

Catégorisez automatiquement des questions

Projet n°6

Parcours Openclassroom Data Science

Introduction

L'objectif de ce projet est de développer un système de suggestion de tags pour une question posée sur le site Stack Overflow. Le but est d'aider les membres du site à mieux classer leurs questions et avoir des réponses potentiellement plus pertinentes.

Dans un premier temps, je vais récupérer les données à partir d'une API du site Stack Overflow, puis je vais les analyser et traiter en utilisant des méthodes propres au traitement du langage naturel afin d'en tirer tout leur potentiel.

Dans un second temps, je vais mettre en œuvre 2 approches différentes de recommandation de tags. La première, non supervisée, visera à trouver le sujet principal d'une question et à proposer des mots relatifs au sujet détecté. La seconde, supervisée, visera à généraliser, à des questions non classifiées, les tags des questions déjà classifiées fournis par l'API Stack Overflow.

Le système de recommandation de tags mettant en œuvre les 2 approches sera intégré au travers d'une simple application web. Pour finir, des ouvertures à l'amélioration seront proposées.

Récupération des données

- Récupération de **91 947 questions** en utilisant l'API StackOverflow

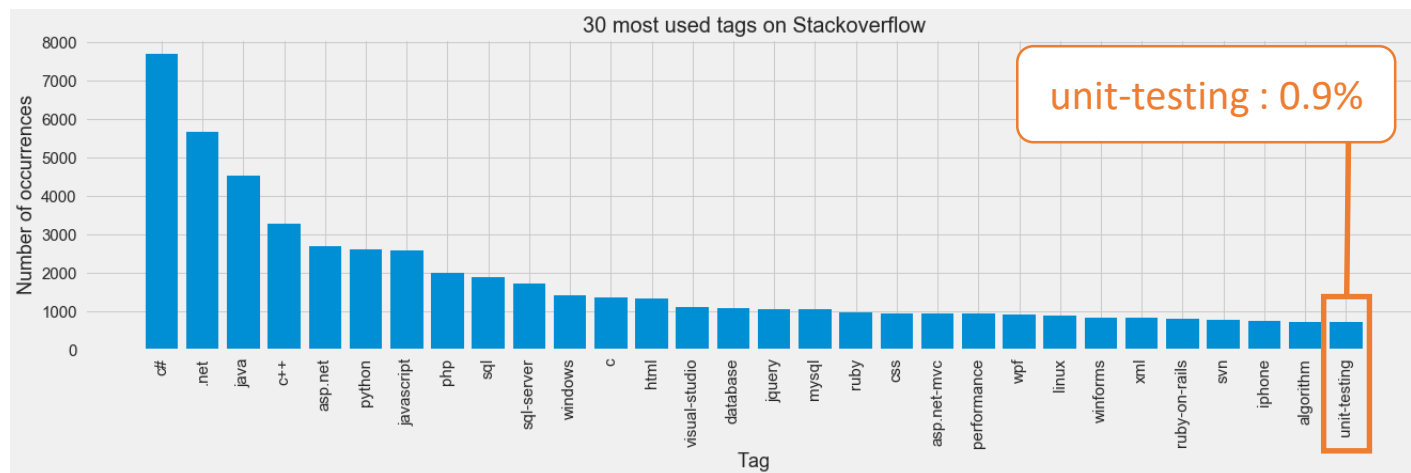
```
select Posts.Id,  
       Name,  
       Score,  
       Body,  
       Tags  
from Posts  
  inner join PostTypes on Posts.PostTypeId = PostTypes.id  
where PostTypes.Name = 'Question'  
  and Posts.Id >= 550000  
  and Posts.Id < 600000
```

Id	121656
Name	Question
Score	3
Body	<p><p>I have the following string and I would like to remove <code>&lt;bpt *&gt;*&lt;/bpt&gt;</code> and <code>&lt;ept *&gt;*&lt;/ept&gt;</code> (notice the additional tag content inside them that also needs to be removed) without using a XML parser (overhead too large for tiny strings).</p>\r\n\r\n<pre><code>The big &lt;bpt i="1" x="1" type="bold"&gt;&lt;b&gt;&lt;/bpt&gt;black&lt;ept i="1"&gt;&lt;/b&gt;&lt;/ept&gt; &lt;bpt i="2" x="2" type="ulined"&gt;&lt;u&gt;&lt;/bpt&gt;cat&lt;ept i="2"&gt;&lt;/u&gt;&lt;/ept&gt; sleeps.\r\n</code></pre>\r\n\r\n<p>Any regex in VB.NET or C# will do.</p>\r\n</p>
Title	Regular expression to remove XML tags and their content
Tags	<c#><.net><xml><vb.net><regex>

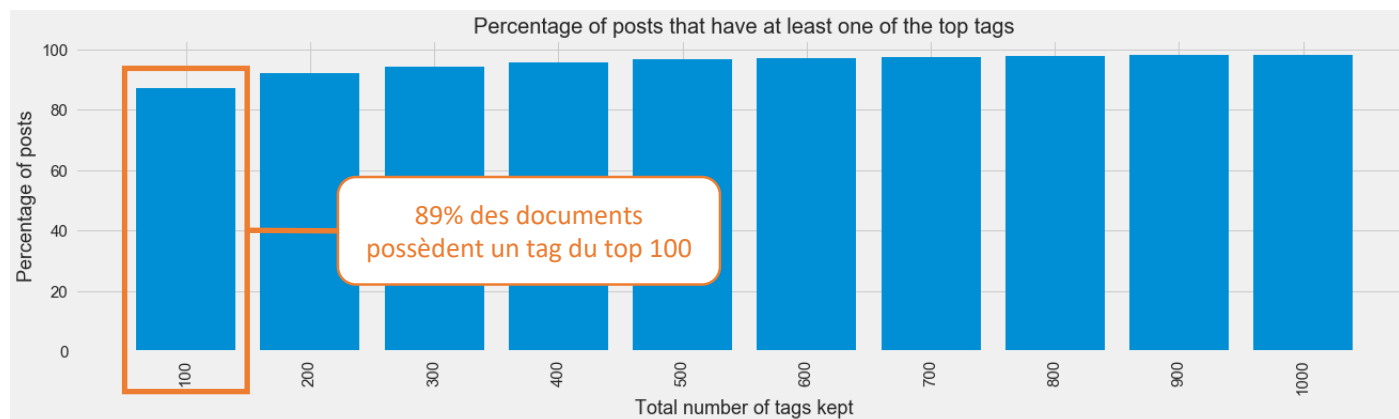
- Aucune valeur manquante et aucune question en doublon
- La colonne *Body* est parsemée de sauts de ligne ou balises HTML
- Fusion des colonnes *Body* et *Title*
- Conservation des posts dont le score est supérieur ou égal à 3, soit **55 598 documents**

Analyse de la variable Tags

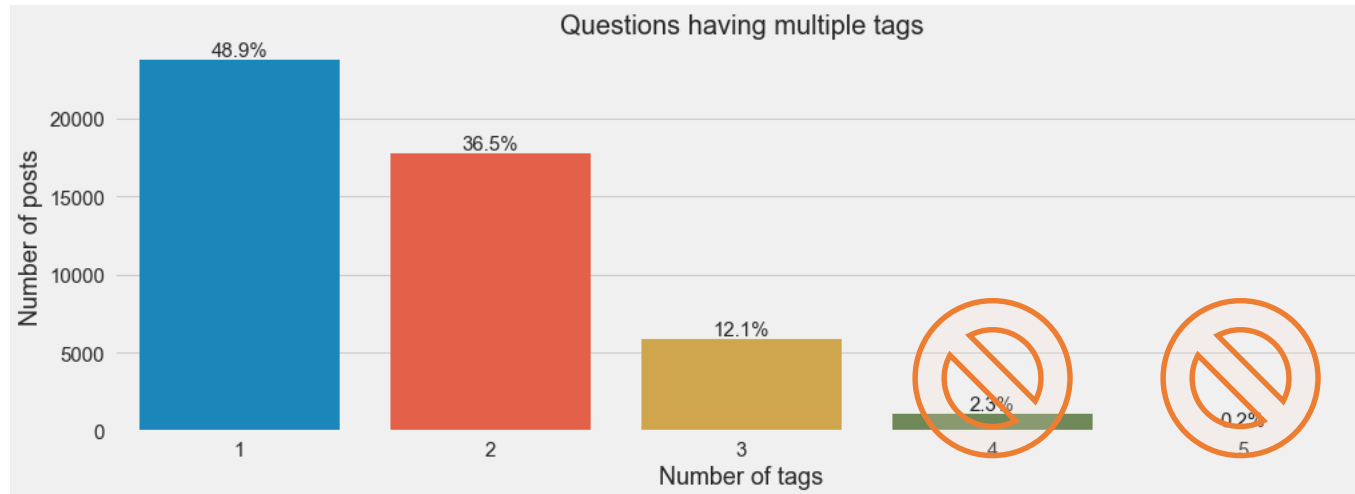
- **9 712 tags distincts**, grande diversité du vocabulaire avec de nombreux termes peu fréquents



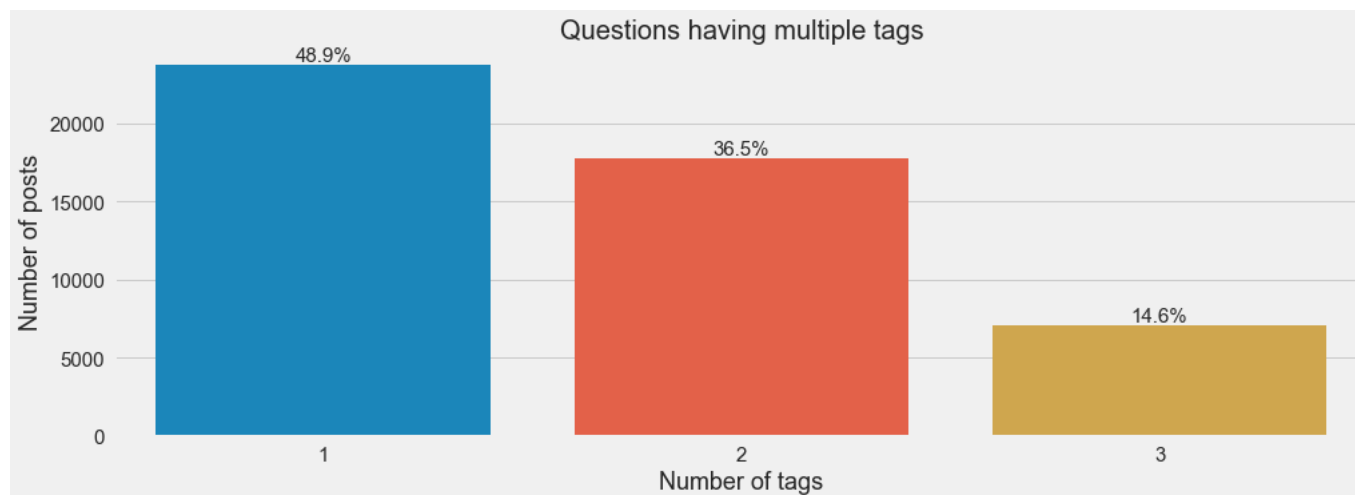
- Ne conserver que les 100 tags les plus fréquents me permet de m'affranchir en partie du fléau de la dimension tout en restant suffisamment exhaustif (**48 527 documents conservés**)



Analyse de la variable Tags



Peu de documents (1 209) ont plus de 3 tags



Je ne conserve dans la colonne tags que les 3 tags les plus fréquents pour des raisons de parcimonie et considérant que 3 tags sont suffisants pour décrire une question

Body : Natural Language Processing

40240 <p>I have the following string and I would like to remove <code><bpt *>*</bpt></code> and <code><ept *>*</ept></code> (notice the additional tag content inside them that also needs to be removed) without using a XML parser (overhead too large for tiny strings).</p>\r\n\r\n<pre><code>The big <bpt i="1" x="1" type="bold"></bpt>black<ept i="1"></ept><bpt i="2" x="2" type="ulined"><u></bpt>cat<ept i="2"></u></ept> sleeps.\r\n</code></pre>\r\n\r\n<p>Any regex in VB.NET or C# will do.</p>\r\nRegular expression to remove XML tags and their content

- Mise en minuscules du texte, suppression des caractères « whitespace » et du code au motif que le vocabulaire contenu entre des balises code est à mon sens trop spécifique

40240 <p>i have the following string and i would like to remove and (notice the additional tag content inside them that also needs to be removed) without using a xml parser (overhead too large for tiny strings).</p> <pre></pre> <p>any regex in vb.net or c# will do.</p> regular expression to remove xml tags and their content

- Suppression du format HTML avec le package Beautiful Soup

40240 i have the following string and i would like to remove and (notice the additional tag content inside them that also needs to be removed) without using a xml parser (overhead too large for tiny strings). any regex in vb.net or c# will do. regular expression to remove xml tags and their content

- Recodage des top tags possédant des caractères spéciaux dans les documents pour éviter des effets de bord indésirables avec la suppression de la ponctuation ou la tokenisation + Suppression de la ponctuation

40240 i have the following string and i would like to remove and notice the additional tag content inside them that also needs to be removed without using a xml parser overhead too large for tiny strings any regex in xyzspecialtags16zyx or xyzspecialtags26zyx will do regular expression to remove xml tags and their content

Body : Natural Language Processing

- Suppression des stopwords proposés par les modules NLP (NLTK et Spacy)
- Lemmatisation pour réduire les mots à leur forme neutre canonique
- Suppression des mots qui ne sont pas des noms (POS tagging) considérant que les verbes ou adverbes n'apportent pas de valeur ajoutée pour la problématique

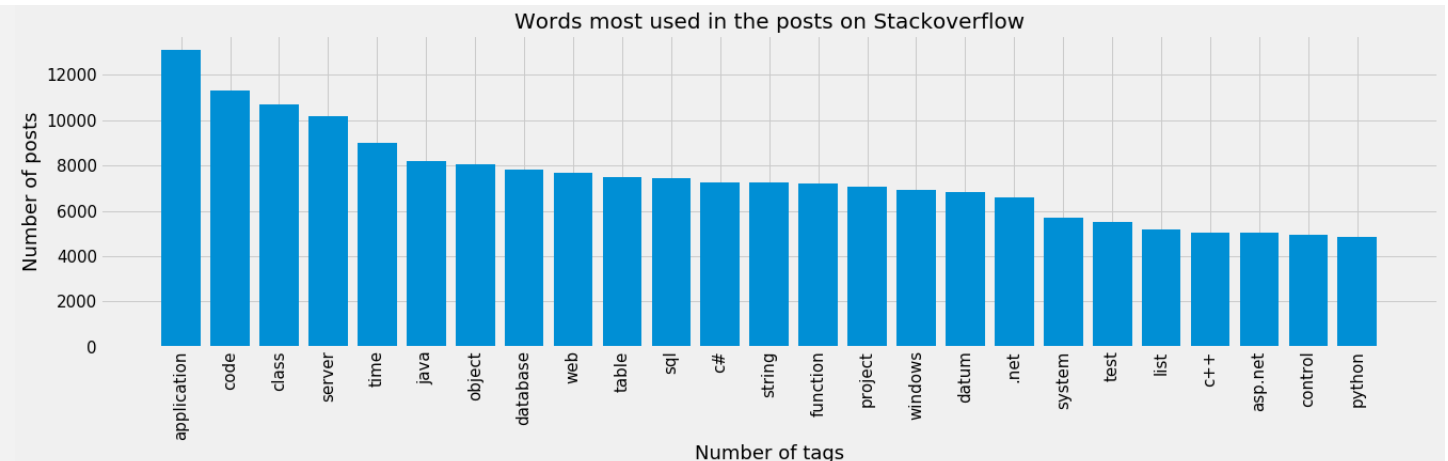
40240 string notice tag content xml parser string regex xyzspecialtags16zyx xyzspecialtags26zyx expression xml tag

- Suppression de stopwords manuels. Certains des 200 premiers mots par occurrence sont très génériques et n'apportent pas de valeur ajoutée, ils sont supprimés

'file', 'way', 'user', 'use', 'problem', 'work', 'example', 'method', 'question', 'value', 'thank', 'solution', 'thing', 'number', 'change', 'idea', 'answer', 'issue', 'update', 'lot', 'message', 'information', 'people', 'reason', 'help', 'want', 'run', 'need', 'end', 'default', 'difference', 'suggestion', 'approach', 'task', 'implementation', 'check', 'e', 'custom', 'place', 'practice', 'support', 'experience', 'product', 'stuff', 'comment', 'note', 'argument', 'year'

Conclusion après NLP :

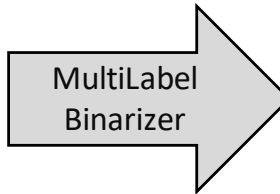
- Tous les documents possèdent au moins un mot.
- Il y a **24 103 mots distincts**, **914 109 en tout** et environ **19 mots par document en moyenne**.



Préprocessing des données avant modélisation

- Je transforme la variable *Tags* à l'aide d'un *MultiLabelBinarizer* pour la modélisation supervisée

	Tag 1	Tag 2	Tag 3
Document 1	Python		
Document 2	Pandas	Numpy	Python
Document 3	Database	SQL	

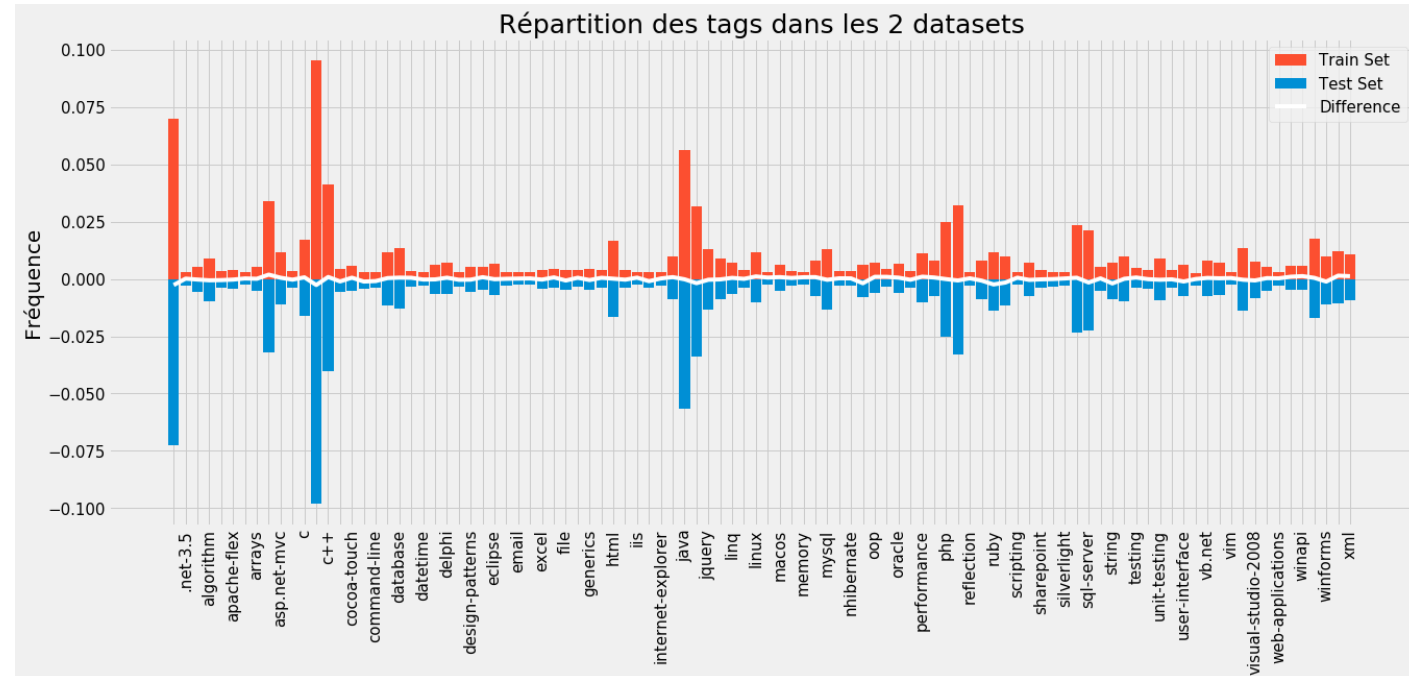
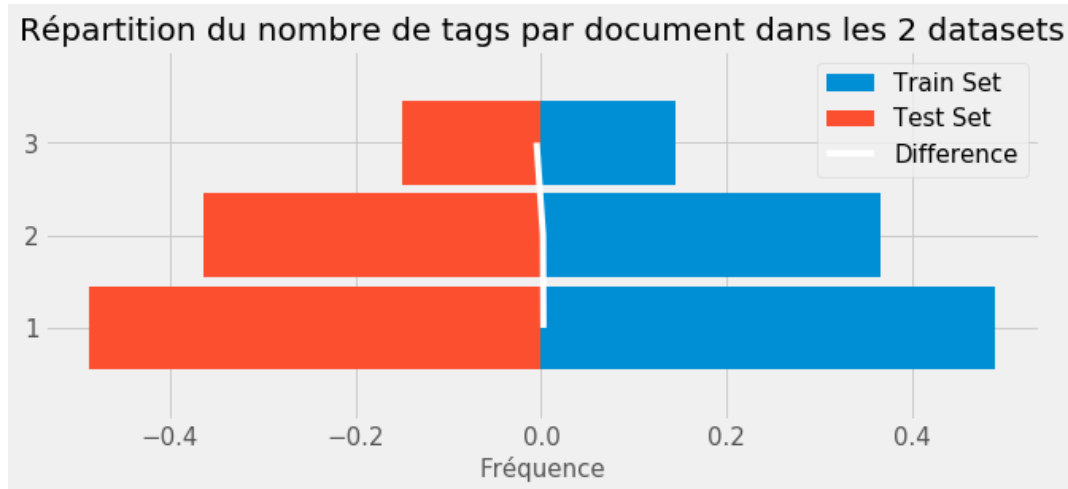


	Python	Pandas	Numpy	Database	SQL
Document 1	1	0	0	0	0
Document 2	1	1	1	0	0
Document 3	0	0	0	1	1

- J'obtiens une matrice de taille (documents x tags) soit **(48 527 x 100) de valeurs binaires** indiquant la présence ou non d'un ou plusieurs tags pour chaque document

Préprocessing des données avant modélisation

- Je splitte les jeux de données *Tags* et *Body* en jeux d'entraînement (80% soit **38 821 documents**) et validation (20% soit **9 706 documents**)



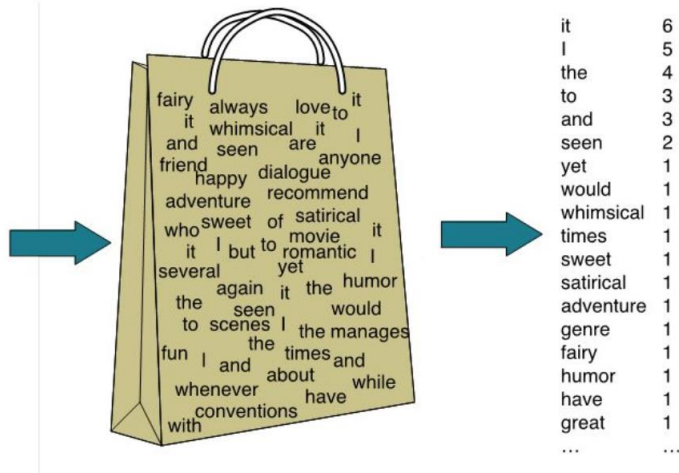
Les jeux de données sont bien équilibrés

Préprocessing des données avant modélisation

- Je transforme la variable *Body* préprocessée en « bag of words » à l'aide de la méthode *CountVectorizer* qui se présente sous la forme d'une matrice où chaque document est représenté par un vecteur de la même dimension que le vocabulaire, dont la composante *i* indique le nombre d'occurrences du *i*-ème mot du vocabulaire dans le document.

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

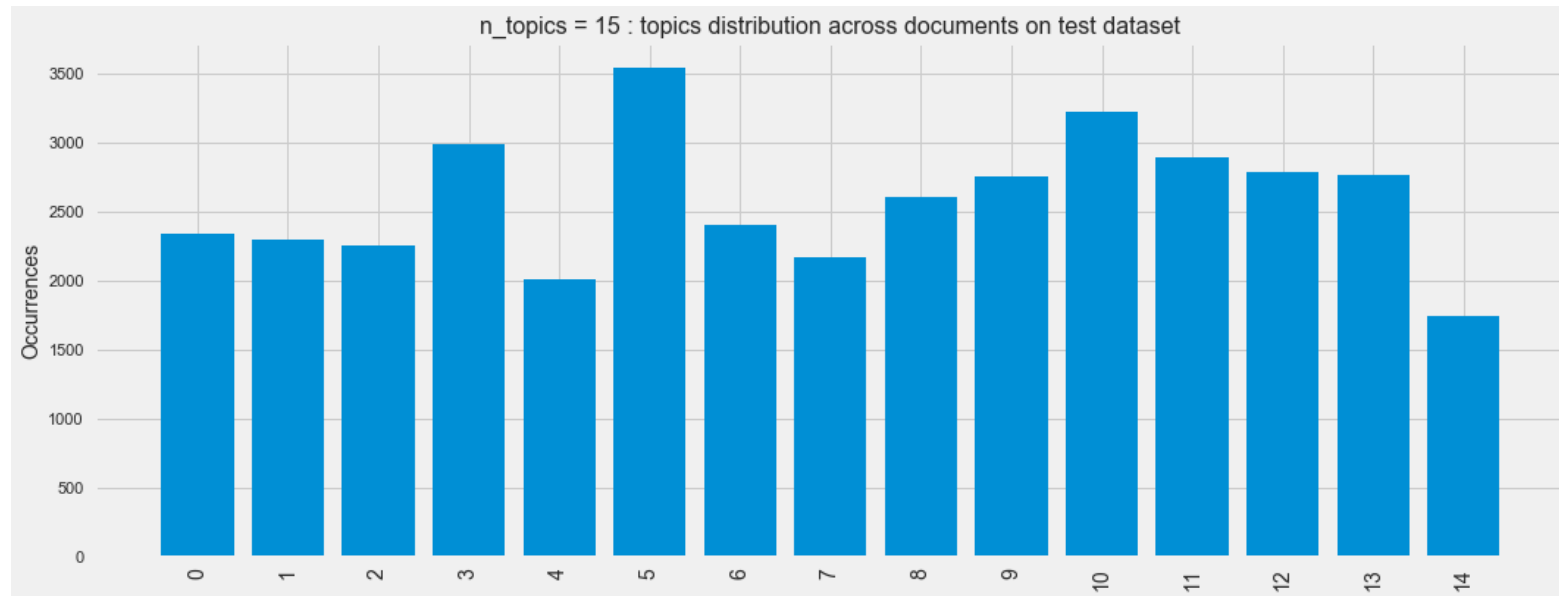


15

- Je modifie le *token_pattern* par défaut pour conserver les termes d'un seul caractère
 - Je ne prends en compte que les unigrams
- Après plusieurs itérations, je retiens **min_df = 150** ce qui permet de réduire mon vocabulaire à 585 mots, soit une matrice d'entraînement de dimension **38 821 x 585**

Non supervisé : Latent Dirichlet Allocation

- Méthode qui cherche à déduire la structure des topics du corpus en fonction des mots et des documents
- L'indicateur de perplexité semble souffrir d'un bug avec Sklearn
- Je choisis le nombre de topics de façon à ce qu'il discrimine le mieux possible les documents



- A mon sens, le meilleur compromis entre l'homogénéité de la répartition des topics dans les documents, la variété des topics et leur cohérence s'obtient avec un **nombre de topics égal à 15**

Non supervisé : Latent Dirichlet Allocation

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
Topic 0	database	datum	sql	server	access	mysql	data	store
Topic 1	list	time	date	item	search	algorithm	linq	regex
Topic 2	c++	c	language	session	point	compiler	code	tree
Topic 3	javascript	html	function	jquery	event	content	browser	css
Topic 4	string	property	character	path	field	format	size	system
Topic 5	windows	studio	project	process	command	script	folder	service
Topic 6	application	xml	web	.net	app	iphone	document	delphi
Topic 7	c#	php	interface	library	git	perl	code	class
Topic 8	server	test	unit	thread	memory	testing	client	connection
Topic 9	table	sql	column	row	query	datum	database	index
Topic 10	class	object	exception	code	instance	collection	constructor	reference
Topic 11	asp.net	web	site	view	http	request	config	service
Topic 12	control	image	form	button	window	ruby	wpf	variable
Topic 13	java	Project	source	eclipse	code	version	svn	subversion
Topic 14	python	Input	django	parameter	bit	procedure	function	integer

Non supervisé : Latent Dirichlet Allocation

Topic 0

Base de données

Topic 1

Expressions régulières et temporelles

Topic 3

Développement web

Topic 6

Développement d'applications graphiques

Topic 8

Tests, performance et architecture

Topic 9

Utilisation d'objets tabulaires

Topic 10

Langage objet

Topic 13

Développement Java et versioning

Recommandation non supervisée de tags

- Le modèle LDA appliqué à une matrice TF permet de produire 2 matrices :
 - Une de dimension $(d \times t)$ contenant les probabilités des topics sachant le document
 - Une de dimension $(t \times w)$ contenant les probabilités des mots sachant le topic
- Le produit matriciel entre les 2 matrices pour obtenir une matrice de dimension $(d \times w)$ contient les probabilités des mots sachant le document
- On peut désormais choisir les N mots les plus probables pour proposer pour un document donné, les N mots les plus liés au topic latent du document

	Mots proposés	Cleaned Body
44492	python, c++, c, language, code	c c++ loop statement
7894	control, image, form, button, window	parentusercontrol host load parentusercontrol access property parentusercontrol childusercontrol time property parent control child control
36398	class, object, exception, code, instance	class asset class class definition asset getdefinition class definition asset getdefinitionbyname
17197	java, project, source, eclipse, code	java effect point operation java
44492	python, c++, c, language, code	c c++ loop statement

Préprocessing des données avant modélisation

- Je transforme la variable *Body* préprocessée en une matrice TF-IDF à l'aide de la méthode *TfidfVectorizer*

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{\text{df}_x} \right)$$

TF-IDF

Term x within document y

$\text{tf}_{x,y}$ = frequency of x in y

df_x = number of documents containing x

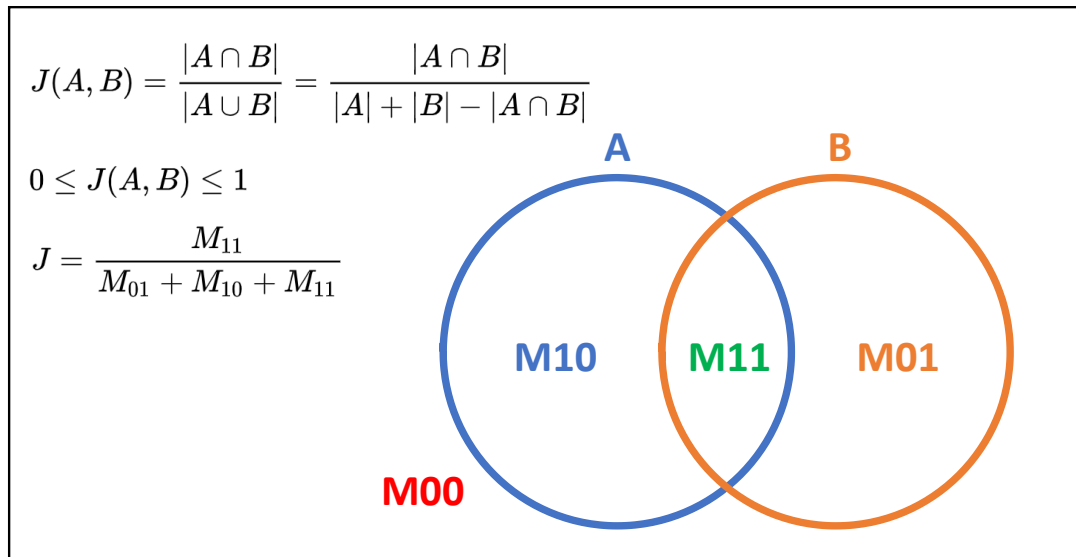
N = total number of documents

- Je modifie le *token_pattern* par défaut pour que TF-IDF conserve les termes d'un seul caractère
- Je ne prends en compte que les unigrams pour le calcul de la matrice TF-IDF
- Tuning des différents paramètres *min_df*, *max_df* et *max_features* de *TfidfVectorizer* à l'aide d'une recherche par grille visant à maximiser l'indice *jaccard_weighted* qui me sert de métrique d'évaluation pour l'analyse supervisée.
- Le meilleur résultat est donné pour **min_df=30**, **max_df=0.1** et **max_features=1000**, ce qui porte mon vocabulaire à 1 636 mots, soit une matrice d'entraînement de dimension **38 821 x 1636**.

Analyse supervisée

Indice de Jaccard

- Comparer un ensemble de labels prédits pour un échantillon aux labels réels correspondant
- Indice de Jaccard pondéré détermine la moyenne des métriques calculée pour chaque label, pondérée par leur distribution réelle observée



Binary Relevance (*OneVsRestClassifier*)

- Décomposition de l'apprentissage multilabel en plusieurs apprentissages binaires indépendants (une par label)
- Présomption d'indépendance des labels entre eux

Producing Meta Labels

dataset		binary relevance			label powerset	meta labels	
instance	labels	Y_A	Y_B	Y_C	$Y_{A,B,C}$	$Y_{A,C}$	$Y_{B,C}$
1	B	0	1	0	B	\emptyset	B
2	B,C	0	1	1	BC	\emptyset	BC
3	C	0	0	1	C	\emptyset	C
4	B	0	1	0	B	\emptyset	B
5	A,C	1	0	1	AC	AC	C
6	A,C	1	0	1	AC	AC	C
7	A,C	1	0	1	AC	AC	C
8	A,B,C	1	1	1	ABC	\emptyset	BC
9	C	0	0	1	C	\emptyset	C

- *binary relevance*: 9 exs, 3×2 binary classes
- *label powerset*: 9 exs, 1×5 multi-class

Analyse supervisée

Dummy Classifier : prédiction en respectant la distribution des labels dans l'échantillon d'entraînement

Jaccard weighted Train	0.024
Jaccard weighted Test	0.025

MultinomialNB

Paramètres testés	alpha : [0.0001, 0.001, 0.01, 0.1] fit_prior : [True, False]
Meilleurs paramètres	alpha : 0.001 fit_prior : False
Jaccard weighted Train	0.211
Jaccard weighted Test	0.195

- Plus performant que le classifieur naïf
- Léger sur-apprentissage

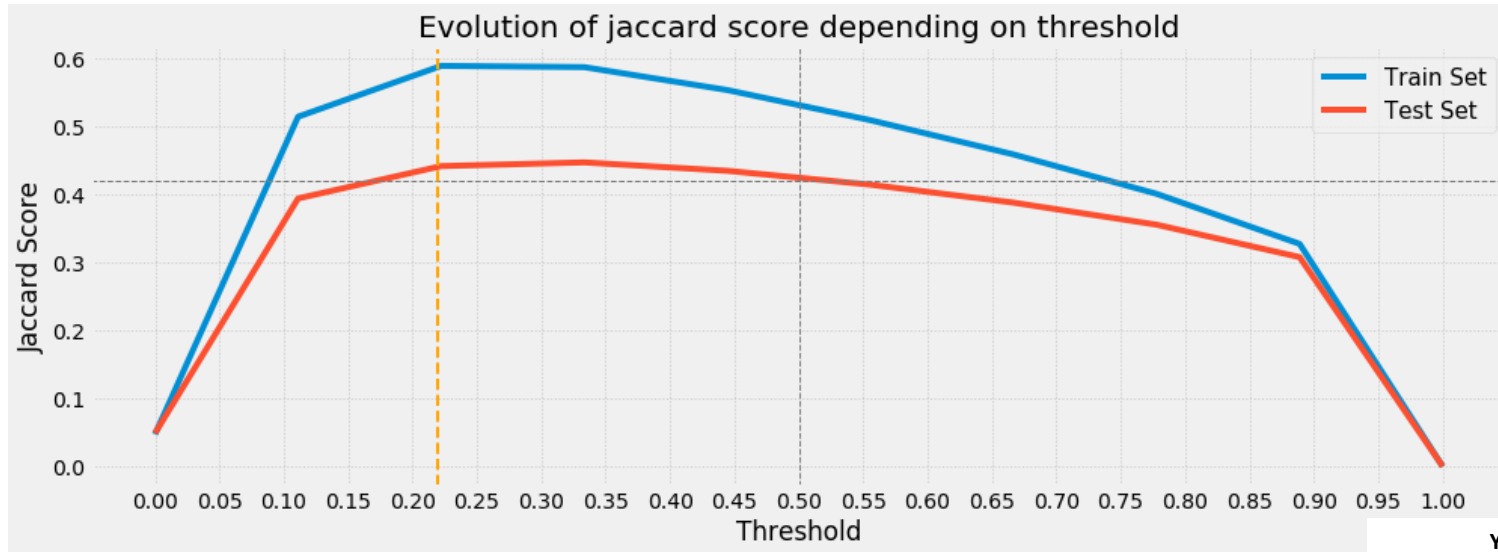
LogisticRegression

Paramètres testés	penalty : ['l1', 'l2'] C : [0.1, 1, 10, 100, 1000]
Meilleurs paramètres	penalty : l1 C : 10
Jaccard weighted Train	0.532
Jaccard weighted Test	0.424

- Performance considérablement augmentée
- Sur-apprentissage prononcé
- Modèle retenu pour l'API

Analyse supervisée

Optimisation du threshold

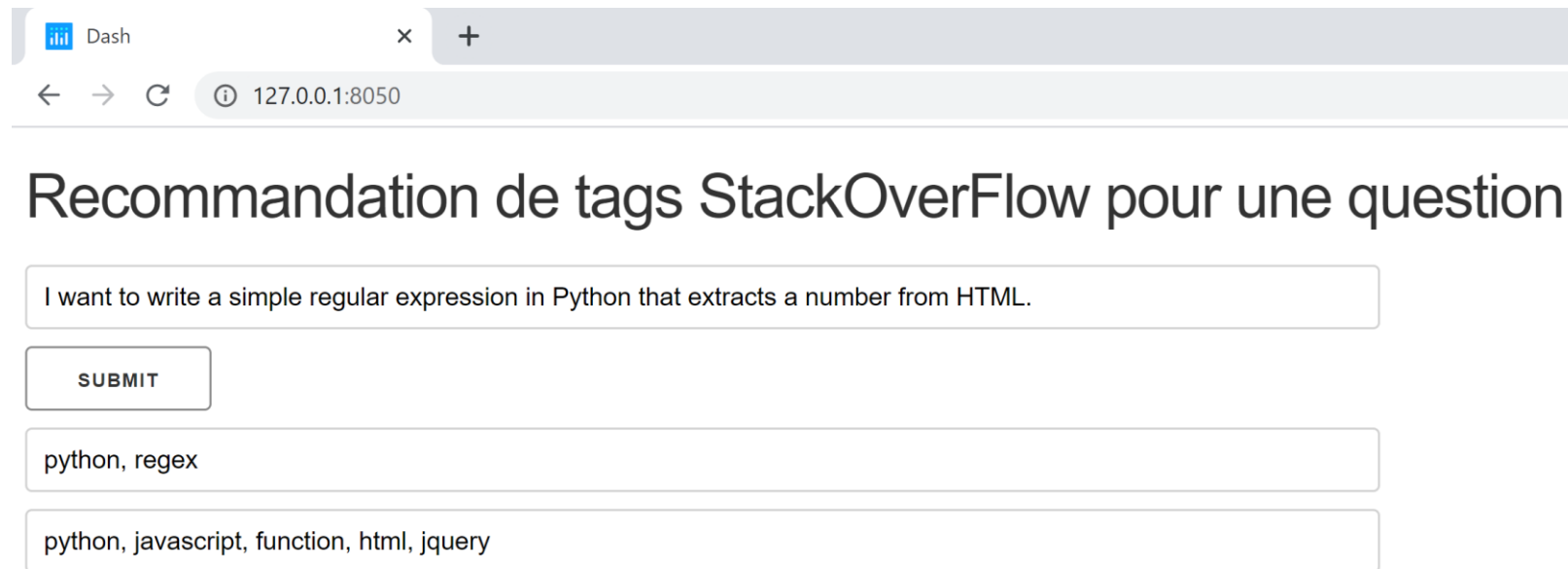


- Modifier le seuil par défaut (= 0.5) n'améliore pas la capacité du modèle à mieux généraliser
- Réduire le seuil à 0.22 a pour effet de favoriser la capacité de mon modèle à fournir une prédiction sans en dégrader la performance

Y_true	Y_pred_0.5	Y_pred_0.22
(ruby, ruby-on-rails)	(performance, ruby, ruby-on-rails)	(performance, ruby, ruby-on-rails)
(c#,,)	()	(c#,,)
(version-control,,)	(algorithm,,)	(algorithm,,)
(.net, c++)	(c++,)	(c++,)
(javascript,,)	(javascript,,)	(internet-explorer, javascript)
(command-line, linux)	()	(shell,,)
(vb.net,,)	(vb.net,,)	(.net, vb.net)
(.net, user-interface, vb.net)	(.net,,)	(.net, multithreading)
(css, html, internet-explorer)	()	(java,,)
(c#, xml)	(c#, xml)	(c#, xml)

API de recommandation de tags

- Propose une liste de tags StackOverflow (jusqu'à 3 prédicts par l'approche supervisée, et 5 prédicts par l'approche non supervisée) relatifs à une question saisie traitant de sujets informatiques et en Anglais



The screenshot shows a web browser window with a single tab titled 'Dash'. The address bar displays '127.0.0.1:8050'. The main heading of the page is 'Recommandation de tags StackOverFlow pour une question'. Below the heading is a text input field containing the question: 'I want to write a simple regular expression in Python that extracts a number from HTML.'. A 'SUBMIT' button is located below the input field. Underneath the button, there are two rows of recommended tags. The first row contains 'python, regex' and the second row contains 'python, javascript, function, html, jquery'.

- Le texte saisi passe par toutes les étapes de preprocessing NLP puis est transformé en matrices TF ou TF-IDF avant application respectivement des modèles supervisés et non supervisés
- Non disponible sur Heroku : Error R15 (Memory quota vastly exceeded)

Pistes d'amélioration

- Pour le modèle non supervisé, retirer les mots les plus fréquents par topic pourrait permettre d'amener un peu plus de spécificité. Je pourrais gagner en spécificité en intégrant des n-grams ou en utilisant des techniques de plongements de mots pour prendre en compte le contexte.
- Il faudrait arriver à gérer l'effet de bord induit par la faible taille des documents qui n'aide pas à bien discriminer les mots importants dans chaque document dans les matrices TF et TF-IDF.
- Je pourrais éviter de prédire les mêmes tags entre supervisé et non supervisé et donc intégrer les tags comme stopwords pour la matrice TF-IDF.
- Au lieu d'avoir un unique modèle supervisé pour prédire l'ensemble des tags, il ne serait pas inintéressant d'isoler les tags identifiant une technologie des autres tags. Je spécialiserais alors un classifieur à prédire la technologie concernée par la question tandis qu'un autre classifieur se concentrerait plutôt à décrire la nature du problème.