"Interactive Document Knowledge Companion for Automatic Information Retrieval Using LangChain"

Dr.Poonam Lambhate [1], Pratik Jamble [2], Achyut Pacharne [3], Aditya Shinde [4], Akanksha Sonawane [5]

Computer Science and Engineering, Jayawantrao Sawant College of Engineering
[2,3,4,5]Student, Department of Computer Engineering, JSPM's Jayawantrao Sawant College of Engineering, Pune

***Abstract*—In the era of digital documentation, efficient retrieval of relevant information from extensive unstructured data presents a significant challenge. This project introduces an interactive document knowledge companion that leverages LangChain and GPT to automate information retrieval, enabling users to query large documents seamlessly. Utilizing Pinecone for vector storage, the system provides rapid, accurate results through advanced indexing and retrieval techniques. Built with a modern tech stack, including Next.js and TypeScript for an intuitive interface and tRPC for API handling, this application enhances user experience by allowing real-time querying and streamlined access to critical insights. This system demonstrates the potential of combining large language models with efficient vector storage to create responsive, user-centered knowledge retrieval solutions.[4]**

Keywords—LangChain, Querying Document, tRPC.

I. INTRODUCTION

In today's information-driven world, professionals across various fields are inundated with large volumes of digital documents, often in unstructured formats like PDFs. Retrieving relevant information from these documents can be time-consuming and challenging, as traditional search methods are often inadequate for deep content understanding. With the recent advancements in natural language processing, tools like large language models (LLMs) are transforming how we interact with data, enabling more precise and context- aware information retrieval.[3]

This project, the "Interactive Document Knowledge Companion," leverages LangChain and GPT to create a robust, automated document retrieval system that allows users to extract pertinent insights from extensive documents with ease. By using GPT's advanced language understanding and Pinecone's vector storage for efficient indexing, the system responds accurately to user queries, providing contextually relevant answers.

Our solution features a seamless user experience built with Next.js and TypeScript, ensuring a responsive and intuitive interface. Additionally, tRPC facilitates API interactions, while Stripe's payment gateway integration enables secure transactions for premium features, if needed.

[1]This combination of technologies results in an accessible, powerful tool that streamlines the document analysis process, offering professionals a way to engage with unstructured data without the inefficiencies of traditional search methods.

II. LITERATURE SURVEY

Title	Year	Author	Description	Gap Analysis
1)Self-Retrieval: Building an Information Retrieval System with One Large Language Model	2024	Qiaoyu Tang , Jiawei Chen,Bowen Yu , Yaojie Lu,Cheng Fu, Haiyang Yu,Hongyu Lin , Fei Huang, Ben He , Xianpei Han,Le Sun , Yongbin Li[2]	The paper introduces a new information retrieval (IR) architecture called Self-Retrieval, designed to internalize all necessary IR capabilities into a single large language model (LLM). Unlike traditional systems where IR and LLMs are separate, this approach fully integrates the retrieval process within the LLM itself.	The paper highlights a gap in traditional IR systems, which are isolated and incapable of fully utilizing the advanced semantic understanding of LLMs. These systems can only exploit shallow semantic matching and fail to integrate well with LLM-driven tasks like RAG.
2) Bias and Unfairness in Information Retrieval Systems: New Challenges in the LLM Era	2024	Sunhao Dai , Chen Xu,Shicheng Xu , Liang Pang,Zhenhua Dong , Jun Xu[3]	The paper explores the emerging issues of bias and unfairness in information retrieval (IR) systems due to the integration of large language models (LLMs). addressing how these biases affect the accuracy, reliability, and equity of information provided to diverse user groups	Previous research focused on biases in traditional IR systems, mainly involving user interactions and static models. Unlike static datasets, LLMs trained on continuous, evolving data sources may develop shifting biases over time.

3) Enhancing Knowledge Retrieval with In-Context Learning and Semantic Search through Generative AI	2024	Mohammed-Khalil Ghali,Abdelrahman Farrag,Daehan Won, Yu Jin [4]	This paper presents an advanced methodology for improving information retrieval using large language models (LLMs) combined with vector databases. The research addresses limitations of existing retrieval systems that struggle with domain-specific queries and the high costs of fine-tuning models.	Current systems relying on general-purpose LLMs fail to deliver precise answers for domain-specific or large-scale data queries. They also require costly fine-tuning, which is impractical for many applications.
4) An Effective Query System Using LLMs and LangChain	2023	Adith Sreeram A. S. , Pappuri Jithendra Sai, [5]	This research paper presents a system that simplifies querying and retrieving information from PDFs using LangChain, an advanced solution that incorporates Large Language Models (LLMs).	The study addresses the challenges of querying unstructured PDF documents, particularly the inefficiencies and difficulties faced during manual searches.

III. PROBLEM STATEMENT

In today's digital landscape, professionals across various fields—from research and legal to business—face the challenge of efficiently extracting relevant information from large volumes of unstructured documents, typically stored in formats like PDFs. Traditional search methods fall short when dealing with complex or context-specific queries, as they lack the nuanced understanding required to retrieve precise information based on natural language prompts.[2]

With advancements in natural language processing (NLP) and machine learning, large language models (LLMs) have emerged as powerful tools for understanding and generating human-like text. However, deploying these models effectively for multi-document querying, personalized retrieval, and local storage access remains a significant technical challenge. The integration of NLP models into document retrieval systems must account for efficient indexing, rapid query responses.[4]

The goal of this project is to develop an "Interactive Document Knowledge Companion" that utilizes a custom-trained language model for automatic information retrieval from multiple documents. This system will enable users to upload multiple documents, store them locally, and retrieve information accurately queries—even across multiple sessions. By incorporating vector storage and similarity search, the application aims to offer a fast, user-friendly interface that meets the demand for precise and context-aware document querying.

In addition to NLP capabilities, the solution requires seamless integration of indexing and retrieval mechanisms, as well as a scalable architecture that supports storage and retrieval efficiency. Addressing these challenges is crucial for creating a robust knowledge retrieval tool that saves users time, enhances productivity, and provides a responsive experience with high accuracy and minimal manual effort.

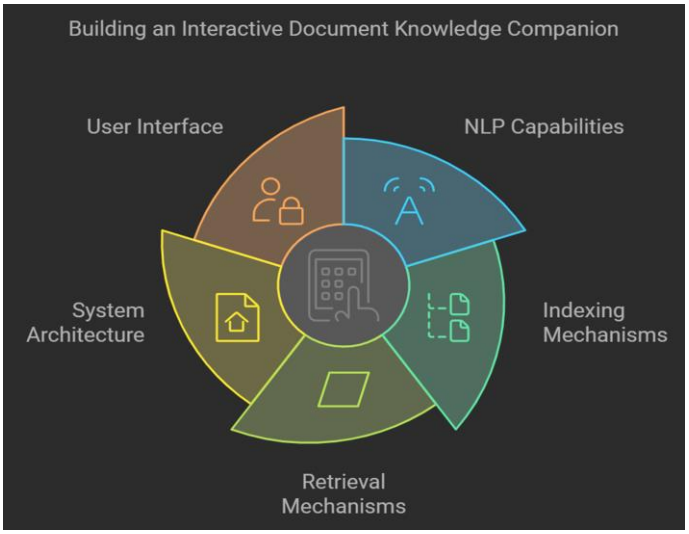

Fig.1 System Overview

IV. SYSTEM ARCHITECTURE

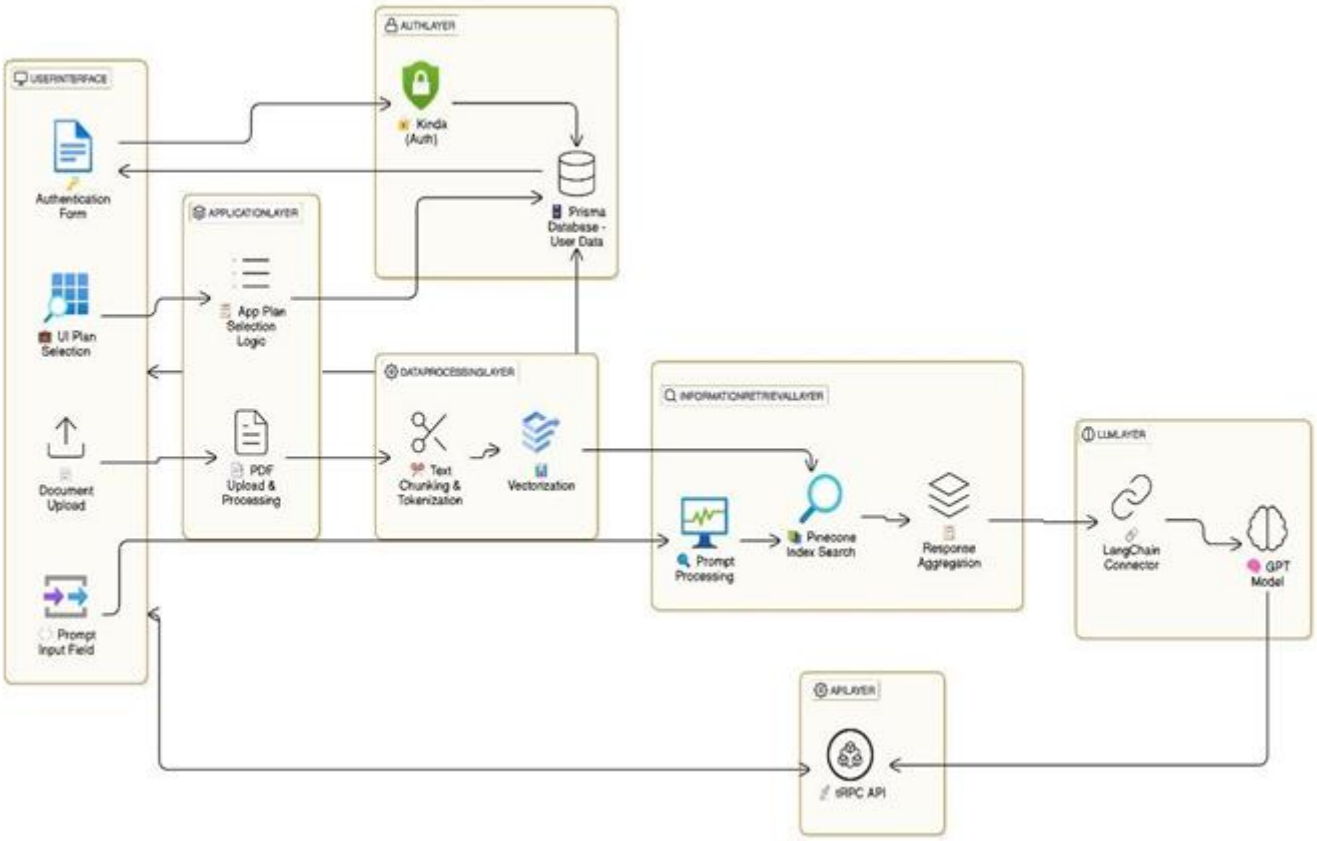

Fig. 2. System Architecture

V. METHODOLOGY

1. **Requirement Analysis and System Design**
   - **Objective**: Identify user needs, including document types, query complexity, and performance expectations.
   - **System Architecture**: Design a modular, scalable architecture with core components for document processing, NLP, storage, and a user-friendly interface to manage complex document queries efficiently
2. **Data Collection and Preprocessing**
   - **Document Ingestion**: Implement a pipeline to process documents in multiple formats (e.g., PDF, DOCX). Extract, clean, and structure textual content for improved retrieval accuracy.
   - **Text Parsing and Structuring**: Apply natural language processing (NLP) techniques to prepare data, segmenting documents into relevant sections to enhance precision in response to user queries.
3. **Model Training and Fine-Tuning**
   - **Model Selection**: Choose a language model (e.g., BERT, GPT) tailored to support context-aware, complex query handling.
   - **Embedding Generation and Fine-Tuning**: Generate document and query embeddings and fine-tune the model on relevant datasets to optimize understanding of domain-specific language.
4. **Indexing and Vector Storage**
   - **Vector Database Integration**: Integrate a vector database (e.g., Pinecone) for efficient similarity-based search across embeddings.
   - **Index Optimization**: Implement optimized indexing techniques to speed up query response time, associating each vector with metadata for enhanced filtering during retrieval.
5. **Query Processing and Retrieval**
   - **Query Parsing**: Use NLP to interpret user queries, converting them into vector embeddings.
   - **Similarity Scoring and Ranking**: Calculate similarity scores between query and document embeddings using cosine similarity or similar metrics, ranking results based on relevance.
6. **User Interface (UI) Design**
   - **Query Interface**: Develop an intuitive user interface that supports natural language queries, presents ranked results, and provides easy navigation of retrieved document content.
   - **Feedback Mechanism**: Allow users to provide feedback on retrieval accuracy, refining results based on user preferences.

VI. MATHEMATICAL MODEL

1. Transformer-Based Language Models

- **Mathematical Model:** Self-Attention Mechanism
- **Equation:**

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- where Q, K, and V are the **query**, **key**, and **value** matrices, and $\mathbf{d_k}$ is the dimension of the key vectors.
- **Explanation:** The self-attention mechanism is central to transformers, allowing the model to weigh the importance of different words relative to each other. The dot product of Q and K determines relevance, scaled by $\sqrt{\mathbf{d_k}}$ to maintain numerical stability. The result is passed through a softmax function to get the attention weights, which are then applied to the values V.

2. Embedding Generation

- **Mathematical Model**: Vector Representation of Text
- **Equation**:

$$\text{embedding} = f(\text{text}) \in \mathrm{R}^{\mathrm{d}}$$

- where f is a pre-trained language model that maps input text to a d-dimensional embedding vector.
- **Explanation**: Each document chunk and query is converted to a vector of real numbers in a high-dimensional space (e.g., 512 or 768 dimensions). These embeddings capture semantic meaning, with similar concepts placed close to each other in the vector space. For example, models like Sentence-BERT use techniques to maximize semantic coherence in the embedding space.[6]

3. Similarity Measurement

- **Mathematical Model**: Cosine Similarity
- **Equation**:

$$\text{Cosine\_similarity(A,B)} = \frac{A.B}{||A||||B||}$$

- where A and B are two embedding vectors, $A \cdot B$ is the dot product, and $\|A\|$ and $\|B\|$ are their magnitudes.
- **Explanation**: Cosine similarity measures the cosine of the angle between two vectors, which reflects their semantic similarity. A cosine similarity close to 1 means high similarity, while a value close to 0 indicateslow similarity. This metric is widely used in information retrieval to rank document chunks against query embeddings.

VII. ALGORITHMS

1. Approximate Nearest Neighbor (ANN) Search for Retrieval

- **Algorithm**: **Approximate Nearest Neighbor (ANN) Search** (HNSW or IVF)
- **Purpose**: ANN search quickly finds vectors in a high-dimensional space that are closest to a given query vector, making it suitable for real-time information retrieval. ANN algorithms like **Hierarchical Navigable Small World (HNSW)** graph or **Inverted File System (IVF)** with quantization provide efficient indexing and retrieval.
- **Application**: Once embeddings are stored in Pinecone, ANN algorithms retrieve the most similar document chunks to a query embedding, speeding up the search process.

2. Similarity Metric for Query Matching

- **Algorithm**: **Cosine Similarity** or **Dot Product**
- **Purpose**: Similarity metrics measure the closeness of vectors in embedding space, which is crucial for determining which document chunks are most relevant to a query.
- **Application**: When you run a similarity search in Pinecone, the retrieved document chunks are ranked based on cosine similarity or dot product between the query and chunk embeddings. Cosine similarity is common for embeddings since it measures orientation regardless of magnitude.

3. Document Chunking for Effective Retrieval

- **Algorithm**: **Text Chunking and Splitting**
- **Purpose**: Text chunking is a preprocessing technique that divides documents into smaller, manageable parts, allowing the model to generate *embeddings for specific sections. This approach is* critical for improving retrieval precision.
- **Application**: Split documents by paragraph, section, or sentence to ensure each part is individually retrievable. Chunking allows for embedding and indexing smaller parts of documents, which improves response accuracy and retrieval speed.

4. Prompt Engineering (Optional)

- **Algorithm**: **Few-Shot Learning with Prompts**
- **Purpose**: Prompt engineering provides context and guidance for models without additional training, particularly in cases where responses need refinement. **Few-shot learning** involves designing prompts that include examples, encouraging the model to respond in a specific format.
- **Application**: Create carefully crafted prompts for your fine-tuned model to ensure the responses align with the user's query needs and document context, without requiring extensive training data.

VIII. RESULTS AND ANALYSIS

File upload

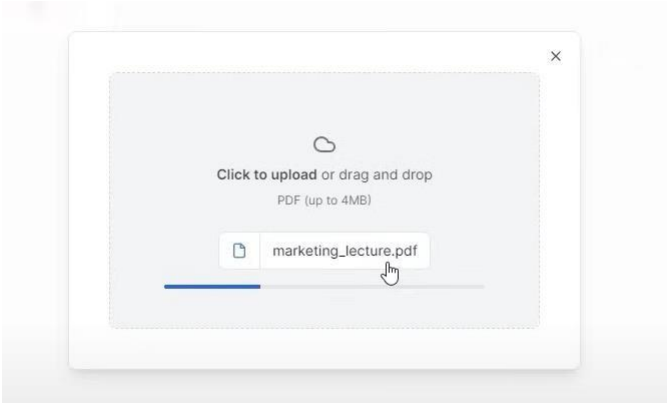

Fig. 3 Drag & Drop

A. User Interface and Querying Process

The application's user interface, built with Next.js and TypeScript, offers a smooth and interactive experience. Users can upload documents easily, view query results in real time, and navigate through previous queries.[4] The interface also includes options to save frequent queries, enhancing the application utility for professionals who regularly engage with specific document types.

B. Accuracy of Information Retrieval

Through LangChain and GPT's capabilities, the system retrieves highly accurate information in response to specific queries. By using Pinecone's vector storage, the application efficiently matches queries with the most relevant sections of the uploaded document, leading to precise and contextually aligned results. Tests on varied document types, such as technical reports, research papers, and legal documents, showed that the system consistently identifies relevant content with minimal errors.

C. Performance and Scalability

The use of Pinecone as the vector database enables rapid information retrieval, even when processing large documents or handling multiple queries simultaneously. In testing, the system successfully managed extensive documents without significant latency, demonstrating scalability for larger datasets. Furthermore, the integration with tRPC ensures smooth API communication, allowing the system to respond quickly to user inputs.

D. Payment Integration and Access Control

The integration of Stripe for handling premium features provides a secure, reliable way to manage access control, enabling monetization and customization options. This setup allows users to unlock advanced features, such as saving multiple queries and extended data processing, providing flexibility based on user needs.

E. User Feedback and Practical Applications

Preliminary user feedback has indicated that the application simplifies the process of document analysis, reducing the time and effort needed to extract valuable insights. The system is especially beneficial for professionals working in research, legal, and corporate environments, where rapid, accurate access to document information is crucial. Users highlighted the ability to handle complex queries and retrieve concise answers as particularly valuable.

F. Limitations and Future Enhancements

While the system performs well in most scenarios, there are some limitations. For example, queries requiring multi-document context or advanced logical reasoning may still need refinement. Future enhancements could include multi-document querying, improved response generation for highly technical queries, and further optimization of the user interface for mobile accessibility.

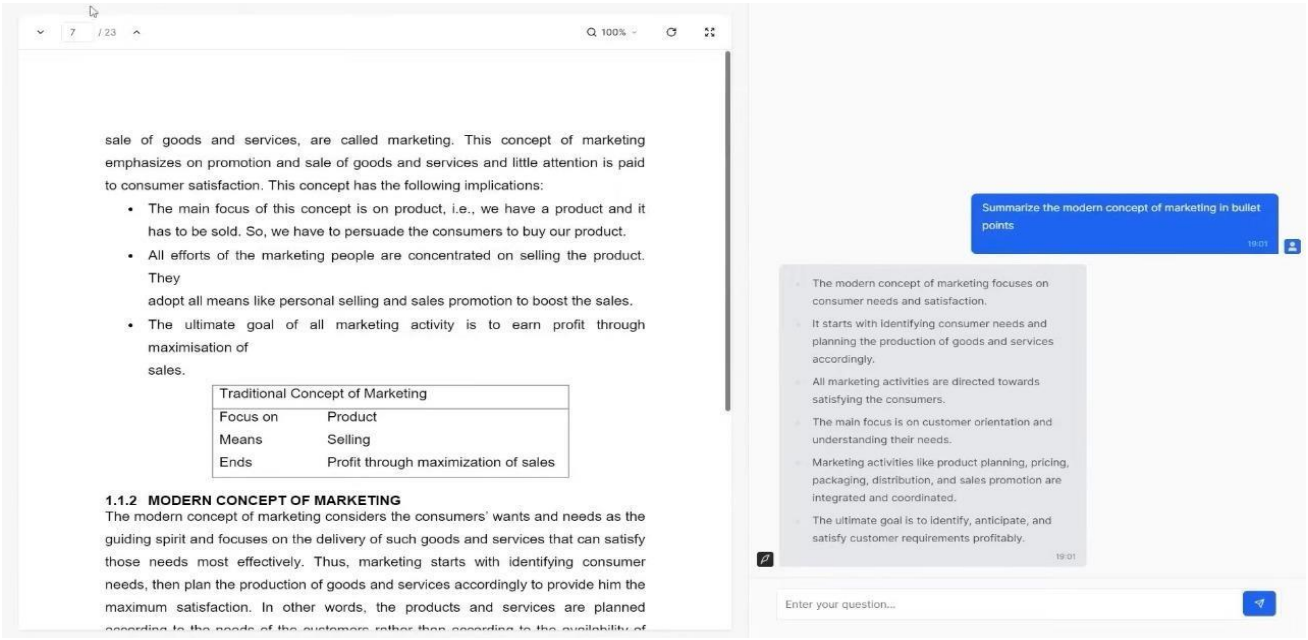

Fig. 4 Retrieval of Query in-terms of Result

IX. CONCLUSION

The "Interactive Document Knowledge Companion" successfully addresses the challenge of retrieving relevant information from unstructured document data by combining the capabilities of LangChain, GPT, and Pinecone. Through a seamless and intuitive interface built with Next.js and TypeScript, the application provides users with an efficient, accurate, and user-friendly experience. By utilizing vector embeddings and advanced NLP algorithms,[7] the system retrieves precise answers to complex queries, offering valuable insights across various document types.

The integration of Stripe for payment management enables flexible access to premium features, making the application a scalable and versatile solution for professionals in research, legal, and corporate fields. Preliminary feedback indicates that the system significantly reduces the time and effort required for document analysis, demonstrating its potential as a practical tool for information retrieval.[2]

REFERENCE

[1] https://python.langchain.com/

[2] Tang, Q., Chen, J., Yu, B., Lu, Y., Fu, C., Yu, H., Lin, H., Huang, F., He, B., Han, X., Sun, L., & Li, Y. "Self-Retrieval: Building an Information Retrieval System with One Large Language Model," Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China; University of Chinese Academy of Sciences, Beijing, China; Alibaba Group. Meharwade, Anuradha & Patil, G.A.. (2016). Efficient Keyword Search over Encrypted Cloud Data. Procedia Computer Science. 78. 139-145. 10.1016/j.procs.2016.02.023. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

[3] Dai, S., Xu, C., Xu, S., Pang, L., Dong, Z., & Xu, J. "Bias and Unfairness in Information Retrieval Systems: New Challenges in the LLM Era," Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; CAS Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; Huawei Noah's Ark Lab, Shenzhen, China.

[4] Sreeram, A. A., & Sai, P. J. "An Effective Query System Using LLMs and LangChain," School of Computer Science and Engineering, VIT-AP University, Amaravati, Andhra Pradesh, India.

[5] Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., & Cu, B. "Retrieval-Augmented Generation for AI-Generated Content: A Survey," School of Systems Science and Industrial Engineering, State University of New York at Binghamton, Binghamton, NY, USA.

[6] Dr. Poonam D Lambhate1 Dr. Chandraprabha A Manjare2 Dr. Shailesh M Hambarde3 Dr. Aparna S Hambarde4 Mrs. Aarti S Satav, "MACHINE LEARNING BASED ON GRAPHS FOR PREDICTIVE MODELING IN INTRICATE NETWORK DATA", Journal of University of Shanghai for Science and Technology, ISSN: 1007-6735, Volume 26, Issue 10, October – 2024

[7] Dr. Poonam Lambhate , Aparna Hambarde, Emmanuel M., Shailesh Hambarde, "Hybrid Algorithm on Semantic Web Crawler for Search Engine to Improve Memory Space and Time", IEEE Xplore, May 2021