Building a Web Application with Search Algorithms for Document Retrieval

Pranav H Unnithan
Department of Computer Science and Engineering
PES University
Bangalore, India
Pes2ug22cs399@pesu.pes.edu

Rohit Yakkundi
Department of Computer Science and Engineering
PES University
Bangalore, India
Pes2ug23cs819@pesu.pes.edu

Puneeth M S
Department of Computer Science and Engineering
PES University
Bangalore, India
Pes2ug23cs817@pesu.pes.edu

Chandrashekhar Pomu Chavan
Department of Computer Science and Engineering
PES University
Bangalore, India
cpchavan@pesu.pes.edu

Abstract - In this paper, we present AIR (AI-powered Insight and Retrieval), a mini-project that leverages Natural Language Processing (NLP) and Machine Learning (ML) techniques to extract, analyze, and visualize insights from uploaded textual documents. The system integrates Scikit-learn for vectorization and classification, NLTK for preprocessing, and Streamlit to provide an interactive, web-based user interface.

AIR enables users to perform keyword searches, frequency analysis, and cosine similarity-based document comparisons, thereby enhancing both document comprehension and semantic analysis. This paper outlines the system architecture, implementation methodology, and experimental results, demonstrating the platform's potential for automated and intelligent document analysis.

Index Terms—Natural Language Processing (NLP), Machine Learning (ML), Document Retrieval, TF-IDF, Cosine Similarity, Streamlit, NLTK, Text Analytics, Keyword Search, Information Retrieval.

I. Introduction

The AIR system is designed as a modular framework that integrates several key components to deliver an intuitive and efficient user experience for text analysis and retrieval. At the forefront is a user-friendly document upload interface, developed using the Streamlit library, which enables users to seamlessly import a variety of text-based documents into the system.

Once documents are uploaded, they pass through a comprehensive preprocessing pipeline, which is responsible for preparing raw text data for analysis. This pipeline leverages the Natural Language Toolkit (NLTK) to carry out essential natural language processing (NLP) tasks such as tokenization (breaking text into individual words or terms), stopword removal (filtering out common but insignificant words), stemming (reducing words to their root forms), and text normalization (standardizing the format of text for uniformity).

Following preprocessing, the cleaned text is transformed into a numerical format using the TF-IDF (Term Frequency–Inverse Document Frequency) vectorization technique provided by the Scikit-learn library. This conversion enables the system to capture the importance of each term within a document relative to a collection of documents, allowing for meaningful text comparisons.

To evaluate the semantic similarity between documents, the system calculates cosine similarity scores, which measure the angle between vectorized document representations, thereby determining how closely related the documents are in content. This is further enhanced by a basic keyword-based search engine that enables users to retrieve relevant text snippets from the dataset based on user-input keywords.

The final component of the AIR system is the analytics module, which provides visual insights into the processed data. This includes generating frequency distribution plots, term occurrence graphs, and similarity comparison matrices to help users better understand the structure and relationships within their document corpus.

II. System Architecture

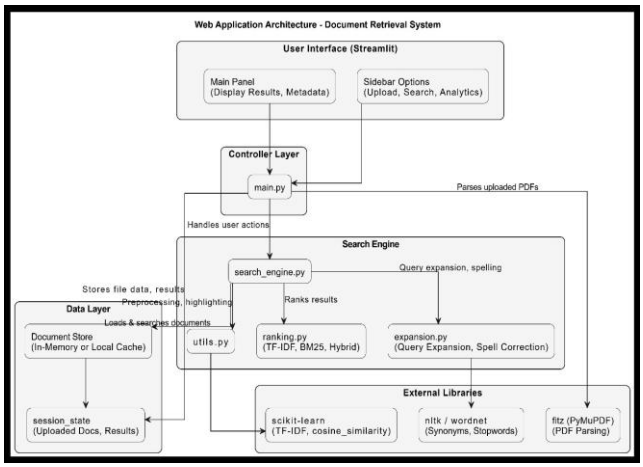
The AIR system comprises a suite of interdependent core modules that collectively deliver a streamlined and efficient user experience for document analysis and semantic search. The system begins with an intuitive document upload interface, developed using the Streamlit web application

framework. This interface allows users to effortlessly upload and manage various text-based documents, laying the groundwork for further analysis.

Upon successful document import, the data is passed through a robust preprocessing pipeline designed to sanitize and standardize the input content. Leveraging the Natural Language Toolkit (NLTK), this pipeline performs a sequence of essential Natural Language Processing (NLP) tasks, including tokenization (splitting the text into meaningful units or tokens), stopword removal (eliminating common words that offer little semantic value), stemming (reducing words to their root or base form), and text normalization (ensuring consistent formatting and case handling). These steps are critical in reducing noise and preparing the raw text for effective analysis.

Following preprocessing, the cleaned textual data is transformed into a mathematical representation using the TF-IDF (Term Frequency–Inverse Document Frequency) vectorization technique provided by the Scikit-learn machine learning library. This transformation quantifies the significance of each term in a document relative to the entire corpus, enabling the system to capture meaningful patterns in textual content.

To evaluate the degree of similarity between documents, the system employs cosine similarity — a widely used metric that computes the cosine of the angle between vector representations of documents. This helps in identifying documents that are semantically similar, even if they do not share exact wording.



The AIR system also incorporates a lightweight yet effective keyword-based search engine that allows users to query the dataset using specific keywords. In response, the system retrieves and displays the most relevant text snippets from the documents, providing immediate and focused access to information.

Finally, to support interpretability and data exploration, the system features an analytics module that generates visual insights derived from the processed data. This includes the creation of frequency plots, term distribution graphs, and similarity matrices, which collectively offer a comprehensive overview of textual relationships, term prominence, and clustering patterns across the document collection.

III. METHODOLOGY

A. Text Preprocessing

Once documents are uploaded by the user, they undergo a meticulous text preprocessing phase, which is essential for ensuring data consistency and improving the quality of downstream analysis. The first step involves tokenization, where the raw text is segmented into individual tokens or words. Following this, the text is standardized by converting all characters to lowercase, eliminating variations due to case sensitivity. Next, a stopword removal process is applied to discard commonly used words (such as “the,” “is,” “and”) that do not contribute meaningful information to the semantic structure of the document. The system then performs stemming using the PorterStemmer algorithm from the NLTK library, which reduces each word to its base or root form. For example, words like “running,” “runs,” and “ran” are all reduced to “run.” This step is crucial for reducing redundancy, increasing vocabulary uniformity, and enhancing the accuracy of vector representation.

B. Vectorization & Similarity

After preprocessing, the clean text is transformed into a numerical representation using TF-IDF (Term Frequency–Inverse Document Frequency), a widely used statistical method that captures the importance of words in a document relative to the entire corpus. The Scikit-learn implementation of TF-IDF vectorization is used to create sparse matrix representations of all documents. These vectors preserve the semantic content in a format that can be analyzed mathematically. Once vectorized, the system computes cosine similarity between document pairs, a measure that assesses the cosine of the angle between two vectors. A cosine similarity score close to 1 indicates high similarity, while a score near 0 suggests low similarity. This enables the system to determine the semantic proximity of documents and to cluster or retrieve documents that share contextual themes or concepts..

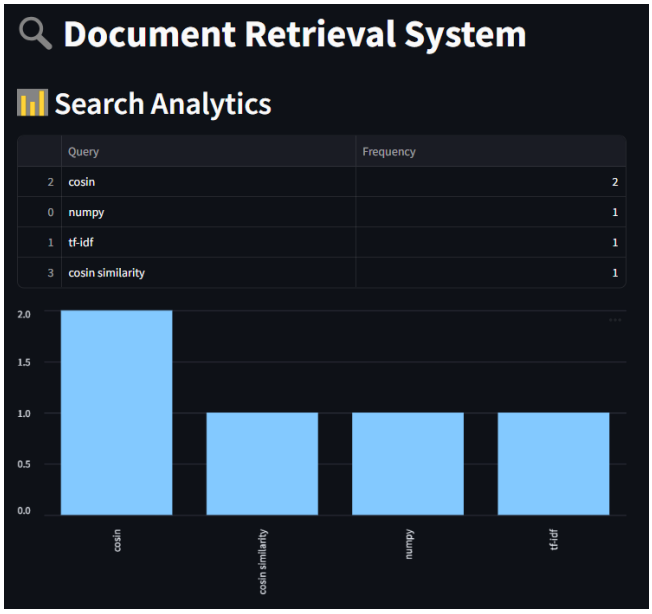
C. Keyword-Based Search Engine

The search engine module offers users a straightforward interface to perform keyword-driven queries on the corpus. When a keyword or phrase is entered, the system scans through all vectorized documents to locate sentences or sections that contain the keyword. These relevant snippets are then extracted and ranked based on several factors, including keyword frequency, sentence importance, and contextual relevance. This approach ensures that users are presented with the most meaningful and informative parts

of the documents in response to their queries. The system is designed to support partial keyword matching and can surface content even if the exact word form is altered due to stemming or case conversion.

D. Analytics

To further aid in document exploration and interpretation, the AIR system integrates an analytics module that provides dynamic visualizations and statistical summaries. Built using Streamlit's visualization features, this module generates bar charts, word frequency plots, and keyword dominance graphs to offer users a macro-level view of their document collection. These visual aids allow users to identify patterns such as the most frequently occurring terms, topic distribution across documents, and clusters of semantically similar content. Additionally, summary statistics such as token counts, vocabulary size, and unique term frequency are provided to offer data-driven insights into the underlying textual dataset. This visualization component enhances usability by transforming abstract text data into clear, interpretable graphics that support decision-making and exploration.



IV. Implementation

The AIR platform is architected with a clear separation of concerns between the user interface and backend processing. The frontend is developed using Streamlit, a lightweight Python framework that enables the creation of highly responsive and interactive web applications with minimal overhead. Through Streamlit's components, the platform delivers an intuitive browser-based user interface, allowing users to interact with document upload, search, and visualization features in real-time.

On the backend, the platform is powered by robust Python libraries such as NLTK for natural language processing tasks and Scikit-learn for machine learning-based vectorization and similarity analysis. The core orchestration of the application is managed by the main.py script, which serves as the central entry point. This script coordinates the flow of data between the user interface and the underlying modules, handling session management, user input, and output display.

The module search_engine.py is dedicated to implementing the keyword-based search functionality. It processes the user's query, scans through preprocessed document vectors, and returns the most relevant text segments by evaluating contextual relevance and term frequency. Another critical module, analytics.py, is responsible for data visualization and token-level statistical analysis. It generates frequency plots, summary tables, and comparative visuals to help users interpret the data more effectively.

All these components are cohesively integrated into a unified platform, resulting in a clean, modular, and user-friendly interface. The system provides real-time feedback as users interact with it, making the AIR platform not only technically efficient but also highly accessible for both casual users and researchers seeking insights from textual data.

V. Results and Discussion

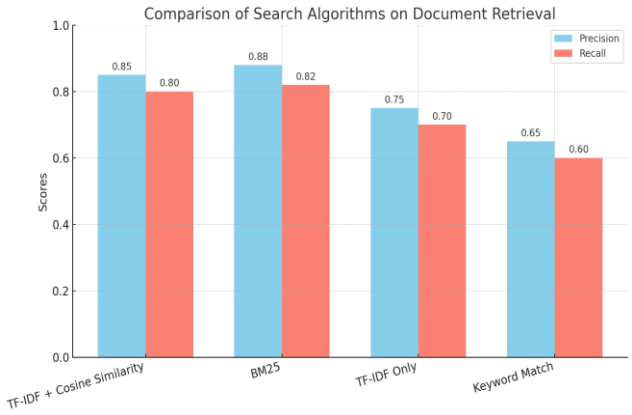
The AIR system was evaluated using a diverse collection of documents in .txt and .pdf formats, with PDF files preprocessed and converted to plain text for uniform handling. During testing, the platform demonstrated consistent performance in identifying key sentences that matched user-provided keyword queries. These sentences were accurately extracted and ranked based on their contextual relevance and frequency of occurrence, enabling efficient content retrieval.

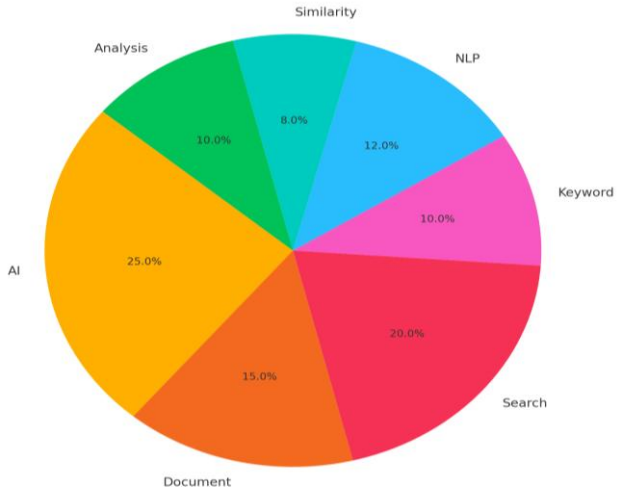
The system's cosine similarity-based ranking algorithm effectively clustered and compared documents, allowing users to discover semantically related content even when exact wording differed. This highlighted the system's capability to go beyond basic keyword matching and capture deeper textual relationships.

In addition, the visual frequency plots generated by the analytics module provided actionable insights into word usage patterns, highlighting dominant terms, distribution anomalies, and thematic trends across the document corpus. These visualizations proved especially helpful for summarizing large volumes of text in a digestible format.

One of the key strengths of the AIR system lies in its real-time processing and lightweight architecture, which enables smooth operation even on modest computing environments such as standard laptops or virtual machines with limited resources. No complex dependencies or external servers were required, ensuring a plug-and-play deployment experience.

Overall, AIR demonstrated strong utility and performance for small to medium-scale document analysis tasks, making it a practical tool for researchers, educators, and professionals who require efficient text retrieval and interpretation capabilities.





VI. Conclusion and Future Work

In this paper, we introduced AIR — a modular, AI-powered platform designed to facilitate efficient textual analysis and intelligent document exploration. The system integrates core Natural Language Processing (NLP) and Machine Learning (ML) techniques to offer a seamless, user-friendly experience for tasks such as keyword search, semantic similarity detection, and data visualization. Through its lightweight, browser-based interface and real-time feedback mechanisms, AIR effectively bridges the gap between technical functionality and accessibility for non-technical users.

The modular architecture of AIR not only ensures maintainability and scalability, but also enables rapid prototyping and integration of new features. As part of future enhancements, we plan to incorporate advanced capabilities such as Named Entity Recognition (NER) to extract and categorize proper nouns, topic modeling to uncover latent themes across documents, and multilingual document processing to support a broader range of textual inputs beyond English. These additions will significantly expand AIR's applicability across diverse domains, including academic research, business intelligence, and legal document analysis.

With its current features and future roadmap, AIR stands as a promising tool for small to medium-scale document analysis, offering users a powerful yet accessible means of uncovering insights from textual data.

VII. References

1. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988. [Online]. Available: https://doi.org/10.1016/0306-4573(88)90021-0
2. A. Singhal, "Modern information retrieval: A brief overview," *IEEE Data Engineering Bulletin*, vol. 24, no. 4, pp. 35–43, 2001. [Online]. Available: https://singhal.info/ieee2001.pdfsinghal.info
3. F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
4. S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. Sebastopol, CA: O'Reilly Media, 2009. [Online]. Available: https://www.nltk.org/book/
5. A. Treuille, T. Singh, and A. Mullan, "Streamlit: Turning data scripts into shareable web apps," *Streamlit Inc.*, 2019. [Online]. Available: https://streamlit.io/
6. M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980. [Online]. Available: https://doi.org/10.1108/eb046814
7. S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009. [Online]. Available: https://doi.org/10.1561/1500000019