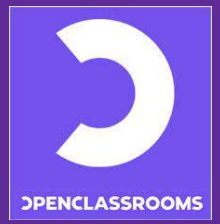
Projet 5 : Parcours Machine Learning
Catégorisez automatiquement des questions



Juillet 2023
Clara Yaïche
Étudiante en alternance NXP - OpenClassrooms
Parcours Machine Learning

La problématique

Natural language processing (NLP)

Topic modelling

How to make good reproducible pandas examples

Asked 9 years, 7 months ago Modified 5 months ago Viewed 50k times

220

This question's answers are a community effort. Edit existing answers to improve this post. It is not currently accepting new answers or interactions.

Having spent a decent amount of time watching both the `r` and `pandas` tags on SO, the impression that I get is that `pandas` questions are less likely to contain reproducible data. This is something that the R community has been pretty good about encouraging, and thanks to guides like this, newcomers are able to get some help on putting together these examples. People who are able to read these guides and come back with reproducible data will often have much better luck getting answers to their questions.

How can we create good reproducible examples for `pandas` questions? Simple dataframes can be put together, e.g.:

```
import pandas as pd
df = pd.DataFrame({'user': ['Bob', 'Jane', 'Alice'],
                   'income': [40000, 50000, 42000]})
```

But many example datasets need more complicated structure, e.g.:

- `datetime` indices or data
- Multiple categorical variables (is there an equivalent to R's `expand.grid()` function, which produces all possible combinations of some given variables?)
- MultiIndex or Panel data

For datasets that are hard to mock up using a few lines of code, is there an equivalent to R's `dput()` that allows you to generate copy-pasteable code to regenerate your datastructure?

`python` `pandas`

Sommaire

I. Les données, leur traitement et leur représentation
   1. Traitements des posts
   2. Approche du type “bag-of-word”
   3. Représentation en grande dimension : id2word, TF-IDF, sparse matrix, LSA

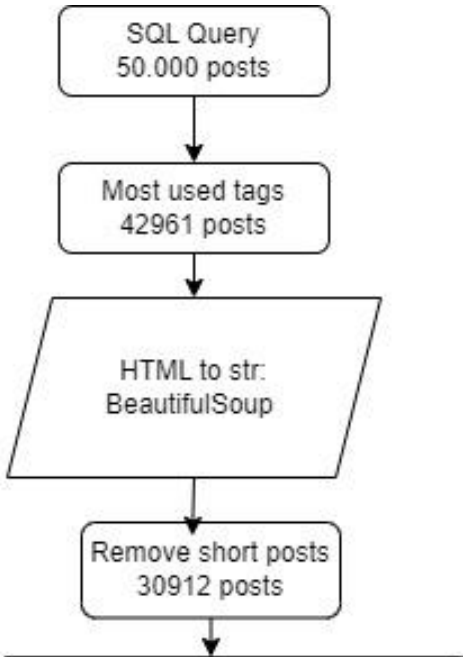
II. Le choix du modèle
   1. Approche non supervisé : LSA, LDA, K-means evaluation
   2. Approche supervisée : Régression logistique, SVM, arbres aléatoires
   3. Réseau de neurones avec embeddings : Word2Vec, USE and BERT

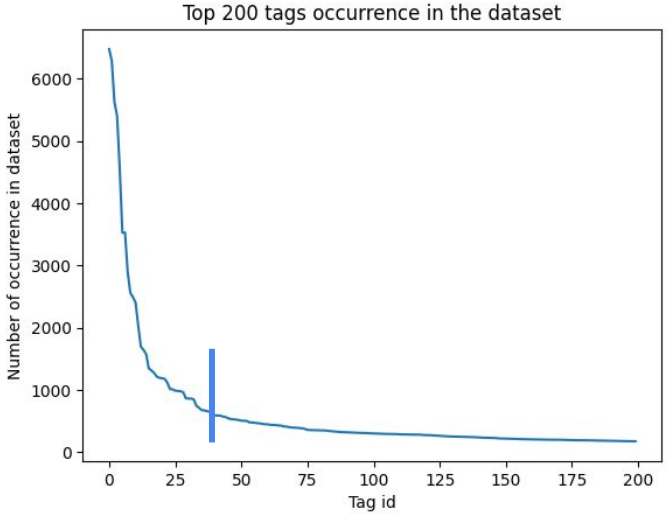
III. La Mise en production :

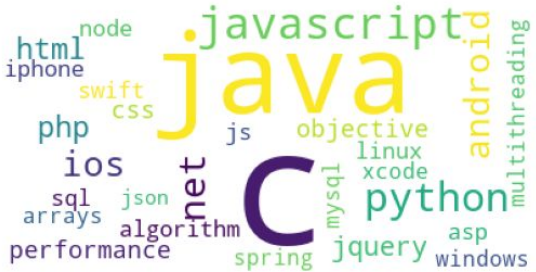
   1. Gestion de version : Git
   2. Streamlit et MLFlow

Les données, leur traitement et leur représentation

Traitements des posts

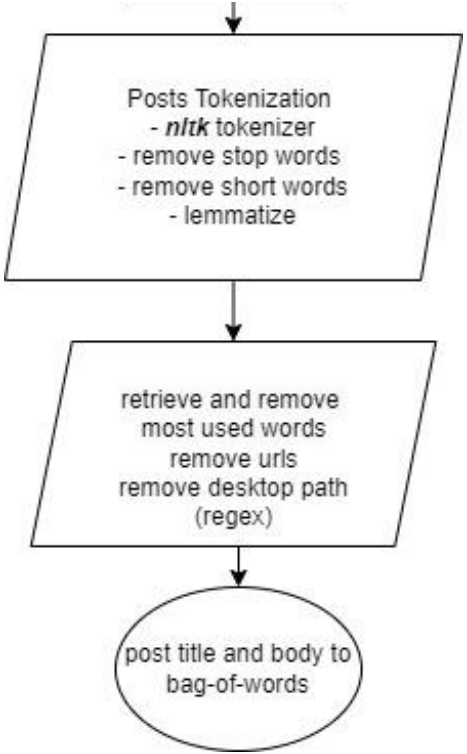






Les données, leur traitement et leur représentation

Approche du type “bag-of-word”



Let's tokenize this sentence !! This will make a great example

['tokenize', 'sentence', 'This', 'make', 'great', 'example']

['tokenize', 'sentence', 'great']

Nombre de tokens par post :

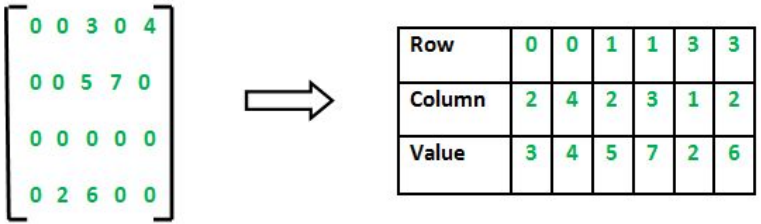
count	30912
mean	122
50%	87
min	15
max	2368

Choix particulier au projet : garder les majuscules et les minuscules, lemmatisation plutôt que stémmatisaton

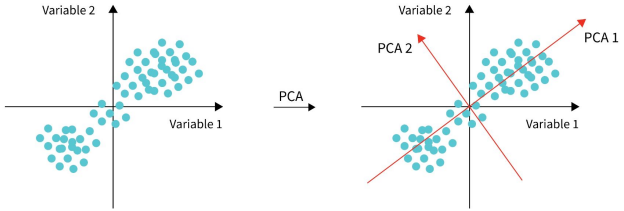
Les données, leur traitement et leur représentation

Représentation en grande dimension : id2word, TF-IDF, matrice creuse, LSA

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0



Représentation mémoire des matrices creuses



Réduction de dimension

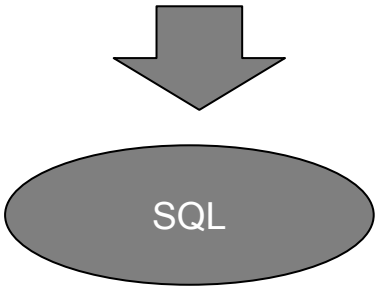
Le choix du modèle

Le choix du modèle

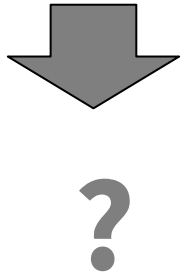
Approche non supervisé : LSA, LDA, K-means

LSA :

```
Component 4 ['query', 'database',
'date', 'connection', 'record',
'MySQL', 'column', 'SELECT',
'WHERE', 'FROM', 'table', 'row',
'month', 'Date', 'day', 'Server',
'session', 'product',
'transaction', 'timestamp']
```



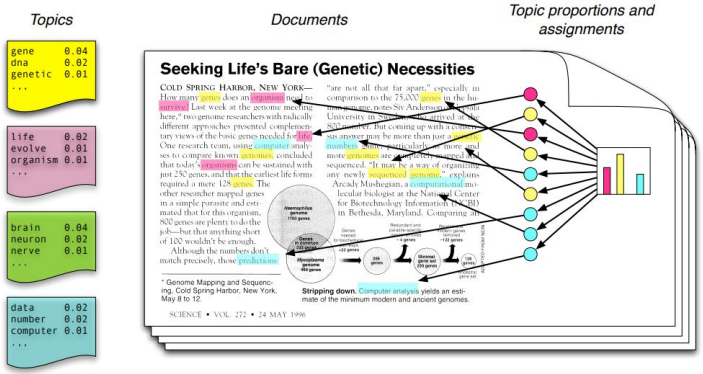

```
Component 3 ['dependency',
'Studio', 'compile', 'build',
'Visual', 'assembly', 'module',
'artifactId', 'groupId', 'package',
'Android', 'directory', 'file',
'install', 'service', 'folder',
'Windows', 'INFO', 'project',
'machine']
```



Le choix du modèle

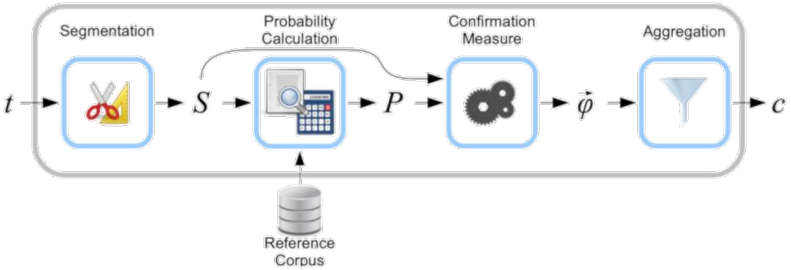
Approche non supervisé : LSA, LDA, K-means

LDA : Latent Ditrichlet Allocation



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

Mesure l'interprétabilité :



Choix des mesures de cohérence : c_v et u_mass

Le choix du modèle

Approche supervisée : Régression logistique, SVM, arbres aléatoires

How to make good reproducible pandas examples

Asked 9 years, 7 months ago Modified 5 months ago Viewed 50k times

220

This question's answers are a community effort. Edit existing answers to improve this post. It is not currently accepting new answers or interactions.

Having spent a decent amount of time watching both the `r` and `pandas` tags on SO, the impression that I get is that `pandas` questions are less likely to contain reproducible data. This is something that the R community has been pretty good about encouraging, and thanks to guides like this, newcomers are able to get some help on putting together these examples. People who are able to read these guides and come back with reproducible data will often have much better luck getting answers to their questions.

How can we create good reproducible examples for `pandas` questions? Simple dataframes can be put together, e.g.:

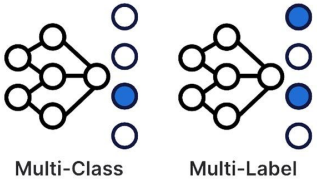
```
import pandas as pd
df = pd.DataFrame({'user': ['Bob', 'Jane', 'Alice'],
                   'income': [40000, 50000, 42000]})
```

But many example datasets need more complicated structure, e.g.:

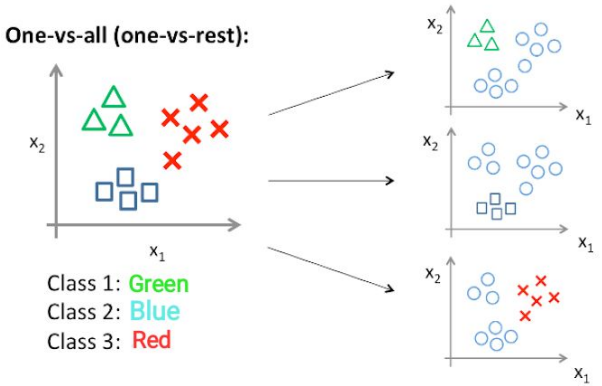
- `datetime` indices or data
- Multiple categorical variables (is there an equivalent to R's `expand.grid()` function, which produces all possible combinations of some given variables?)
- MultiIndex or Panel data

For datasets that are hard to mock up using a few lines of code, is there an equivalent to R's `dput()` that allows you to generate copy-pasteable code to regenerate your datastructure?

python pandas



MultiLabelBinarizer()

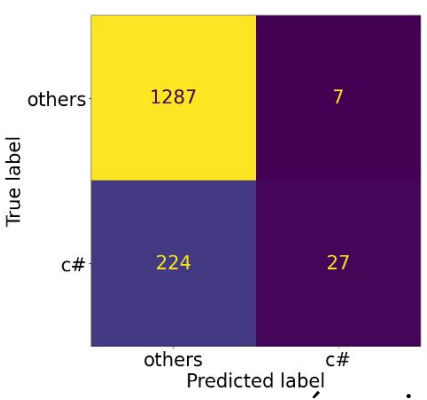


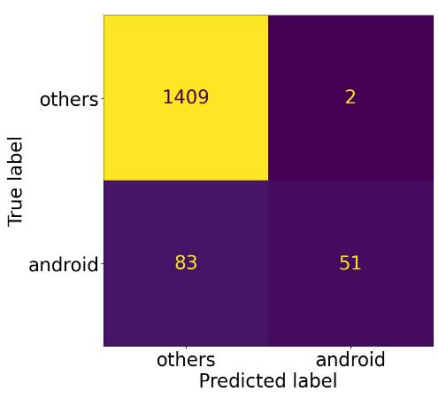
Regression logistique : One Vs Rest

SVM : SVC, One Vs One

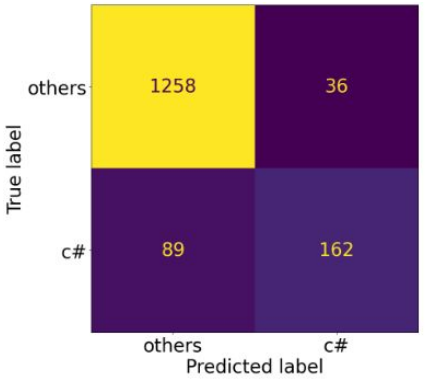
Le choix du modèle

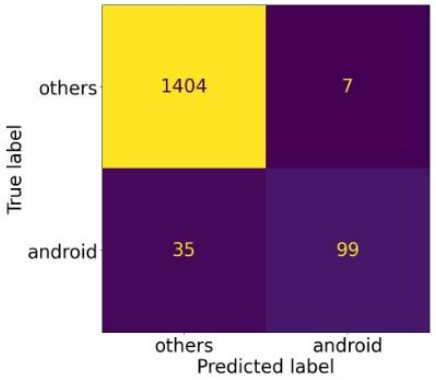
Approche supervisée : Régression logistique, SVM, arbres aléatoires



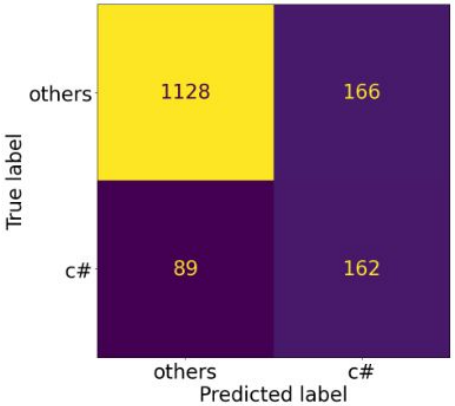


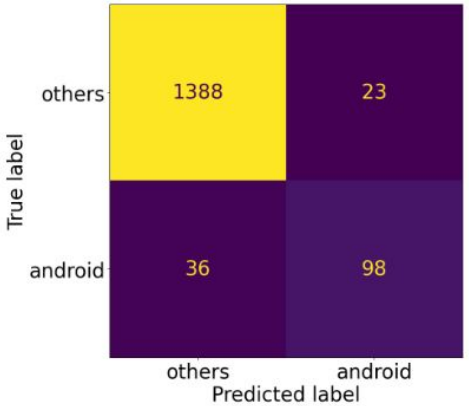
régression logistique





SVC

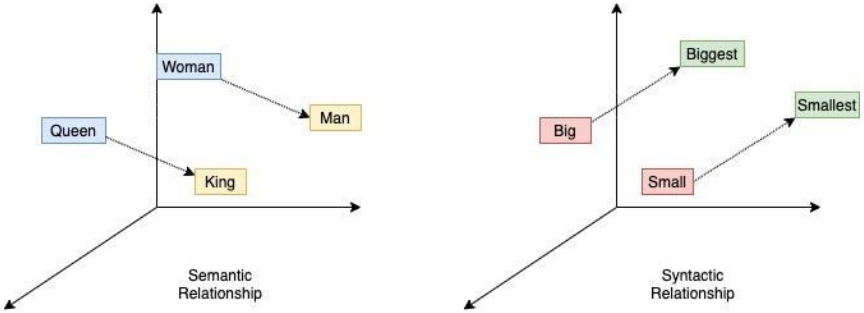




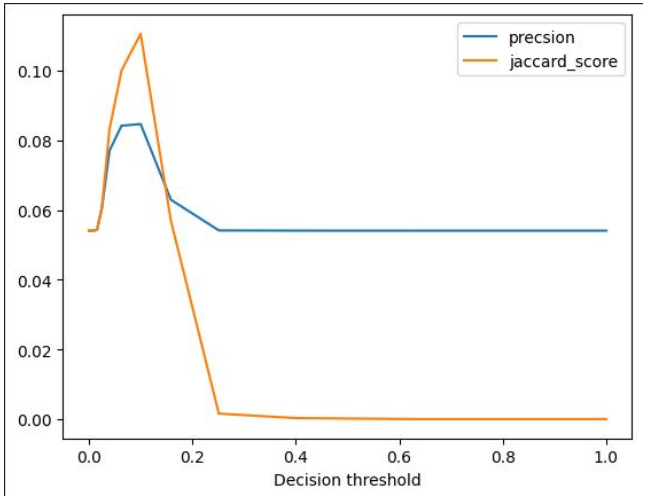
arbres aléatoires

Le choix du modèle

Approche supervisée : Neural Networks with embeddings : Word2Vec, USE and BERT

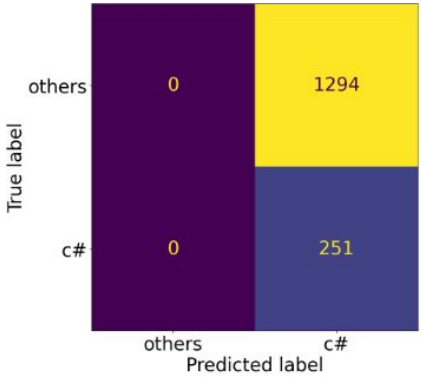


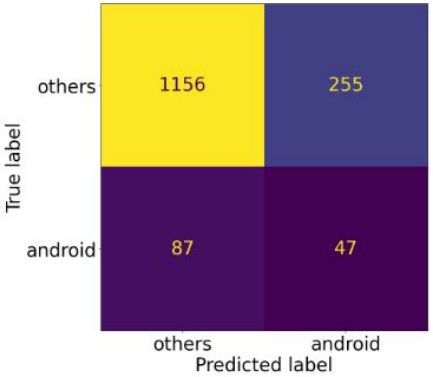
De la probabilité d'appartenance à l'appartenance à la classe : le choix du seuil



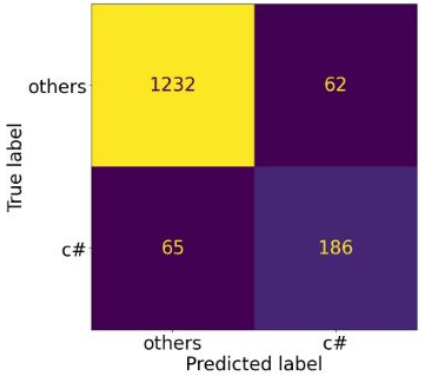
Le choix du modèle

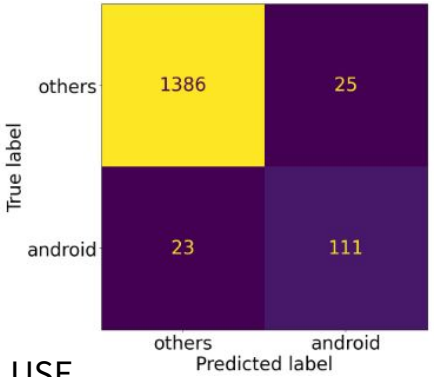
Approche supervisée : Neural Networks with embeddings : Word2Vec, USE and BERT



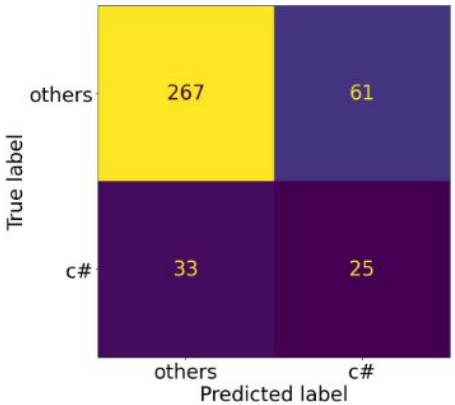


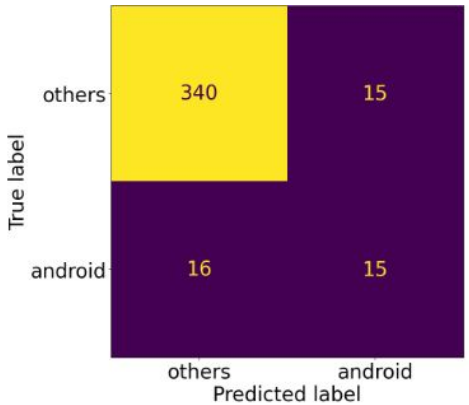
Word2Vec





USE





BERT

Le choix du modèle

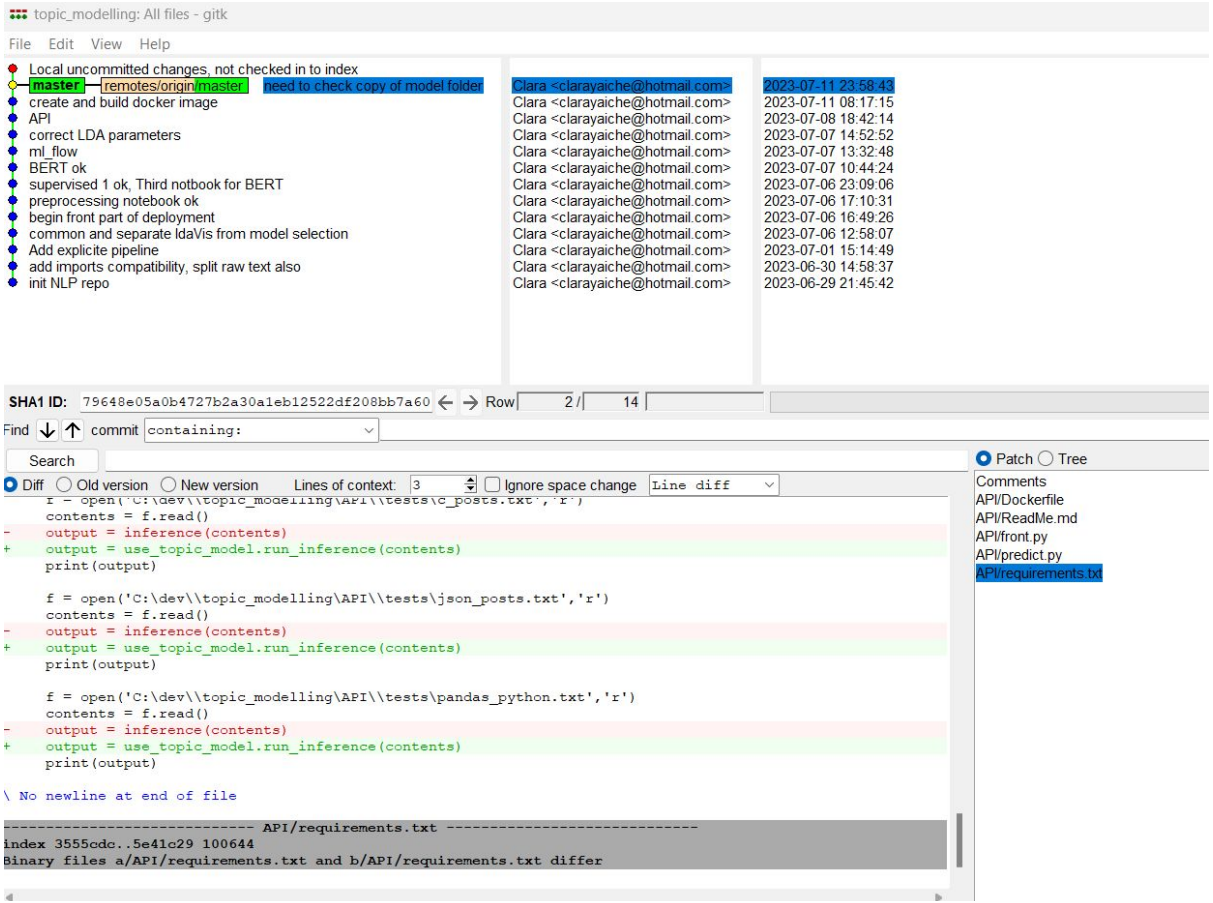
	u_mass_scores	c_v_scores
lsa	-7.234623	0.349950
lda	-7.871950	0.475715
kmeans	-12.306075	0.485754

	model	average_precision_scores	jaccard_scores
0	Dummy classifier	0.055033	0.000000
1	logistic : concatenation	0.139004	0.097722
2	SVM : concatenation	0.373852	0.382614
3	RandomForest	0.296040	0.345479
0	Word2Vec	0.084533	0.111220
1	USE	0.471676	0.508555
0	BERT	0.140975	0.188865

Déploiement

Mise en production : gestion de version

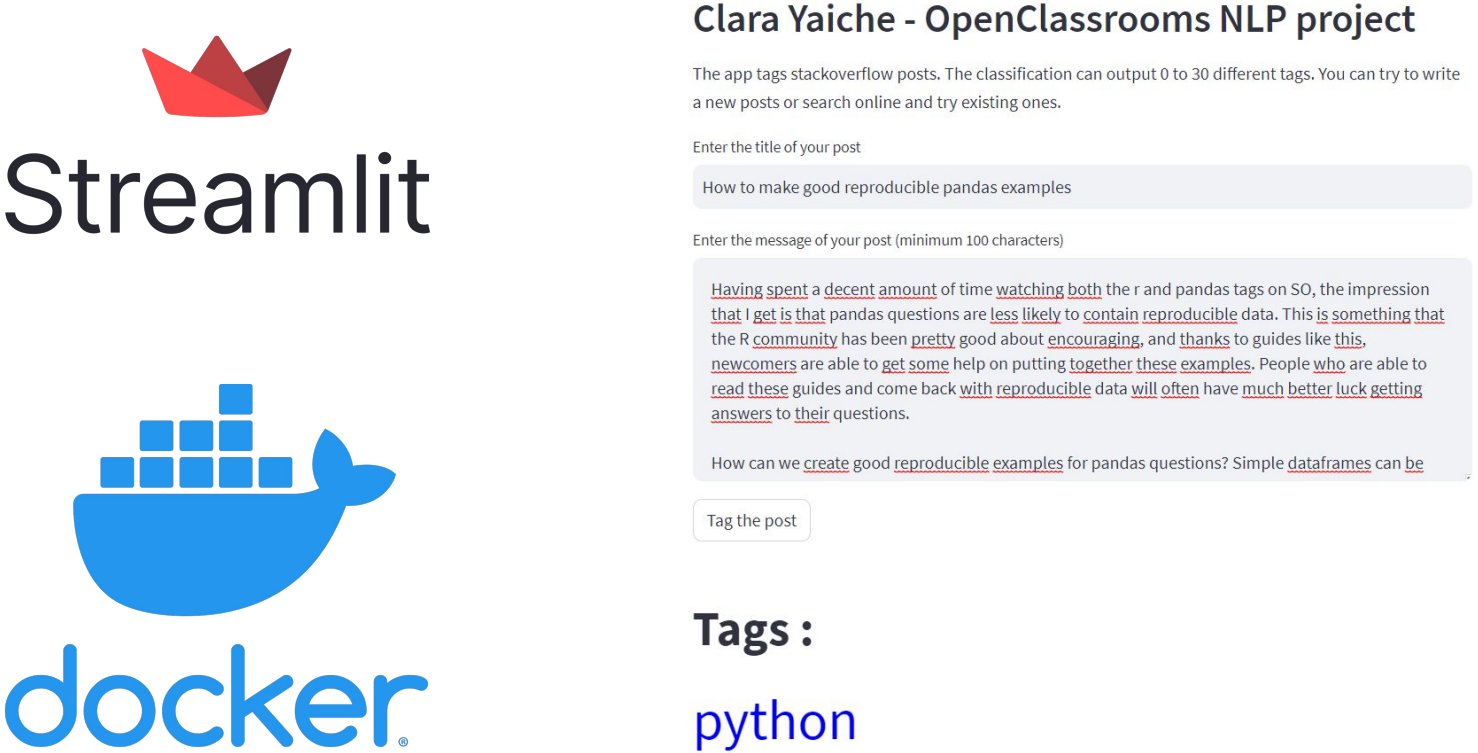
https://github.com/CYaiche/nlp_topic_modelling



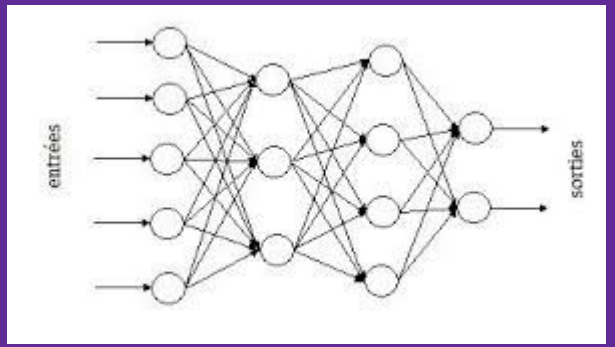
Mise en production : streamlit, Docker, Azur

Streamlit

docker



Perspectives



Annexes

Sources utiles : [Understanding Topic Coherence Measures]

$$(10.1)\quad Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$(10.2)\quad Precision = \frac{T_p}{T_p + F_p}$$

$$(10.3)\quad Recall = \frac{T_p}{T_p + T_n}$$

$$(10.4)\quad F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$