Voici un **résumé clair, structuré et accessible** que tu peux intégrer dans ton mémoire de thèse pour expliquer le fonctionnement de ton script Python de calcul des scores TF-IDF et BM25 à partir d'expressions régulières appliquées à un corpus de textes.

📌Calcul des scores de pertinence TF-IDF et BM25

Afin d'évaluer la pertinence lexicale d'un ensemble de concepts (représentés par des expressions régulières), nous avons mis en place un script Python qui calcule deux scores standards de pondération lexicale : **TF-IDF** (Term Frequency–Inverse Document Frequency) et **BM25** (Best Matching 25). Le calcul repose sur une analyse document par document, de manière à refléter la répartition des termes dans l'ensemble du corpus.

📁Structure des données en entrée

- **Corpus** : un ensemble de fichiers `.txt`, chacun correspondant à un document.
- **Liste de termes candidats** : fournie sous forme de **regex** (expressions régulières), permettant de capturer des variantes flexibles (formes fléchies, pluriels, etc.).

🔍Étapes du traitement

1. **Lecture des expressions régulières**
   Chaque ligne du fichier `liste_concepts_regex.txt` est lue et compilée en expression régulière insensible à la casse. Ces motifs servent à repérer les occurrences des termes dans les documents.

2. **Parcours du corpus**
   Chaque document est traité individuellement. Pour chaque texte :
   - le nombre total de mots est compté (approximation du "document length"),
   - on cherche combien de fois chaque regex apparaît dans ce document,
   - si une regex est trouvée, on enregistre :
     - son nombre d'occurrences,
     - le fait qu'elle apparaît dans ce document (utile pour l'IDF).

3. **Calcul de l'IDF (Inverse Document Frequency)**

Pour chaque regex, on détermine dans combien de documents elle apparaît (`df`).
L'IDF est ensuite calculé selon la formule :

$$\text{IDF}(r) = \log\left(\frac{N}{\text{df}(r)}\right)$$

où N est le nombre total de documents. Cela permet de réduire le poids des termes très fréquents dans le corpus.

4. **Calcul du TF-IDF par document**
   Pour chaque document où une regex apparaît, on calcule :

$$\text{TF-IDF}(r, d) = \frac{f(r, d)}{|d|} \cdot \text{IDF}(r)$$

   où $f(r, d)$ est le nombre d'occurrences de la regex dans le document, et $|d|$ la longueur du document.
   Le score TF-IDF final pour une regex est obtenu en **sommant ses scores documentaires** sur tout le corpus.

5. **Calcul du score BM25 par document**
   Le score BM25 est une amélioration du TF-IDF qui introduit une fonction de saturation (pour éviter que les termes très fréquents soient surévalués), ainsi qu'une normalisation plus fine par la longueur du document :

$$\text{BM25}(r, d) = \text{IDF}(r) \cdot \frac{f(r, d) \cdot (k_1 + 1)}{f(r, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$$

   avec :

   - $k_1 = 1.2$: paramètre de saturation,
   - $b = 0.75$: pondération de la longueur du document,
   - avgdl : longueur moyenne des documents dans le corpus.
     Comme pour TF-IDF, les scores BM25 sont ensuite sommés sur tous les documents.

6. **Normalisation des scores**
   Les scores TF-IDF et BM25 sont normalisés entre 0 et 1 (via **min-max scaling**) pour faciliter la comparaison visuelle.

7. **Export des résultats**
   Un fichier `.csv` est généré contenant, pour chaque regex :
   - sa fréquence totale dans le corpus,
   - son score TF-IDF et BM25,
   - leur version normalisée.

✅Avantages de l'approche

- L'utilisation de **regex** permet de détecter des variantes lexicales complexes.
- Le calcul est **document-based**, respectant les définitions classiques de TF-IDF et BM25.
- L'agrégation des scores permet de **comparer les termes candidats** selon leur saillance statistique dans le corpus.
- L'export CSV permet des visualisations ultérieures (Flourish, matplotlib, etc.).

Souhaites-tu une version abrégée de ce résumé pour une légende ou un encart méthodologique ?