Case Study Assignment-II for IR: Implement a Basic Document Ranking System

Assignment Steps:

Step 1: Set Up Your Environment:

- Ensure you have Python installed on your computer.
- Choose a code editor or integrated development environment (IDE) for writing Python code (e.g., Visual Studio Code, PyCharm, or Jupyter Notebook).

Step 2: Gather Your Documents:

- Create a folder with a few sample text documents (e.g., .txt files) that you want to rank based on user queries. These documents should have some common keywords or themes.

Step 3: Create a Query Function:

- Write a Python function that takes a user query as input and returns a list of relevant documents based on some simple criteria. You can start by using keyword matching or TF-IDF scoring for simplicity.

Step 4: Implement Keyword Matching:

- In your query function, implement a basic keyword matching approach. Split the query into keywords and compare them with the content of each document. Rank the documents based on the number of matched keywords.

Step 5: Implement TF-IDF Scoring (Optional):

- To enhance your ranking system, you can implement TF-IDF (Term Frequency-Inverse Document Frequency) scoring. Calculate TF-IDF scores for each keyword in the query and the documents and use these scores to rank the documents.

Step 6: Display the Ranked Documents:

- Write a function to display the ranked documents to the user in order of relevance. You can show document titles and/or snippets of text.

Step 7: User Interaction:

- Create a simple user interface or command-line interface (CLI) that allows users to input their queries.

Step 8: Test Your Document Ranking System:

- Test your system with various queries and documents to ensure it ranks documents accurately based on relevance.

Step 9: Documentation and Submission:

- Write a brief report or documentation explaining how your basic document ranking system works, the techniques used, and any challenges you faced during implementation.
- Submit your Python code along with the documentation.

Step 10: Bonus (Optional):

- If you want to take your project further, consider exploring more advanced techniques like cosine similarity or using pre-trained word embeddings (e.g., Word2Vec or GloVe) for better document ranking.


Any
Questions