

Projet 5 : Catégorisation automatique de questions

Rapport



Auteur : Antoine Chesnais

Date : 06/12/2019

Objectif du projet :

Issu de la page du projet :

« 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. »

Le but du projet est donc d'établir un système de suggestion de tags qui proposera des tags en fonction du contenu de la question de l'utilisateur. Pour cela on testera deux approches : une approche purement supervisée et une approche hybride. Dans la première on utilisera les tags déjà associés à des questions comme cible. Dans la seconde on utilisera une approche non supervisée pour associer des tags aux questions et l'on créera ensuite un modèle supervisé à partir de cette nouvelle cible pour pouvoir ensuite prédire les tags sur de nouvelles questions.

Table des matières

I.Introduction :	3
A) Les questions StackOverflow.....	3
B) Collecte des données.....	3
C) Le corps d'une question.....	4
II.Nettoyage des données :	5
A) Processus de nettoyage.....	5
B) Récupération du texte de la question.....	6
C) Retrait des stopwords.....	6
D) Lemmatisation et retrait POS (Part Of Speech).....	7
E) Retrait du vocabulaire partagé.....	7
III.Modélisation purement supervisée :	8
A) Stratégie de modélisation.....	8
B) Réduction du nombre de tags cibles.....	9
C) Choix de la représentation des données.....	9
D) Modèles testés	10
E) Performances sur le test set.....	10
IV.Modélisation hybride.....	11
A) Processus de modélisation.....	11
B) Sortie des algorithmes non supervisés.....	12
C) Association de tags à chaque sujet	12
D) Création d'une matrice questions / tags :.....	13
E) Transformation en un problème de classification.....	13
F) Visualisation des nouvelles cibles.....	14
G) Résultats de la modélisation supervisée	14
V.Conclusion.....	15
A) Exemples d'utilisation.....	15
B) Conclusion et perspectives.....	15

I. Introduction :

Comme mentionné sur la page de garde, l'objectif du projet est la création d'un système de suggestion de tags à partir des questions posées sur stackoverflow. Dans cette partie d'introduction nous allons voir comment se présentent les questions sur stackoverflow, comment les collecter et quelles sont les problématiques à relever afin de pouvoir les exploiter.

A) Les questions StackOverflow

Les questions sur StackOverflow se présentent de la manière suivante (voir figure 1) :

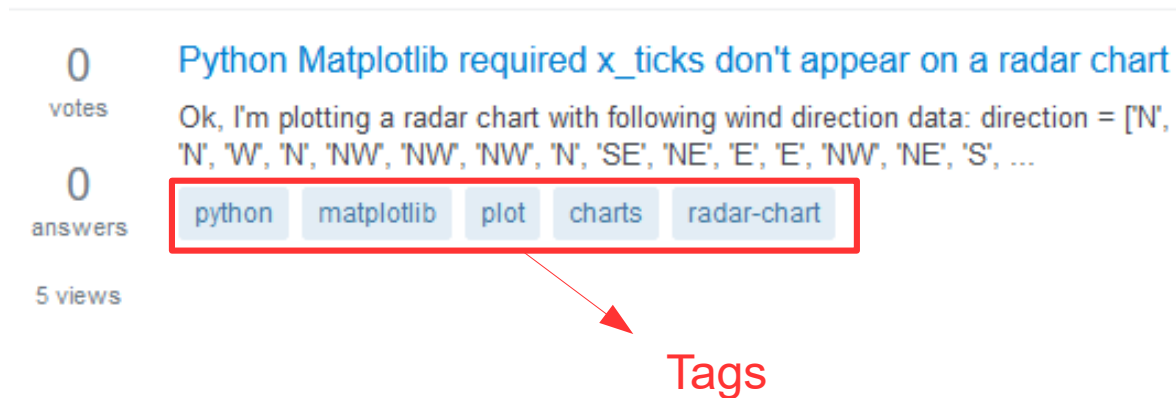


Figure 1: Exemple de question sur le site StackOverflow

On distingue en particulier un titre de question (en bleu), le corps de la question (en noir) et les tags associés (encadré rouge). Le titre est généralement une description succincte des éléments principaux du problème rencontré. Le corps du texte lui donne plus de détails, on peut notamment lui associer du code pour illustrer le problème rencontré. Les tags eux permettent d'identifier facilement le sujet de la question et permet de mettre en avant celle-ci auprès d'utilisateurs ayant déjà répondu à des problématiques similaires. Les tags sont souvent des termes techniques comme le langage utilisé, le framework, les librairies ...

B) Collecte des données

Les données peuvent être collectées depuis l'API StackExchange.

Pour la récupération, on précise que l'on souhaite que les posts soient récupérés sur StackOverflow uniquement, sur la période de début Juillet à fin Octobre 2019 (soit environ 176 000 questions).

Les questions collectées ont également été restreintes à celles ayant une réponse acceptée, en considérant que si une question a obtenu une réponse satisfaisante, c'est qu'elle a été repérée par des utilisateurs compétents et que les tags sont donc appropriés.

De nombreuses données sont collectées via l'API, et seuls le titre de la question, son corps et les tags qui lui sont associés ont été récupérés. Ci-dessous en figure 2 un aperçu du dataset de travail :

title	body	tags
"Column ambiguous defined" error in ...	<p>I am working on a homework problem. I am su...	[oracle]
How to get the first day of the next month in ...	<p>How can I get the first date of the next mo...	[python, 'python-datetime']
Visual Studio 2019 Change code template for ne...	<p>I have done this in the past but cannot see...	[visual-studio-2019, 'code-editor', 'code-te...
Webapi inherited controllers are ignoring Rout...	<p>I want to have a Controller Hierarchy assem...	[c#, 'asp.net', 'asp.net-web-api']
How to install Font Awesome in ASP.NET Core 2....	<p>I am struggling to find any up to date inst...	[asp.net-core]

Figure 2: Aperçu des données collectées

C) Le corps d'une question

Il n'est pas possible d'exploiter directement les questions récupérées. En effet, en même temps que le texte de la question sont récupérés entre autres sa mise en page et le code inséré. Ci dessous en figure 3 un exemple :

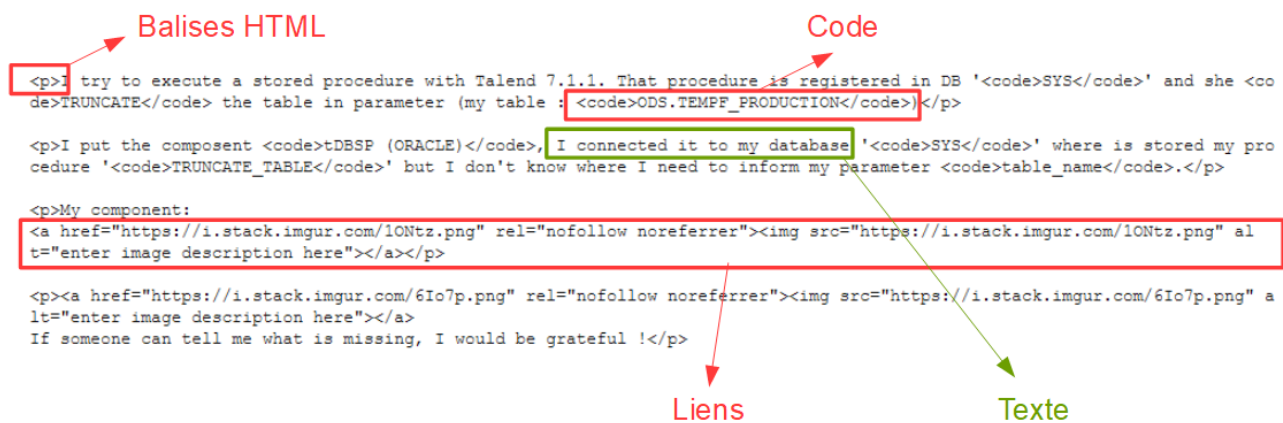


Figure 3: Exemple de corps de question collectée depuis l'API

De nombreux éléments polluent le corps de la question, dont il faudra se débarrasser, ce que nous verrons dans la partie suivante intitulée 'Nettoyage des données'.

II. Nettoyage des données :

Dans cette partie nous aborderons toutes les étapes qui ont transformé les questions afin de les rendre exploitables par la suite par des algorithmes de machine learning.

A) Processus de nettoyage

Ci dessous (figure 4) un diagramme résumant le processus de nettoyage utilisé :

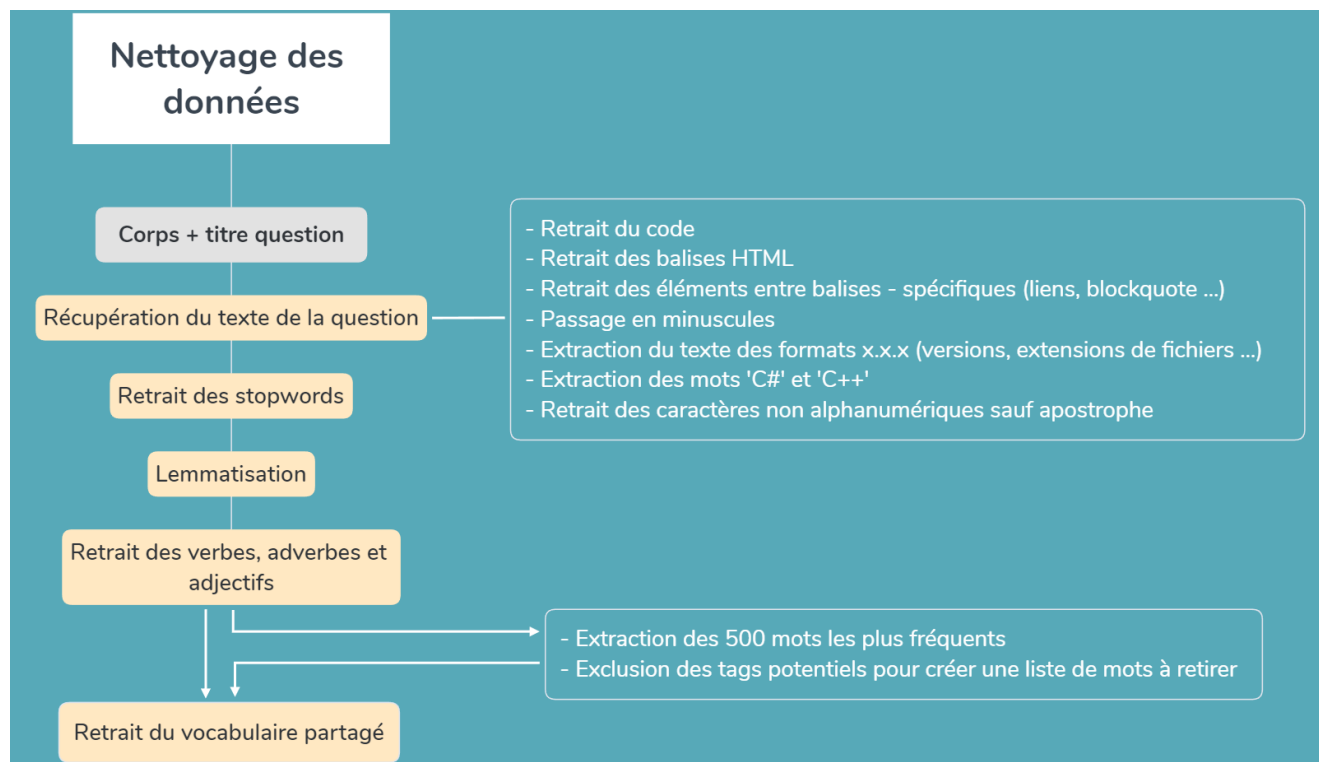


Figure 4: Processus de nettoyage des données

Pour la suite, chaque partie correspondra à une étape du nettoyage, avec la description des actions menées et une visualisation des mots les plus fréquents à la fin de celle ci.

III. Modélisation purement supervisée :

Dans cette partie on testera une approche purement supervisée pour la prédiction de tags. C'est à dire transformer les questions en features que l'on pourra fournir à un algorithme de classification qui aura pour cible les tags associés. Il s'agit là d'un problème de classification multi-label, car plusieurs tags peuvent être associés à une même question. A noter que pour des questions de temps de calcul le dataset pour la modélisation a été limité à un quart, soit environ 44 000 questions.

A) Stratégie de modélisation

Ci dessous en figure 9 un résumé des différentes étapes de modélisation :

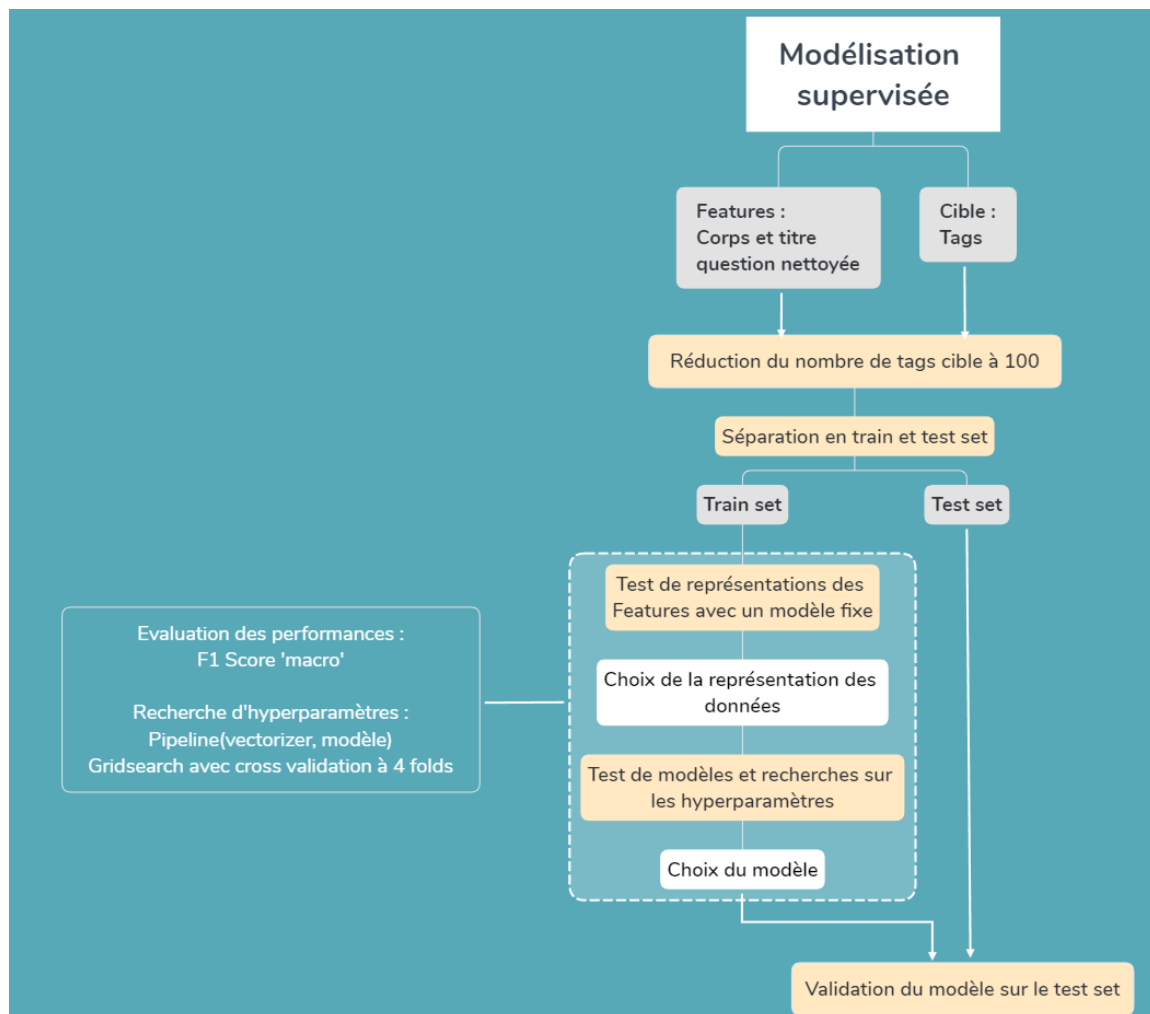


Figure 9: Processus de modélisation supervisée

Les données utilisées comme features sont sous la forme d'une chaîne de caractères composée du titre de la question et de son corps post nettoyage. La cible est l'ensemble des tags, sous la forme d'une matrice creuse dans laquelle chaque tag est une variable et cette variable prend la valeur 1 ou 0 pour une question si respectivement le tag lui est associé ou non.

Les modèles sont construits en utilisant l'outil Pipeline et évalués par Cross validation. Cela permet respectivement d'éviter le 'dataleakage' entre les sets d'entraînement et de test et de limiter le sur-apprentissage.

Les modèles sont évalués via le F1 score, qui prend à la fois en compte la précision et le rappel. Il est moyenné sur tous les Tags de manière 'Macro', c'est à dire que le score obtenu pour chaque tag compte avec le même poids. Ce choix a été fait malgré le déséquilibre dans la fréquence d'apparition des tags car une sélection préliminaire a déjà été faite, chaque tag est donc important.

B) Réduction du nombre de tags cibles

Dans notre dataset réduit il existe environ 11 650 tags uniques. Pour pouvoir prédire ce nombre très important de tags, il serait nécessaire d'employer des outils de calcul puissants que nous n'avons pas à disposition. La cible a donc été réduite en s'assurant que la grande majorité des questions conservaient toujours au moins un tag. A noter également que la distribution des tags est très déséquilibrée, puisque si l'on regarde le nombre d'occurrences de chaque tag parmi les 1000 les plus fréquents, on passe d'un compte de presque 6000 à moins de 100 en passant du premier au 200ème tag.

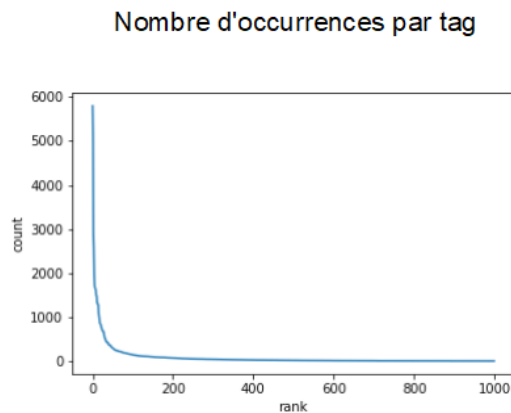


Figure 11: Nombre d'occurrences par tag

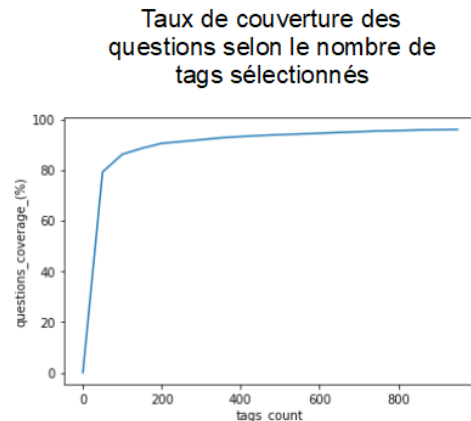


Figure 10: Taux de couverture

Au final on retient pour cible les 100 tags les plus représentés, ce qui permet de conserver environ 86% des questions, soit 38 000 échantillons.

C) Choix de la représentation des données

Afin de pouvoir utiliser la question et son titre en tant que features d'un algorithme de machine learning il est nécessaire de construire un 'sac de mots' (Bag Of Words). Avec cette représentation, chaque mot devient une feature. Il existe plusieurs variantes de cette représentation, lesquelles associent des valeurs différentes aux variables pour une question :

- **Term Frequency** : La valeur correspond au nombre d'apparition d'un mot dans la question
- **Binary** : La valeur prise est 1 ou 0, selon que le mot soit présent ou non dans la question
- **Term Frequency – Inverse Document Frequency** : La valeur prise correspond au nombre d'apparition d'un mot au sein de la question, pondéré par son apparition dans d'autres questions. Ainsi si le terme est fréquent dans la questions sa valeur augmente, mais s'il est également présent dans beaucoup d'autres questions sa valeur va fortement décroître.

Ces différentes représentations on été testées avec un classifieur Ridge. Ci dessous en figure 12 les résultats :

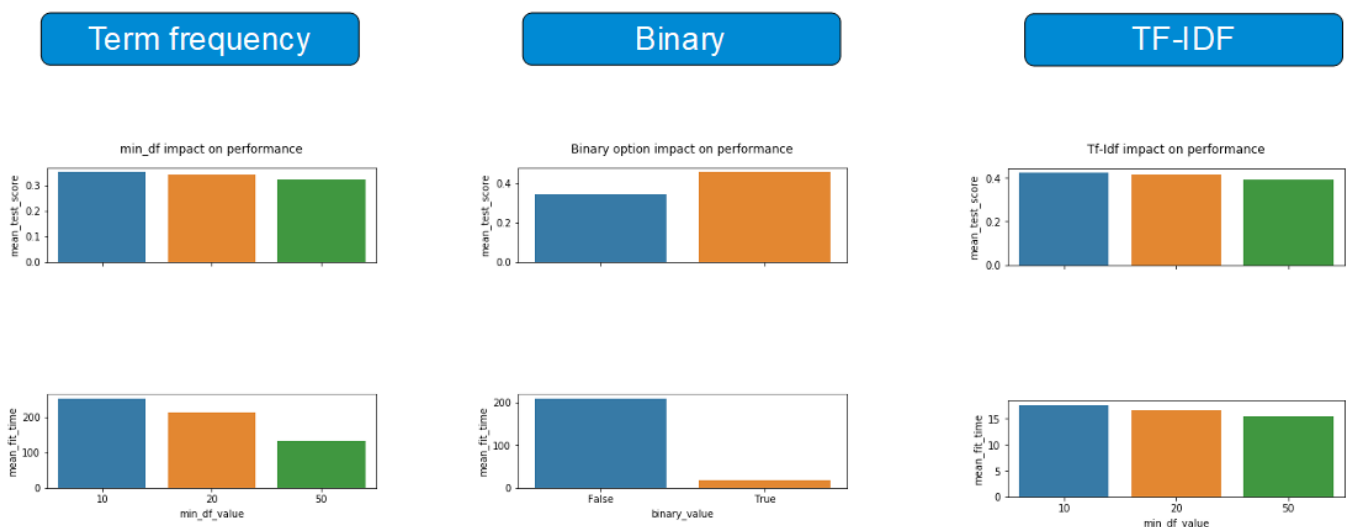


Figure 12: Impact de la vectorisation sur les performances

La représentation 'Binary' donne les meilleurs résultats et permet des calculs beaucoup plus rapide, c'est donc celle ci qui a été retenue.

D) Modèles testés

Ci dessous (figure 13) les modèles testés, ainsi que les hyperparamètres qui ont été ajustés :

- **Classifieur Ridge** : paramètres de régularisation
- **Régression Logistique** : paramètres de régularisation
- **Classifieur linéaire SGDC** : fonction de perte
- **Classifieur bayésien naïf** : paramètre de lissage
- **Random Forest** : nombre d'estimateurs et profondeur de l'arbre

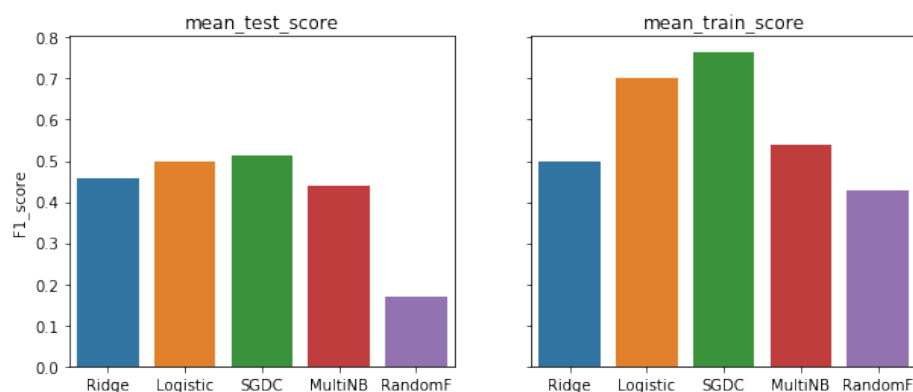


Figure 13: F Score des différents modèles supervisés

Le classifieur SGDC avec perte 'Hinge' donne les meilleurs résultats, avec un F1 score de 0.51.

E) Performances sur le test set

Le F1 score sur le test set est de 0.51. Ci dessous (figure 14) le détails des 20 tags les mieux et les moins bien classés :

TOP 20						BOTTOM 20					
rank	tag	F1_score	precision	recall	tag_rank	rank	tag	F1_score	precision	recall	tag_rank
98	ggplot2	0.89	0.85	0.94	98	93	visual-studio	0.24	0.19	0.36	93
24	django	0.87	0.84	0.91	24	53	linux	0.22	0.15	0.40	53
54	git	0.87	0.86	0.89	54	81	algorithm	0.20	0.14	0.42	81
86	swiftui	0.87	0.76	1.00	86	70	android-studio	0.20	0.16	0.28	70
85	elasticsearch	0.86	0.78	0.96	85	69	.net-core	0.18	0.12	0.38	69
60	wordpress	0.83	0.80	0.86	60	16	arrays	0.18	0.14	0.25	16
66	unity3d	0.83	0.80	0.86	66	30	dataframe	0.17	0.10	0.39	30
75	keras	0.82	0.77	0.88	75	76	windows	0.16	0.12	0.26	76
34	docker	0.82	0.84	0.81	34	59	loops	0.08	0.04	0.50	59
48	mongodb	0.82	0.76	0.89	48	95	date	0.05	0.03	0.25	95
99	flask	0.82	0.84	0.79	99	77	for-loop	0.05	0.02	0.50	77
22	laravel	0.82	0.73	0.93	22	62	asp.net	0.04	0.02	0.17	62
64	kubernetes	0.81	0.76	0.86	64	42	string	0.02	0.01	0.11	42
32	vue.js	0.81	0.69	0.97	32	7	python-3.x	0.01	0.01	0.14	7
50	powershell	0.80	0.78	0.81	50	91	tsql	0.00	0.00	0.00	91
84	azure-devops	0.77	0.73	0.82	84	61	database	0.00	0.00	0.00	61
10	r	0.77	0.69	0.87	10	63	function	0.00	0.00	0.00	63
14	pandas	0.76	0.69	0.86	14	78	api	0.00	0.00	0.00	78
31	vba	0.76	0.65	0.92	31	51	.net	0.00	0.00	0.00	51
94	google-cloud-firestore	0.75	0.70	0.82	94	43	list	0.00	0.00	0.00	43

Figure 14: Précision, rappel et F1 score pour différents tags

Au final il est intéressant de noter les points suivants :

- Les tags plus précis, comme les noms de logiciel et frame work particuliers ont souvent un bon score
- Les tags plus 'généraux' comme des termes techniques partagés par plusieurs technologies / langages (function, dataframe, windows, api, database ...) eux ont des scores moins bons.
- Les tags initialement les plus représentés ne sont pas forcément les mieux prédits comme on peut le voir avec la colonne 'rank'.

IV. Modélisation hybride

Dans cette partie on utilisera dans un premier temps une approche non supervisée sur les questions afin de découvrir des sujets potentiels et les associer à chaque question. On associera ensuite des tags à chaque sujet découvert et ainsi au passage aux questions également. Cela donnera une nouvelle cible découverte de manière non supervisée à utiliser ensuite dans un algorithme supervisé pour la suggestion de tags.

A) Processus de modélisation

Le processus de modélisation utilisé est le suivant (figure 15) :

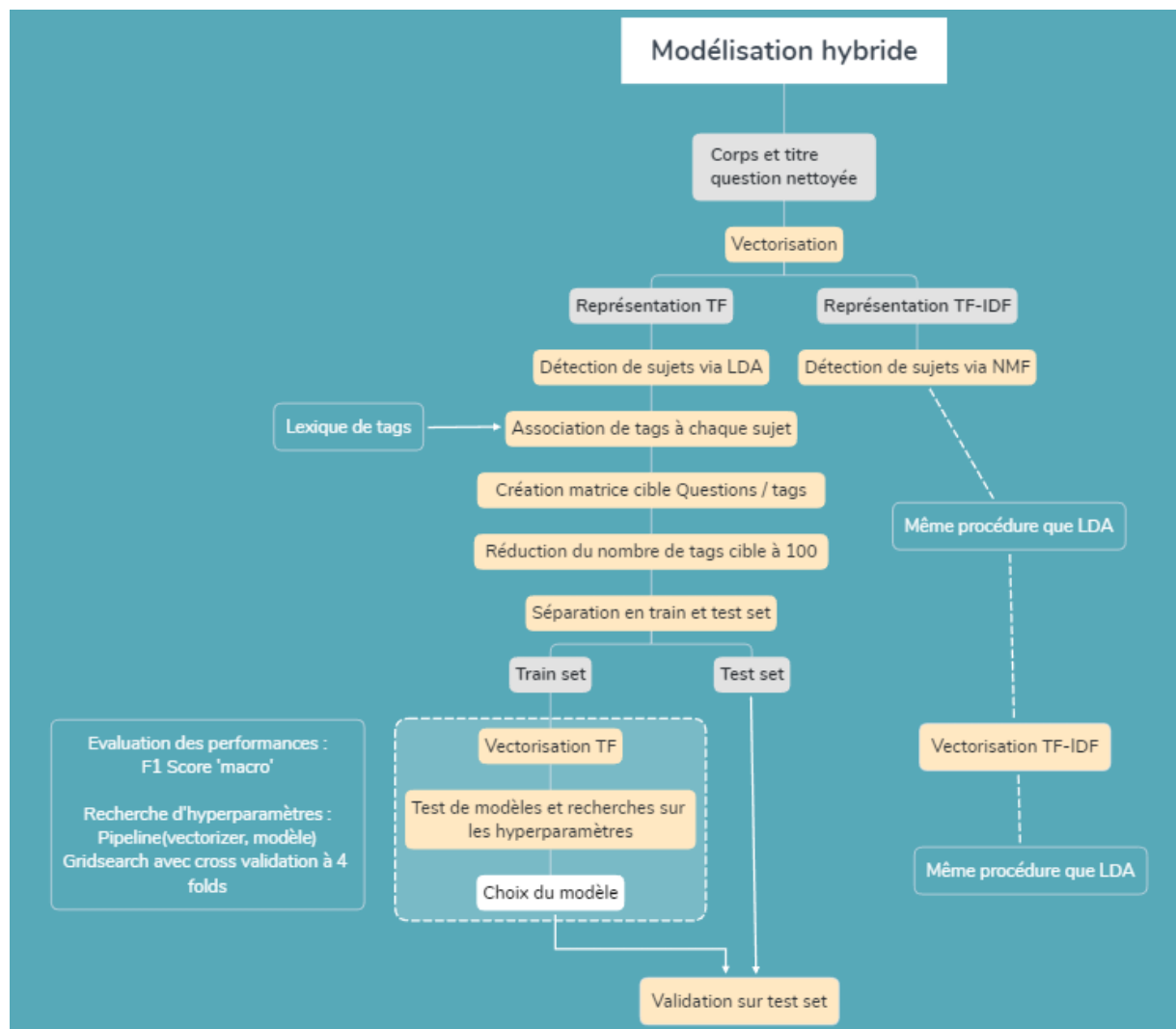


Figure 15: Processus de modélisation hybride

Pour la modélisation non supervisée des sujets deux méthodes sont testées : la Factorisation en Matrices Non Négatives (NMF) et Latent Dirichlet Allocation. Ces deux méthodes donneront une liste de sujets latents, définis chacun par une liste de mots ayant un poids plus ou moins important. Elles donnent également pour chaque question les proportions associées à chaque sujet. Avec ces données de sorties de l'algorithme non supervisée est construite une cible de Tags associés à chaque question. Les étapes de cette construction sont détaillées dans les parties suivantes. Une fois la cible obtenue, la modélisation supervisée qui s'en suit est très similaire à ce qui a pu être fait dans la partie III.

B) Sortie des algorithmes non supervisés

Ci dessous un exemple de ce que l'on peut obtenir en sortie de LDA, avec les sujets et les mots les plus importants associés :

```
Topic #0: array object json form property map datum return ajax data
Topic #1: string python model dataframe column character match panda dictionary regex
Topic #2: server application node client log spring connection entry load resource
Topic #3: class x java index c++ filter reference vector section syntax
Topic #4: document exception link website memory sum location pattern menu store
Topic #5: api app html javascript css device rest video category endpoint
Topic #6: instance print process thread background console event app ios time
Topic #7: command android window studio pointer js constructor widget extension windows
Topic #8: datum time query date loop data day attribute format month
Topic #9: text c statement r show point n xml plot collection
Topic #10: update response google module package tag dependency cloud firebase status
```

Figure 16: Extrait des résultats d'une modélisation LDA avec 20 sujets

On voit que pour les sujets détectés certains termes techniques ressortent et pourraient être utilisés comme tags, néanmoins la distinction entre les sujets est qualitative et certains regroupent parfois plusieurs thèmes ensemble.

Il est possible de faire varier le nombre de sujet à détecter par l'algorithme. Plusieurs valeurs ont été testées et le choix s'est fait sur des critères qualitatifs :

- Consistance des sujets
- Limiter la redondance dans les sujets
- Nombre de tags potentiels par sujets
- Absence de sujets sans tags potentiels

Au final le nombre de 100 sujets a été retenu. Le processus est similaire pour la LDA et NMF.

C) Association de tags à chaque sujet

Une fois les sujets détectés, il était nécessaire de leur associer des tags. Pour cela il a été utilisé une liste de 1000 tags potentiels établie au début du projet en sélectionnant les 1000 tags les plus populaires. Selon l'algorithme, les 10 ou 20 termes les plus importants de chaque sujets ont été retenus dans la matrice sujets / features obtenue en sortie de LDA (figure 17), ainsi que leur poids puis filtrés pour ne conserver que ceux qui étaient présent dans la liste de tags populaires. Le résultat final est représenté par la figure 18:

		Features					
		Database	pandas	time	user	...	css
Sujets	Sujet 1	1,2	4	8,5	0,1	...	0,04
	Sujet 2	4,1	5,2	7,3	0,001	...	2,6
	Sujet 3	0,01	0,2	1,7	5,8	...	0,3

	Sujet X	5,2	14,2	0,1	0,003	...	3,7

Figure 17: Matrice exprimant les sujets en fonctions des features obtenue en sortie de LDA



		Tags potentiels + poids
Sujet 1	{ time : 8.5, date : 7.3 }	
...	...	
...	...	
...	...	
Sujet X	{ pandas : 14.2, dataframe : 13.9, python : 12.1 }	

Figure 18: Extrait du dictionnaire associant des tags à chaque sujet

D) Création d'une matrice questions / tags :

L'autre donnée de sortie des algorithmes non supervisés est une matrice donnant pour chaque question la proportion associée à chaque sujet (figure 19). A partir de ce résultat on peut ensuite transformer cette matrice en une matrice questions / tags en utilisant le résultat obtenu au IV.C. Il suffit de créer une nouvelle variable pour chaque tag potentiel de chaque sujet. La valeur associée pour une question à chacune de ces nouvelles variables est le produit entre le poids du sujet pour la question (figure 19) et le poids du tag pour le sujet concerné (figure 20). Le résultat est une matrice questions / tags (figure 21), chaque tag ayant un poids différent. Un même tag pouvant être partagé par différents sujets, il est possible que des variables soient en double. Dans ce cas les doublons sont agrégés en effectuant la moyenne, considérant que si un tag est présent dans un sujet avec un poids important et dans un autre avec un faible poids il n'est peut être pas pertinent, son poids en est alors réduit.


Les chiffres donnés dans les figures sont purement fictifs et ne servent qu'à illustrer le processus de transformation.

		Sujets					
		Sujet 1	Sujet 2	Sujet 3	Sujet 4	...	Sujet X
Questions	Q1	5,8	0,1	1,7	3,2	...	1,2
	Q2	0,3	0,2	1,2	0,001	...	3,2
	Q3	4,1	1,7	0,001	0,01	...	0,04
	QX	0,5	0,1	2,6	1,2	...	7,3

Figure 20: Matrice questions / sujets

		Tags potentiels + poids		
Sujet 1		{ time : 8.5, date : 7.3		
...		...		
...		...		
Sujet X		{ pandas : 14.2, dataframe : 13.9, python : 12.1		

Figure 19: Dictionnaire Sujets / tags



		Tags potentiels					
		time	date	...	pandas	dataframe	python
Questions	Q1	49,3	42,34	...	17,04	16,68	14,52

	QX	4,25	3,65	...	103,66	101,47	88,33

Figure 21: Matrice questions / tags

E) Transformation en un problème de classification

Une fois la matrice questions / tags établie, il reste à transformer celle ci afin que le problème devienne non plus un problème de régression, mais un de classification. Pour cela on conserve uniquement les 10 tags ayant le poids le plus important pour chaque question . On assigne ainsi à ces tags la valeur 1 et la valeur 0 aux autres. Ci dessous en figure 22 une illustration :

		Tags potentiels					
		time	date	...	pandas	dataframe	python
Questions	Q1	49,3	42,34	...	17,04	16,68	14,52

	QX	4,25	3,65	...	103,66	101,47	88,33

		Tags potentiels					
		time	date	...	pandas	dataframe	python
Questions	Q1	1	1	...	0	0	0

	QX	0	0	...	1	1	1




Figure 22: Processus de transformation en problème de classification multi-label

F) Visualisation des nouvelles cibles

Ci dessous en figures 23 et 24 une représentation des nuages de mots obtenus à la fin des différentes étapes :

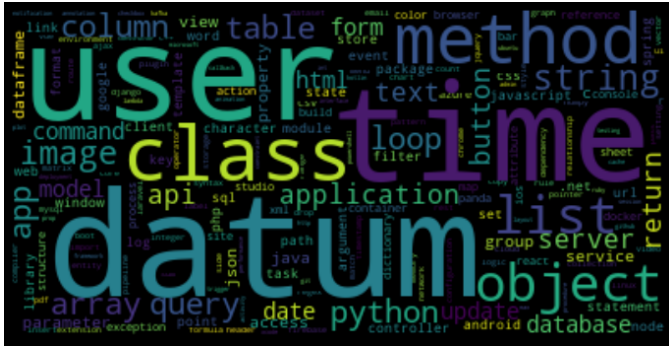


Figure 24: Tags les plus fréquents en sortie de LDA



Figure 23: Tags les plus fréquents en sortie de NMF

On note que les tags retenus sont souvent plus généralistes que ceux observés avec l'apprentissage purement supervisé, se basant moins sur les langages et les frameworks.

G) Résultats de la modélisation supervisée

La dernière étape consiste en la création d'un modèle supervisé de suggestion de tags à partir des nouvelles cibles établies. Le processus est quasi identique à celui de la partie purement supervisée (voir détails en IV.1). Ci dessous en figure les résultats :

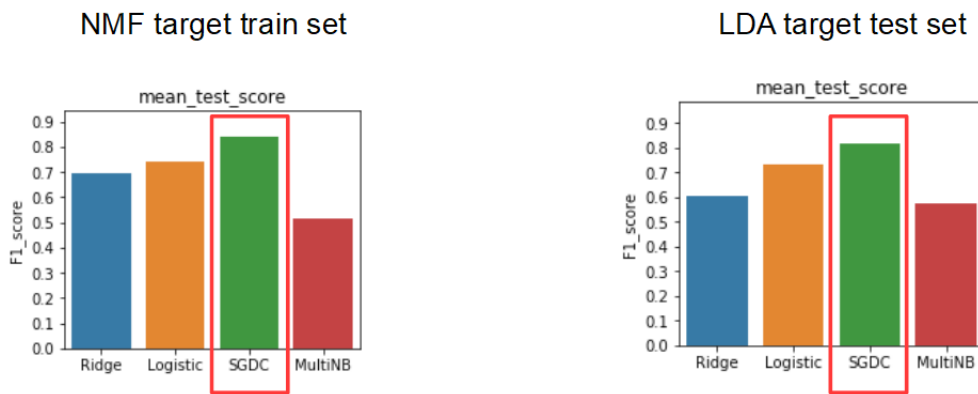


Figure 25: Performances des modèles supervisés à partir des cibles LDA et NMF

Comme pour l'étude avec apprentissage purement supervisé le classifieur linéaire avec SGDC donne les meilleures performances. Les valeurs de F1 score sont plus élevées dans ce cas, mais cela ne veut pas dire grand chose puisque la cible est totalement différente et les mots clés même prédits correctement peuvent ne pas avoir de sens puisque issus de l'approche non supervisée.

La performance obtenue sur le test set est similaire à celle obtenue avec cross validation, aux alentours de 0,84 pour les deux modèles.

V. Conclusion

A) Exemples d'utilisation

Ci dessous 3 exemples qui comparent les 3 modèles obtenus, celui purement supervisé et ceux obtenus avec une construction non supervisée de la cible (NMF et LDA). Le titre original de la question et les tags employés sont donnés à chaque fois :

```
Titre de la question :
how to update spyder on anaconda

Tags utilisés :
['python', 'python-2.7', 'anaconda', 'spyder']

Tags suggérés :
LDA : ['string', 'python', 'command', 'update', 'character', 'csv']
NMF : ['window', 'python', 'command', 'update', 'shell']
Supervised : ['python']
```

Figure 26: Premier exemple d'utilisation des 3 modèles

```
Titre de la question :
Insert Data with pymysql using inputs

Tags utilisés :
['python', 'mysql', 'python-3.x', 'pymysql']

Tags suggérés :
LDA : ['time', 'string', 'array', 'query', 'database', 'sql', 'character', 'attribute']
NMF : ['database', 'table', 'string', 'query', 'update', 'sql', 'statement', 'php']
Supervised : ['mysql']
```

Figure 27: Second exemple d'utilisation des 3 modèles

```
Titre de la question :
How to use AOP with Feign calls

Tags utilisés :
['java', 'aop', 'aspectj', 'spring-aop', 'feign']

Tags suggérés :
LDA : ['class', 'method', 'application', 'api', 'update']
NMF : ['application', 'exception', 'method', 'class', 'api', 'server', 'update', 'boot']
Supervised : ['java', 'spring']
```

Figure 28: Troisième exemple d'utilisation des 3 modèles

B) Conclusion et perspectives

Après cette étude, nous avons obtenus 3 modèles de suggestion de tags. Un obtenu de manière purement supervisé et deux autres de manière hybride, croisant méthodes supervisées et non supervisées. Il est difficile de départager ces modèles car les cibles employées ne sont pas les mêmes.

Néanmoins, en se basant sur les exemples précédents, le modèle obtenu de manière purement supervisé paraît plus adéquat. En effet les tags proposés sont souvent plus pertinents, alors que les deux autres modèles vont avoir tendance à proposer de nombreux tags, en relation avec le sujet de manière plus large.

A ce stade il s'agit de toute manière d'une preuve de concept sur la possibilité d'établir un système de suggestion de tags de cette manière. En effet le nombre de tags cibles ayant été limité à 100 alors qu'il en existait plus de 11 650 dans le set de départ, toutes les possibilités sont loin d'avoir été explorées. Pour aller plus loin une puissance de calcul plus importante serait nécessaire. Les perspectives suivantes seraient intéressantes à explorer :

- Augmenter le nombre de tags cibles
- Augmenter le nombre de sujets pour les méthodes non supervisées
- Utiliser les méthodes supervisées comme outils de réduction de dimension pour créer de nouvelles features en entrée de modèle et non plus une cible. Cela permettrait également de comparer les approches supervisées et non supervisées.
- Tester des algorithmes plus poussés dans le problème de classification multi-label de texte à grande échelle. Par exemple l'utilisation de BERT (Bidirectional Encoder Representations from Transformers), Xlnet ...