Objectif : extraction des concepts + fonction de scoring pour la classification en « relevant » et « not relevant » :

1/ Libraries et Téléchargements de modèles :

installation et téléchargement les dépendances nécessaires et les modèles de langage associés.

- **NLTK (Natural Language Toolkit):** la tokenization, la lemmatization et le téléchargement de données comme les stopwords , WordNet et Punkt .
- **Re :** Permet de travailler avec des expressions régulières pour le nettoyage du texte en supprimant les caractères spéciaux
- **NetworkX :** est utilisé pour la manipulation et l'analyse des réseaux, utilisée ici pour construire un graphe de co-occurrence des mots dans le texte pour l'algorithme TextRank.
- **PyTextRank** : est un outil d'extraction de phrases clés basé sur TextRank, qui permet d'extraire des phrases clés à partir de texte en utilisant l'algorithme TextRank.
- **Pandas :** pour la manipulation de données sous forme de tableaux, utilisée ici pour charger et concaténer les données à partir des fichiers CSV .
- **Gensim :** pour le traitement de texte et le traitement de mots, utilisée ici pour charger des embeddings de mots pré-entraînés (GloVe) .
- **Numpy :** pour le calcul numérique, utilisée ici pour effectuer des opérations mathématiques sur les embeddings de mots.
- **cosine_similarity de sklearn.metrics.pairwise**
- **Seaborn :** pour la visualisation de données basée sur matplotlib, utilisée ici pour afficher la matrice de confusion sous forme de heatmap

- **Spacy** est pour charger les modèles de langue, ici "en_core_web_sm" pour l'anglais.

Le modèle "en_core_web_sm" est plus léger que le modèle "en_core_web_md"

NB: le modèle "fr_core_news_sm" est plus rapide à traiter en raison de sa taille réduite.

2/ Chargement des données :

Les données sont lues à partir d'un fichier CSV en utilisant la bibliothèque pandas. Taille : 100 lignes / 31 colonnes

3/ Prétraitement de données (preprocessing) pour la colonne « body » et la colonne « title »:

3.1/ Nettoyage du texte :

* Remplacement des sauts de ligne : utilise l'expression régulière "\n" pour détecter les sauts de ligne dans le texte et les remplace par un espace, de manière à éliminer les retours à la ligne.

* Conversion en minuscules

* Suppression des caractères spéciaux

*Suppression des caractères non alphabétiques

* Suppression des espaces en début de texte

* Suppression des espaces multiples

3.2/ Suppression des stop words

3.3/ Tokenisation et lemmatisation

4/méthodologie :

1/ appliquer un résumé pour la colonne « body » :

Le texte est analysé à l'aide de **SpaCy** pour obtenir le document (doc)

- Le résumé des phrases clés les plus importantes est obtenu à partir du document en utilisant **doc._.textrank.summary.** Cette opération permet de sélectionner **automatiquement** les phrases les plus significatives du texte, en se basant **sur l'algorithme TextRank.**
- Les phrases clés sont converties en une chaîne de texte (summary). Cela permet d'obtenir un résumé du texte original, composé des phrases clés les plus importantes.

NB : pourquoi l'utilisation d'un résumé : pour minimiser le nombre des mots inutiles dans un texte pour extraction des concepts

2 / Algorithme TextRank : (extraction des concepts)

- Le texte est d'abord tokenisé en phrases
- chaque phrase est ensuite tokenisée en mots
- calculer la frequence de chaque mot à l'aide de 'FreqDist'
- construire un graphe à partir des mots et de leurs cooccurrences. Les mots sont représentés par des nœuds et les cooccurrences sont représentées par des arêtes pondérées
- calculer les scores PageRank de chaque mot dans le graphe (en utilisant la bibliothèque NetworkX)

3 / Chargement des embeddings de mots pré-entraînés :

- utilisation de la bibliothèque gensim.downloader pour charger les embeddings de mots pré-entraînés (GloVe en l'occurrence) qui sont des représentations vectorielles des mots.

4 / Appliquer du score à l'ensemble du dataframe : c'est pour calculer **le score de similarité (cosine_similarity)** entre

les mots-clés extraits d'un document (titre et corps) et la liste de mots-clés agricoles fournie (agri_keywords)

- Extrait les mots-clés du titre (colonne title) en utilisant algo textrank (les tops 3)
- Extrait les mots-clés du corps (colonne body) en utilisant algo textrank (les tops 7)
- Recherche les embeddings de mots pré-entraînés (GloVe) pour les mots-clés extraits et les mots-clés agricoles.
- Calcule la similarité cosinus entre les embeddings des mots-clés extraits et les embeddings des mots-clés agricoles.
- Retourne la moyenne des scores de similarité (Les deux scores de similarité sont ensuite pondérés et moyennés pour obtenir le score de similarité final pour chaque mot clé de l'agriculture)

5 / Classifier : utiliser un seuil de 0.25 pour classifier les documents en "Relevant" ou "not Relevant" en fonction de leurs scores de similarité.

6/ Evaluer le modèle : la matrice de confusion pour évaluer la performance de votre modèle de classification. Vous avez calculé la matrice de confusion normalisée pour montrer les taux de vrais positifs, faux positifs, vrais négatifs et faux négatifs.

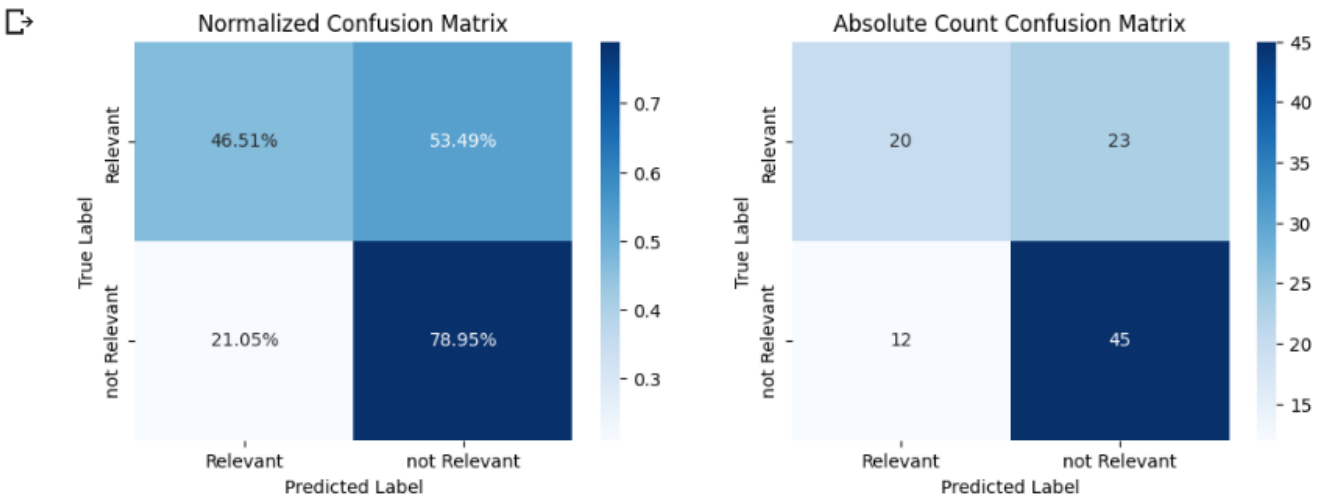
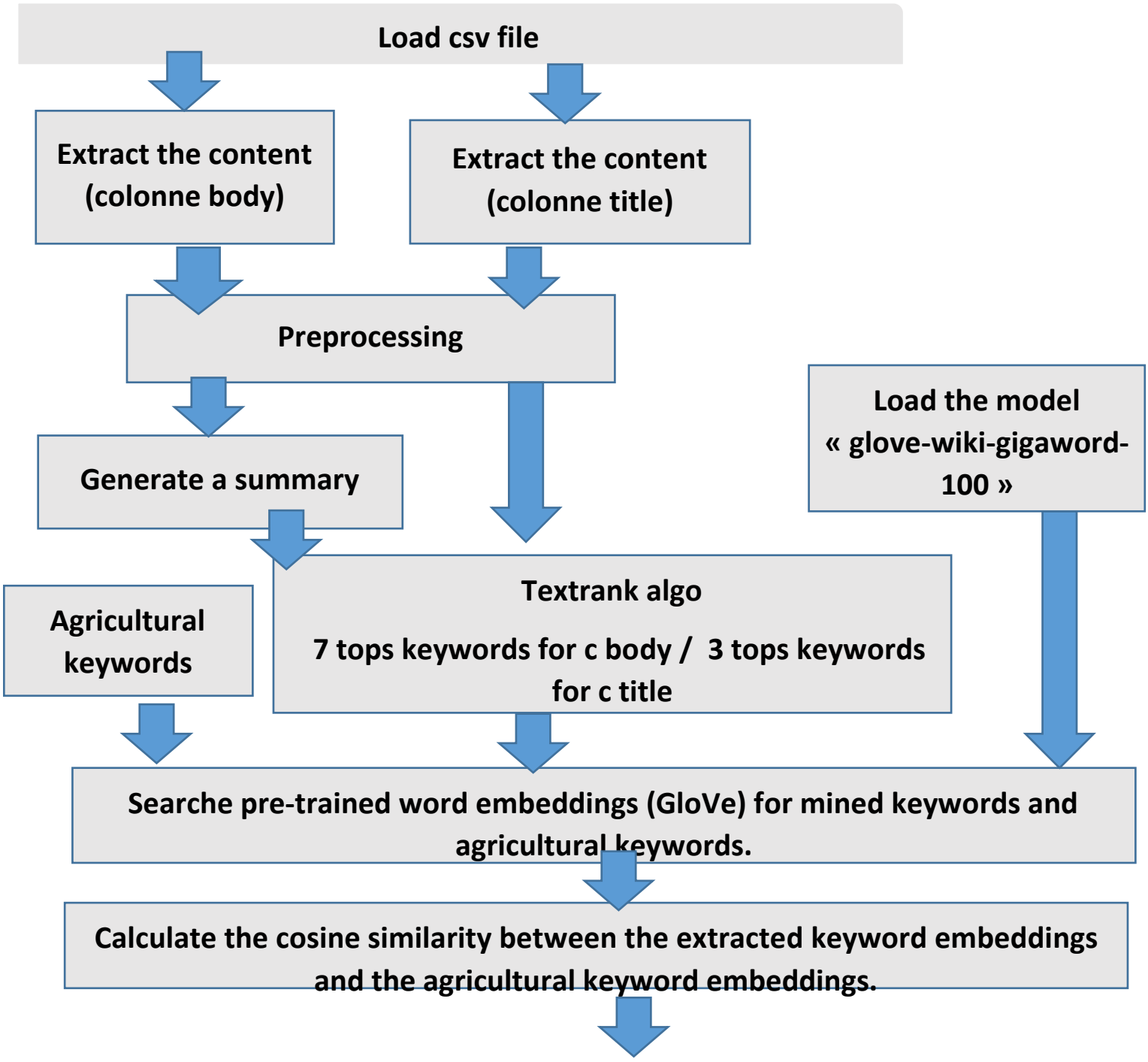
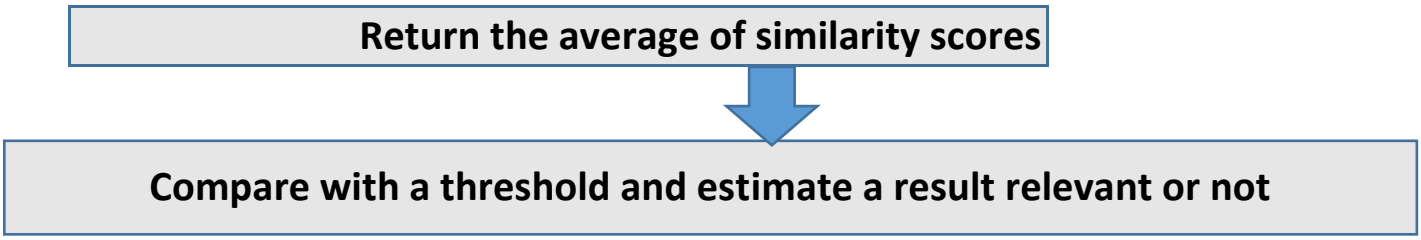


Schéma synoptique :




Return the average of similarity scores
Compare with a threshold and estimate a result relevant or not