StackOverFlow

Catégoriser automatiquement des questions

Presentation

Sofiane Mouhab
Janvier 2022

Sommaire

1 - Généralités

1.1 - Problematique

Stack Overflow est un site célèbre de questions-réponses liées au développement informatique. Pour poser une question sur ce site, il faut entrer plusieurs tags de manière à retrouver facilement la question par la suite. Pour les utilisateurs expérimentés, cela ne pose pas de problème, mais pour les nouveaux utilisateurs, il serait judicieux de suggérer quelques tags relatifs à la question posée.

Amateur de Stack Overflow, qui vous a souvent sauvé la mise, vous décidez d'aider la communauté en retour. Pour cela, vous développez **un système de suggestion de tag** pour le site. Celui-ci prendra la forme d'un algorithme de machine learning qui assigne automatiquement plusieurs tags pertinents à une question.

1 - Généralités

1.2 - Objectifs

- Mettre en œuvre une approche non supervisée.
- Utiliser une approche supervisée ou non pour extraire des tags à partir des résultats précédents.
- Comparer ses résultats à une approche purement supervisée, après avoir appliqué des méthodes d'extraction de features spécifiques des données textuelles.
- Mettre en place une méthode d'évaluation propre, avec une séparation du jeu de données pour l'évaluation.

1 - Généralités

1.3 - Condition de mise en oeuvre

Pour pouvoir sereinement réaliser ses objectifs, il nous faut donc diverses informations qui pourrait se trouver dans notre base de données.

À nous donc, d'examiner celle-ci, de déterminer à quel point les informations sont viables, ou perfectible.

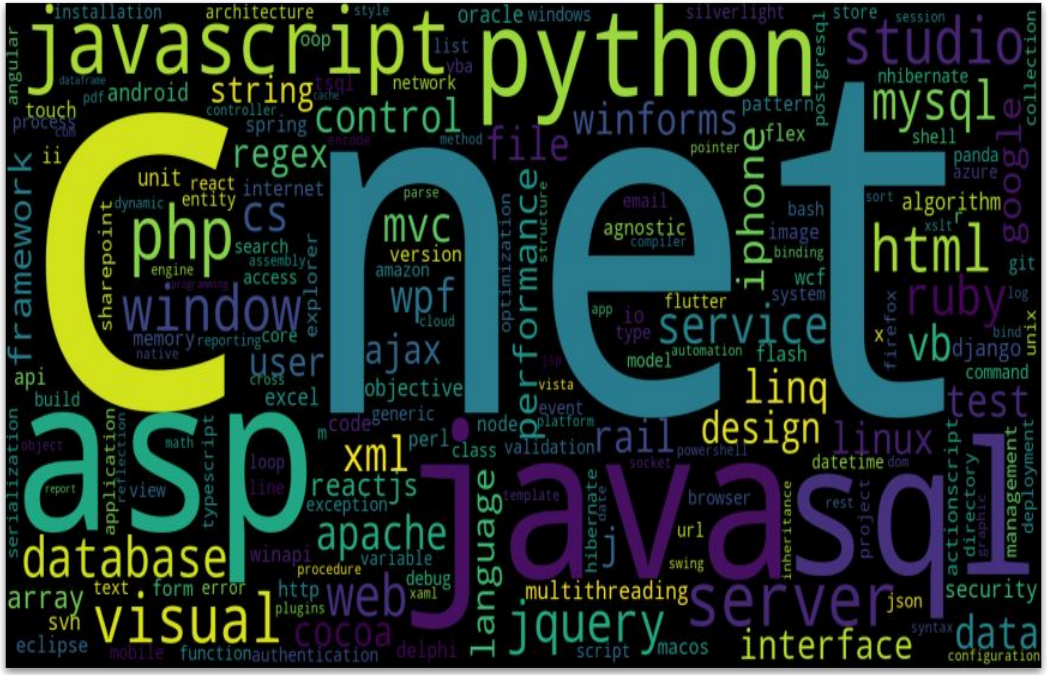
Il y a donc 3 grandes interrogations :

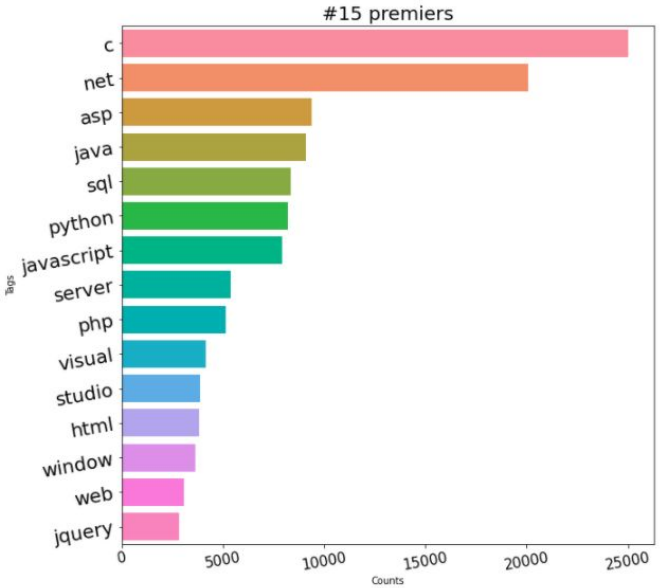
- A-t-on assez de données ?
- Est-il possible d'effectuer des classifications : Non-Superivsée, Semi-SuperVisée, SuperVisée
- Le cas échéant, quel est le meilleur modèle pour répondre à notre problématique ?

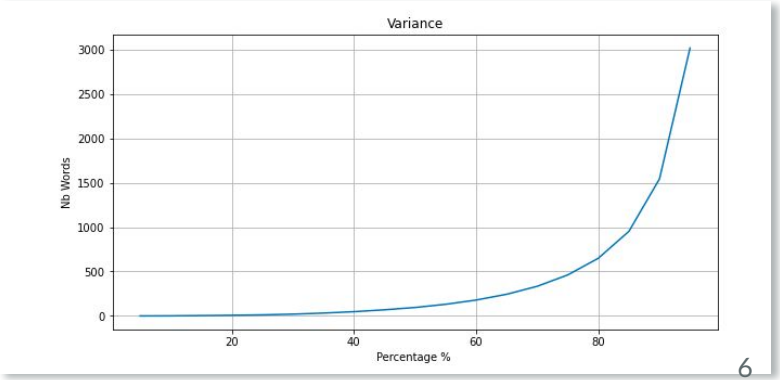
Passons de suite à ce travail, en commençant par rapidement prendre connaissance des données en présence...

2 - Les données

2.1 - Analyses des tags 1/2

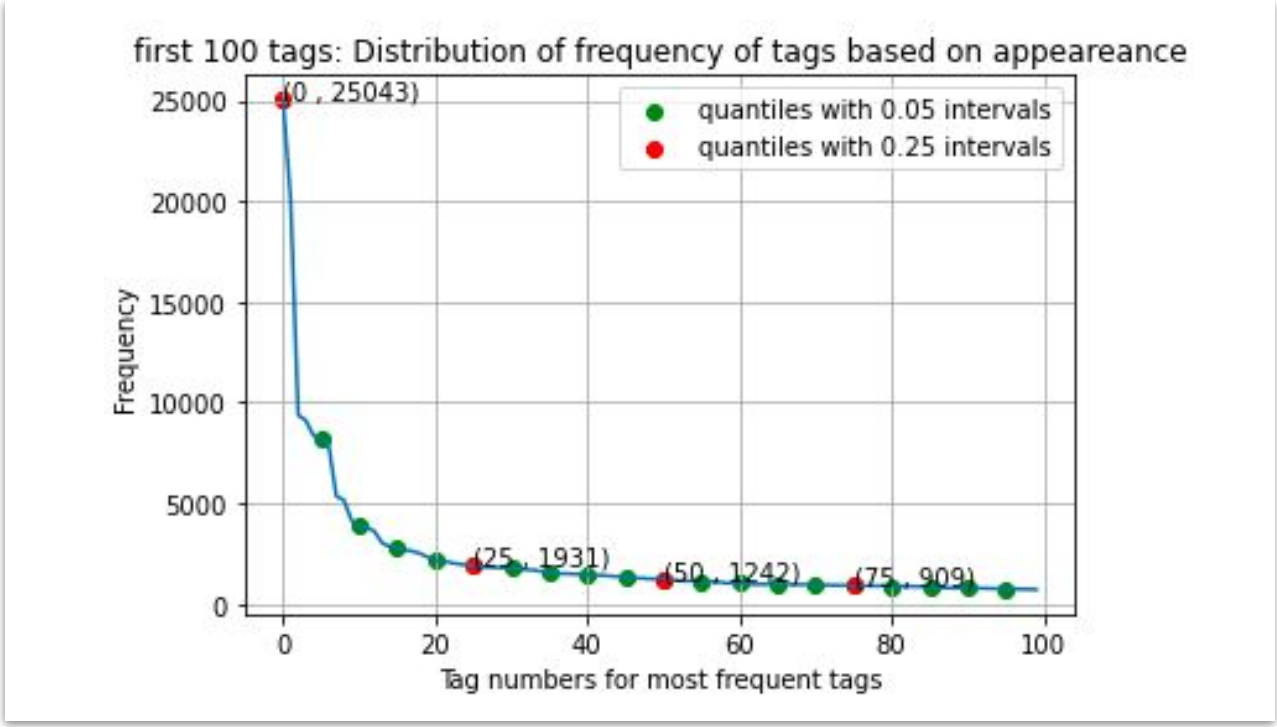


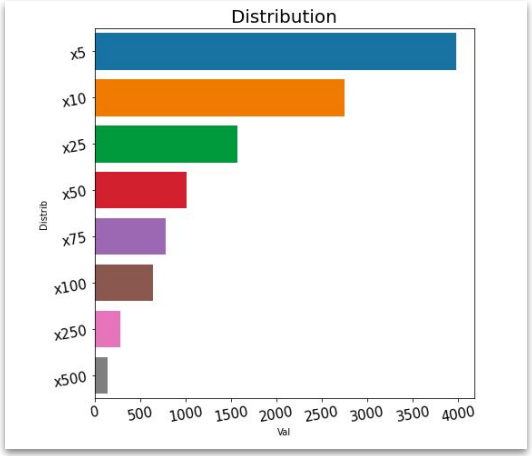


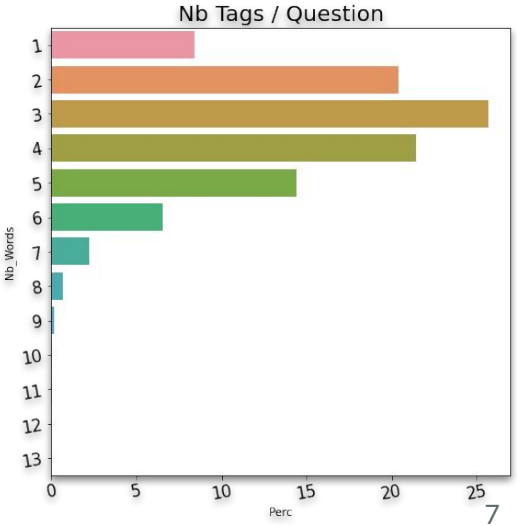


2 - Les données

2.1 - Analyses des tags

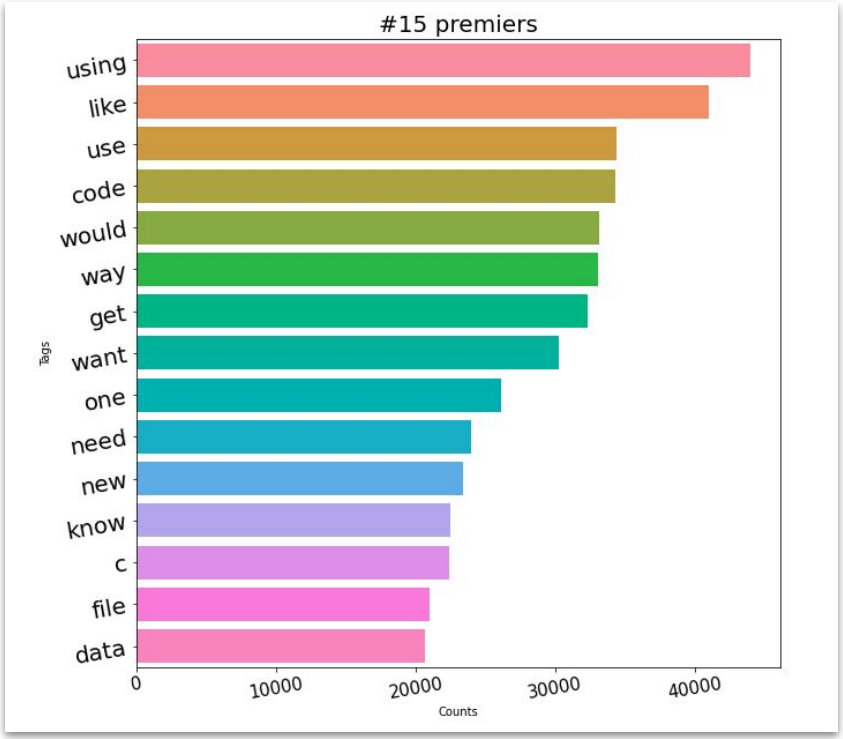


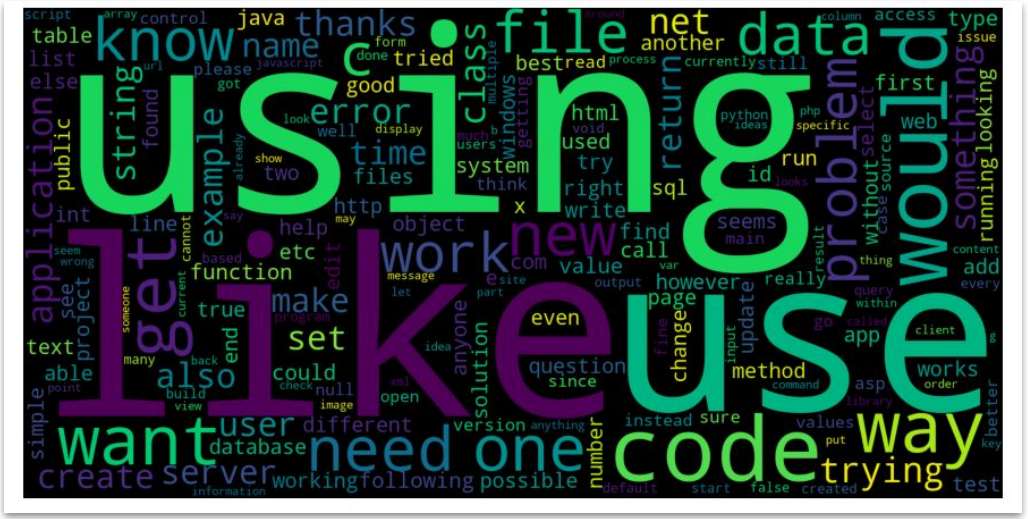




2 - Les données

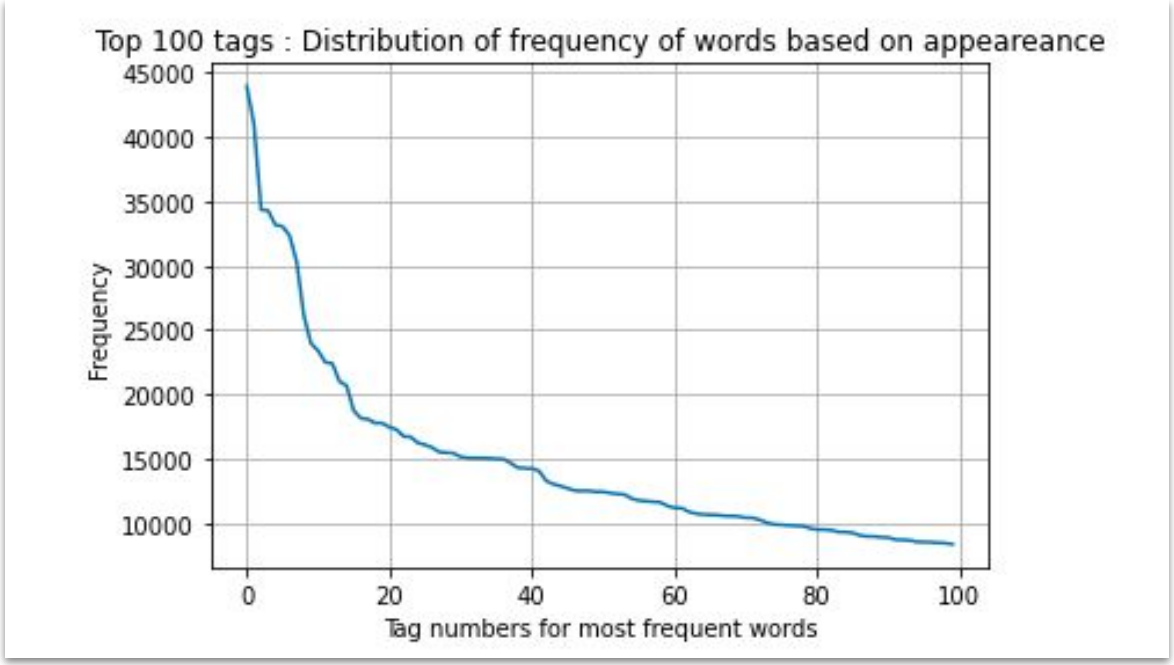
2.2 - Analyses des titres et questions

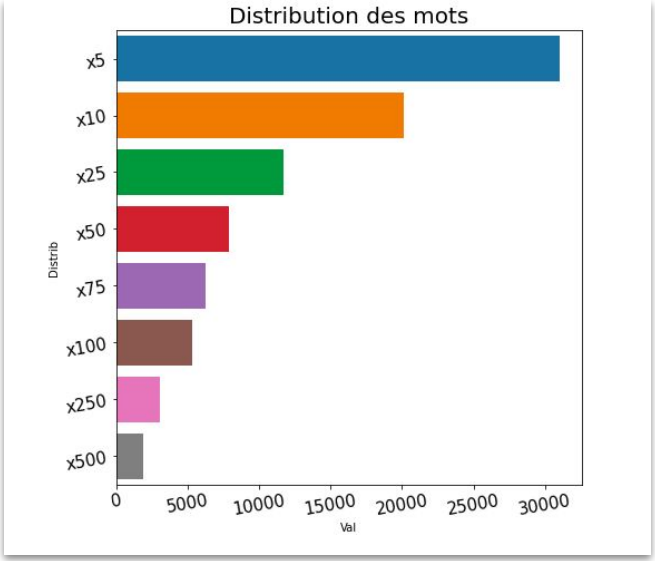


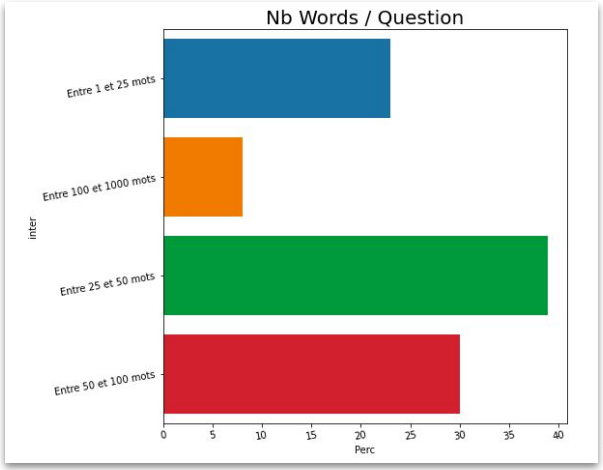


2 - Les données

2.2 - Analyses des titres et questions

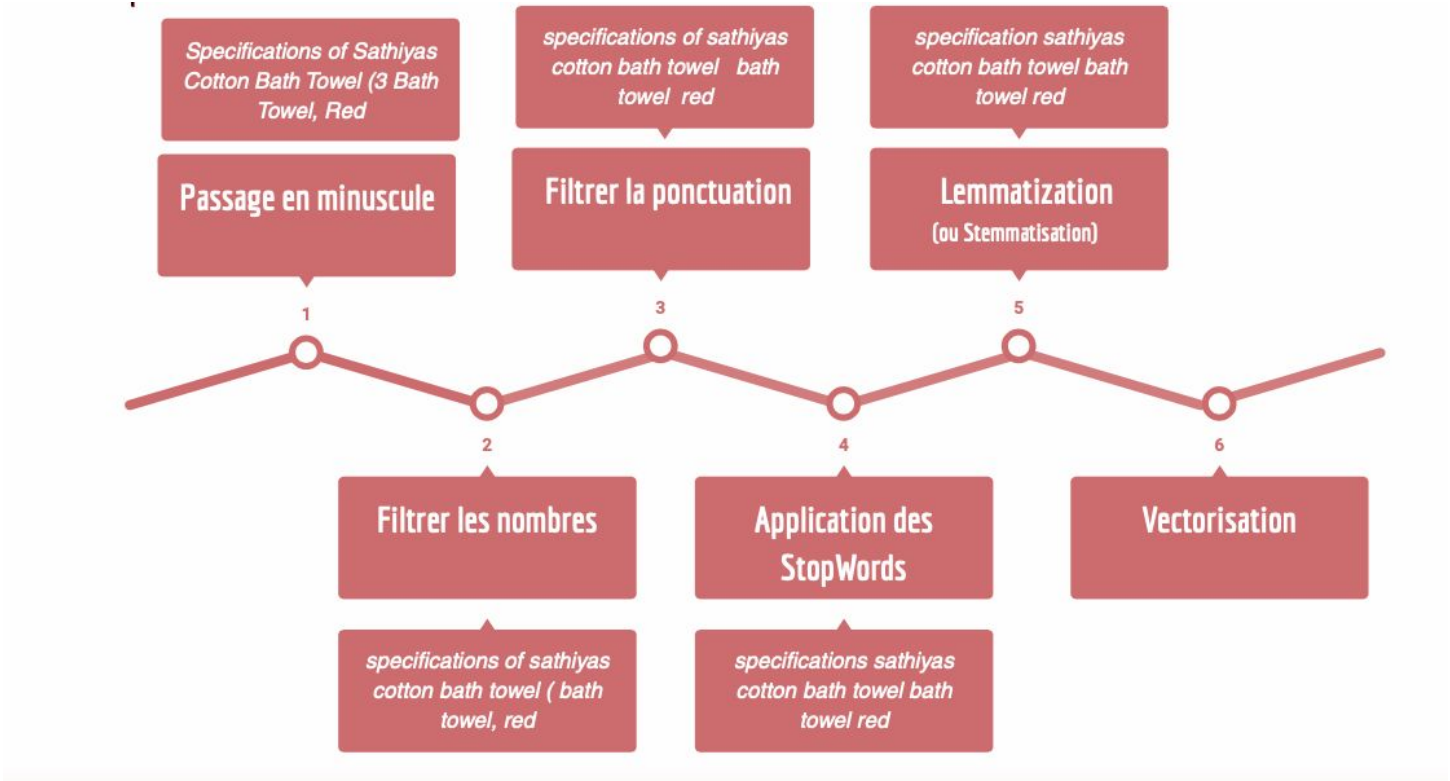






2 - Les données

2.3 - Prétraitement

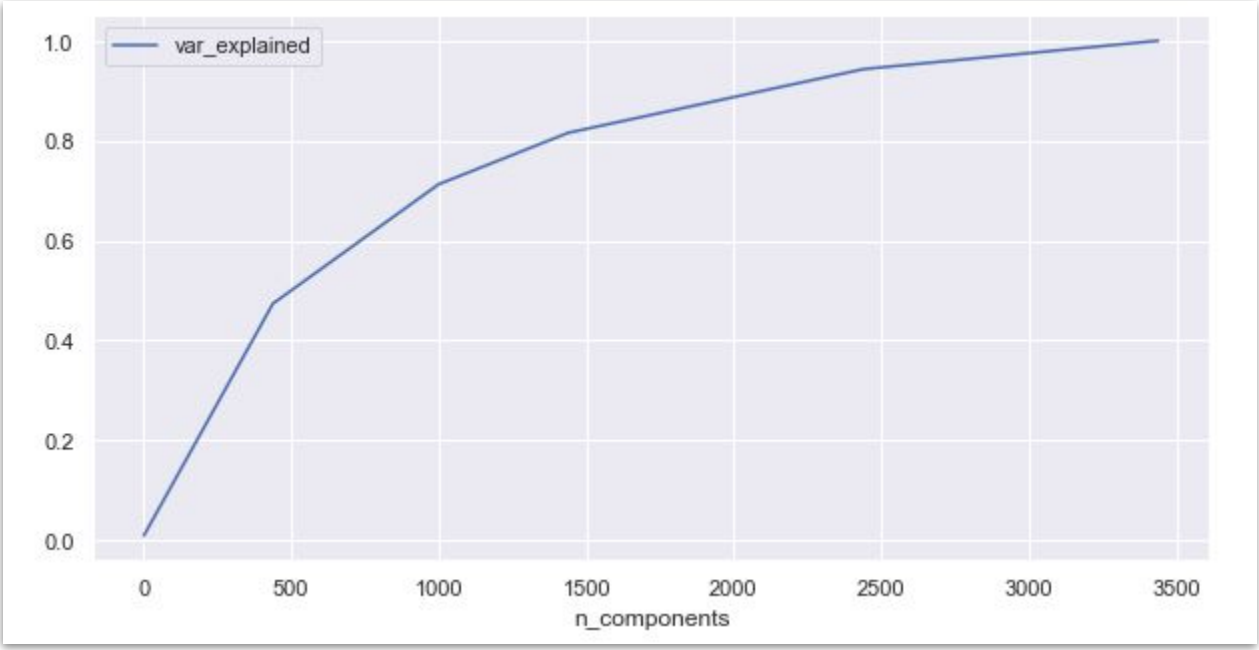


2 - Les données

2.4 - Reduction de dimension

```
n_components :  1000
Total variance explained: 0.71
```

```
X before TruncatedSVD (32455, 3439)
X after TruncatedSVD (32455, 1000)
```



A ce stade, on constate que 1000 variables sur 3439 expliquent 70% des données. On applique donc ce seuil afin de réduire notre dataset, et ainsi faciliter le traitement des données

3 - Approche non supervisée

3.1 - Acteurs

3.1.1 - Métriques

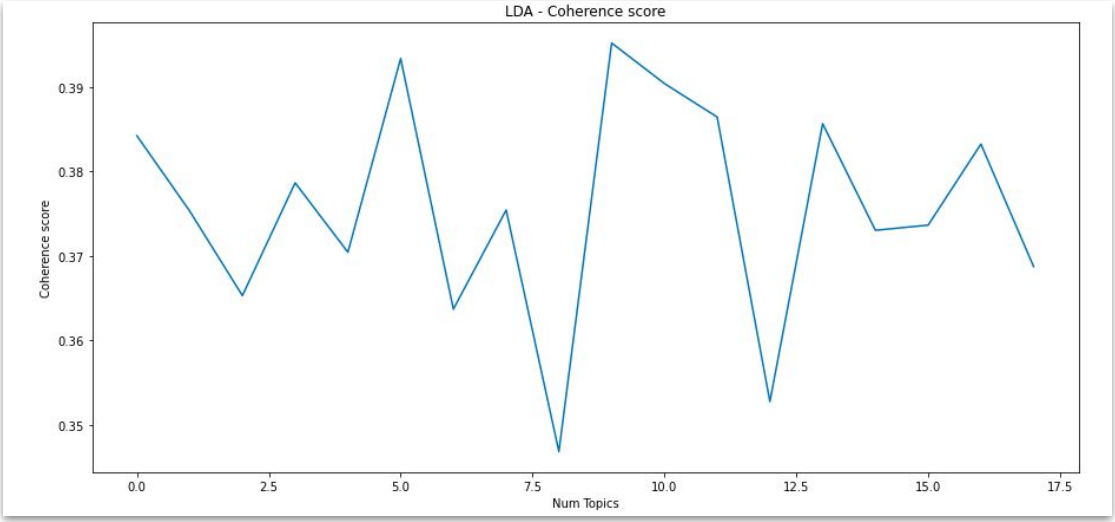
- Coherence Score

3.1.2 - Modeles

- LDA : Latent Dirichlet Allocation
- NMF : Non-Negative Matrix factorization

3 - Approche non supervisée

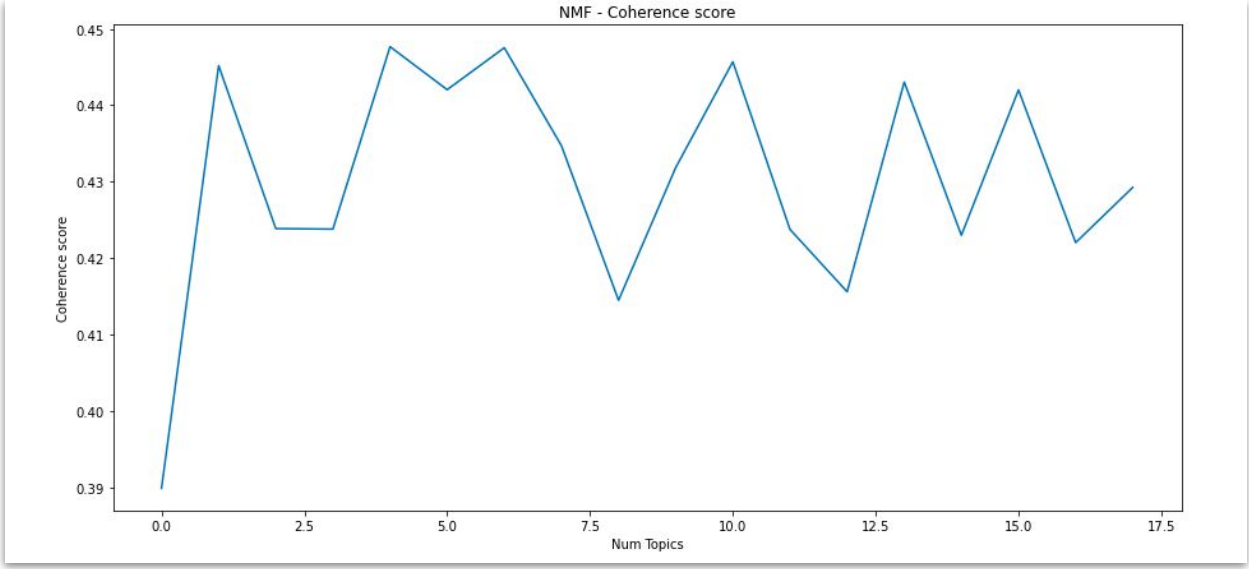
3.2 - LDA



Topics	Word_1	Word_2	Word_3	Word_4	Word_5	Word_6	Word_7	Word_8	Word_9	Word_10
Topic_1	would	good	look	control	question	service	think	library	request	different
Topic_2	user	server	web	page	error	message	system	access	content	site
Topic_3	application	run	project	net	build	version	machine	feature	thread	development
Topic_4	table	column	row	performance	setting	css	root	duplicate	windows_form	focus
Topic_5	class	object	call	method	function	result	query	log	contain	implement
Topic_6	window	interface	print	click	task	parameter	path	button	child	security
Topic_7	test	process	take	tool	order	tag	record	software	insert	comment
Topic_8	value	type	string	return	name	public	d	key	property	select
Topic_9	use	file	way	work	want	code	need	set	get	create

3 - Approche non supervisée

3.3 - NMF



Topics	Word_1	Word_2	Word_3	Word_4	Word_5	Word_6	Word_7	Word_8	Word_9	Word_10
Topic_1	class	object	public	return	method	code	private	function	string	call
Topic_2	use	run	work	code	server	test	try	project	application	get
Topic_3	user	would	way	want	good	page	control	application	need	list
Topic_4	table	database	row	d	column	datum	record	select	query	value
Topic_5	file	project	use	open	want	directory	way	read	line	include
Topic_6	name	value	false	type	error	system	web	property	new	right

4 - Approche supervisée et semi-supervisée

4.1 - Acteurs

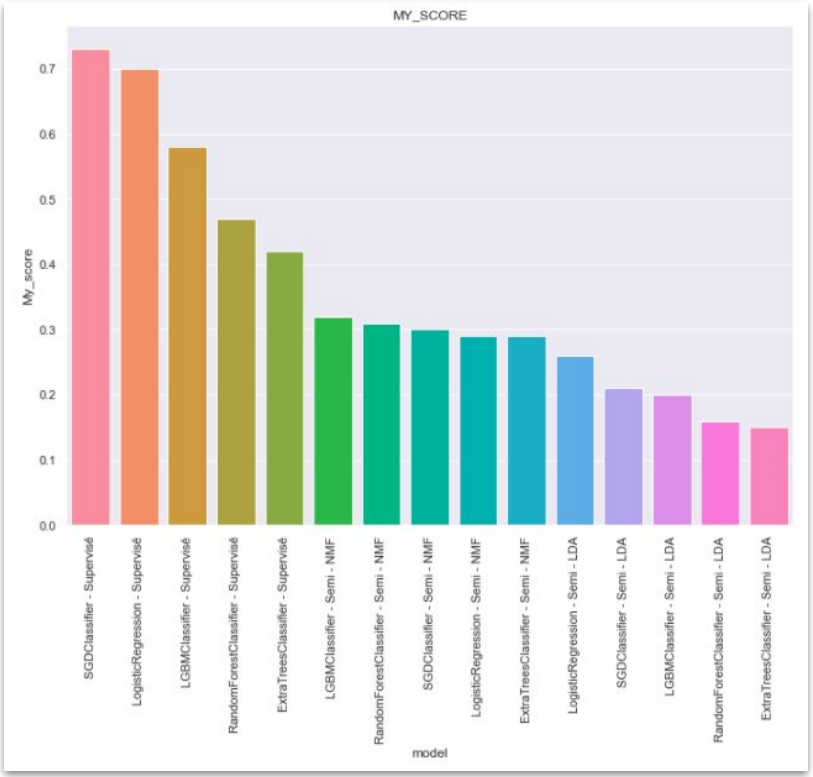
4.1.1 - Métriques

- Accuracy
- Timing
- My Score
- Jaccard Score

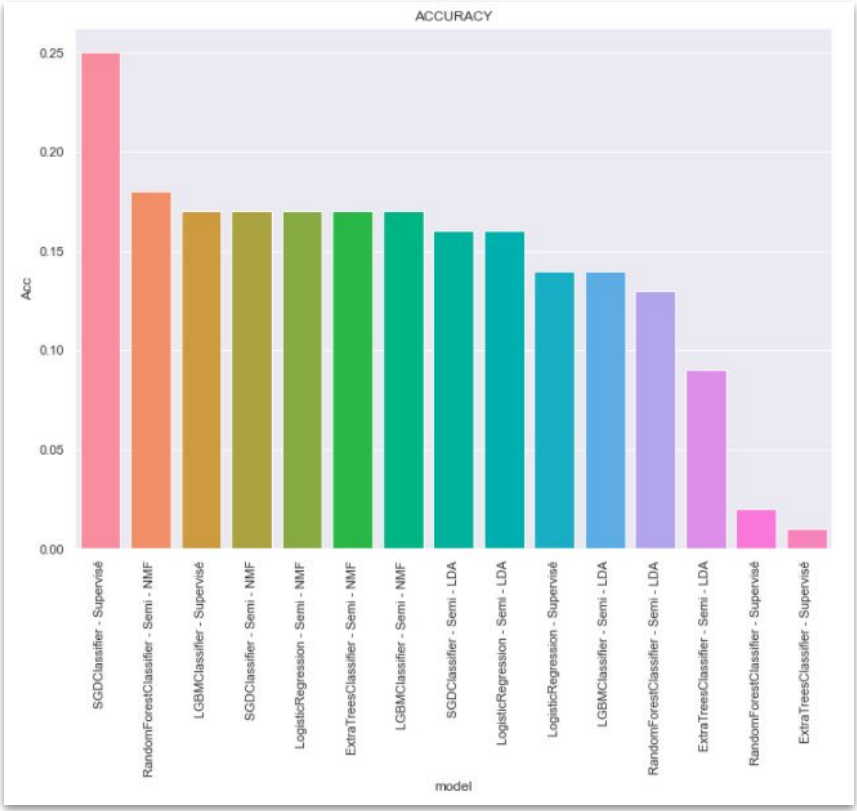
4.1.2 - Modeles

- SGDClassifier
- LogisticRegression
- LinearSVC
- RandomForestClassifier
- ExtraTreesClassifier
- LGBMClassifier

4.2 - Comparaison Accuracy

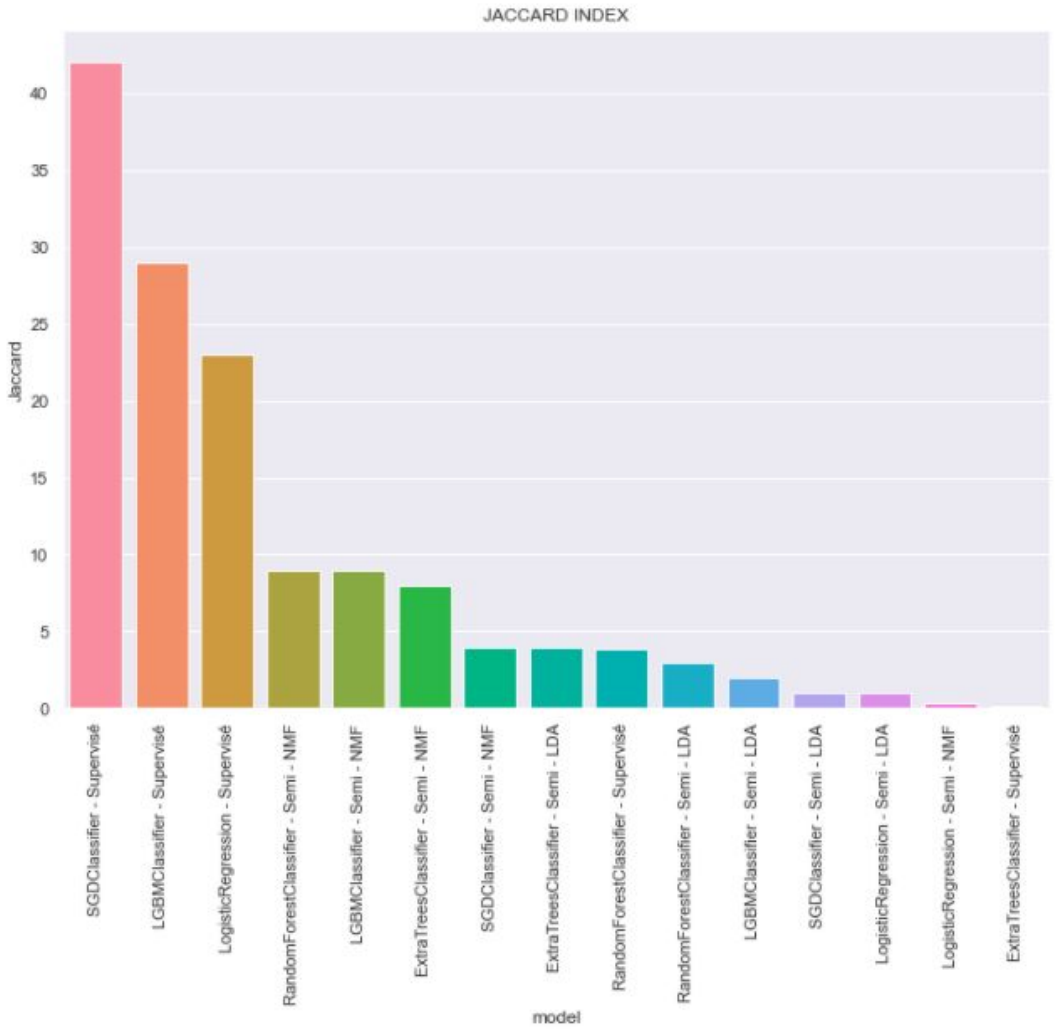


4.2 - Comparaison MyScore



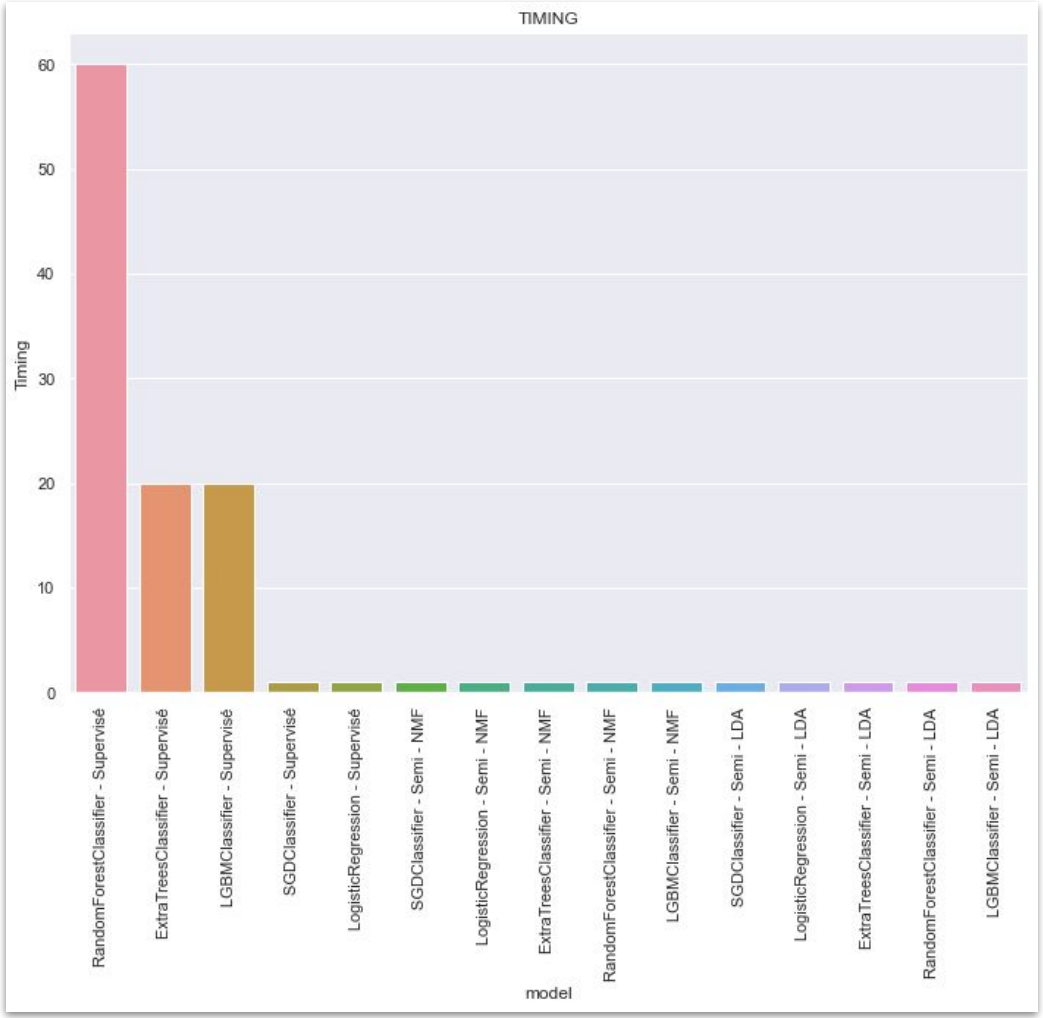
4.2 - Comparaison Jaccard index

On comparer nos differents modeles (semi-supervisée et supervisée) avec pour référence le Jaccard index.



4.4 - Comparaison Timing

Au dela du Jaccard index, il est important de prendre en compte le temps de traitement des différents modèles



4.3 - Meilleur modèle avec GridSearchCV

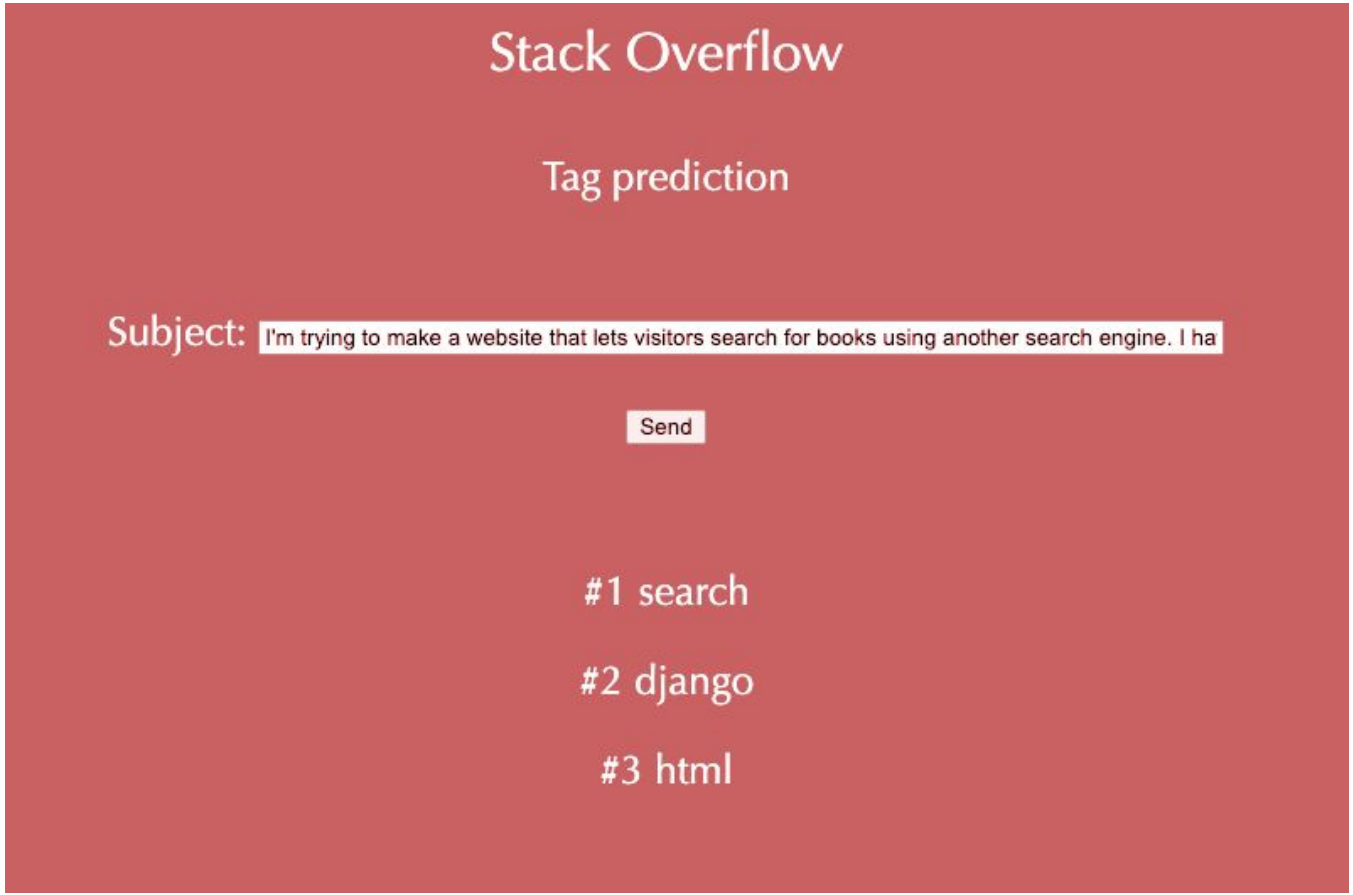
```
SGDClassifier(class_weight={0: 0.3, 1: 0.7}, eta0=10)
Accuracy : 0.3629
Macro f1 score : 0.31057442889545 94
Micro f1 score : 0.5586267325155329
Jacard score: 41.31890072176892
Hamming loss: 1.1082
---
time: 17.6 s (started: 2021-12-31 14:42:35 +01:00)
```

5 - API

GitHub

django

HEROKU



6 - Conclusion

Après avoir tenté différentes approches, il nous semble que la classification supervisée apportent les meilleurs résultats.

Le choix du SDG Classifier est un compromis entre les différents score mais aussi le temps de calcul relativement rapide.

Les tags obtenus semblent cohérent.

La création de l'Api et son déploiement répondent au cahier des charges

7 - Perspective

Axes d'amélioration:

- Gestion de la corrélation entre tags
- Explorer d'autres méthodes (Bow, Pos, word embedding...)
- Améliorer nos filtres (nombres et ponctuation en particulier)
- Approfondir le Deep Learning