



# Multi-label Text Classification

Approches supervisées et non supervisées pour la suggestion automatique de tags



---

# Problématique

Stack Overflow est une plateforme essentielle pour les développeurs, mais la gestion manuelle des tags est souvent imprécise, surtout pour les débutants. Le projet vise à développer un système automatique de suggestion de tags pour améliorer l'organisation et l'expérience utilisateur.



# Extraction des Données

01



## 1. Extraction des données :

- Utilisation d'une requêtes SQL sur StackExchange Data Explorer pour extraire 50.000 questions.

```
SELECT TOP 50000 Title, Body, Tags, Id, Score, ViewCount, AnswerCount  
FROM Posts  
WHERE PostTypeId = 1 AND ViewCount IS NOT NULL  
AND AnswerCount > 0 AND LEN(Tags) - LEN(REPLACE(Tags, '<', '')) >= 5
```

- Et aussi test the l'API Stack Exchange API pour la récupération des données (50 questions).



# Préparation des Données

02


1. Nettoyage des tags
2. Nettoyage des corps et titres des questions

## 1. Nettoyage des tags :

Suppression des caractères non pertinents (spéciaux, chiffres, tirets inutiles) sauf certains mots-clés techniques (ex. : `c#`, `node.js`).

## 2. Nettoyage des corps et titres des questions :

Création d'une fonction plus complexe, pour supprimer les tags HTML, les caractères non pertinents, les contractions et "stop words" et pour faire la lemmatisation pour ramener les mots à leur forme racine (ex. : "running" → "run").



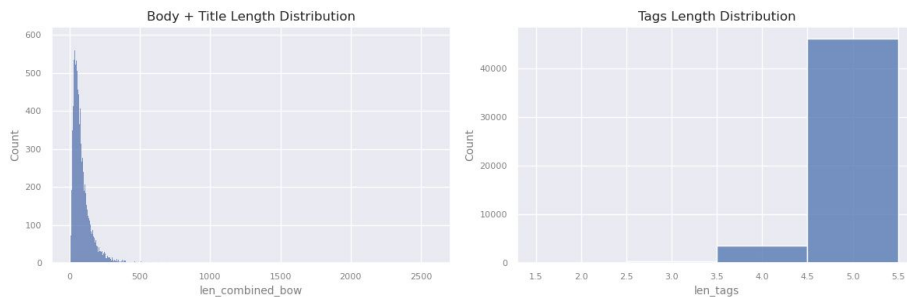
	Title	Body	Tags
0	How to convert Decimal to Double in C#?	<p>&lt;p&gt;I want to assign the decimal variable &amp;quot;trans&amp;quot; to the double variable &amp;quot;this.Opacity&amp;quot;;&lt;/p&gt;\n&lt;pre class="lang-cs prettyprint-override"&gt; &lt;code&gt;decimal trans = trackBar1.Value / 5000;\nthis.Opacity = trans;\n&lt;/code&gt; &lt;/pre&gt;\n&lt;p&gt;When I build the app it gives the following error:&lt;/p&gt;\n&lt;blockquote&gt;\n&lt;p&gt;Cannot implicitly convert type decimal to double&lt;/p&gt;\n&lt;/blockquote&gt;\n</p>	[c#, floating-point, type-conversion, double, decimal]
	title_clean_bow	body_clean_bow	Tags
0	[convert, decimal, double, c#]	[want, assign, decimal, variable, trans, double, variable, opacity, decimal, trans, value, opacity, trans, build, app, give, follow, error, implicitly, convert, type, decimal, double]	[c#, floating-point, type-conversion, double, decimal]



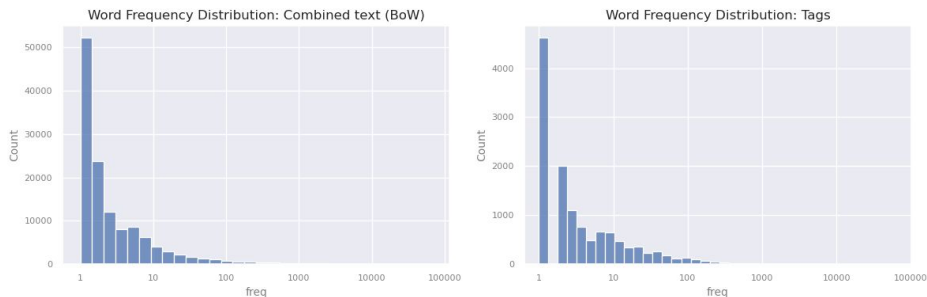
# Analyse Exploratoire

## 03

1. Distributions nombre de mots par document
2. Fréquence des mots
3. Nuage des mots par fréquence



## 2. Fréquence des mots par document



### 3. Nuage des mots par fréquence







# Vectorization

## 04

1. MultiLabelBinarizer
2. CountVectorizer
3. TFIDF
4. Word2vec, BERT, USE

## 1. MultiLabelBinarizer pour la vectorization des Tags - Top 100 tags les plus fréquentes

[illegible]



## 2. CountVectorizer

```
doc=["One Cent, Two Cents, Old Cent,  
New Cent: All About Money"]
```

	about	all	cent	cents	money	new	old	one	two
doc	1	1	3	1	1	1	1	1	1



Sentence A : The car is driven on the road.

Sentence B: The truck is driven on the highway.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2) = 0$	0	0
Car	1/7	0	$\log(2/1) = 0.3$	0.043	0
Truck	0	1/7	$\log(2/1) = 0.3$	0	0.043
Is	1/7	1/7	$\log(2/2) = 0$	0	0
Driven	1/7	1/7	$\log(2/2) = 0$	0	0
On	1/7	1/7	$\log(2/2) = 0$	0	0
The	1/7	1/7	$\log(2/2) = 0$	0	0
Road	1/7	0	$\log(2/1) = 0.3$	0.043	0
Highway	0	1/7	$\log(2/1) = 0.3$	0	0.043

$$TF = \frac{\text{Number of times a word "X" appears in a Document}}{\text{Number of words present in a Document}}$$

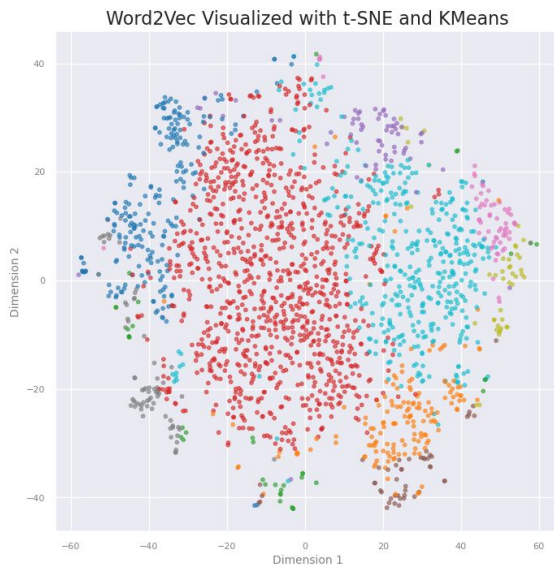
$$IDF = \log \left( \frac{\text{Number of Documents present in a Corpus}}{\text{Number of Documents where word "X" has appeared}} \right)$$

$$TF\ IDF = TF * IDF$$

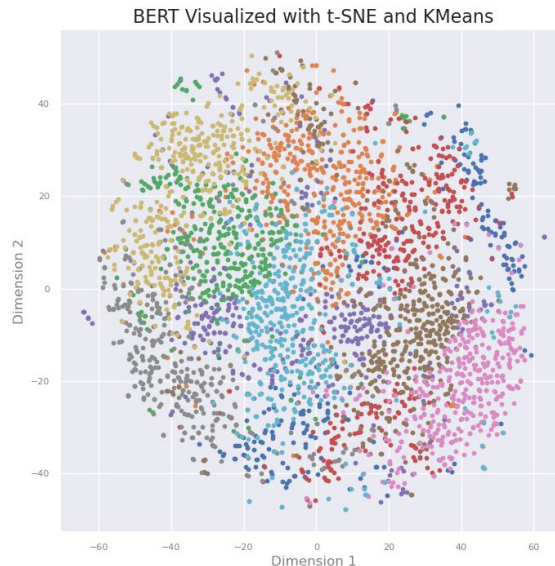


#### 4. Test de techniques avancées de vectorization - 5000 lignes

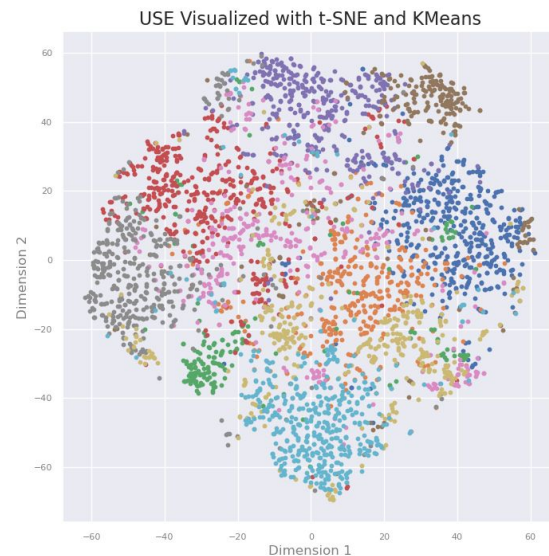
**Word2Vec** : Crée des vecteurs basés sur le contexte local des mots (voisinage) dans une phrase.



**BERT** : Génère des vecteurs **contextuels** en tenant compte de toute la phrase grâce à un modèle bidirectionnel.



**USE** : produit des vecteurs pour des phrases ou des textes entiers, optimisés pour mesurer les similarités **sémantiques** et le sens global des textes.



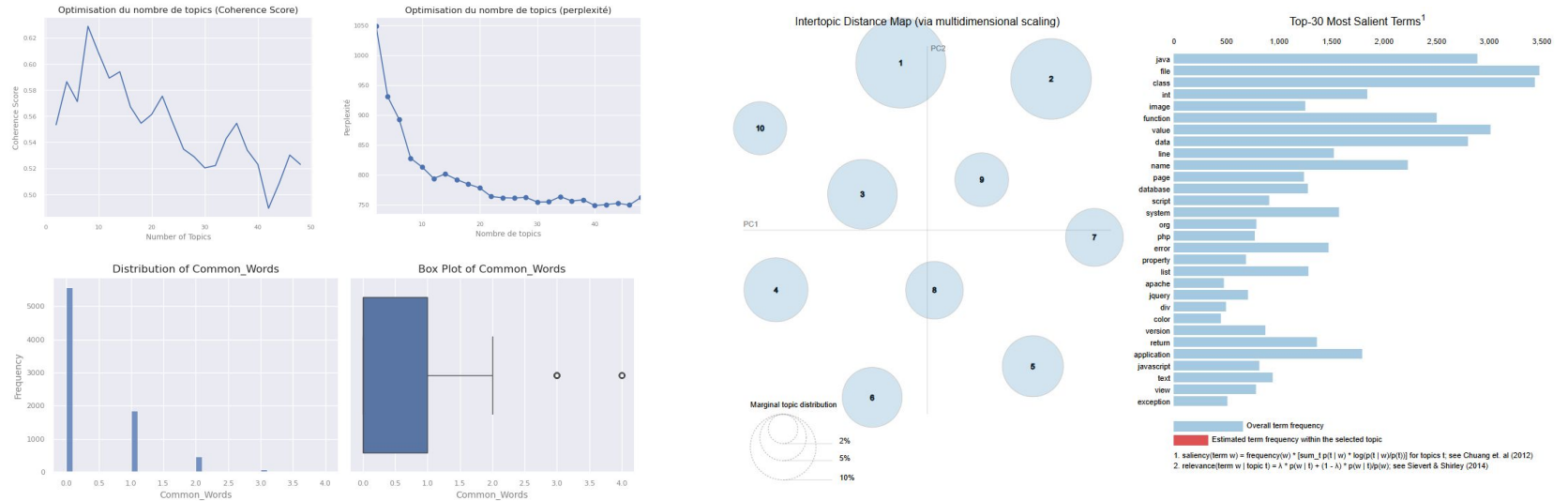


# Modélisation non supervisée

05

1. LDA

1. **LDA (Latent Dirichlet Allocation)** est une méthode utilisée pour analyser un ensemble de textes et découvrir les thèmes principaux (appelés "topics")





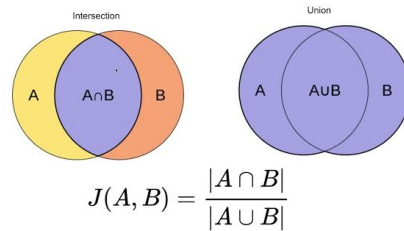
# Modélisation supervisée (TF IDF)

## 06

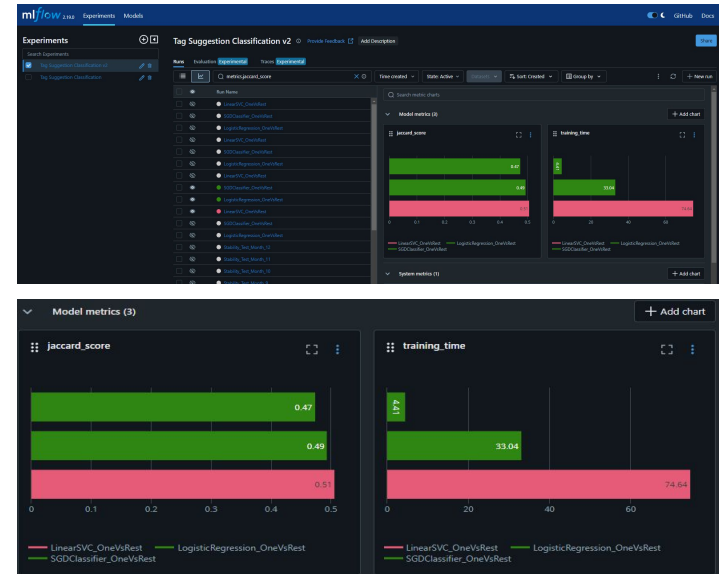
1. OneVsRestClassifier (Logistic Regression, SGDClassifier / LinearSVC)
2. Jaccard Score
3. MLFlow



1. L'approche **OneVsRestClassifier** a été utilisée en combinaison avec les algorithmes **Logistic Regression**, **SGDClassifier**, et **LinearSVC**.
2. **Jaccard Score** a été utilisée comme principal indicateur de performance des modèles.



3. Suivre de l'expérimentation avec **MLFlow**





# Modélisation supervisée

(Embeddings)

07

1. Results avec Word2Vec, BERT, USE



## 1. Results avec Word2Vec, BERT, USE

BERT-----

SGDClassifier() Jaccard Score: 0.20926223651304038

LogisticRegression(max\_iter=200, solver='saga') Jaccard Score: 0.24870617057112235

LinearSVC(C=1.5, dual=False, max\_iter=2000, penalty='l1') Jaccard Score: 0.2150151414620868

WORD2VEC-----

SGDClassifier() Jaccard Score: 0.1995088304573835

LogisticRegression(max\_iter=200, solver='saga') Jaccard Score: 0.14377456234369415

LinearSVC(C=1.5, dual=False, max\_iter=2000, penalty='l1') Jaccard Score: 0.15605573419078242

USE-----

SGDClassifier() Jaccard Score: 0.349746082784668

LogisticRegression(max\_iter=200, solver='saga') Jaccard Score: 0.38597713469096107


LinearSVC(C=1.5, dual=False, max\_iter=2000, penalty='l1') Jaccard Score: 0.32787092328892975

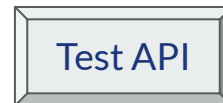


# API de prédiction

## 08

1. Les données et le modèle
2. GitHub pages
3. Architecture AWS
4. Test API

- 
1. Utilisation du modèle supervisée avec **LinearSVC** entraîné avec **100 tags** et **25000** lignes de données
  2. Front-end sur **Github** pages
  3. L'API a été déployée sur le cloud **AWS**
    - **Amazon EC2** : Déploiement des composants backend sur une instance EC2. Une image **Docker** du projet a été créée localement, puis poussée sur l'instance EC2.
    - **API Gateway** : gérer les requêtes POST et les rediriger vers le backend.
    - **AWS Lambda** : pour traiter les requêtes API et exécuter la logique de prédiction des tags.
    - **CloudWatch** : pour les journaux, le débogage et la surveillance des performances d'API Gateway et de la fonction Lambda.





**Merci!**