PROJET 5 : CATÉGORISER AUTOMATIQUEMENT DES QUESTION

SOUTENANCE OPENCLASSROOMS, LE 06/07/2022

ERWAN CHESNEAU

PLAN :

I. Contexte
II. Prétraitement des données
III. Exploration
IV. Tests de segmentations non supervisées
V. Tests de segmentations supervisées
VI. Déploiement du modèle
VII. Conclusions
VIII. Améliorations à envisager

I. CONTEXTE:

- Le site StackOverFlow permet de poser des questions sur le thème large de la programmation
- Pour faciliter la recherche et la classification des questions, il est nécessaire d'utiliser des tags
  - Définir le bon tag peut être complexe pour les nouveaux utilisateurs
  - Proposer automatiquement une suggestion de tags permet d'améliorer l'utilisation du site
- L'outil StackExchange permet d'exporter des données du site
  - Possibilité de développer une segmentation sur des données réelles
- Démarches :
  - Effectuer un traitement du langage naturelle
  - Explorer les données
  - Effectuer des tests de segmentations non supervisées pour déterminer des clusters
  - Effectuer des tests de segmentations supervisées pour prédire les tags des questions existantes
  - Déployer le meilleur modèle

II. PRÉTRAITEMENT DES DONNÉES

- Les données sont exportées grâce à l'outil StackExchange avec la requête SQL suivante :
  - SELECT TOP 500000 Title, Body, Tags, Id, Score, ViewCount, FavoriteCount, AnswerCount
  - FROM Posts
  - WHERE PostTypeId = 1 AND ViewCount > 10 AND FavoriteCount > 10
  - AND Score > 5 AND AnswerCount > 0 AND LEN(Tags) - LEN(REPLACE(Tags, '<','')) >= 5

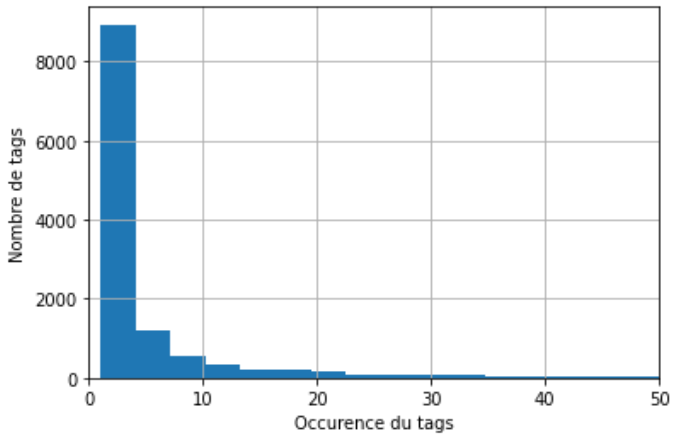
- Traitement : tokenisation
  - Création d'une liste de mot pour chaque document du corpus
  - Suppression des balises (et)
  - Suppression des chiffres : n'aide pas à la compréhension du thème global
  - Suppression des majuscules
  - Suppression des underscores

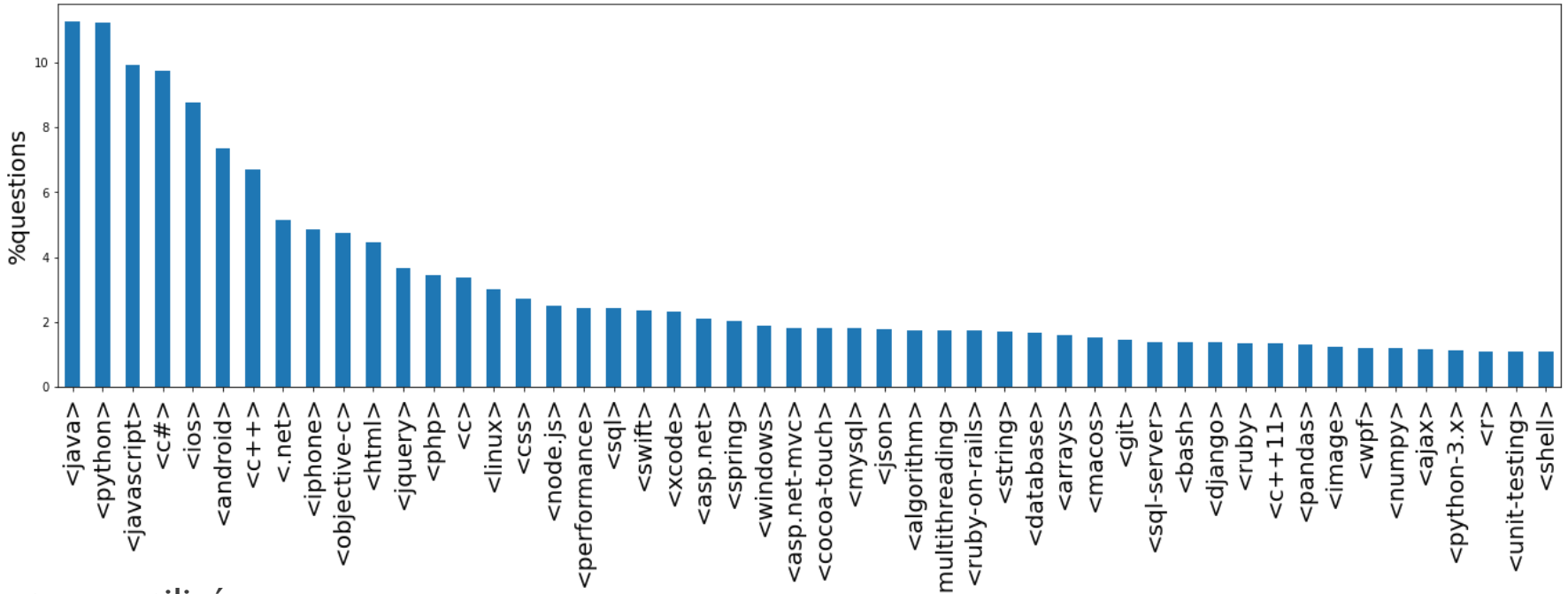
II. TRAITEMENT DES DONNÉES : RÉDUCTION DE DIMENSIONS

- Lemmatisation :
  - Retourner la forme canonique des mots : ne pas tenir compte des accords, de la conjugaison etc
  - Appliquée aux verbes, adjectifs et adverbe
  - Permet du supprimer 3,9% des mots de vocabulaire
- Bags of words :
  - Suppression du vocabulaire des mots trop peu présent et trop souvent présent
  - Suppression des mots présents dans 95% ou plus des documents et présent dans un seul document
  - Permet de supprimer environ 70% des mots
- Stopwords
  - Suppression des mots communs du vocabulaire
  - Comme « et », « ou » etc
  - La bibliothèque NLTK contient 179 stopwords en anglais

III. EXPLORATION DES DONNÉES

- Utilisation des tags :

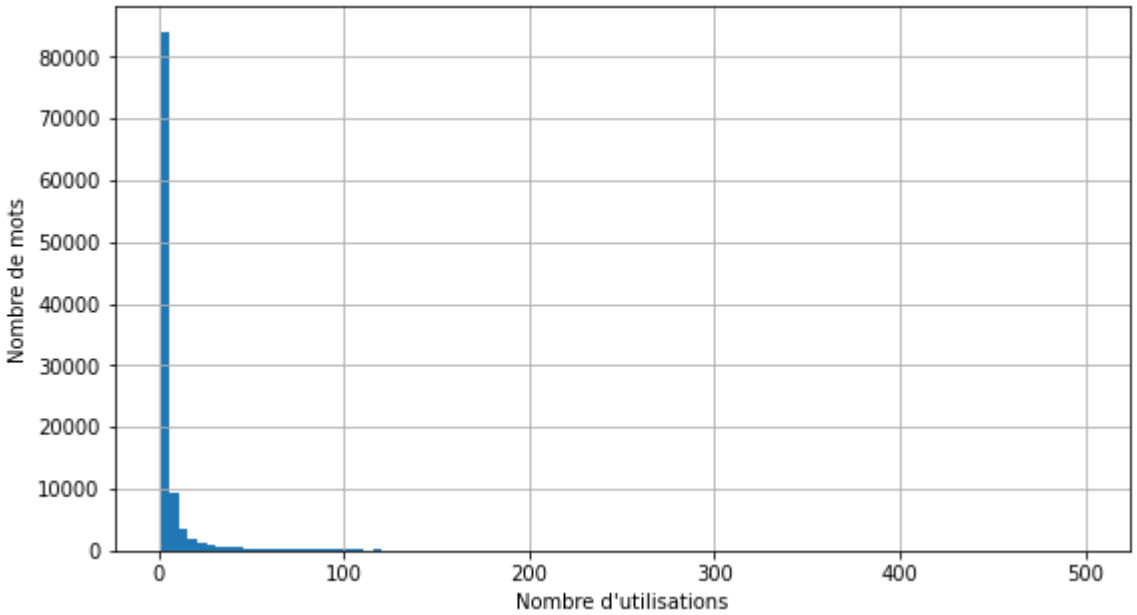


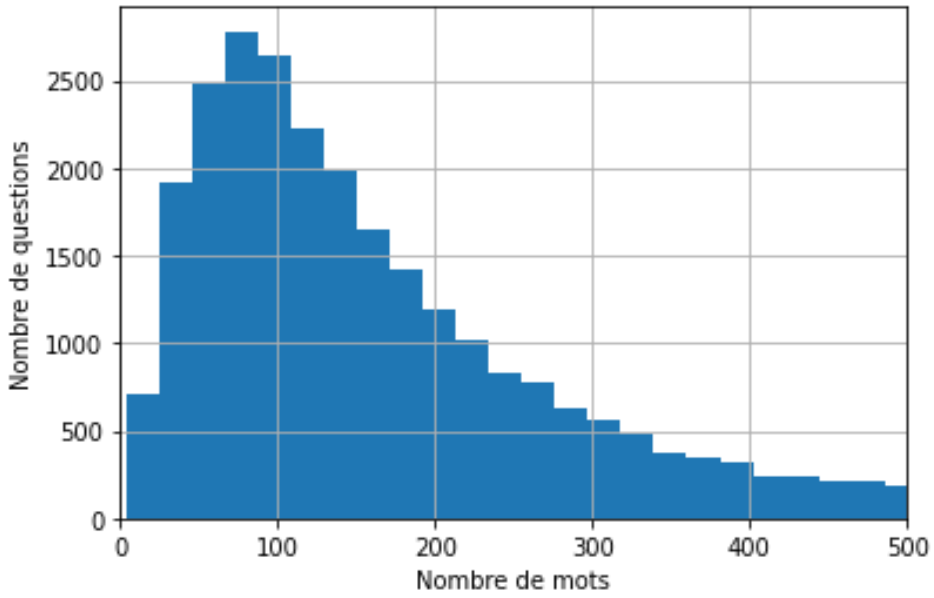


- Un grand nombre de tags sont peu utilisé
  - Permet de réduire le nombre de tags caractérisables
- Les tags les plus fréquents sont des tags génériques en rapport avec des thèmes larges
  - Permet de comprendre le profil des utilisateurs de StackOverFlow

III. EXPLORATION DES DONNÉES

- Utilisation des mots :

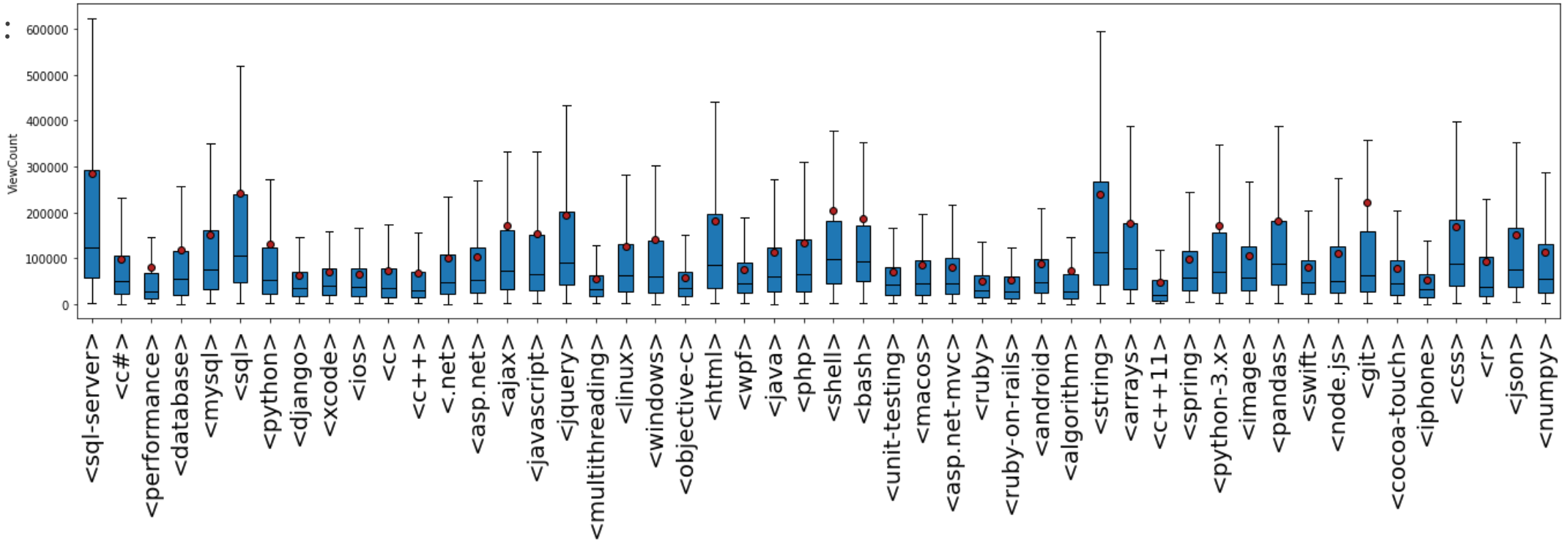




- Un nombre important de mots n'est que peu utilisé
- Une question comporte en environ 100 mots, mais certaines sont beaucoup plus longue
  - Traduit la complexité du problème

III. EXPLORATION DES DONNÉES

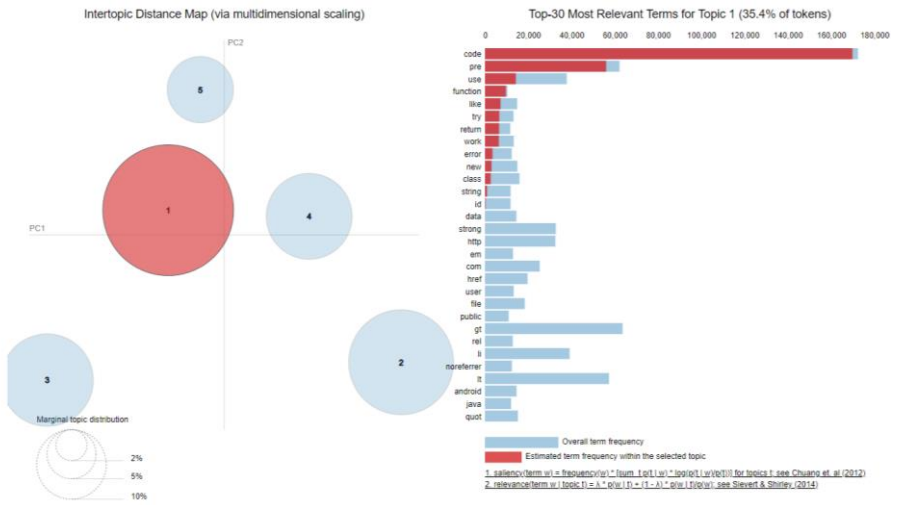
- ANOVA :

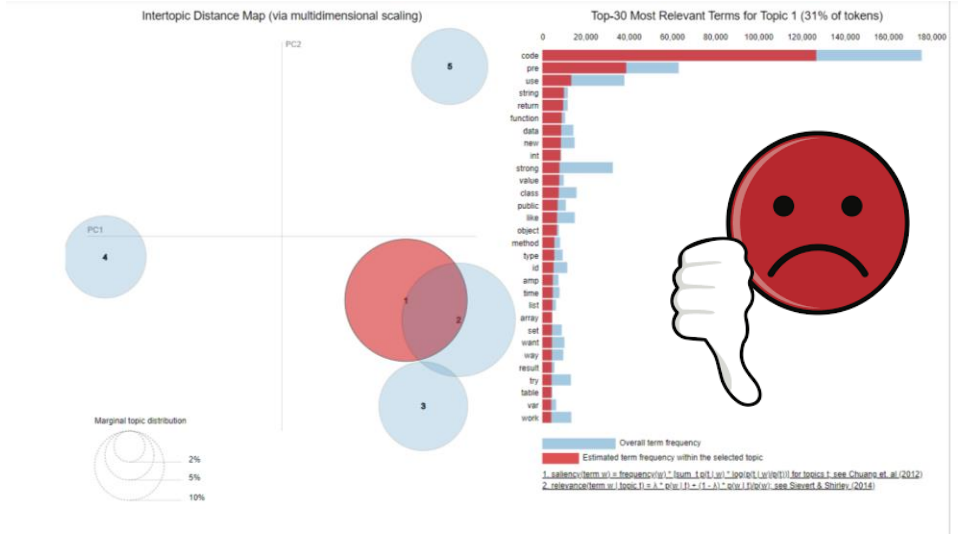


- Le nombre de vue par question varie en fonction des tags
  - Traduit les thèmes favoris

IV. SEGMENTATION NON SUPERVISÉE : LDA

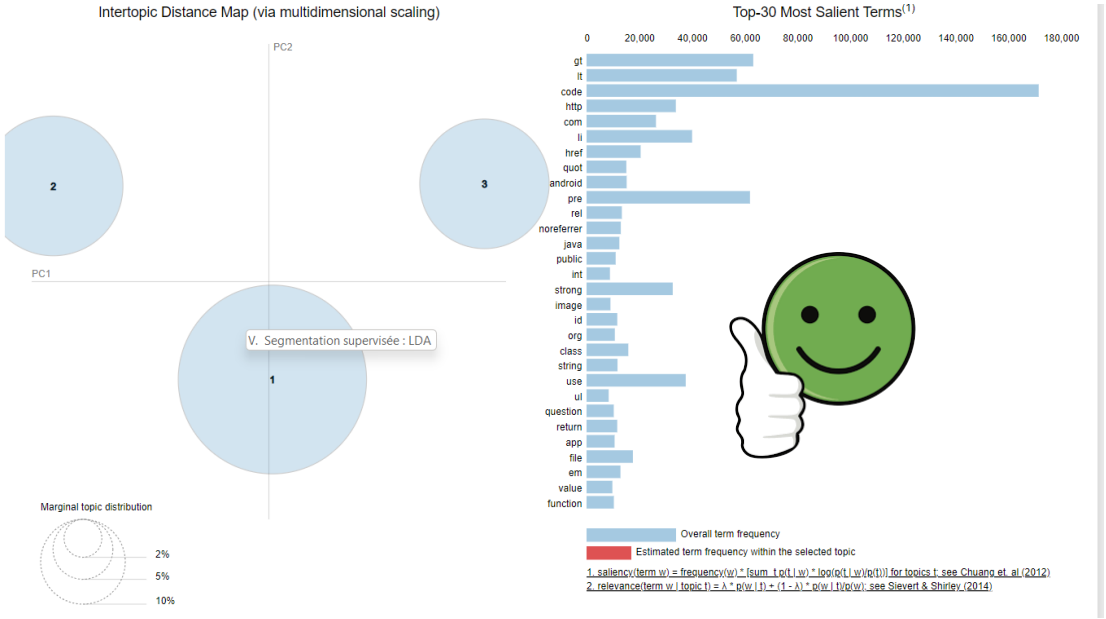
- Optimisation du max_df : 0.9
- Optimisation du min_df : entre 2 et 5
- Optimisation du max_features :
  - Pas de valeurs optimales
  - Choix d'une valeur pour éliminer au maximum les verbes et les adjectifs
  - Valeurs entre 30 et 50 (très faibles)

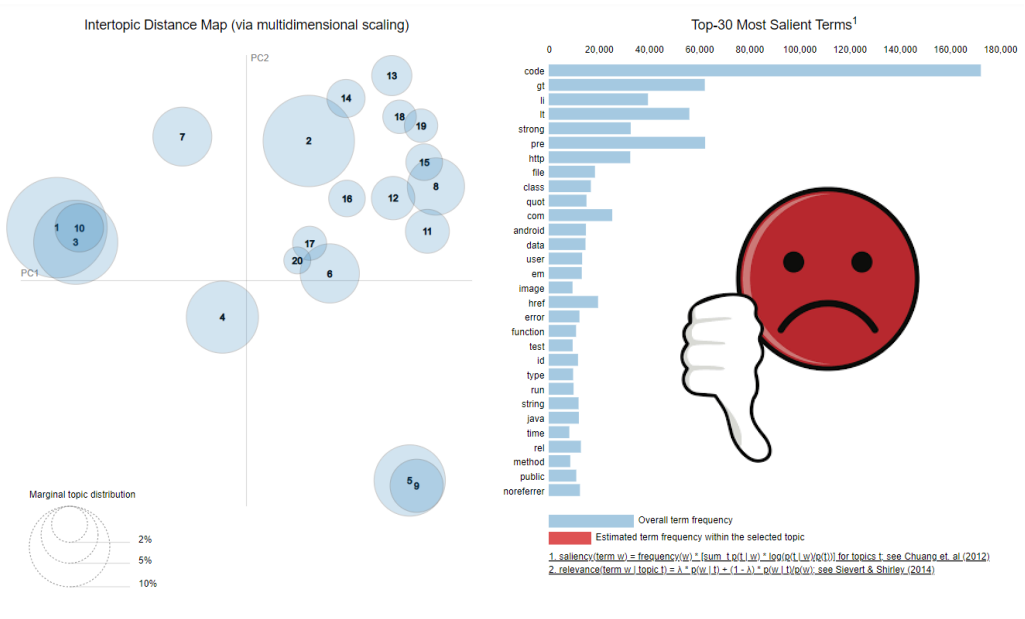




V. SEGMENTATION SUPERVISÉE : LDA

- Nombre de sujets détectables :
  - Nombre de sujets distincts qui sont interprétables facilement à partir des top_words
  - Un nombre de sujets trop important rend difficile l'interprétation





V. SEGMENTATION SUPERVISÉE : LDA

- Interprétations :
  - Topic 0: code pre use strong file data like user li try error work new function em want way return create need
  - Topic 1: http li com href android strong rel noreferrer java use org image ul question app em like application work run
  - Topic 2: gt lt quot pre code public id class int string return new type value data set function add error org

  - Topic 0 : base de données
  - Topic 1 : développement d'application
  - Topic 2 : questions de programmations générales, debugage

V. SEGMENTATION SUPERVISÉE : TF-IDF NMF

- Nombre de sujets détectables:
  - Topic 0: code pre use file like try want function work error string way new return class value data strong run method
  - Topic 1: li http strong com href use rel noreferrer ul question em file image like org app application user need work
  - Topic 2: gt lt pre class public id int string type return new value data android function quot strong error user test

- Dans notre cas les sujets détectés par cette méthode sont similaires à ceux détectés par la méthode LDA

- Grâce à la segmentation non supervisée, on peut obtenir une idée du sujet de la question
  - Peut on avoir une meilleure précision grâce à la segmentation supervisée?

V. SEGMENTATION SUPERVISÉE :

- Création de la variable à prédire :
  - on conserve uniquement les 20 tags les plus fréquents
  - On conserve 1 seul tag pour les questions avec plusieurs tags (arbitrairement)
  - On supprime les questions qui ne sont plus catégorisées
  - On encode les tags pour obtenir un int (entre 0 et 19)
- Création des features :
  - Bag of word countvectorizer et TF-IDF
  - BERT
  - USE
  - word2vec
- Choix des modèles à tester :
  - SVM, KNN, NaiveBayes

V. SEGMENTATION SUPERVISÉE : RÉSULTATS

features	classifier	accuracy_train	precision_train	accuracy_test	precision_test	jaccard_train	jaccard_test
bow_cv	SVC	0.5988	0.5988	0.4135	0.4135	0.427348	0.260637
bow_cv	NB	0.9392	0.9392	0.4275	0.4275	0.885370	0.271860
bow_cv	KNN	0.4998	0.4998	0.2585	0.2585	0.333156	0.148435
bow_tfidf	SVC	0.9550	0.9550	0.5540	0.5540	0.913876	0.383126
bow_tfidf	NB	0.9516	0.9516	0.4205	0.4205	0.907669	0.266223
bow_tfidf	KNN	0.6224	0.6224	0.3920	0.3920	0.451800	0.243781
word2vec	SVC	0.5192	0.5192	0.3830	0.3830	0.350621	0.236858
word2vec	NB	0.1878	0.1878	0.1520	0.1520	0.103631	0.082251
word2vec	KNN	0.4644	0.4644	0.2150	0.2150	0.302423	0.120448
BERT_uncased	SVC	0.4050	0.4050	0.2880	0.2880	0.253918	0.168224
BERT_uncased	NB	0.2086	0.2086	0.1370	0.1370	0.116445	0.073537
BERT_uncased	KNN	0.4494	0.4494	0.1965	0.1965	0.289823	0.108955
BERT_HF_notrain	SVC	0.2282	0.2282	0.1930	0.1930	0.128796	0.106807
BERT_HF_notrain	NB	0.0474	0.0474	0.0430	0.0430	0.024275	0.021972
BERT_HF_notrain	KNN	0.3986	0.3986	0.1380	0.1380	0.248907	0.074114
BERT_HF_train	SVC	0.5966	0.5966	0.5555	0.5555	0.425110	0.384562
BERT_HF_train	NB	0.4950	0.4950	0.4905	0.4905	0.328904	0.324942
BERT_HF_train	KNN	0.6306	0.6306	0.4970	0.4970	0.460494	0.330672
USE	SVC	0.8568	0.8568	0.6755	0.6755	0.749475	0.510004
USE	NB	0.6016	0.6016	0.5505	0.5505	0.430206	0.379786
USE	KNN	0.7034	0.7034	0.5590	0.5590	0.542496	0.387925

V. SEGMENTATION SUPERVISÉE : RÉSULTATS

- Interprétations:
  - Le choix du modèle n'est pas le facteur principal de précision
  - Le choix des features influence fortement les résultats
  - Prédire un seul tag permet d'obtenir une bonne précision
  - Overfitting en utilisant les features TF-IDF : peut être amélioré en optimisant les hyperparamètres ou en ajoutant des données
  - Très bon résultat avec les features USE
  - Le modèle BERT est très sensible au sur-entrainement du modèle
    - Non efficace sans sur entrainement
    - Très bonne performance après 10 epochs

VI. DÉPLOIEMENT :

- Choix du modèle TF-IDF + SVC :
  - Bonne performance
  - Facilité à créer un pipeline Sklearn

- 1ère partie du code pour créer les pipelines et les mettre à jour
- 2ème partie pour la création d'une entrée API grâce à FastAPI
  - Point d'entrée en local sur le 127.0.0.1

VI. DÉPLOIEMENT : PIPELINE

- Fonctions du code :
  - Lecture en argument de la base de données exportée
  - Lecture en argument du modèle à utiliser (pour déployer plusieurs modèles)
  - Mode Verbose et Debug
  - Vérification des inputs
  - Création des variables à prédire
  - Sauvegarde de l'encodeur pour conserver la correspondance
  - Création du pipeline :
    - Transformation du texte
    - Modèle de segmentation
  - Fit du pipeline
  - sauvegarde

VI. DÉPLOIEMENT : API

- Utilisation de FastAPI
- Chargement du pipeline au démarrage
- Chargement de l'encodeur au démarrage
- Ajout d'une fonction prédict
  - Lecture d'un string
  - Renvoi un int et un str : tag number et tag
- Démonstration !

VII. CONCLUSIONS

- La base de données exportées a été caractériser :
  - Nombre de tags important : on ne conserve que les plus fréquents
  - Nombre de mots dans le vocabulaire important
    - Réduction du nombre de dimensions grâce à la lemmatisation et en supprimant les mots trop communs
- La segmentation non supervisée permet d'avoir une première indication sur les thèmes abordés sur le site
  - Complexe à interpréter lorsque l'on augmente le nombre de sujets
- La segmentation supervisée permet de prédire les tags les plus fréquents avec une bonne précision
  - Les transformations USE et TF-IDF donnent les meilleurs résultats dans notre cas
  - Il est nécessaire de sur-entrainer le modèle de BERT
- Un code a été développé pour mettre à jour facilement la segmentation
- Une entrée API a été créée grâce à FastAPI

VIII. AMÉLIORATIONS

- Sauvegarder des pipelines avec des modèles Tensorflow pour pouvoir utiliser ces modèles dans une API
- Optimiser les prédictions pour obtenir une prédiction de plusieurs tags et non pas un seul
- Optimiser les hyperparamètres des modèles
- Tester le modèle de BERT pour la prédiction et non pas uniquement pour la création des features
- Ajouter la transformation inverse de l'encodeur directement dans le pipeline
  - Éviter de charger deux fichiers lors de la création de l'API

MERCI POUR VOTRE ATTENTION !