1. Introduction et Architecture

Le projet *TextIndexerPy* est un outil en ligne de commande conçu pour indexer efficacement un ensemble de documents texte. Il permet à l'utilisateur de charger des fichiers, les prétraiter, les indexer, rechercher des termes, afficher des extraits pertinents et fournir des statistiques sur le corpus. Il repose sur une architecture modulaire répartie sur cinq composants, chacun assigné à un membre du groupe, garantissant une séparation claire des responsabilités et une extensibilité du système.

2. Description des Composants

a. Prétraitement des textes (Module 1)

Ce module charge les fichiers texte d'un dossier donné. Il applique un nettoyage linguistique (passage en minuscules, suppression de la ponctuation) suivi d'une tokenisation. Chaque document est transformé en liste de mots, stockée dans un dictionnaire {nom_fichier: [liste_mots]}.

b. Structure d'index inversé (Module 2)

L'index inversé associe chaque mot aux documents et aux positions où il apparaît sous la forme : mot → [(doc_id, [positions])]. Le module calcule également la fréquence des termes (TF) pour chaque document.

c. Algorithmes de recherche & scoring (Module 3)

Ce module prend en charge les recherches simples et multiples avec des modes *union* ou *intersection*. Le calcul de la pertinence repose sur la fréquence d'apparition des mots-clés. Les résultats sont renvoyés sous forme d'une liste triée par score décroissant.

3. Analyse de Complexité

a. Indexation

- **Prétraitement** : Complexité linéaire O(N), où *N* est le nombre total de mots.
- **Construction de l'index** : Pour chaque mot w dans un document de longueur L, l'ajout à l'index prend O(1). Sur tous les documents, c'est O(N).

b. Recherche

- **Recherche simple** : Accès direct à l'index → O(1) par mot, mais combinée sur tous les mots d'une requête de taille k, c'est O(k).
- **Scoring et tri** : Score basé sur TF, le tri des résultats est O(D log D) pour D documents correspondants.

4. Tests, Évaluation de la Pertinence

a. Tests Réalisés

- Requêtes simples d’un mot et plusieurs autres mots sur la collection de textes.
- Vérification des extraits affichés et des documents retournés.

b. Évaluation de la pertinence

Les résultats obtenus sont satisfaisants pour des requêtes simples, avec des extraits bien contextualisés. L’algorithme de score, bien que basique, fournit un classement cohérent. Des pistes d’amélioration incluent l’ajout de pondération TF-IDF.

5. Conclusion

Le projet *TextIndexerPy* a permis de mettre en pratique des notions essentielles d’algorithmique, de structure de données et d’architecture logicielle. L’approche modulaire a facilité le travail en équipe. Chaque membre a contribué à une fonctionnalité clé, rendant l’outil fonctionnel, structuré et évolutif.