Introduction

The aim of project to implements a **Document Ranking System** designed for searching and ranking textual documents based on user queries. The system uses two distinct methods for ranking:

1. **Keyword Matching**
2. **TF-IDF (Term Frequency-Inverse Document Frequency) with Cosine Similarity**

The project aims to process a predefined set of text documents and return the most relevant results for a user's query, displayed in ranked order.

Features and Steps

1. Document Loading

The documents are loaded from a static folder path. The function load_documents read a predefined set of filenames (doc1.txt to doc5.txt) and stores their content in a dictionary. Files that cannot be found are skipped with a message.

2. Keyword Matching

- The keyword_match function identifies matches by splitting the user query and document contents into lowercase keywords.
- The match score is calculated as the number of query keywords found in the document.
- Documents are ranked in descending order of match scores.

3. TF-IDF Scoring

The script implements TF-IDF manually to ensure a fine-grained understanding of document ranking.

TF-IDF Components:

- **Term Frequency (TF):** Measures how often a word appears in a document.
  - Formula: TF(word) = (Number of occurrences of the word) / (Total number of words in the document)
- **Inverse Document Frequency (IDF):** Penalizes common words across documents.
  - Formula: IDF(word) = log(Total number of documents / Number of documents containing the word)
- **TF-IDF Vector:** Computed for each document and the query by combining TF and IDF values.

Ranking: Documents are ranked in descending order of cosine similarity scores.

4. Displaying Results

The function display_ranked_docs shows the top N ranked documents along with:

- Their scores
- A snippet of their content for user context

5. User Interaction

The script provides a command-line interface with options:

1. **Keyword Matching**: Ranks documents based on query keyword occurrences.
2. **TF-IDF Ranking**: Uses TF-IDF and cosine similarity for ranking.
3. **Exit**: Exits the system.

Implementation Analysis

Strengths:

1. **Multiple Ranking Strategies:** Supports both keyword matching and advanced TF-IDF ranking.
2. **Custom TF-IDF Calculation:** Offers a deeper understanding of ranking processes.
3. **Scalable Framework:** The modular design can handle additional features like stemming or stopword removal.

Limitations:

1. **Static File Paths:** The file paths are hardcoded, limiting usability across different environments.
2. **Stopwords and Preprocessing:** Common words (e.g., "the," "is") are not removed, which may skew results in both ranking methods.
3. **Snippet Generation:** The current snippet displays only the first 100 characters, which may not always highlight the query-relevant content.

Potential Improvements:

1. **Dynamic File Loading:** Allow users to dynamically specify a folder for document loading.
2. **Stopword Removal:** Use libraries like NLTK to filter out stopwords.
3. **Query Expansion:** Enhance queries with synonyms or related terms using NLP techniques.
4. **Improved Snippets:** Highlight query-related parts of the document instead of the first 100 characters.
5. **Performance Optimization:** Optimize TF-IDF computation for large datasets by using libraries like Scikit-learn.

Conclusion

The Document Ranking System is a well-structured script demonstrating basic IR techniques. While functional, its results can be further refined with advanced preprocessing and NLP integration. It provides a solid foundation for understanding IR concepts and developing more sophisticated search engines.