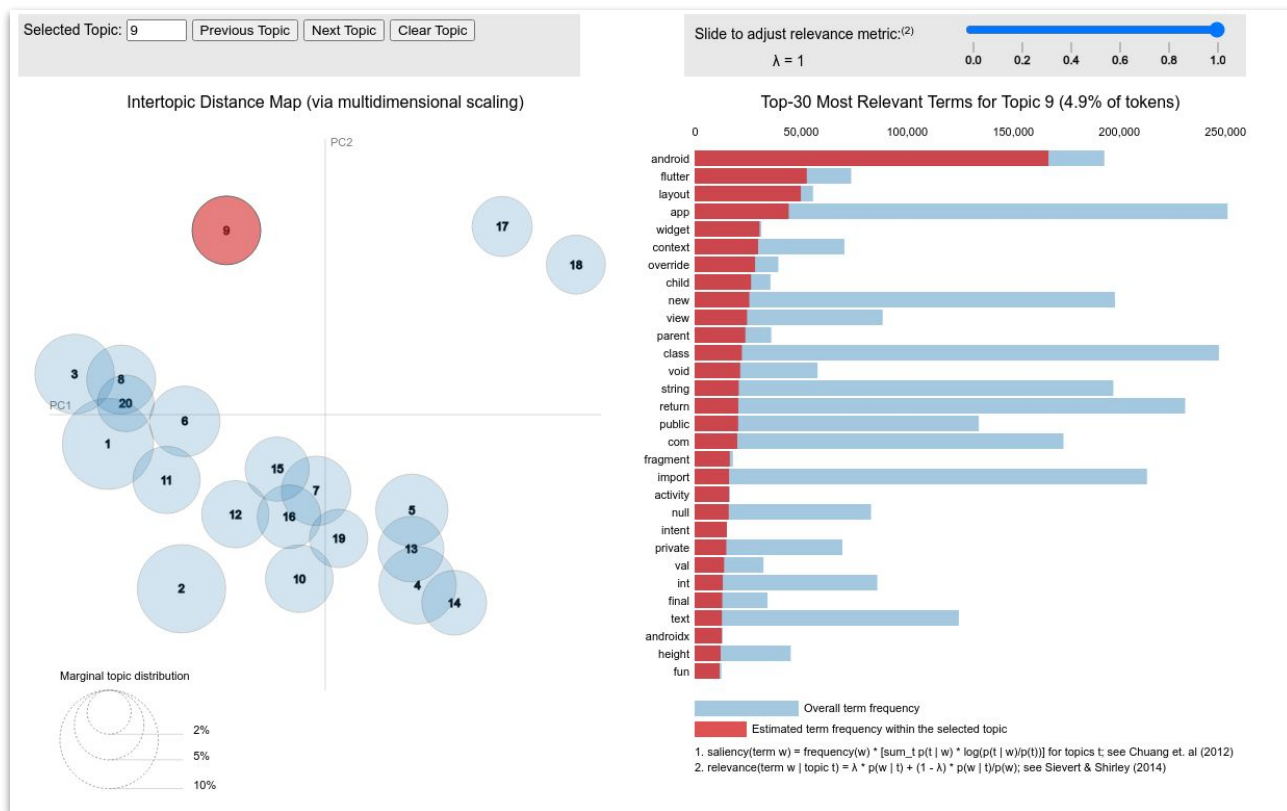
- ❑ Nous allons utiliser une approche LDA pour obtenir nos résultats, en utilisant une base de données de type BoW en input.
- ❑ L'implémentation de Gensim, nous permet d'utiliser la parallélisation lors de l'entraînement de l'algorithme et ainsi, raccourcir très fortement le temps de calcul.
- ❑ Nous faisons une boucle for pour déterminer la valeur de num_topics à utiliser en fonction de la cohérence_value obtenus pour chaque valeurs: 20 topics semble être une bonne valeur à utiliser

```
texts = db['Post'].apply(lambda x : x.split(' '))
id2word = corpora.Dictionary(texts)
id2word.filter_extremes(no_below=100)
bow_corpus = [id2word.doc2bow(text) for text in texts]
```

```
model = LdaMulticore(corpus=bow_corpus,
                    id2word=id2word,
                    num_topics=20,
                    random_state=42,
                    passes=10,
                    workers=7)
```



Visualisation des Topics / Words avec pyLDAvis



NN avec Keras

Embedding layer avec Keras

- ❑ Dans cette partie, nous optons pour une approche “intégrée” dans le réseau de neurones:
l'apprentissage se fait directement dans le réseau de neurones avec la couche Embedding. Nous apprenons donc un plongement de mots directement avec notre database. Cela permet d'avoir un réseau de neurones entraînés spécifiquement pour notre tâche.
- ❑ Nous avons une quantité de données suffisantes pour pouvoir avoir un plongement de mots robuste avec les couches Embeddings de Keras.
- ❑ Cette approche est différente des deux autres approches possibles avec Word2vec et GloVe

Structure de notre NN

- ❑ Cet exercice est notre première approche avec les réseaux de neurones. Nous avons choisi une structure simple largement basé sur la documentation disponible avec Keras et Tensorflow

```
[75] modelNN = Sequential()
    modelNN.add(Embedding(max_words, 300, input_length=maxlen))
    modelNN.add(Flatten())
    modelNN.add(Dense(2048,activation='relu'))
    modelNN.add(Dropout(0.5))
    modelNN.add(Dense(1024,activation='relu'))
    modelNN.add(Dense(y_encoded.shape[1]))
    modelNN.add(Activation('softmax'))
    modelNN.compile(optimizer='adam', loss='binary_crossentropy', metrics=['categorical_accuracy'])

    modelNN.summary()

    history = modelNN.fit(X_train_embed, y_train_embed,
                          epochs=7,
                          batch_size=1000,
                          validation_data = (X_val_embed,y_val_embed)
                          )
```

Comparaison des résultats des modèles

- ❑ Du fait de la nature de notre exercice de tagging, nous allons donner plus d'importance à Jaccard, Recall et Précision, notamment du fait que nous sommes dans un exercice multiclass et multilabel
- ❑ L'implémentation avec Keras nous donne un triplet de performance plus optimale. Nous choisissons ce modèle pour déployer notre API

Résultats sur toute la BDD

	DummyClassifier	SGDClassifier	LDA	Keras
Accuracy	0.0	0.167295	0.063539	0.182490
F1	0.0	0.689912	0.336445	0.613120
Jaccard	0.0	0.338312	0.093766	0.417798
Recall	0.0	0.366058	0.245234	0.554563
Precision	0.0	0.647289	0.130146	0.595299

Résultats sur 50,000 observations

	DummyClassifier	LogisticRegression	LinearSVC	SGDClassifier	Grid_opt_clf	LDA
Accuracy	0.0	0.138260	0.209015	0.163732	0.162264	0.067867
F1	0.0	0.591588	0.670009	0.675427	0.677179	0.331939
Jaccard	0.0	0.314760	0.434074	0.338838	0.336734	0.096840
Recall	0.0	0.340187	0.483031	0.366759	0.363957	0.244765
Precision	0.0	0.696379	0.727796	0.654149	0.653717	0.135055

Bonus:
Transformers avec
HuggingFace 🤗

Des résultats intéressants avec BERT

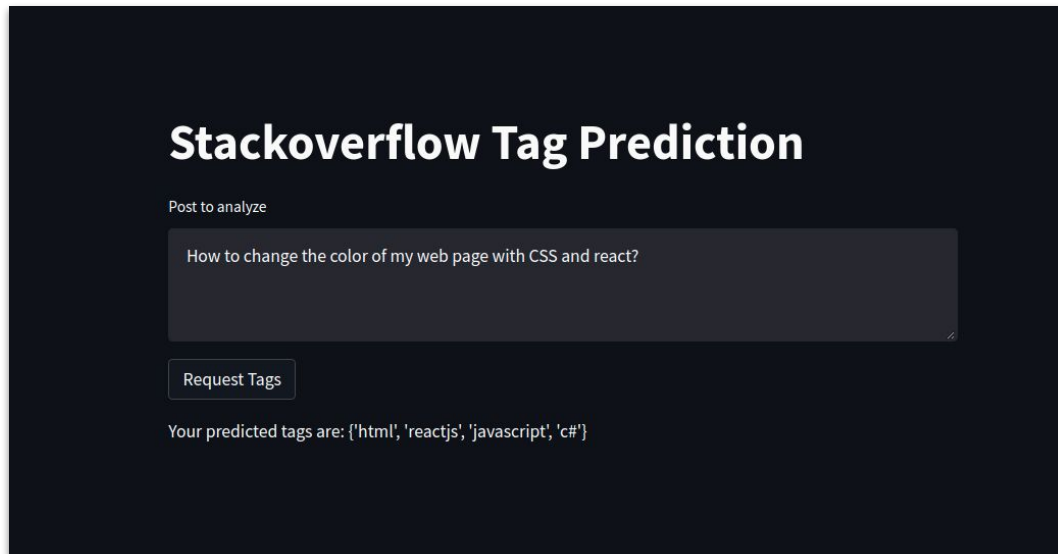
Nous avons fait cet exercice plus par curiosité, que par attente de performance. En effet, les transformers de Hugging Face ont connu une grande attention récemment, nous avons jugé opportun de faire un test. Nos résultats sont basés sur BERT

	Labels	Predicted_Labels
0	[javascript, html]	[error-handling, for-loop, testing, oop]
1	[python, python-3.x, tensorflow, keras, tensor...	[installation, tensorflow, multiprocessing, go]
2	[selenium, google-chrome, selenium-webdriver, ...	[exception, error-handling, colors, google-chr...
3	[angular, google-cloud-firestore]	[exception, error-handling, import, oop]
4	[java, intellij-idea]	[error-handling, java, testing, express]
5	[javascript, reactjs]	[error-handling, testing, dom, module]
6	[javascript, reactjs, redux, visual-studio-cod...	[error-handling, exception, testing, string]
7	[ios, xcode]	[iphone, mobile, ios, networking]
8	[java, spring-boot]	[error-handling, java, class, groovy]
9	[android, android-studio, kotlin]	[gradle, error-handling, kotlin, performance]

Déploiement avec MLFlow et Streamlit

Dashboard

Streamlit et MLflow nous permettent d'avoir un développement simple et rapide de notre API



The screenshot shows a web interface with a dark background. At the top, the title 'Stackoverflow Tag Prediction' is displayed in white. Below the title, there is a label 'Post to analyze' followed by a text input field containing the question 'How to change the color of my web page with CSS and react?'. A 'Request Tags' button is positioned below the input field. At the bottom, the output text reads 'Your predicted tags are: {'html', 'reactjs', 'javascript', 'c#'}

Stackoverflow Tag Prediction

Post to analyze

How to change the color of my web page with CSS and react?

Request Tags

Your predicted tags are: {'html', 'reactjs', 'javascript', 'c#'}

**Merci de votre
attention!**