Catégorisez Automatiquement des Questions

L.khellouf[1] Mentor: B.Beaufils[2]

[12]OpenClassRooms

Projet 21, mai 2021

Sommaire

Problèmatique

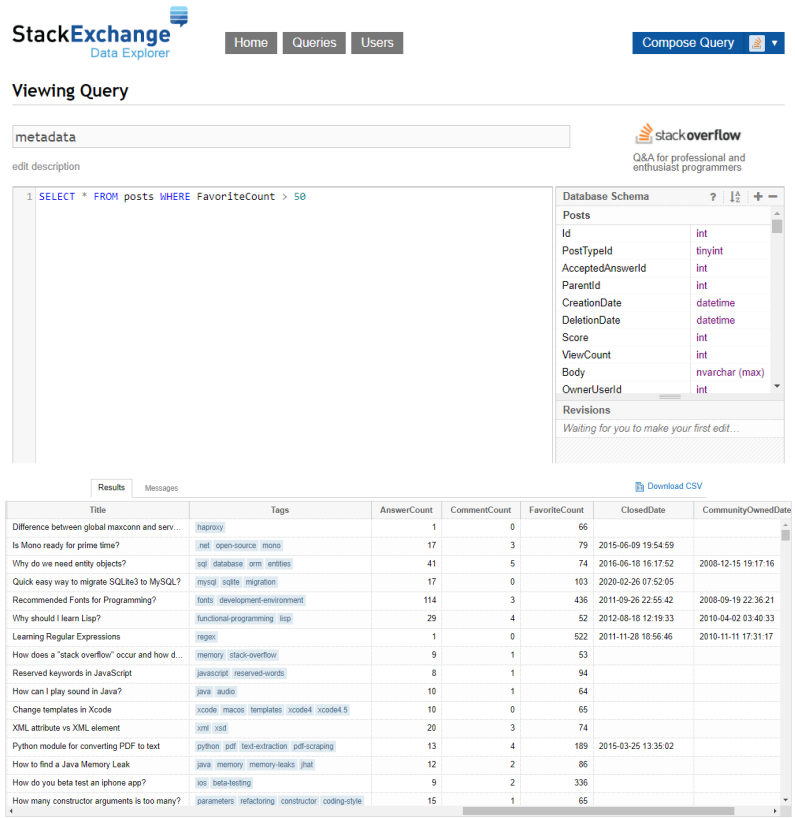
Stack Overflow est un site de questions_réponses liées au développement informatique. Pour poser une question sur ce site, il faut entrer plusieurs étiquettes liées à la question. L'objectif de ce projet est de développer un système de suggestion de tags pour une question posée sur le ce site. Le but est d'aider les membres et aussi les débutants du site à proposer une classifications préliminaire et automatique à leurs questions afin de gagner du temps et ainsi avoir des réponses potentiellement plus pertinentes.

Introduction

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, supervisée, pour suggérer les étiquettes ou tags La seconde, non supervisée, pour suggérer les mots clés ou keywords.
Le système de recommandation de tags mettant en œuvre les 2 approches sera intégré au travers d'une simple application web. Pour finir, des points d'améliorations seront proposées.

Récupération Des Données

- Récupération de 11935 questions pour 23 colonnes en utilisant l'API Stack exchange
- Aucun valeur manquante et aucune question en doublon.
- Fusion des colonnes Body et Title dans une colonne appelée Text



Data Profiling

Analyse de la variable Body

Body
Categorical

HIGH CARDINALITY
UNIFORM
UNIQUE

Distinct	11935
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	93.4 KiB

I'm trying to perform some offline m…	1
<a href="https://stackoverflow.com/…	1
Sealed classes are described in 'Pr…	1
It is supposed to be generally prefer…	1
In terms of Java, when someone as…	1
Other values (11930)	11930

Toggle details

Overview | Categories | Words | Characters

Length

Max length	24750
Median length	512
Mean length	829.1379137
Min length	23

Characters and Unicode

Total characters	9895761
Distinct characters	408
Distinct categories	22
Distinct scripts	11
Distinct blocks	23

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Unique

Unique	11935
Unique (%)	100.0%

Sample

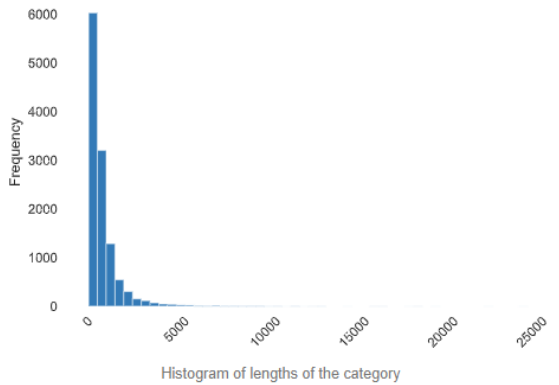
1st row I found an example in the VS2008 Examples for Dynamic LINQ that allows you to use a sql-like string (e.g. <code>OrderBy("Name, Age DESC")</code> for ordering. Unfortunately, the method included only works on <code>IQueryable<T></code>. Is there any way to get this functionality on <code>IEnumerable<T></code>?

2nd row RequireJS seems to do something internally that caches required javascript files. If I make a change to one of the required files, I have to rename the file in order for the changes to be applied. The common trick of appending a version number as a querystring param to the end of the filename does not work with requirejs <code><script src="jsfile.js?v2"></script></code>

Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
I'm trying to perform some offline maintenanc…	1	< 0.1%
<a href="https://stackoverflow.com/questions/…	1	< 0.1%
Sealed classes are described in 'Programmin…	1	< 0.1%
It is supposed to be generally preferable to us…	1	< 0.1%
In terms of Java, when someone asks: <…	1	< 0.1%
Once upon a time, we can watch the most po…	1	< 0.1%
Is there a destructor for Java? I don't seem to …	1	< 0.1%
I would like to know if there is an alternative to…	1	< 0.1%
I am learning about Big O Notation running ti…	1	< 0.1%
After I did brew update and brew upgrade, my…	1	< 0.1%
Other values (11925)	11925	99.9%



Histogram of lengths of the category

Data Profiling

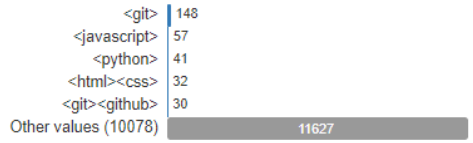
Analyse de la variable Tags

Il existe **5407** tags distincts, et nous avons concervé que **20** tags les plus fréquents.

Tags
Categorical

HIGH CARDINALITY

Distinct	10083
Distinct (%)	84.5%
Missing	0
Missing (%)	0.0%
Memory size	93.4 KiB



Toggle details

Overview | Categories | Words | Characters

Length

Max length	102
Median length	31
Mean length	32.78583997
Min length	3

Characters and Unicode

Total characters	391299
Distinct characters	42
Distinct categories	5
Distinct scripts	2
Distinct blocks	1

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Unique

Unique	9439
Unique (%)	79.1%

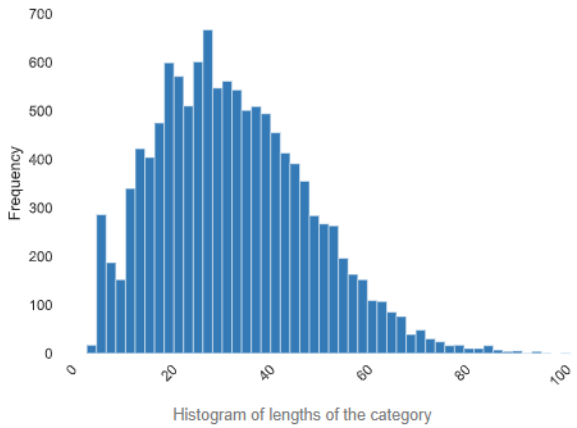
Sample

1st row	<c#><linq><linq-to-objects>
2nd row	<javascript><jquery><requirejs>
3rd row	<algorithm><matrix><multidimensional-array>
4th row	<sql><sql-server><tsql>
5th row	<android><github><intellij-idea><android-studio>

Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
<git>	148	1.2%
<javascript>	57	0.5%
<python>	41	0.3%
<html><css>	32	0.3%
<git><github>	30	0.3%
<javascript><jquery>	28	0.2%
<docker>	27	0.2%
<python><pandas><dataframe>	23	0.2%
<javascript><arrays>	21	0.2%
<node.js>	20	0.2%



Nettoyage de la donnée

- Dans la nouvelle dataset, nous avons gardé uniquement les variables Body, Title, Tags
- Suppression des balises HTML pour la variable Body avec le package Beautiful Soup
- Mise en minuscules de l'ensemeble de champs avec la fonction lower()
- Remplacer les formes contractées par leur forme longue par l'utilisation d'un dictionnaire de correspondance.
- Suppression des stop words et ponctuation avec Spacy, usage des attributs 'is_stop' et 'is_punst' sur les tokens

Nettoyage de la donnée

- Suppression des chiffres y compris à l'interieur des chaine.
- Suppression des espace multiples et les mots de tailles faibles (inférieu à 2)
- Lemmatisation pour réduire les mots à leur forme neutre canonique

Conclusion après NLP :

- Tous les documents possèdent au moins un tag.
- Il y a 8 763 questions

Transformation de la variable Tags à l'aide d'un MultilabelBinarizer pour la modélisation supervisée.

	Tags
Doc 1	[c#]
Doc 2	[Python]
Doc 3	[javascript, jquery]

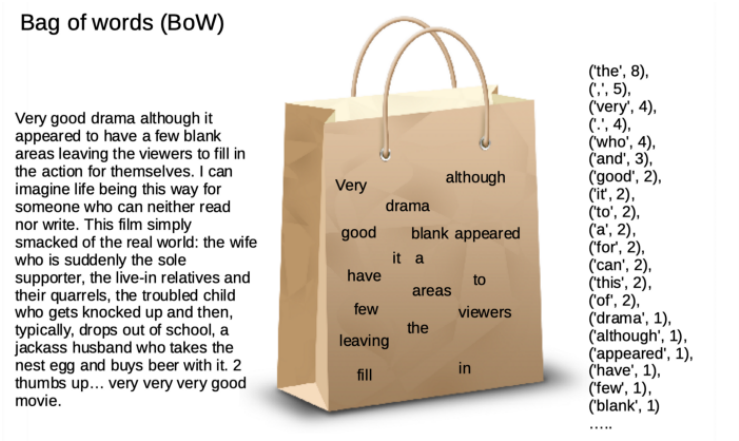
MultilabelBinarizer

	C#	python	javascript	jquery
Doc 1	1	0	0	0
Doc 2	0	1	0	0
Doc 3	0	0	1	1

Préprocessing

Bag of Words

Actuellement, nous avons les messages sous forme de listes de tokens (également appelés lemmes) et nous devons maintenant convertir chacun de ces messages en un vecteur avec lequel les modèles d'algorithme de SciKit Learn peuvent fonctionner.




```
1  from sklearn.feature_extraction.text import CountVectorizer
2  vectorizer = CountVectorizer()
3  corpus = [
4      'This is the first document.',
5      'This is the second second document.',
6      'And the third one.',
7      ]
8
9  X = vectorizer.fit_transform(corpus)
10 print(X)
```

```
(0, 8)        1
(0, 3)        1
(0, 6)        1
(0, 2)        1
(0, 1)        1
(1, 8)        1
(1, 3)        1
(1, 6)        1
(1, 1)        1
(1, 5)        2
(2, 6)        1
(2, 0)        1
(2, 7)        1
(2, 4)        1
```

Cette méthode consiste à compter le nombre d'occurrences des tokens présents dans le corpus pour chaque texte, que l'on divise ensuite par le nombre d'occurrences total de ces même tokens dans tout le corpus.

Words	TF (for A)	TF (for B)	IDF	TFIDF (A)	TFIDF (B)
Jupiter	1/5	0	ln(2/1) = 0.69	0.138	0
Is	1/5	1/8	ln(2/2) = 0	0	0
The	1/5	2/8	ln(2/2) = 0	0	0
largest	1/5	0	ln(2/1) = 0.69	0.138	0
Planet	1/5	1/8	ln(2/2) = 0	0.138	0
Mars	0	1/8	ln(2/1) = 0.69	0	0.086
Fourth	0	1/8	ln(2/1) = 0.69	0	0.086
From	0	1/8	ln(2/1) = 0.69	0	0.086
Sun	0	1/8	ln(2/1) = 0.69	0	0.086

Le but de cette méthode est réduire l'importance relative de certains mots très présents mais peu sinifigatifs.

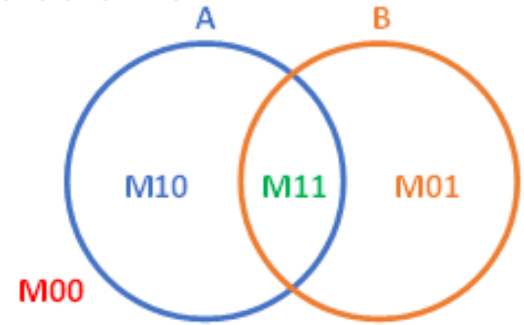
Métrique utilisée est Indice de Jaccard:

- comparer un ensemeble de labels prédits à un échantillon aux labels réels correspondant.
- Indice de Jaccard pondéré détermine la moyenne des métriques calculeé pour chaque label, pondérée par leur distriction réelle observée.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$0 \leq J(A,B) \leq 1$$

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

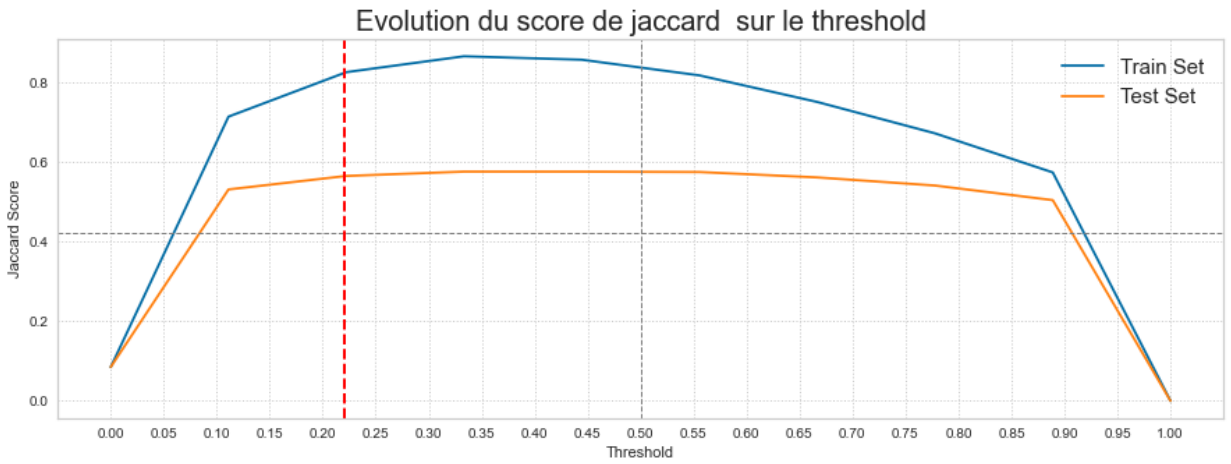


- Mise en place d'une optimisation d'hyperparamètres sur la reprèsentation **Bag_of_words** et **TF_IDF** Pour la reprèsentation vectorielles de mots.
- L'algorithme utilisé est Regression logistic et il dispose dans sa documentation les meilleurs paremètres à optimiser.

	Les hyperparamètres à optimisé sur Bag_of_words et TF_IDF	Best hyperparamètres sur TF_IDF
✓ OneVsRestClassifier ✓ KFold ✓ GridSearch ✓ LogisticRegression	✓ Params= {'estimator__penalty': ['l1', 'l2'], 'estimator__C': [0.1, 1, 10, 100, 1000]} ✓ KFold= 5	✓ {'estimator C': 10, 'estimator_penalty: 'l1'} ✓ Jaccard Score= 63.94

Approche Supervisée

Optimisation du threshold

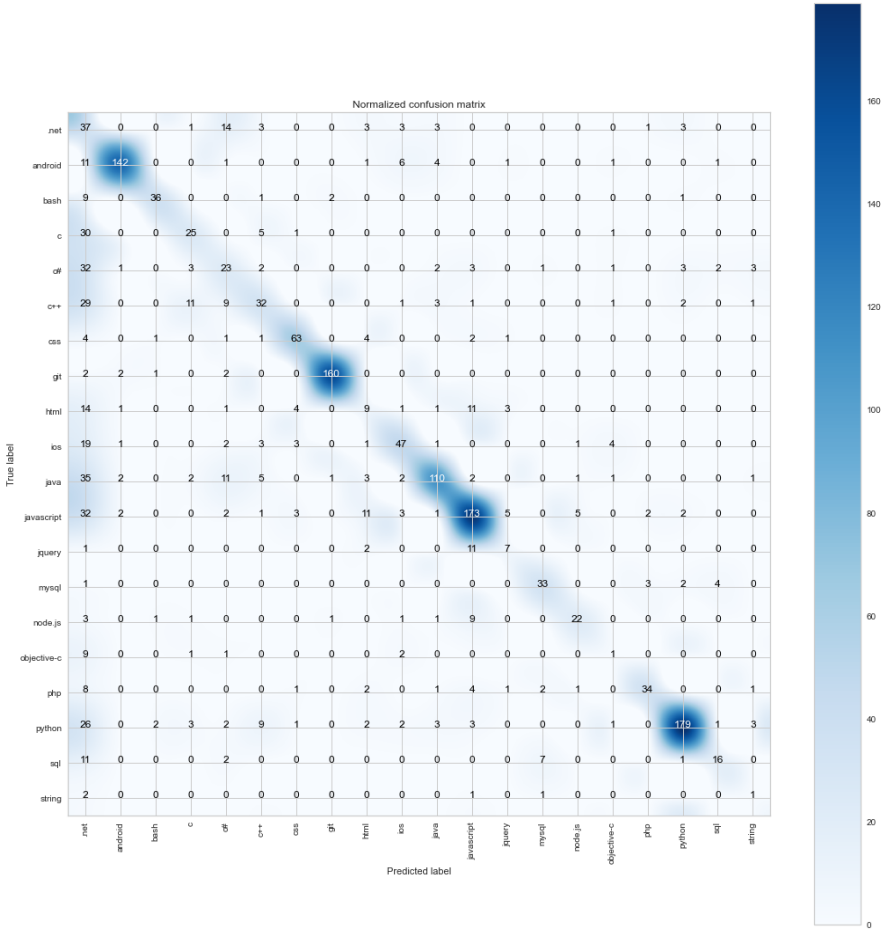


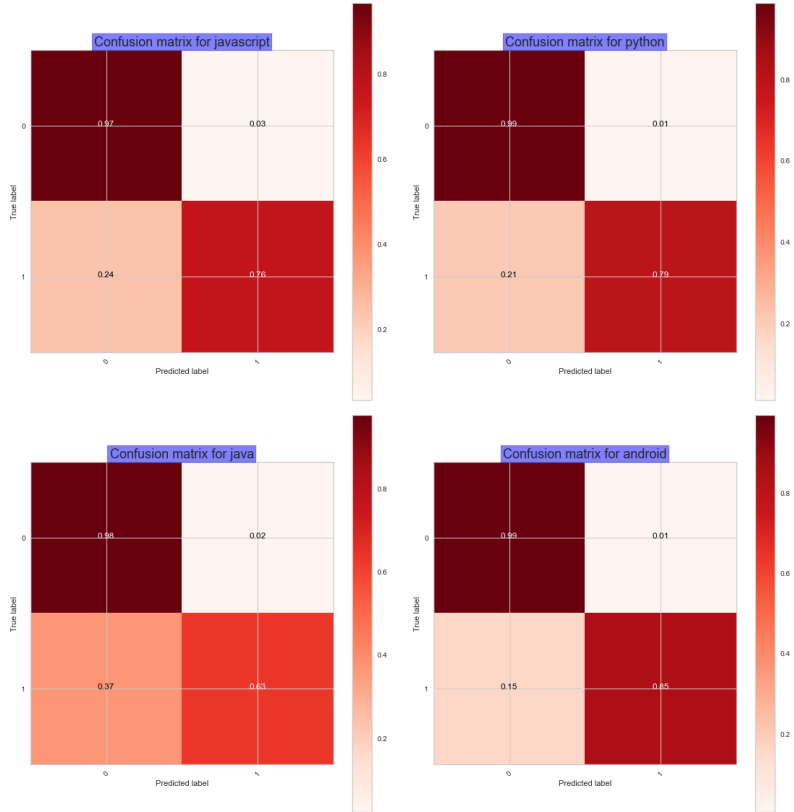
	y_true	y_pred_0.5	y_pred_0.22
20	(css, git, html)	(css, html, javascript)	(css, html, javascript)
21	(java,)	(java,)	(java,)
22	(c, c++)	(c,)	(c, javascript)
23	(git,)	(git,)	(git,)
24	(mysql, python)	(python,)	(python,)
25	(java,)	(java,)	(c#, java)
26	(git,)	(git,)	(git,)
27	(css,)	(css,)	(css, html)
28	(objective-c,)	()	()
29	(mysql, php)	(mysql,)	(mysql,)
30	(android,)	(android,)	(android,)
31	(python,)	(python,)	(python,)
32	(git,)	(git,)	(git,)
33	(ios,)	(objective-c,)	(ios, objective-c, python)
34	(.net, c#)	(c#, string)	(c#, string)
35	(python,)	(python,)	(python,)
36	(ios, objective-c)	()	()
37	(git,)	(git,)	(git,)
38	(css,)	(css, html)	(css, html)
39	(git,)	(git,)	(git,)

- 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 du modèle à fournir une prédiction sans en dégrader la performance.

Approche Supérvisée

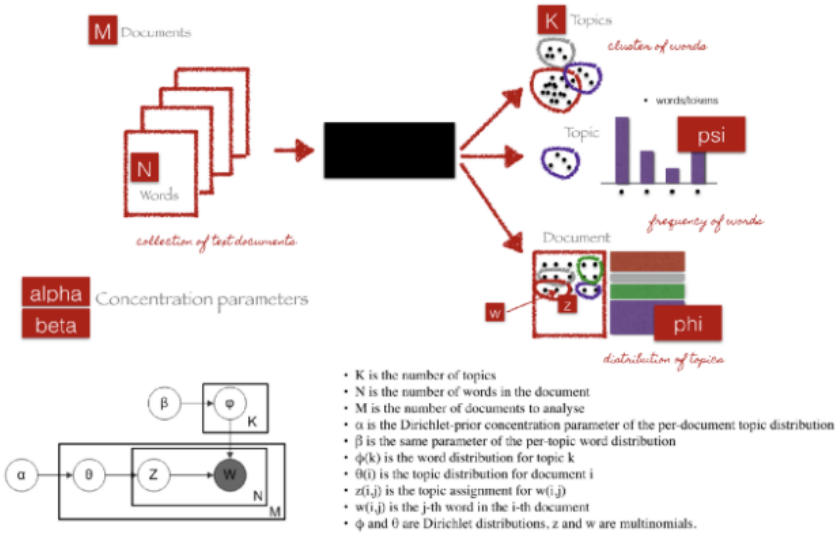
Matrice de Confusion





Approche Non Supervisée

Nous avons utilisé l'algorithme LDA(Latent Dirichlet Allocation) qui est une méthode non supervisée générative pour les mots clés.



http://chdoig.github.io/pytexas2015-topic-modeling/#/3/4

Nous pouvons décrire ce processus génératif comme, étant donné le nombre M de documents, le nombre N de mots et le nombre K préalable de sujets, le modèle s'entraîne à produire:

- psi, la distribution des mots pour chaque sujet K
- phi, la répartition des sujets pour chaque document i.

Approche non supervisée

- Inférence : déterminer les thèmes, les distributions de chaque mot sur les thèmes, la fréquence d'apparition de chaque thème sur le corpus.
- Pour effectuer l'inférence de ce modèle, nous avons utilisé la librairie LDA de gensim.

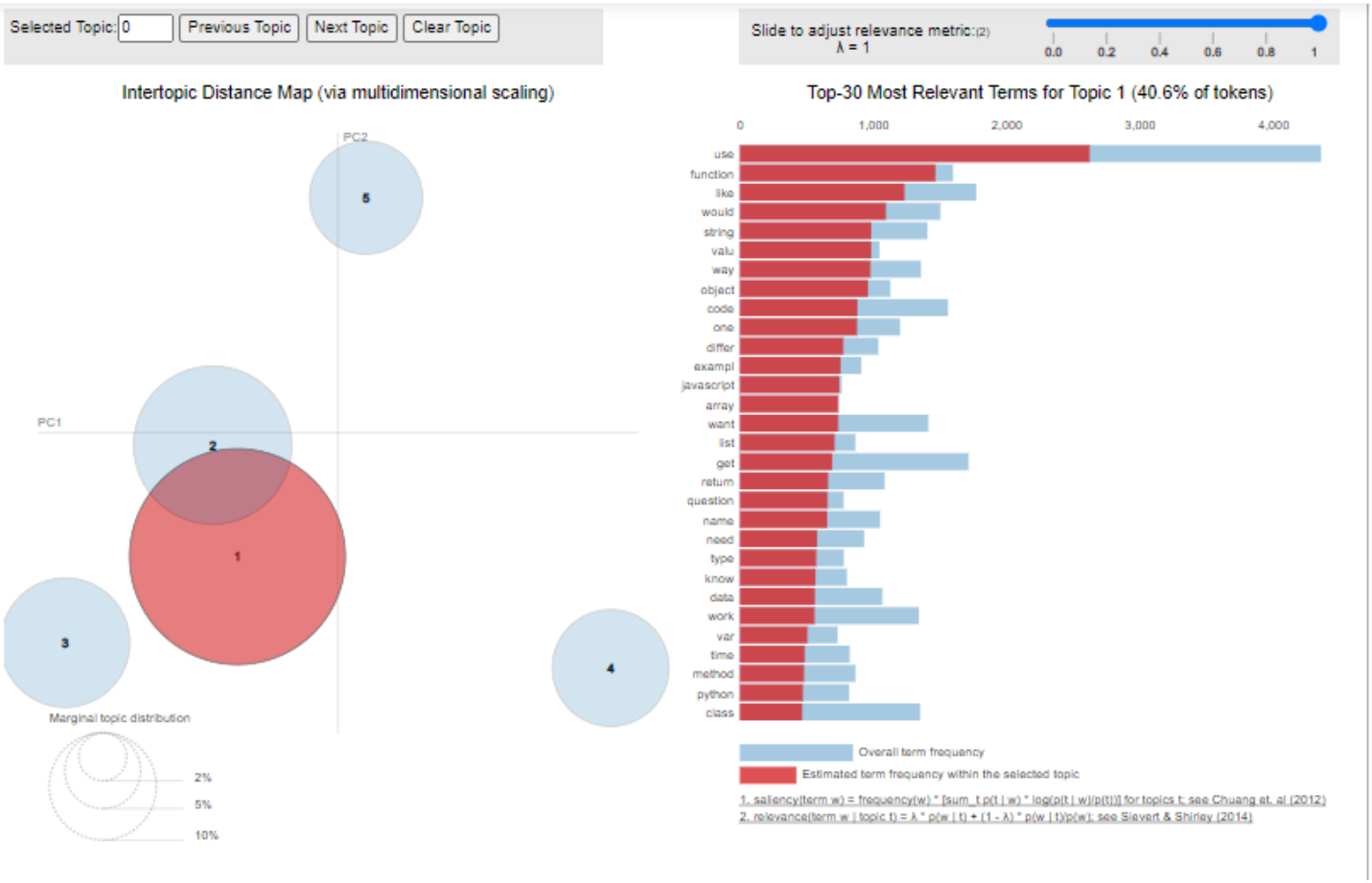
```python
no_top_words = 10
n_topics = 5
def display_topics(model, no_top_words, num_topics):
    for idx, topic in model.show_topics(formatted=False, num_topics=num_topics, num_words= no_top_words):
        print("---------------------------------------------")
        print("Topic %d:" % (idx))
        print(" ".join([w[0] for w in topic]))
        print("---------------------------------------------")

display_topics(lda_model, no_top_words, n_topics)
```

```
---------------------------------------------
Topic 0:
java public int class new void string return static method
---------------------------------------------
---------------------------------------------
Topic 1:
file git use branch commit instal run error work line
---------------------------------------------
---------------------------------------------
Topic 2:
use function like would string valu way object code one
---------------------------------------------
---------------------------------------------
Topic 3:
android view div layout imag button text height width content
---------------------------------------------
---------------------------------------------
Topic 4:
js node request server modul user use error get data
---------------------------------------------
```

- Perplexity: -7.499418620423826
- Coherence Score: 0.520627982382461 6

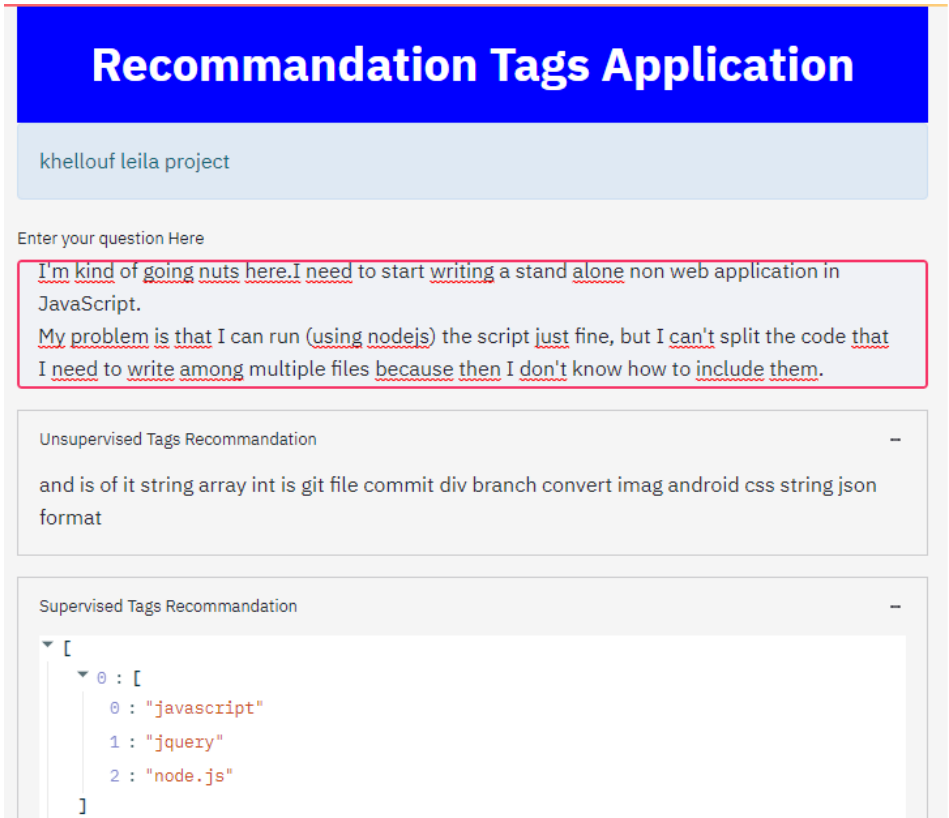
Approche non Supervisée



Le model est un bon topic car les bulles sont assez grosses et non superposées dispersées dans le graphique.

Api de recommandation de tags

- **API** propose une liste de tags Stack Overflow relatifs à une question saisie traitant de sujets informatiques.



- Le text saisi passe par toutes les étapes de préporocessing NLP puis est transformé en matrices TF-IDF avant application respectivement des modèles supervisée et non supervisée.

Conclusion

- **Data set**: entrainer les modèles sur une grande base de donnée.
- Utilisation de plusieurs modèles supervisée.
- **le prétraitement**: Manipulation des URL, emoji par example,..etc
- **Apprentissage des features**: Utiliser les dernières techniques issues du deep learning comme BERT ou ELMo.
- Trouver de nouvelle **métrique d'évaluation**

The End