CATEGORISER AUTOMATIQUEMENT DES QUESTIONS

Openclassrooms **Projet n°5** Mai 2023

Parcours DataScientist Ingénieur Machine Learning

Problématique

Stack Overflow est un site qui permet à la communauté de développeurs d'échanger sur des sujets liés à l'informatique.

Un utilisateur pose une question en renseignant un formulaire contenant 3 champs :

- **Title**
- **Body**
- **Tags** (target)

La problématique : On souhaite proposer automatiquement une liste de tags après saisie du titre et de la question par l'utilisateur.

Modélisation en problème de machine learning

Le corpus est obtenu en **concaténant le titre et la question** de l'utilisateur (title et body)

Approche supervisée

- On a N classes possibles (tags).
- Chaque observation appartient à 1 à 5 classes.
- Chaque classe est affectée à 1 à K observations.

On a un problème de classification multilabels

Approche non-supervisée

Déterminons des "Topics" qui seront dans notre cas un ensemble de tags. Puis affectons un topic aux questions.

La target

Combien de tags ?
Combien de tags unique?
Doit-on conserver tous les tags?

Procédure

- **Acquisition des données**

Requête SQL: *https://data.stackexchange.com/stackoverflow/query/new*

- **Préparation des données**

Cleaning, tokenisation, pré-traitement
Vectorisation des Tags

- **Feature engineering**

Tf-idf, Word2vec, Bert et Use

- **Classification supervisée**

Choix du modèle le plus adapté parmi 4 à l'aide de différentes métriques
Optimisation du modèle retenu
Vérification tag true, tag false et tag empty

- **Classification non supervisée**

Vérification de la performance du modèle (cohérence)
Vérification tag true, tag false et tage empty

- **Choix entre les 2 types de classification**

- **Création d'un point API (à destination de l'utilisateur)**

Métriques d'évaluation

- **Préambule:**
  Un problème de classification binaire est assez simple à évaluer : on a juste vrai ou faux
  Un problème de classification multilabels est un peu plus compliqué:
  On a prédit tous les tags, on n'a prédit qu'une partie des tags, on a prédit une partie vrai et une partie fausse, on a tout faux
  Nous retiendrons les évaluations suivantes: **Tag true** (au moins un tag prédit est juste), **tag false** (tous les tags prédits sont incorrects), **tag empty**(aucun tag n'est prédit)

- **Métriques d'évaluation en classification supervisée:**

  - Micro precision: (en micro car multi-labels) combien de tags prédits correctement parmi les prédictions
  Dans le cadre d'une API de suggestion de tags, la capacité d'un modèle à retourner tous les tags n'est pas essentielle
  - Micro f1: moyenne harmonique entre recall et precision (micro)
  - Hamming loss: fraction des tags prédits incorrectement (doit tendre vers zéro)
  - Tag true, tag false, tag et time_score

- **Métriques d'évaluation en classification non supervisée:**

  - Estmation du nombre de topics le plus adapté
  - **Cohérence du modèle**,
  - **Tag true, tag false, tag empty**

Prétraitement des données

- **Concaténisation** des ‘title’ et des ‘body’

- **Cleaning**
  Suppression des balises html
  Texte en minuscules
  Suppression des ponctuations

- **Tokenisation**
  Chaque observation est transformée en tableau de mots

- **Pré-traitement**
  Stop words: suppression des mots qui n’apportent pas de sens supplémentaires
  POS tagging: associer aux mots d'un texte les informations grammaticales correspondantes comme la partie du discours, le genre, le nombre
  Lemmatisation (lemme des mots)

- **Feature engineering**
  Bag of words (bow): fréquence des tokens
  TF-IDF: fréquence d’apparition du token dans le document et la fréquence inverse du nombre de documents où le token apparaît.
  Word2vec, Bert et Use: prise en compte du contexte linguistique

Présentation des données initiales

	Title	Body	Tags
0	How to convert Decimal to Double in C#?	<p>I want to assign the decimal variable &quot...	<c#><floating-point><type-conversion><double><...
1	Why did the width collapse in the percentage w...	<p>I have an absolutely positioned <code>div</...	<html><css><internet-explorer-7>
2	How do I calculate someone's age based on a Da...	<p>Given a <code>DateTime</code> representing ...	<c#><.net><datetime>

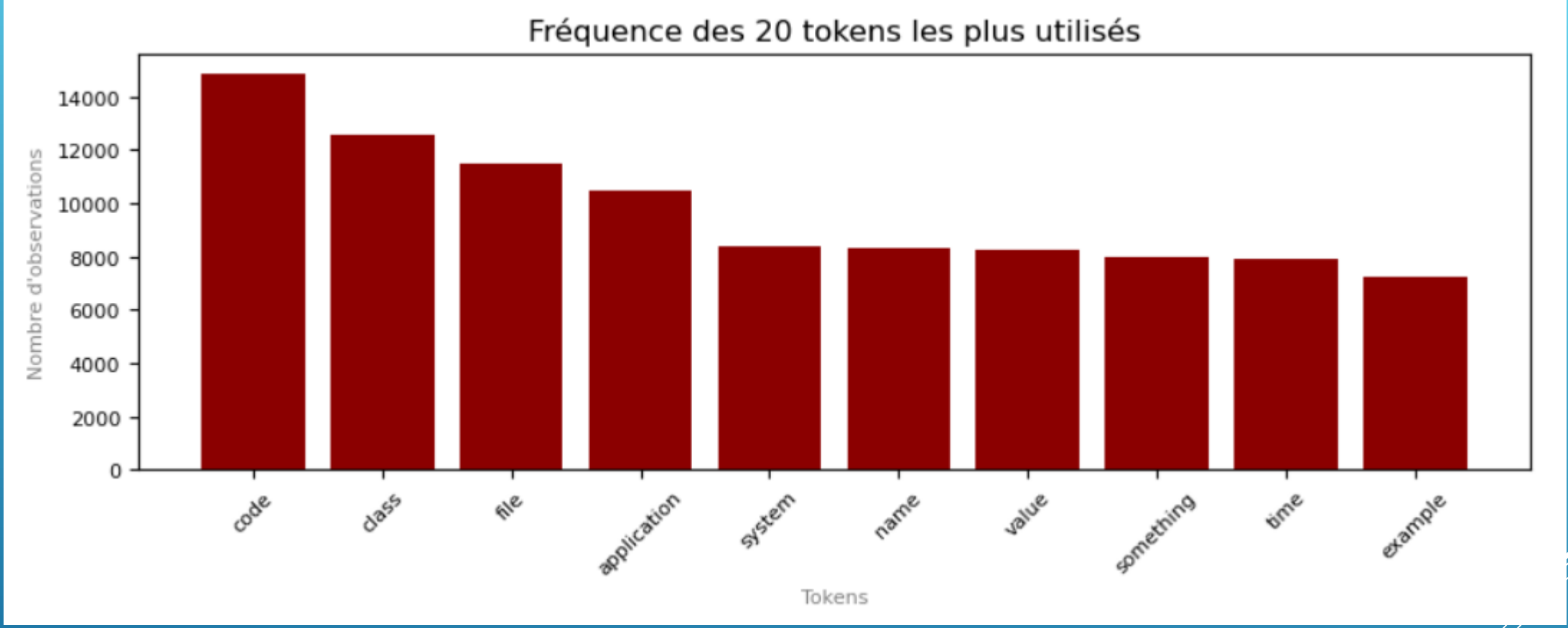
Nombre d'occurrences	46 502

Prétraitement des données (le corpus)

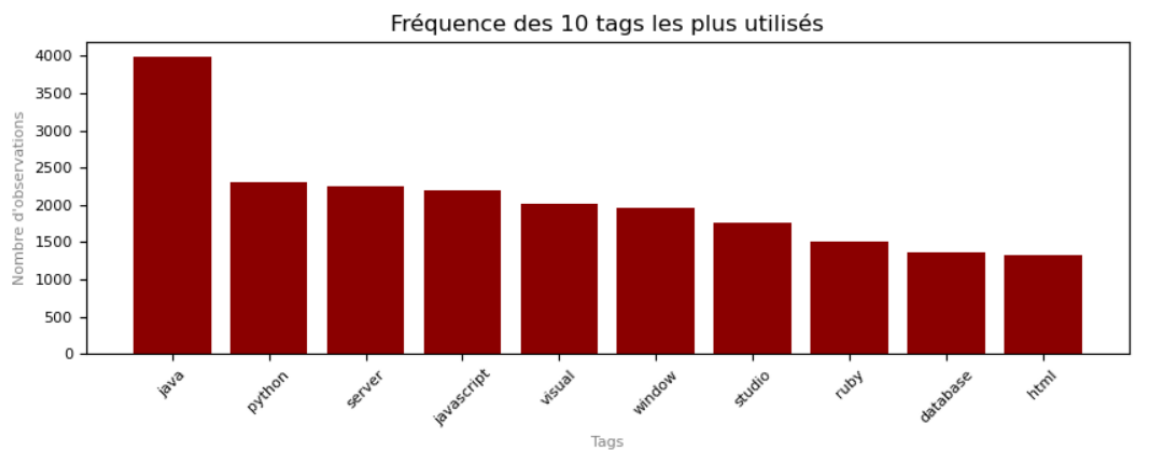
Traitement	Description
Fusion des colonnes 'title' et 'body'	• On obtient une seule colonne contenant toutes les données utilisateurs
Suppression des balises HTML	• Suppression des balises
Nettoyage du texte	• Tout en minuscules • Filtrage des caractères non alphabétiques • Filtrage des termes de moins de 3 caractères
Tokenisation	• Découpage en tokens • Suppression des stopwords
Filtrage à l'aide d'un modèle de POS (Part Of Speech)	• Filtrage des noms communs
Racinisation des tokens	• Lemmatisation
Filtrage des valeurs vides	• Suppression des valeurs vides générées par les traitements précédents
Vectorisation du corpus	• Tf-idf, Word2Vec, Bert et Use

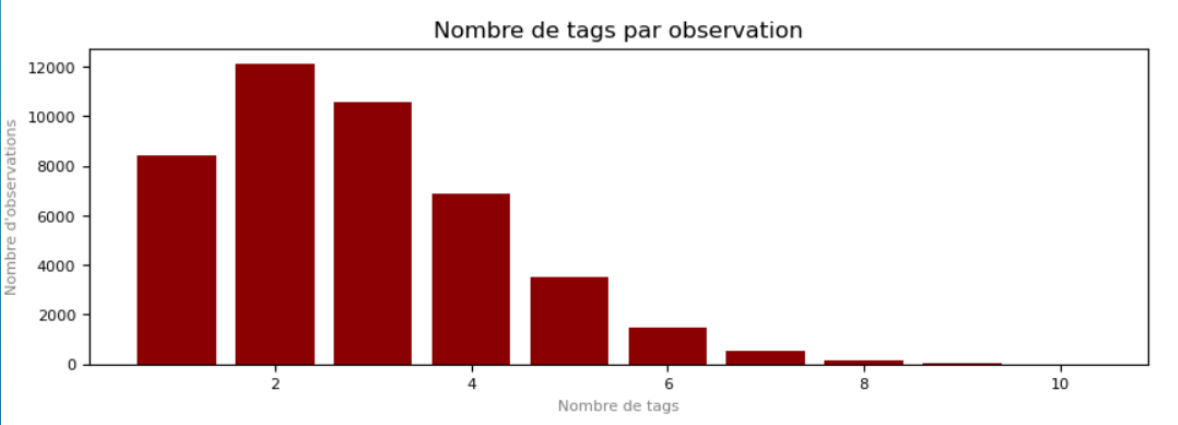
Impact sur le nombre de données	
Avant	**Après**
46 502	**43 798**

Fréquences de distribution des tokens



Fréquences de distribution des tags





Nombre de tags	Nombre de tags uniques
123 917	5742

Pour les analyses futures, nous décidons de ne retenir que les top 200 tags (les plus représentés)
Nous filtrerons les documents qui ne sont pas liés aux top 200 tags

Feature engineering

Word embedding	
Tf-idf	Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le document. Il varie également en fonction de la **fréquence** du mot dans le corpus
Word2vec, Bert et Use	Prise en compte du **contexte** linguistique

Approche supervisée

Modèles utilisés	Description
DummyClassifier()	Ce classificateur est utile comme base de référence simple pour comparer avec d'autres classificateurs
KNeighborsClassifier()	Classificateur mettant en œuvre le vote des k-plus proches voisins
OneVsRestClassifier(LinearSVC)	Classification linéaire par vecteurs de support
RandomForestClassifier()	Assemblage d'arbres de décision

Infos:
*Le design du **KNN** et du **Random Forest** supportent nativement les problèmes de classification multiclasses. Pour la SVM nous avons eu recours à algorithme One vs Rest (son principe est d'entrainer un modèle par classe afin de déterminer si l'observation appartient à la classe sur laquelle il a été entrainé ou non)*

Traitements spécifiques à l'approche supervisée

Traitement	Détails
Dédoublonnage des tags	
Réduction de dimensions des features (PCA)	On conserve 85% de la variance expliquée
Vectorisation des tags	Transformation en valeurs numériques

Impact sur le nombre de features	
Avant	**Après**
165	**119**

Résultats

	Dummy_tfidf	Dummy_w2v	Dummy_bert	Dummy_use	KNN_tfidf	KNN_w2v	KNN_bert	KNN_use	SVM_tfidf	SVM_w2v	SVM_bert	SVM_use	RF_tfidf	RF_w2v	RF_bert	RF_use
time_score	0.40	0.50	0.41	1.46	25.150	25.750	29.100	31.500	25.250	197.590	396.220	55.680	201.910	135.580	587.020	324.510
micro_precision	0.00	0.00	0.00	0.00	0.641	0.620	0.719	0.696	0.776	0.767	0.597	0.789	0.768	0.844	0.911	0.891
micro_f1	0.00	0.00	0.00	0.00	0.226	0.290	0.289	0.419	0.237	0.255	0.365	0.436	0.183	0.140	0.044	0.187
hamLoss	0.01	0.01	0.01	0.01	0.009	0.009	0.009	0.008	0.009	0.009	0.009	0.008	0.009	0.009	0.010	0.009
tag_true	0.00	0.00	0.00	0.00	20.300	26.440	26.170	41.290	22.260	22.450	31.810	42.350	16.560	11.810	3.630	17.490
tag_false	0.00	0.00	0.00	0.00	12.930	19.000	12.250	21.390	7.200	7.360	27.140	14.220	4.890	2.290	0.400	2.430
tag_empty	100.00	100.00	100.00	100.00	66.770	54.560	61.590	37.320	70.540	70.200	41.050	43.420	78.550	85.900	95.980	80.080

Nous choisissons le modèle SMV-classifier avec la tvectorisation USE, nous avons recherché par la suite une optimisation de ce modèle

Pour information, nous avons également testé le modèle GradientBoostingClassifier() qui s'est rélévé très peu performant et très long à entraîner (23 heures)

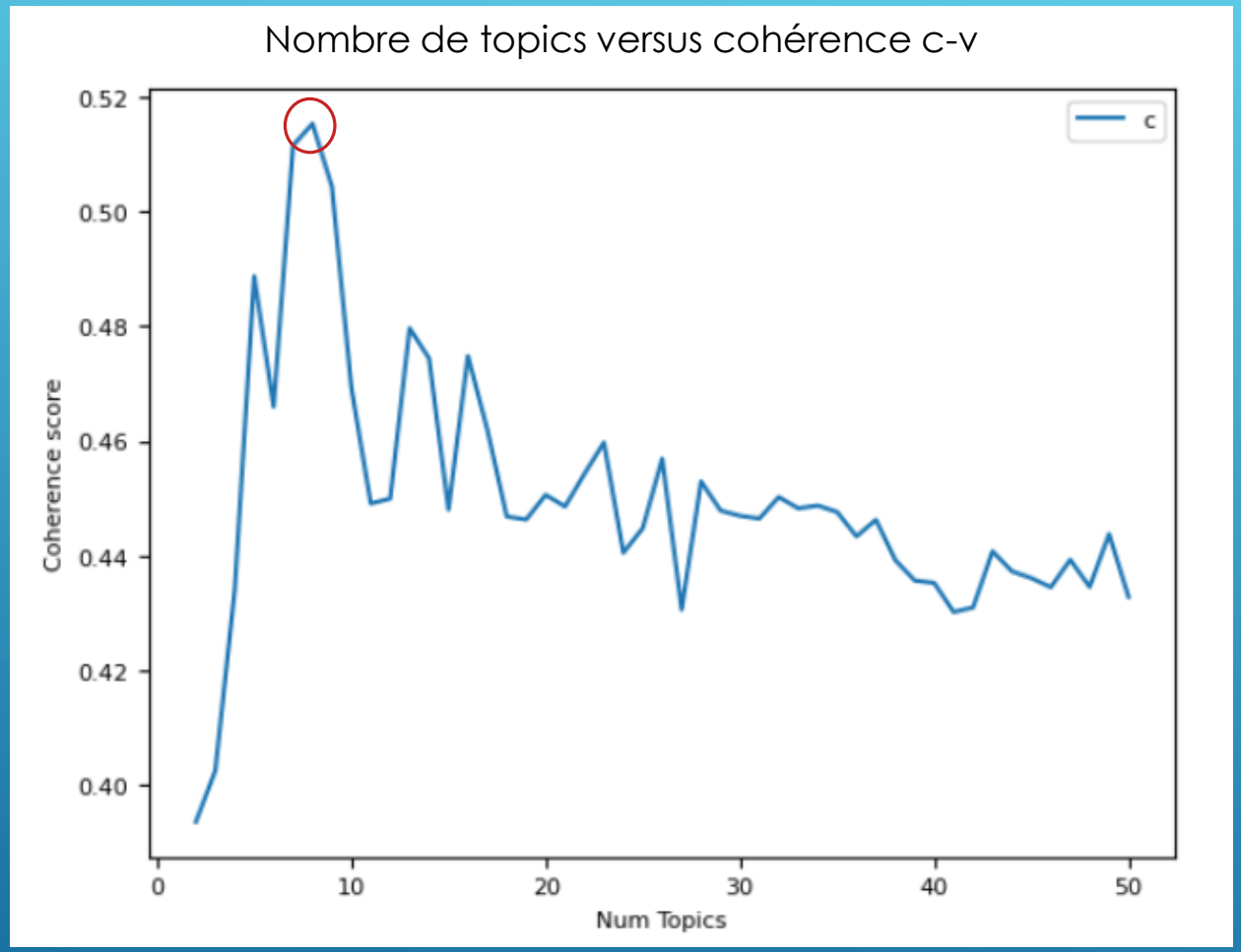
Approche non supervisée

Choix du modèle

Dans le domaine du traitement automatique des langues, l'**allocation de Dirichlet latente** (de l'anglais *Latent Dirichlet Allocation*) ou ***LDA*** est un modèle génératif probabiliste permettant d'expliquer des ensembles d'observations, par le moyen de groupes non observés, eux-mêmes définis par des similarités de données.
Source: *https://fr.wikipedia.org/wiki/Allocation_de_Dirichlet_latente*

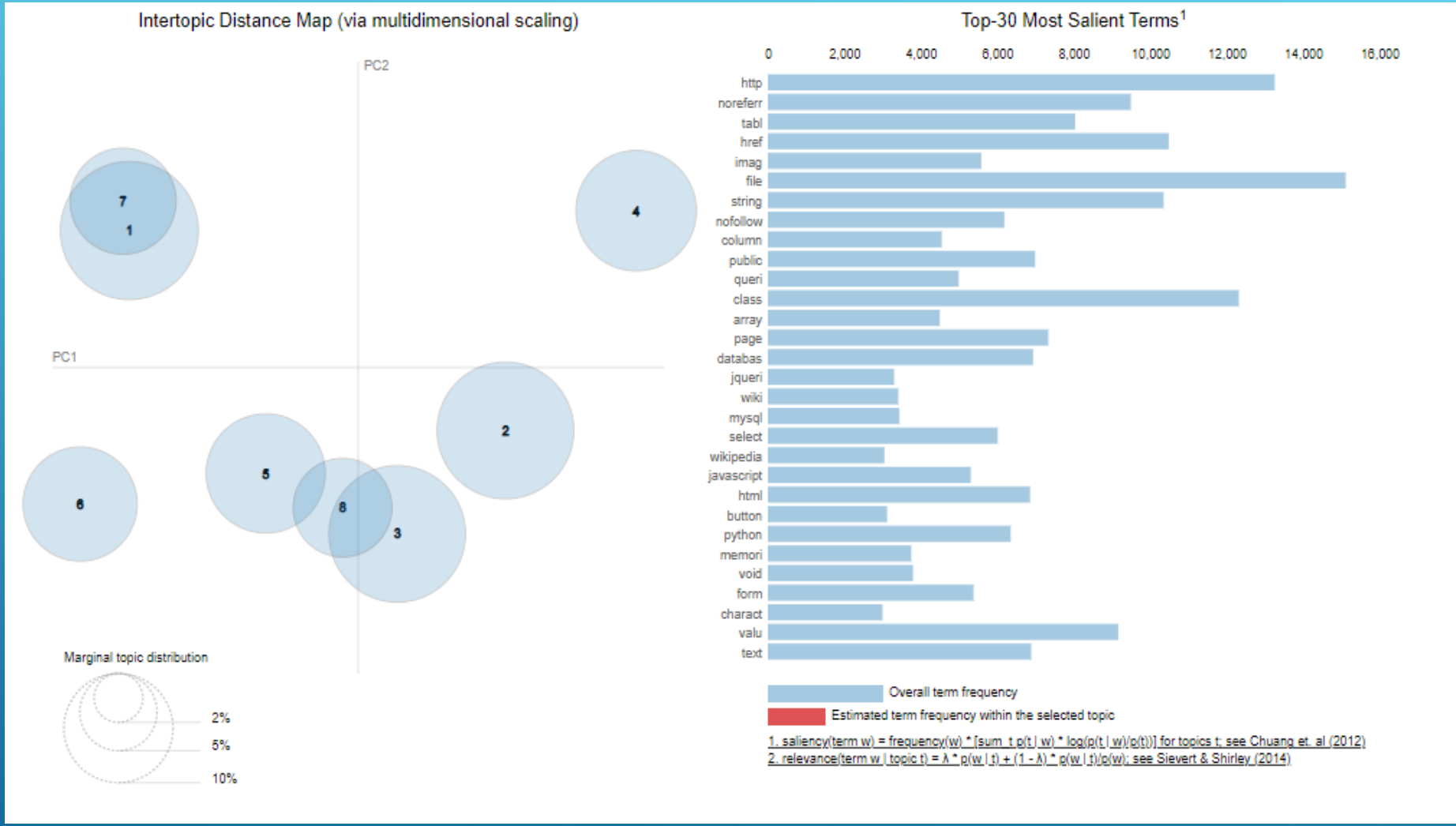
Les **mesures de cohérence** évaluent le **degré de similitude sémantique** entre les mots les mieux notés dans le topics. Ces mesures aident à faire la distinction entre les **topics sémantiquement interprétables** et les **topics dûs à des inférences statistiques.**
Pour un bon modèle LDA la **cohérence** doit être comprise entre **0.4 et 0.7** au-delà et en dessous le modèle est très probablement erroné
Il est important d'estimer préalablement le **nombre de topics optimal**

Recherche du nombre de topics optimal



Nous retenons 8 topics (cohérence max)

Distribution spatiale des topics



Score de cohérence en fonction du word embedding

Coherence score	
BagOfWord	0,498
Tf-idf	0,535

Nous retenons la vectorisation Tf-idf

Résultats

Résultas	
Coherence score	0,524
Tag true	8,75%
Tag false	76,75%
Tag empty	14,51%

Comparatif entre approche supervisée et non supervisée

Comparatif des résultats en classification supervisée et non supervisée

items	Modèle supervisé	Modèle non supervisé
Time score	56	291
Tag true	44,35%	8,75%
Tag false	14,22%	76,75%
Tag empty	43,42%	14,51%

Les seuls indicateurs de performance communs aux modèles supervisés et non supervisés sont:

- **time_score**
- **tag_true**: parmi les tags choisis par l'utilisateur, au moins 1 tag prédit est juste
- **tag_false**: tous les tags prédits sont différents des tags utilisateurs
- **tag_empty**: aucun tag n'est prédit

Nous constatons:

- Une différence nette dans le temps de traitement (time_score), avantage au modèle supervisé (56 secondes vs 291)

- Des performances moyennes dans les prévisions justes (tag_true) avec un avantage pour le modèle supervisé (44% vs 9)

- Les erreurs de prédiction (somme des 'tag_false' et 'tag_empty') sont plus nombreuses avec le modèle non supervisé (91% vs 88)

- Le modèle supervisé prédit peu de tags faux (tag_false) mais beaucoup de tags vides (tag_empty) alors que le modèle non supervisé fait le contraire

- Au-delà des prédictions, il s'avère que l'approche non supervisée est soumise à moins de contraintes que l'approche supervisée

En effet le modèle supervisé nécessite un travail plus important sur le dimensionnement des données et leur pré-traitement avant entrainement
De plus l'approche supervisée nécessite de maintenir plus de ressources (modèle de PCA, modèle de vectorisation des tags, SVM) que l'approche non supervisée (modèle LDA et vocabulary)

Pour toutes ces raisons et avec l'objectif de maximiser les tags prédits 'justes', nous décidons une utilisation complémentaire de ces modèles pour le codage de l'API

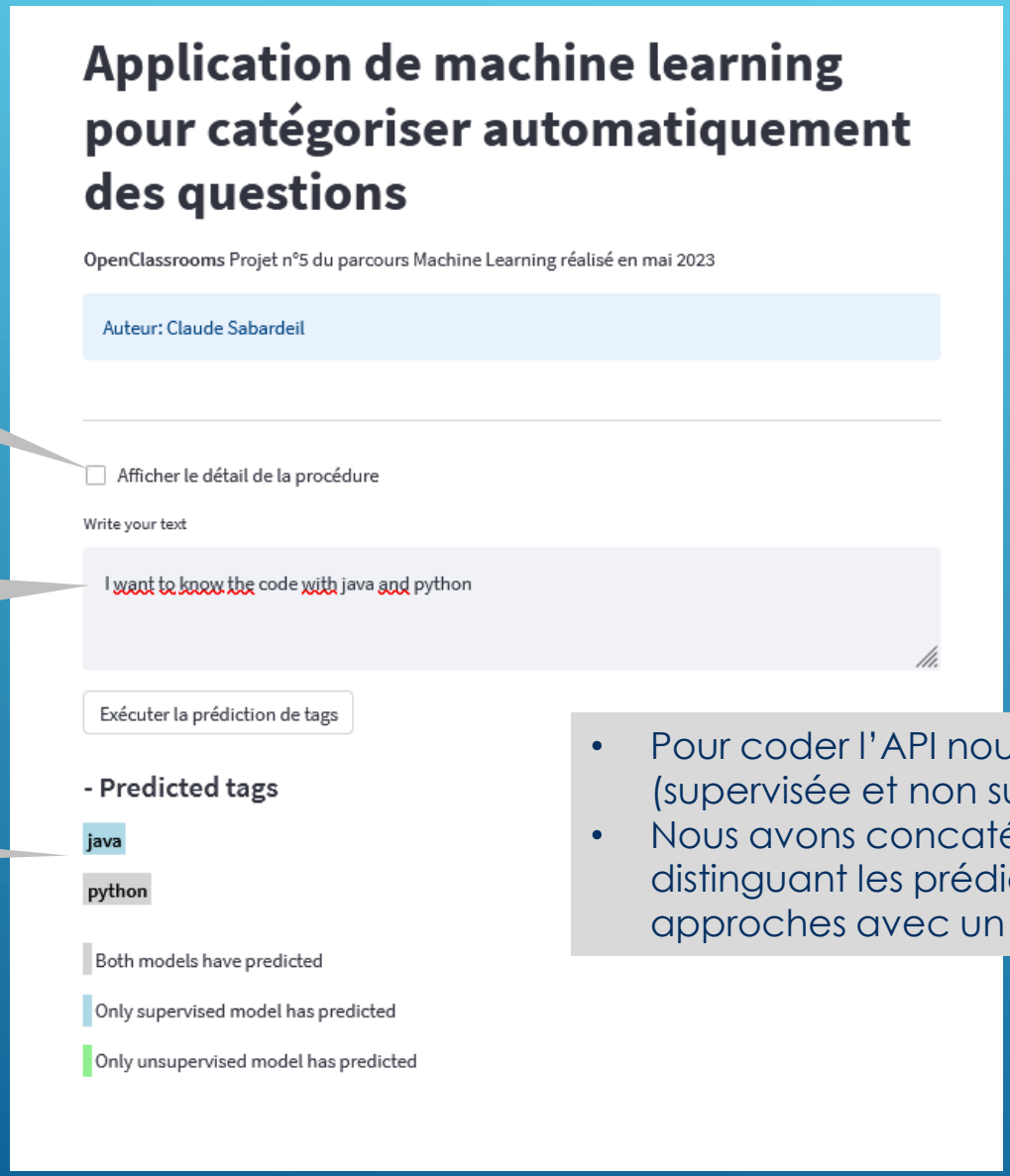
API

Pour créer un point API nous avons utilisé la librairie 'Streamlit' de python (code avec VScode) https://ycs127-opc-api2-app-p5-18s8dz.streamlit.app/

L'utilisateur peut ou pas afficher le détail des modèles utilisés

L'utilisateur saisit la question et clique sur 'exécuter la prédiction de tags'

Les prédictions



- Pour coder l'API nous avons utilisé les 2 approches (supervisée et non supervisée)
- Nous avons concaténer les prédictions des 2 modèles en distinguant les prédictions de l'une ou l'autre des approches avec un code couleur

Conclusion

- Nous avons dans cette étude essayé de prédire des tags à partir d'un corpus de questions stackoverflows avec leur titre en utilisant 2 approches l'une supervisiée et l'autre non supervisée

- Le résultat final est assez satisfaisant avec un taux de tags prédits justes d'environ 50% mais avec une absence de prédiction (tags vides) légèrement majoritaire.

- Avec plus de temps à disposition et plus de ressources (mémoire vive), plusieurs axes d'amélioration peuvent être envisagés:
  - Accorder plus de temps au filtrage des données
  - Pré-traiter les données avec des filtres plus adaptés au domaine d'activité

  (soit de tenter d'améliorer les filtres existants)

- Dans le codage de l'API, plutôt que de choisir l'une ou l'autre de ces approches, nous les avons utilisées de manière complémentaire afin de maximiser le taux de tags prédits justes.