Système de suggestion de tag pour le site Stackoverflow

OPENCLASSROOMS

stackoverflow

Outline

- Introduction
  - Objectif de l'étude
  - Jeux de données
- Nettoyage
  - Extraction du texte à partir des données html
  - Numéros, ponctuation, caractères spéciaux
  - Tokenisation et lemmatisation
- Analyse exploratoire
  - Les tags plus fréquentes
  - Les mots du corpus plus fréquentes
  - Les mots du corpus associés aux 3 tags principaux
- Modélisation du corpus
  - Matrice des termes (TF, TF-IDF)
  - Modèles non-supervisés – optimisation et performance
  - Modèles supervisés
  - Réduction dimensionnelle par ACP
- Conclusions
  - Choix des meilleur algorithmes de prédiction pour une API

Modalités de la soutenance

5 min - Présentation de la problématique, de son interprétation et des pistes de recherche envisagées.

5 min - Présentation du cleaning effectué, du feature engineering et de l'exploration.

10 min - Présentation des différentes pistes de modélisation effectuées.

5 min - Présentation du modèle final sélectionné et résultats.

5 à 10 minutes de questions-réponses.

Objectif du modèle

Dans cette étude avec un apprentissage supervisé ou non, on essaye de trouver le meilleur modèle pour

- Prédire les mots-clés d'une question posée sur le site Stackoverflow

- utiliser ce modèle avec une API

Données du départ

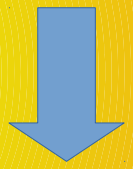
outil d'export de données avec une requête SQL:

"stackexchange explorer"

```
SELECT
      TOP 50000 ViewCount,
    CreationDate,
    Title,
    Body,
    Tags,
    Score,
    CommentCount,
    AnswerCount,
    FavoriteCount

FROM Posts
WHERE
    FavoriteCount > 10
    AND AnswerCount > 1
    AND Score > 100
ORDER BY Score DESC
```

- Le 50000 questions les plus consultées
- Bien notées (note >100)
- Avec au moins une réponse



Title	Body	Tags
Why is processing a sorted array faster than p...	<p>Here is a piece of C++ code that shows some...	<java><c++><performance><optimization><branch-...
How do I undo the most recent local commits in...	<p>I accidentally committed the wrong files to...	<git><version-control><git-commit><undo><pre-c...
How do I delete a Git branch locally and remot...	<p>I want to delete a branch both locally and ...	<git><version-control><git-branch><git-push><g...
What is the difference between 'git pull' and ...	<p>What are the differences between <code>git ...	<git><version-control><git-pull><git-fetch>
What is the correct JSON content type?	<p>I've been messing around with <a href="http...	<json><http-headers><content-type>

Corpus

Mots-clés

Traitement du texte non-structuré

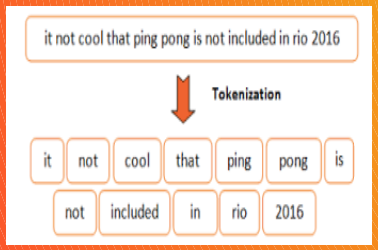
- **BeautifulSoup** :

  extraction du texte de chaque document/question dans le « Body » à partir du contenu en html (« soup »)

- Jointure des colonnes **« Body » et « Title »**

- **Pre-traitement** :

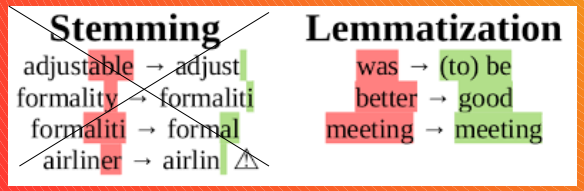
  filtrage des chiffres, ponctuation, caractères spéciaux, mots qui n'ont que peu d'intérêt sémantique (stopwords)



- **Tokenisation** : décomposition du texte en mots

- **Lemmatisation** : remplacement de chaque mot par sa forme canonique

Pas utile dans ce cas la



employée pour deux raisons :

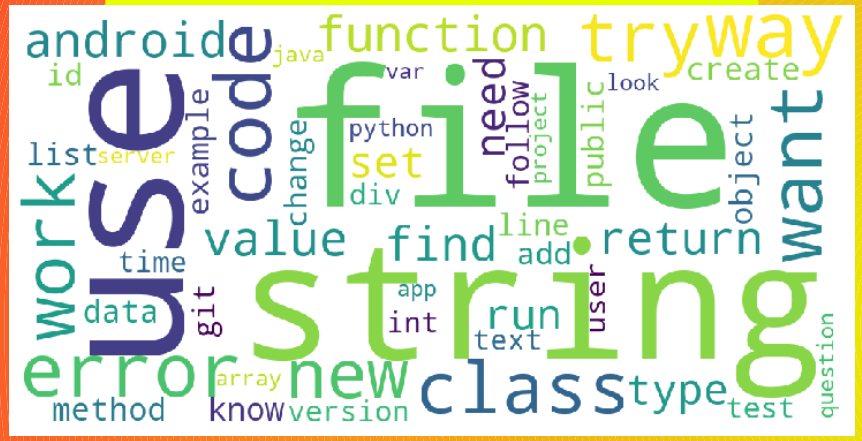
Donner le même sens à des mots très proches mais d'un genre différent (ou éliminer le pluriel, etc…)

Réduire la sparsité des matrices utilisées dans les algorithmes (voir partie suivante sur TF-IDF)

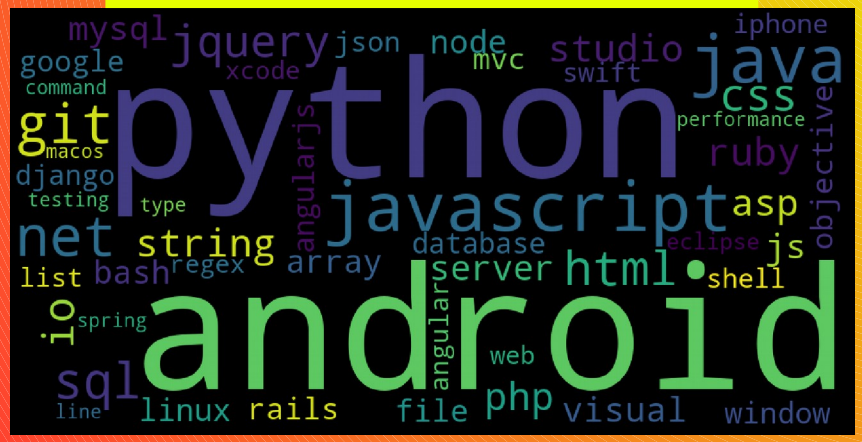
Most cited tags: associated body text

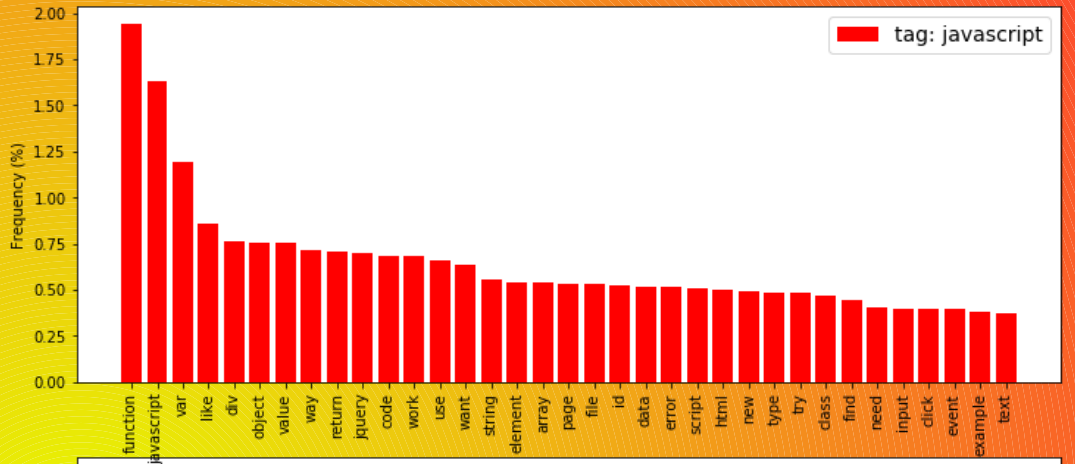
Exploration des données

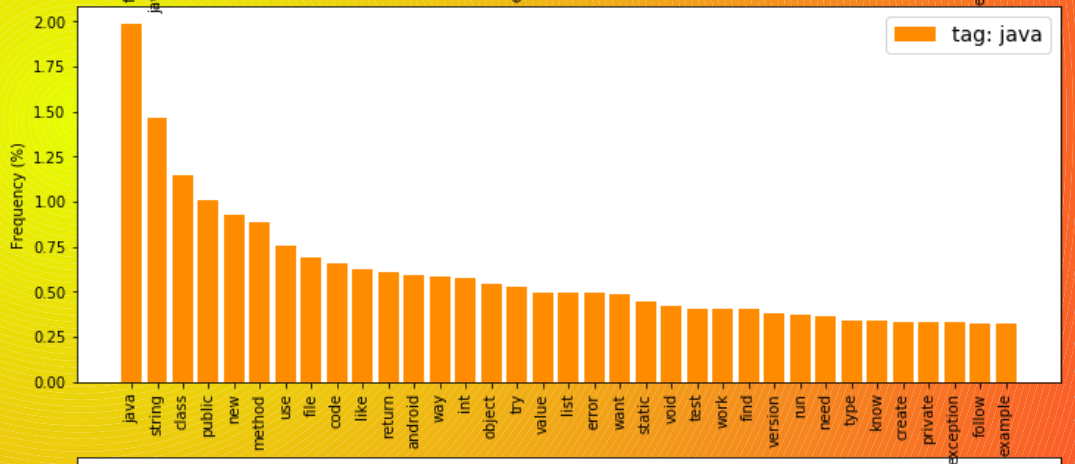
Corpus

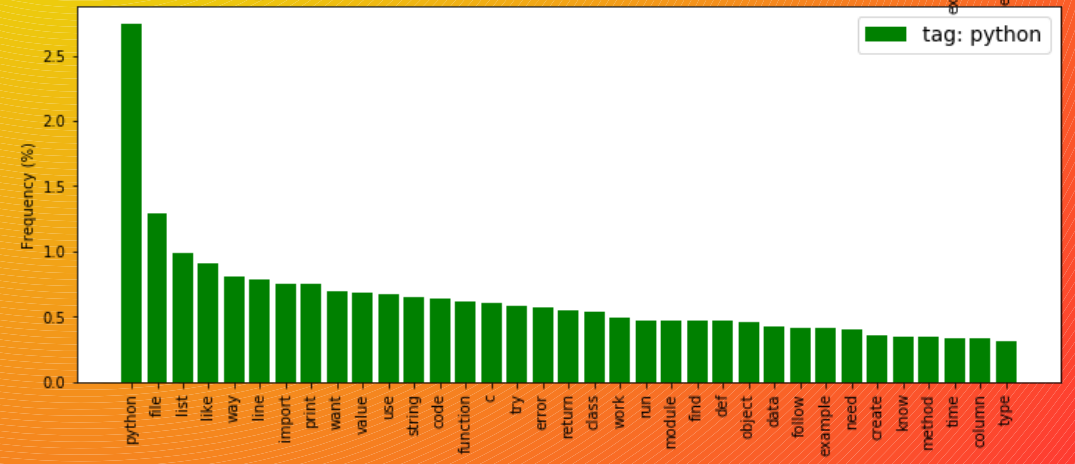


Mots-clé







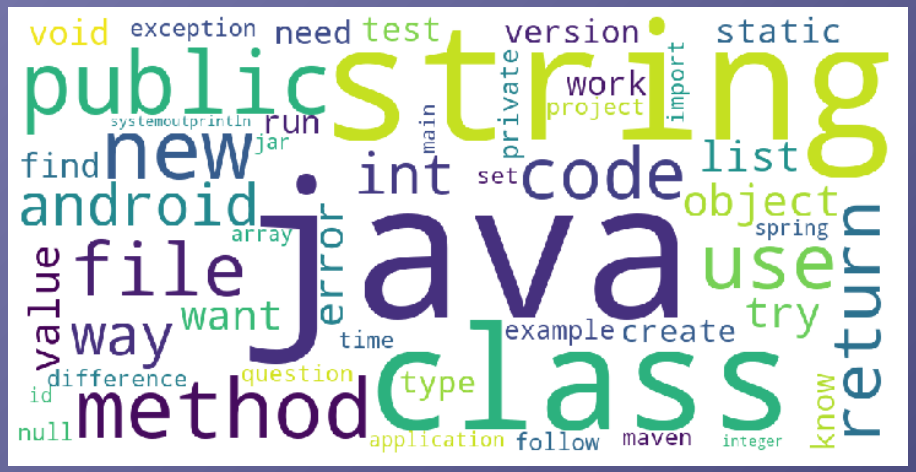


Vectorisation des données

Données texte

Title	Body	Tags
Why is processing a sorted array faster than p...	<p>Here is a piece of C++ code that shows some...	<java> <c++><performance> <optimization> <branch-...
How do I undo the most recent local commits in...	<p>I accidentally committed the wrong files to...	<git><version-control><git-commit> <undo><pre-c...
How do I delete a Git branch locally and remot...	<p>I want to delete a branch both locally and ...	<git><version-control><git-branch> <git-push><g...

Tokens



android	answer	api	app
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	1	1	0

Vecteurs binaires correspondants au tokens

Matrice des mots-clé

Tags

MultilabelBinarizer

+ pandas.Dataframe
columns=one_hot.classes_

```
(37175, 8578)
```

	.a	.bash-profile	.d.ts	.htaccess	.net	.net-2.0	.net-3.0	.net-3.5	.net-4.0	.net-4.5	...	zipfile	zipper	zlib	zombie-process	zoo	zoom
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

Dimensions Matrice creuse (37175, 8578)

Fonction python « filter »

```
y_bin = y_bin.filter(items=popular_tags)

y_bin[:2]
```

On garde que le colonnes correspondants aux 110 tags les plus utilisés

	javascript	python	java	c#	android	html	git	css	jquery	c++	...	types	java-8	dom	svn	entity-framework	variables	mat
0	0	0	1	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0

Dimensions Matrice creuse (37175, 110)

Matrice à prédire, en suite partagée 80 % entraînement (y_{train}), 20 % validation y_{test}

TF-IDF

Approche bag-of-words (bow) permettant de représenter les mots d'un document à l'aide d'une matrice de nombres. Le terme bow signifie que l'ordre des mots dans la phrase n'est pas pris en compte

Sélection des mots par importance dans le corpus

n est le nombre de documents et t∈d le nombre de documents où le terme est présent

$$idf(t) = \log\left(\frac{n}{t \epsilon D}\right)$$

$$tf(t) = \frac{n(t)}{N}$$

fréquence du terme dans un document

	words	idf_weight	count	TF_IDF score
0	file	2.736351	17370	47530.417314
1	android	3.915541	10096	39531.297872
2	string	3.048507	12716	38764.813379
3	class	3.145947	11375	35785.151931
4	error	3.020527	11575	34962.603081
5	use	2.386402	14324	34182.821721
6	like	2.264955	14620	33113.642445
7	function	3.100097	10272	31844.193251
8	value	3.125460	9943	31076.444477
9	code	2.616212	11800	30871.307214

$tfidf(t) = tf(t) \times idf(t)$

Matrice des termes TF-IDF

```
tfidfVectorizer = TfidfVectorizer(norm=None,analyzer='word',min_df = 5, max_df = 0.8, ngram_ran
ge=(1,2),max_features = num_features, use_idf=True)
```

```
# TF-IDF matrices
TF_IDF = tfidfVectorizer.fit_transform(df_base['Lemma'])
```

Sélection des 220 avec le meilleur TF-IDF

(37175, 220)

	able	access	add	allow	android	answer	api	app	application	array	...
0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	20.33771	...
1	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.00000	...
2	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.00000	...
3	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.00000	...
4	0.0	0.0	0.0	0.0	0.0	3.899777	4.660661	0.0	0.0	0.00000	...

Matrice à modéliser, en suite partagée 80 % entraînement (X_{train}), 20 % validation X_{test}

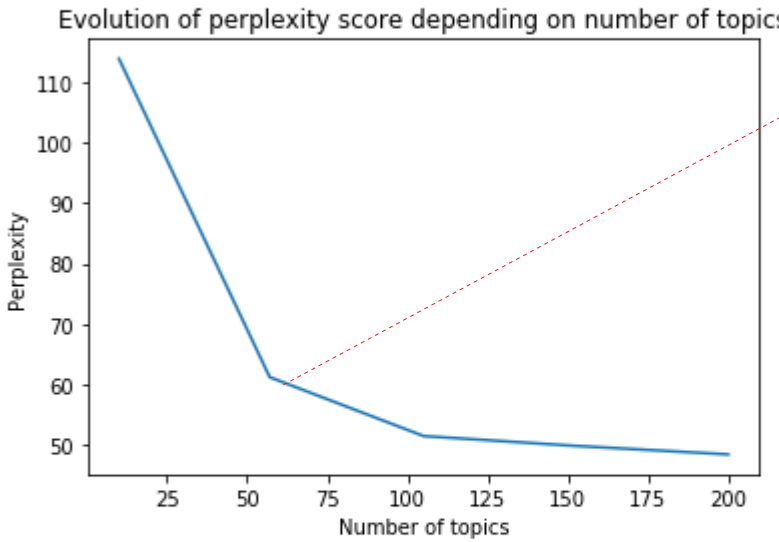
Modélisation non-supervisée

- Extraction des thèmes
- Prédiction des tags à partir des thèmes
- Performance

Extraction non-supervisé des tags : LDA et NMF

```
process_time_starts()
lda(tfidfVectorizer,X_tfidf_train, X_tfidf_test )
time_elapsed()
```

```
Extracting term frequency features for LDA...
```




```
The process took: 0hour:2min:37sec
```

2.1.2 Recommendation algorithm based on unsupervised topic modeling

Topics are automatically assigned based on word and component weight

```
n_topics = 55 # based on perplexity calculations
```

```
def Recommend_tags(text, model):
```

Nombre optimal de thèmes

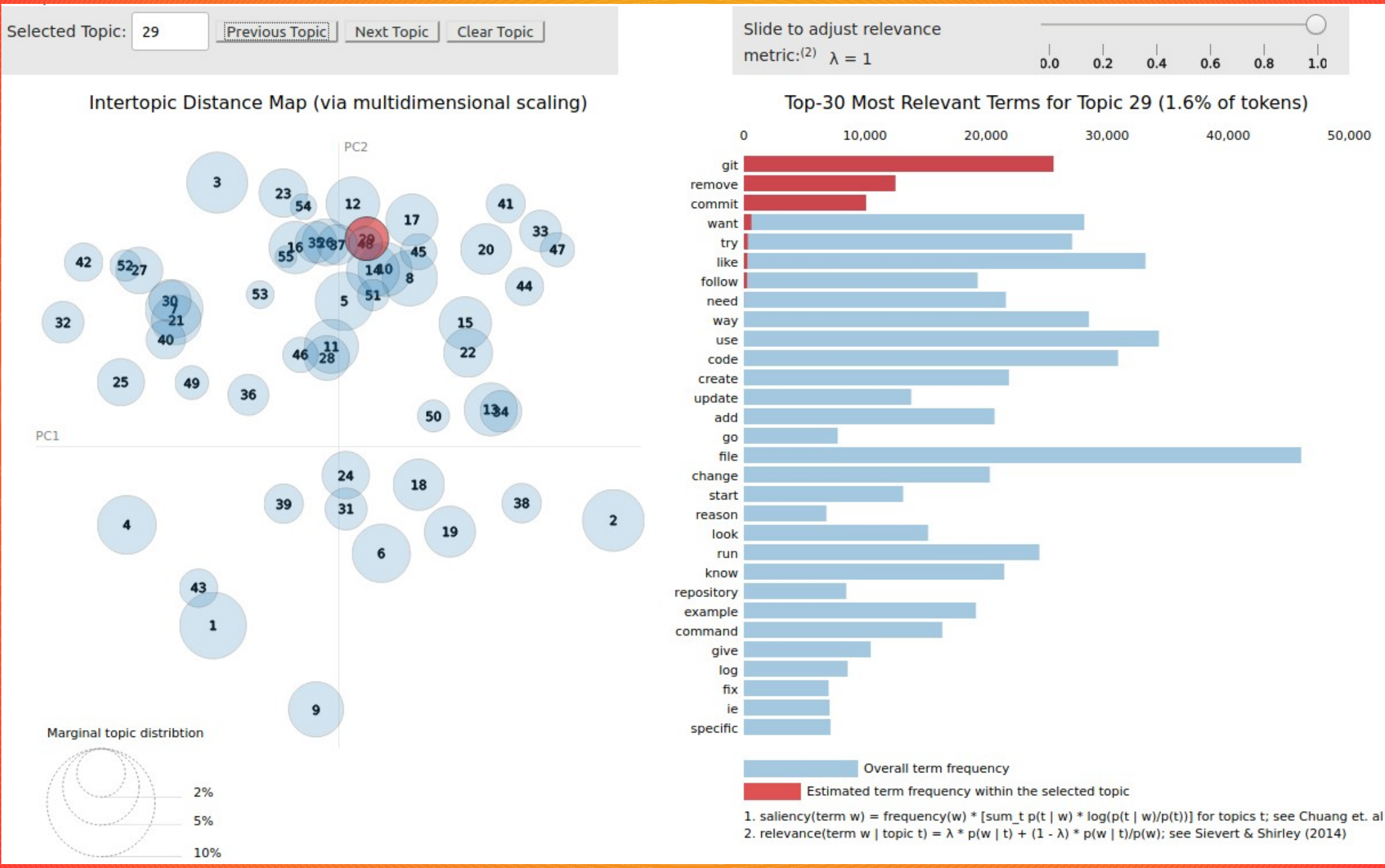
Choix de l'algorithme

- Latent Dirichlet Allocation (LDA)
- Negative Matrix Factorization (NMF)

Assignation automatique des tags

Thème plus probables par le mots du document avec la note TF-IDF plus élevée

Groupement du corpus par thèmes



Modèles non-supervisés : performance

On compare la modélisation avec LDA ou NMF à cela obtenue en utilisant l'extraction automatique des mots-clé avec le paquet «YAKE ! »

	Lemma	taglist (cible)	LDA_tags	NMF_tags	YAKE tags
0	['instal', 'libv', 'gem', 'o', '+', 'try', 'in...	gem rubygems osx-mavericks osx-yosemite libv8	find error build install user	error package compile install option	usersmervmgemsruby target cxx libv error
1	['right', 'leave', 'support', 'twitter', 'boot...	css html twitter-bootstrap twitter-bootstrap-3	create new script difference element	project compile build	bootstrap leave support twitter rtl
2	['operator', 'mysql', 'work', 'code', 'write',...	mysql sql operators spaceship-operator	name mysql source set error	error null server sql thread	mysql query error operator work
3	['warn', 'implicit', 'declaration', 'function'...	c compiler-warnings	function error import library size	function return	warn implicit declaration function compiler
4	['exportimport', 'job', 'jenkins', 'possible',...	jenkins	different test reference app number		job exportimport jenkins exchange search

```
y_tfidf_YAKE = tfidfVectorizer.transform(df_test['YAKE tags'])
print('Dummy score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_dummy.reshape(1,-
1)))
print('YAKE score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_YAKE.reshape(1,-1)))
print('LDA score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_LDA.reshape(1,-1)))
print('NMF score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_NMF.reshape(1,-1)))
```

```
Dummy score: [[0.05342378]]
YAKE score: [[0.29256526]]
LDA score: [[0.18009483]]
NMF score: [[0.18110464]]
```

Le plus proche au tags à prédire

Modélisation supervisée

- Prédiction des tags à partir des questions
- Performance

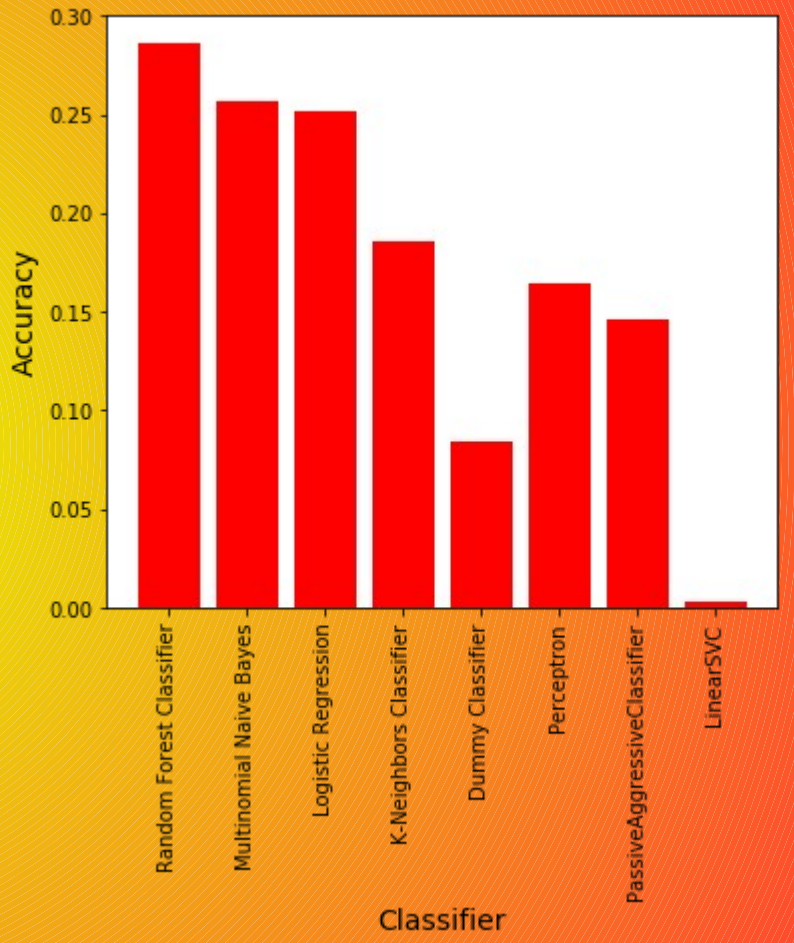
Modèles supervisés : exactitude de prédictions

Optimisation du Random Forest

N. features : 100

Max depth : 30000

Seuil de probabilité : 0.3




```
# check the cleaned-up probability matrix
print(df_probs.shape, y_test_bin.shape)
df_probs.head()
```

```
(7435, 110) (7435, 110)
```

	javascript	python	java	c#	android	html	git	css	jquery	c++	...	types	java-8	dom	svn
0	0.01	0.49	0	0.02	0	0	0.04	0	0	0.02	...	0	0	0	0
1	0.08	0.01	0.15	0.14	0.02	0.01	0.01	0.03	0	0.02	...	0	0	0	0
2	0	0.01	0.05	0.02	0	0.01	0.01	0	0	0	...	0	0	0	0
3	0.0613889	0.11	0	0.01	0	0	0	0	0	0.327778	...	0	0	0	0
4	0.01	0.0139102	0.0161706	0.1	0.0184408	0	0	0	0	0.0895	...	0.05	0	0	0.0153175

5 rows × 110 columns

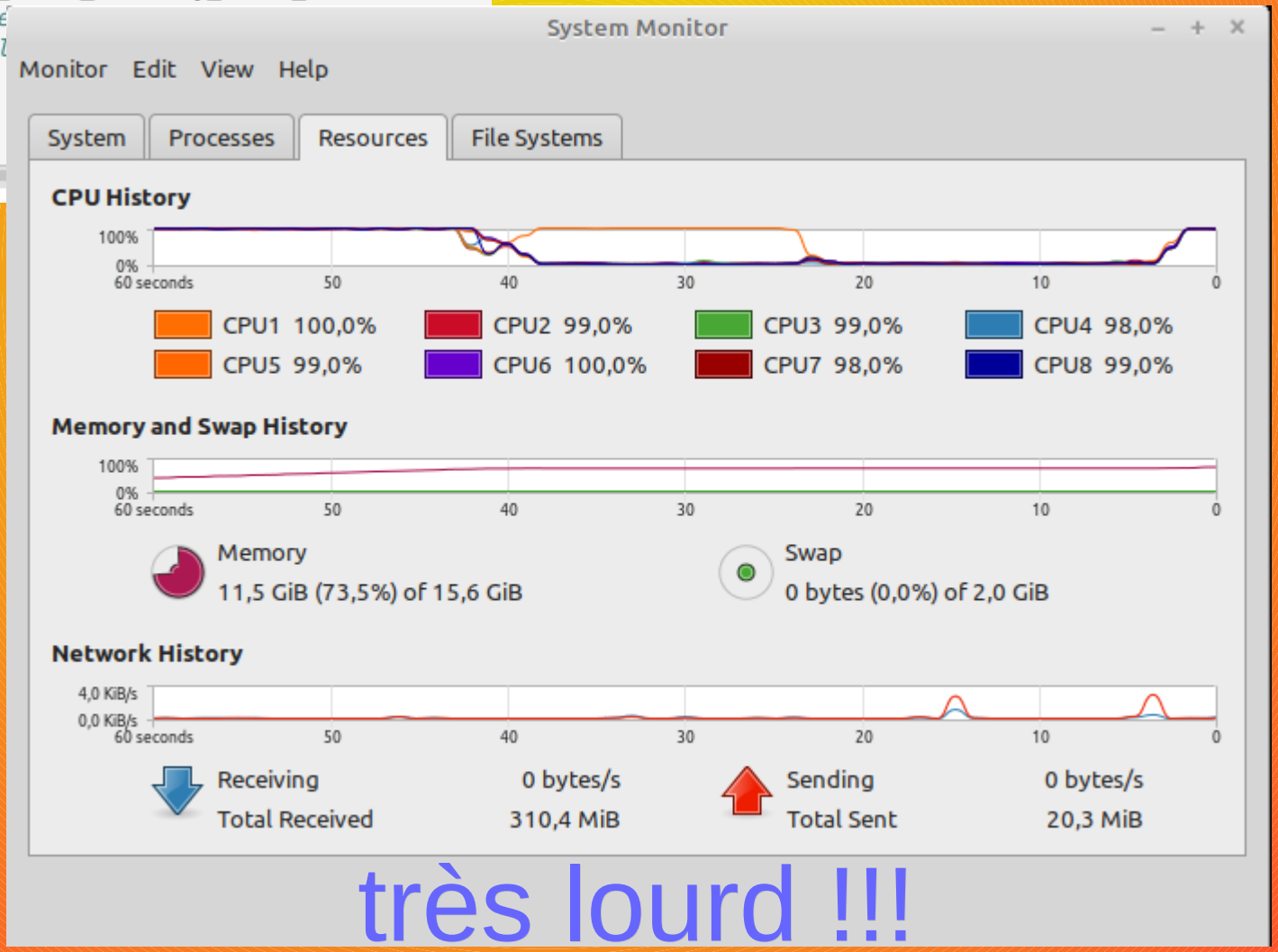
```
y_pred_RF = pd.DataFrame(index=y_test_bin.index,columns=y_test_bin.columns).fillna(0).reset_ind
ex(drop=True) # create a dataframe with zerous only
m, n, = y_pred_RF.shape
absolute_min_threshold = 0.3 # minimum accepted probability to count as tag
```

Le Random Forest est le plus exacte mais...

3.1.4 Run Random Forest Classifier

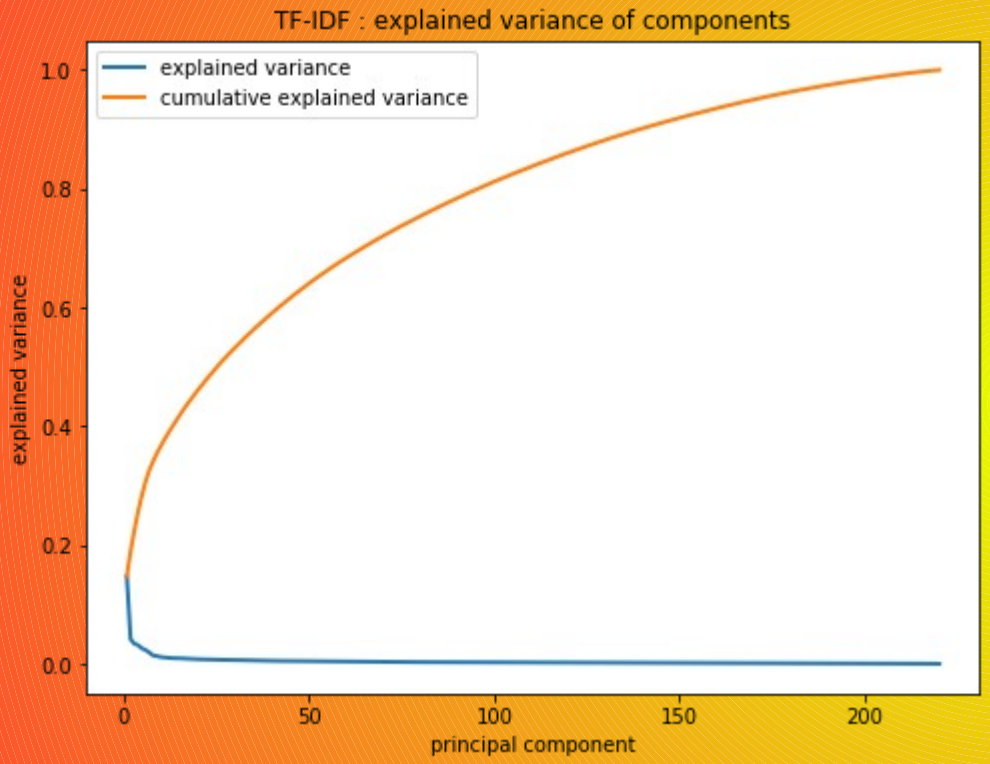
```
[*]: process_time_starts()
rf = RandomForestClassifier(max_depth=30000, random_
rf_clf = rf.fit(X_tfidf_train, y_train_bin)
# apply the traine
#y_pred_RF = rf_cl

time_elapsed()
```



très lourd !!!

Réduction des nombre des variables par ACP



classifiers	220 features Accuracy	100 features (ACP) Accuracy
Dummy Classifier	0.084	0.084
Logistic Regression	0.252	0.062
Perceptron	0.164	0.014
Passive Aggressive Classifier	0.146	0.055

Moins lourd, mais avec une performance nettement inférieure

Modèles supervisés vs. non-supervisés

	Lemma	taglist	LDA_tags	NMF_tags	YAKE tags	Random_Forest_tags	LogisticRegression_tags
0	['instal', 'libv', 'gem', 'o', '+', 'try', 'in...	gem rubygems osx-mavericks osx-yosemite libv8	find error build install user	error package compile install option	usersmervmgemsruby target cxx libv error	python	python
1	['right', 'leave', 'support', 'twitter', 'boot...	css html twitter-bootstrap twitter-bootstrap-3	create new script difference element	project compile build	bootstrap leave support twitter rtl		
2	['operator', 'mysql', 'work', 'code', 'write',...	mysql sql operators spaceship-operator	name mysql source set error	error null server sql thread	mysql query error operator work	mysql	mysql
3	['warn', 'implicit', 'declaration', 'function'...	c compiler-warnings	function error import library size	function return	warn implicit declaration function compiler	c++ c	
4	['exportimport', 'job', 'jenkins', 'possible',...	jenkins	different test reference app number		job exportimport jenkins exchange search		

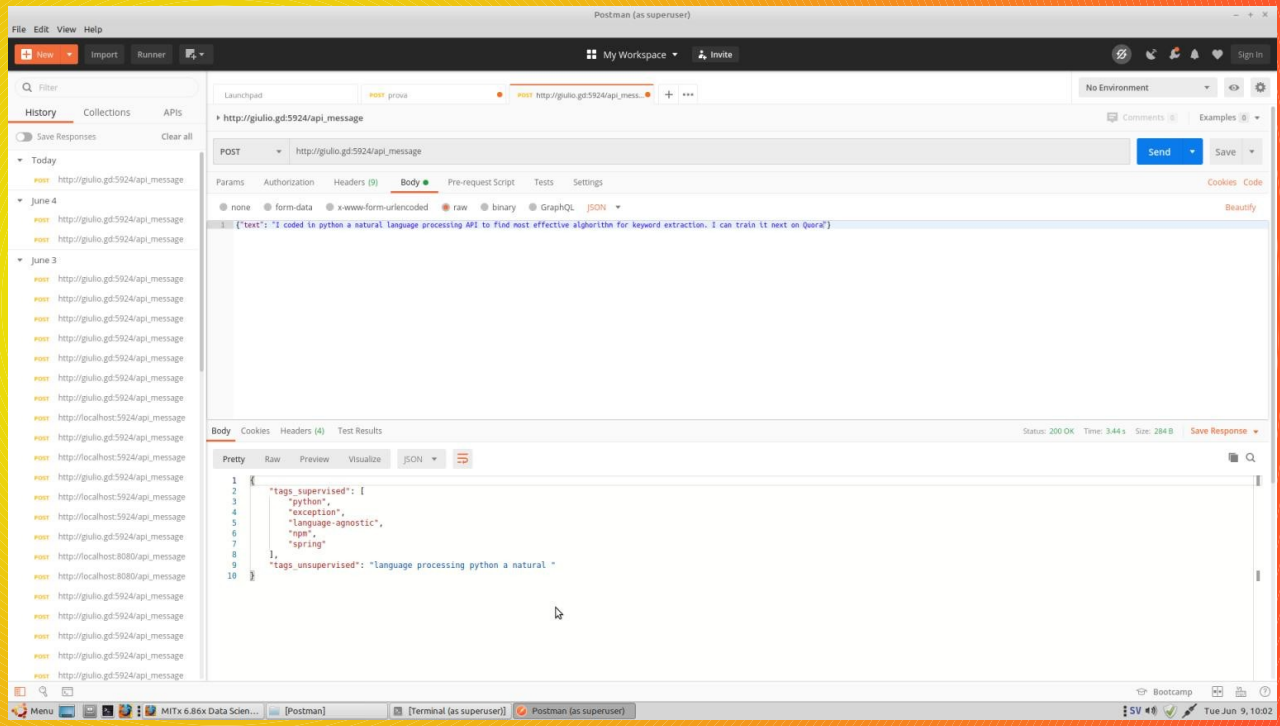
```
y_tfidf_test = tfidfVectorizer.transform(y_test)
y_tfidf_RF = tfidfVectorizer.transform(RF_tags)
y_tfidf_LR = tfidfVectorizer.transform(LR_tags)
print('Dummy score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_dummy.reshape(1,-
1)))
print('RF score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_RF.reshape(1,-1)))
print('LR score:',cosine_similarity(y_tfidf_test.reshape(1,-1), y_tfidf_LR.reshape(1,-1)))
```

```
Dummy score: [[0.05342378]]        YAKE score: [[0.29256526]]
RF score: [[0.61749296]]           LDA score: [[0.18009483]]
LR score: [[0.56390788]]           NMF score: [[0.18110464]]
```

API pour la prédiction des tags

Modèle supervisé : Logistic Regression
un peu moins précis mais beaucoup moins lourd que le Random Forest

Modèle non-supervisé : YAKE



Code source :
https://github.com/opsabarsec/tags_stackoverflow/blob/master/API/app.py

API endpoint :
http://giulio.gd:5924/api_message

Conclusions

- La meilleure performance pour la prédiction des tags était obtenue avec la modélisation Random Forest optimisée
- Une réduction dimensionnelle par ACP de la matrice TF-IDF ne donne pas des bons résultats de modélisation
- Parmi les modèles non-supervises testés le meilleur est celui du paquet « YAKE ! »
- Pour un API endpoint le meilleur compromis entre exactitude et consommation de ressources était obtenu avec la Régression Logistique au lieux des forets aléatoires (Random Forest)