Name: Muhammad Mobeen

Registration No: 200901097

BSCS01-(B)

Presented to: Mam Reeda Saeed

Big Data Analytics

Assignment #1,2

Locality Sensitive Hashing

Potential Challenges:

1. **Handling Large Datasets**: The current implementation assumes that all documents can fit into memory. However, for large-scale applications with extensive document collections, this approach may not be feasible. Modifications would be required to handle data streaming or disk-based operations.
2. **Parameter Tuning**: The choice of parameters, such as the shingle size, the number of minhash vectors, and the number of bands, can significantly impact the performance and accuracy of the similarity search. Finding the optimal parameter values may require extensive experimentation and domain knowledge.
3. **Dealing with Near-Duplicates**: The provided implementation focuses on identifying exact duplicates or highly similar documents. However, in real-world scenarios, it is often desirable to detect near-duplicates or documents with a high degree of overlap. Additional techniques, such as combining LSH with other similarity measures or employing secondary filtering mechanisms, may be required to handle near-duplicates effectively.
4. **Scalability and Performance Optimization**: As the dataset grows, the computational requirements for generating minhash signatures and performing banding operations can increase significantly. Parallelization, distributed computing, or the use of specialized hardware (e.g., GPUs) may be necessary to achieve acceptable performance levels for large-scale deployments.

Potential Improvements:

1. **Incremental Updates**: Implement a mechanism to handle incremental updates to the document collection. This would allow for efficient updating of the minhash signatures and banding structures without having to recompute everything from scratch when new documents are added, or existing ones are modified.
2. **Distributed Implementation**: Consider adapting the implementation to a distributed computing framework, such as Apache Spark or Hadoop, to leverage parallel processing capabilities and handle larger datasets more efficiently.
3. **Hybrid Approaches**: Explore the possibility of combining LSH with other techniques, such as Locality Sensitive Permutation (LSP) or Locality Sensitive Bloom Filters (LSBF), to improve the accuracy and efficiency of the similarity search, especially for near-duplicate detection.
4. **Query-Aware Optimization**: Investigate techniques to optimize the similarity search based on the specific query or use case. For example, if the primary interest is in finding highly similar documents, the parameters and algorithms could be tuned accordingly to prioritize precision over recall or vice versa.
5. **Indexing and Retrieval Optimization**: Implement additional indexing and retrieval optimizations, such as caching frequently accessed minhash signatures or using more efficient data structures for storing and querying the banding structures.
6. **Handling Different Document Formats**: Extend the implementation to handle various document formats beyond the provided Word documents (.docx). This could involve integrating libraries or modules for text extraction from different file types (e.g., PDF, HTML, plain text).
7. **User Interface and Visualization**: Develop a user-friendly interface or visualization tools to facilitate the exploration and interpretation of the similarity search results, making it easier for users to understand and interact with the output.