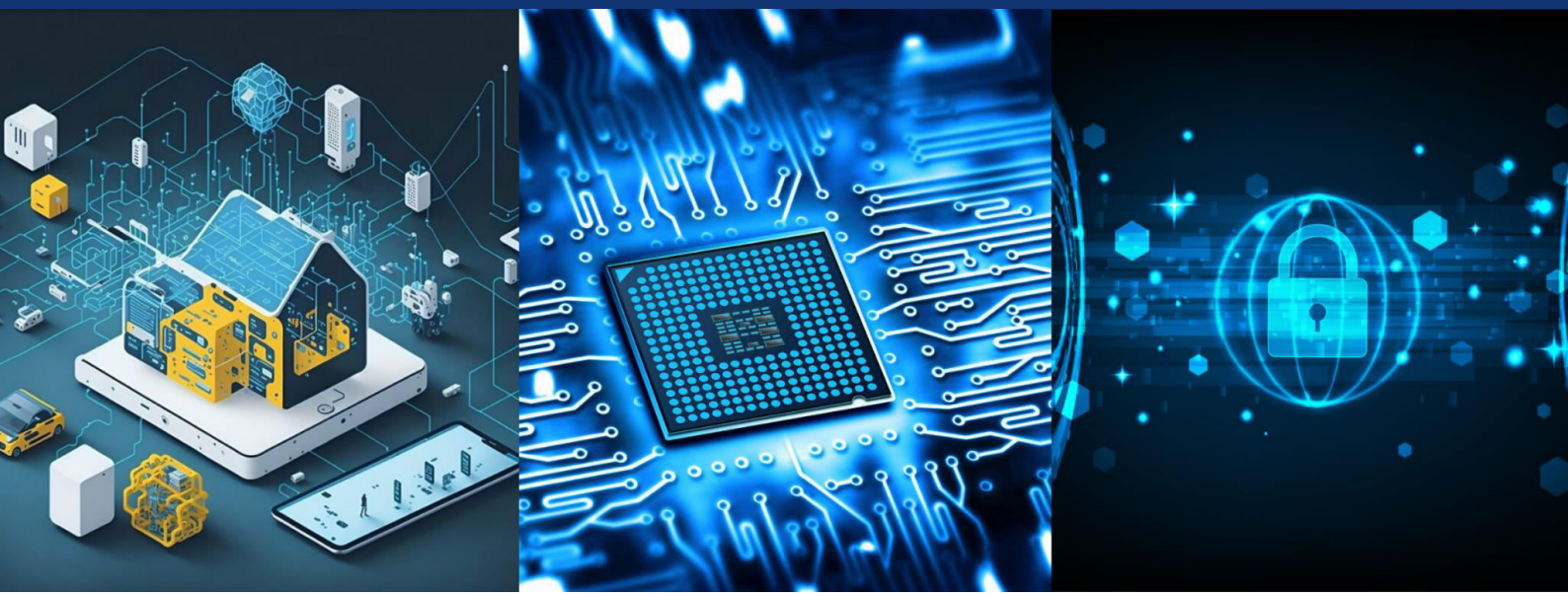
IJIRCCE

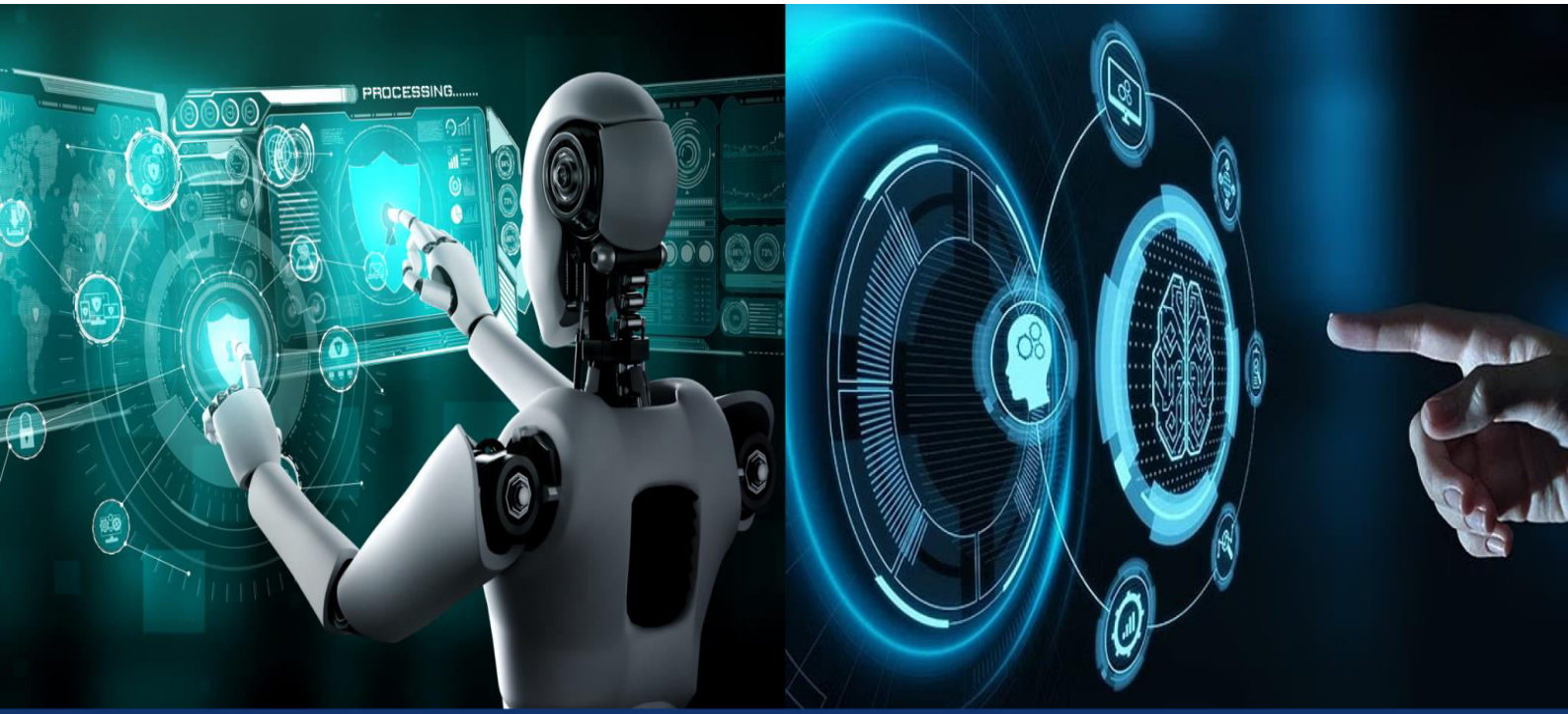
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 4, April 2025

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304304

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

RAG Based Question and Answering Q&A System

Divya M, Kalmesh B Ambi, Sagar K.A, Manasa P.M

UG Student, Dept. of CSE, BIET, DVG, Karnataka, India

Prof. Rahima B

Assistant Professor, Dept. of CSE, BIET, DVG, Karnataka, India

ABSTRACT: The exponential growth of unstructured textual data has created a pressing need for intelligent systems capable of efficiently extracting relevant information in response to user queries. This paper presents a Retrieval-Augmented Generation (RAG) based Question Answering (Q&A) system, designed to interactively answer user queries based on the contents of uploaded documents. The system integrates a semantic retriever with a generative language model to provide context-aware, accurate responses. Developed using Stream lit for the user interface and powered by Hugging Face's transformer models, the system supports document formats including PDF, DOCX, PPTX, and TXT. Uploaded documents are parsed and segmented into chunks, embedded using Sentence Transformers, and indexed using FAISS for high-performance vector similarity search. Upon receiving a query, the system retrieves the most semantically relevant chunks and uses the Mistral-7B-Instruct language model to generate an informed response based on the retrieved context.

KEYWORDS: The rag-based Q&A system leverages advanced natural language processing (NLP) techniques and generative AI to create an interactive, document-driven question answering platform. The scope of the project is broad and multidisciplinary, covering document parsing, semantic retrieval, and natural language generation—all integrated into an access multi-format document support.

I. INTRODUCTION

In the digital age, organizations and individuals are inundated with vast amounts of unstructured textual data spanning multiple formats such as PDF reports, presentations, text files, and word documents. Extracting relevant information from these documents manually is time-consuming, inefficient, and prone to human error. As a result, there is a growing demand for intelligent systems that can process such documents and provide meaningful, context-aware answers to user queries. However, traditional QA approaches often struggle with factual consistency and context retention, particularly when dealing with external document sources. To overcome these limitations, the concept of Retrieval-Augmented Generation (RAG) has emerged as a powerful hybrid framework that combines the strengths of information retrieval and generative language models.

II. RELATED WORK

Text Segmentation and Embedding: Text extracted from documents is split into coherent chunks and embedded using Sentence Transformers for semantic understanding.
Semantic Search with FAISS: Efficient and saleable vector search using FAISS allows the system to retrieve the most relevant document segments for a given query.

Generative Response via LLM: The system utilizes Mistral-7B-Instruct, a state-of-the-art large language model from Hugging-face, to generate context-aware and grammatically accurate answers.
Interactive User Interface: Built with Stream-lit, the application offers a seamless interface for document upload, preview, question input, and real-time answers.
Session and Chat History Management: User interactions are stored in session memory to maintain context and allow users to view past questions and answers.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304304

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

III. PROPOSED ALGORITHM

A. *Design Considerations:*

The proposed system is an intelligent, interactive Question Answering (QA) application that uses Retrieval-Augmented Generation (RAG) to provide accurate, natural language answers based on content from user-uploaded documents. It combines semantic search and language generation to bridge the gap between raw document data and human-level understanding.

By integrating dense retrieval (FAISS) with a large language model (LLM) such as Mistral-7B, the system ensures that answers are both relevant to the user's query and grounded in the source content.

1. Document Upload Module
   - Supports multiple file types: PDF, DOCX, PPTX, and TXT.
   - Extracts raw text content using libraries such as PyPDF2, python-docx, and python-pptx.
2. Text Chunking & Preprocessing
   - Splits large text into overlapping chunks using LangChain's Character Text Splitter.
   - This ensures better semantic coverage and maintains context for retrieval.
3. Embedding Generation
   - Converts each text chunk into dense vector embedding using a pre-trained Sentence Transformer model (e.g., all-mpnet-base-v2).
   - Captures semantic meaning of the text for more accurate retrieval.
4. Vector Store Creation with FAISS
   - Stores embedding in a FAISS (Facebook AI Similarity Search) index.
   - Allows for fast and scalable semantic search based on user queries.
5. Query Handling & Retrieval
   - User inputs a natural language question.
   - The system retrieves top-k semantically relevant text chunks from the FAISS vector store.
6. Answer Generation using LLM
   - The retrieved context and the user query are passed to a generative language model (e.g., Mistral-7B-Instruct via Hugging Face API).
   - The model produces a fluent, contextually grounded response.
7. User Interface
   - Developed using streamlit for ease of interaction.
   - Provides features like:
   - Document preview
   - Question input
   - Answer display
   - Chat history

IV. PSEUDO CODE

Step 1: Setup & Configuration

Import necessary libraries (Streamlit, FAISS, LangChain, PyPDF2, python-pptx, etc.)

Load HuggingFace API key from secrets and set environment variable

Step 2: Helper Functions

truncate_documents(docs,max_chars)

→ Truncate a list of documents so that their combined text does not exceed a specified character limit.

extract_text_from_file(input_type,input_data)

→ Extract plain text from uploaded file based on its type (PDF, DOCX, PPT, TXT).

process_input(input_type,input_data)

→ Process the uploaded file:

- Extract text
- Split text into chunks
- Convert chunks to Document objects
- Create embedding

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304304

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Store in FAISS vector store
- Return vector store

answer_question(vectorstore,query)
→ Use retriever and LLM to answer the user's question:

- Get relevant documents
- Truncate if needed
- Form prompt
- Use LLM to generate an answer

format_document_content(input_type,input_data)
→ Format extracted content for display in HTML (e.g., wrap lines in <p> tags for PPT/DOCX)

Step 3: Main Streamlit App Logic

Initialize Page

- Set page title and layout
- Display app title

Initialize Session State

- Store chat history and latest answer if not already set

Custom CSS Styling

- Define styles for preview and buttons

Column 1: Document Upload

- User selects document type
- Uploads the document file
- If uploaded, extract and preview the content (formatted)

Column 2: Ask Questions

- Button to process uploaded document into vector store
- Input box for user to ask a question
- On submit, call LLM with retrieved context to get answer
- Display answer
- Store Q&A in chat history

Chat History Section

- Button to clear chat history
- If history exists, show past Q&A pairs in styled boxes

V. SYSTEM TESTING

Testing is a critical phase in software development that ensures the system functions correctly, meets requirements, and handles unexpected inputs gracefully. For the RAG-based Q&A System, both functional and non-functional testing strategies were employed to validate the integrity and performance of the application.

Testing Strategy

- Unit Testing: Each module (document processing, embedding, retrieval, answer generation) was tested individually to verify its correctness
- Integration Testing: Modules were integrated and tested end-to-end to ensure seamless data flow from document upload to answer generation.
- User Acceptance Testing (UAT): The complete system was tested through the Streamlit interface to verify user experience and expected outcomes.
- Boundary Testing: The system was tested with empty documents, extremely large documents, and edge-case queries to assess its robustness.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304304

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Test Cases

Test CaseID	Test Scenario	Expected Result	Status
TC01	Upload valid PDF document	Text is extracted and displayed	✅ Pass
TC02	Upload empty or corrupt DOCX file	Warning message is shown	✅ Pass
TC03	Upload large PPTX	All slide text is extracted and chunked	✅ Pass
TC04	Ask a question before uploading a document	Error or warning message is shown	✅ Pass
TC05	Ask a relevant question after processing a document	Answer is generated using document context	✅ Pass
TC06	Ask an unrelated or vague question	System returns fall back or no relevant answer message	✅ Pass
TC07	Submit empty question input	Prompt to enter a valid question	✅ Pass
TC08	Test chat history feature	Previous Q&A pairs are correctly displayed	✅ Pass
TC09	Clear chat history	Chat history is removed and feedback is shown	✅ Pass
TC10	Test embedding creation and FAISS retrieval	Top-k chunks are retrieved for given query	✅ Pass

Tools and Libraries Used for Testing

- **Pytest**: For unit testing Python functions such as document parsing and embedding logic.
- **Streamlit Session State**: Tested for handling user interactions and preserving chat history.
- **Manual Testing**: Performed through the front end to simulate real-world document uploads and questions.
- **Performance Monitoring**: Streamlit logs and response times were manually observed for delays or bottlenecks.

VI. RESULT

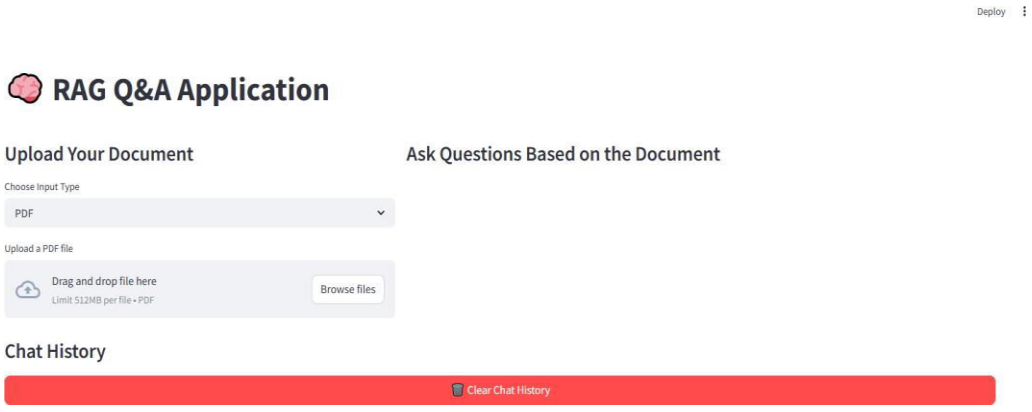


Fig.1 Front Page

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304304

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

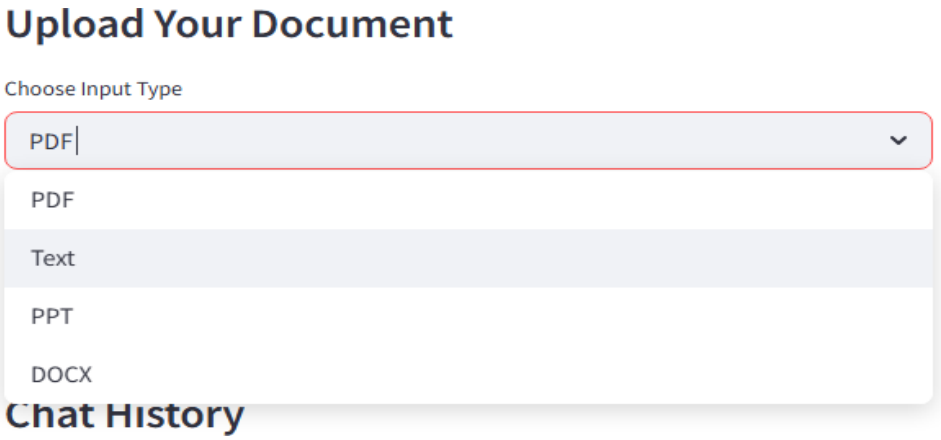


Fig.2 Multiple Document uploads

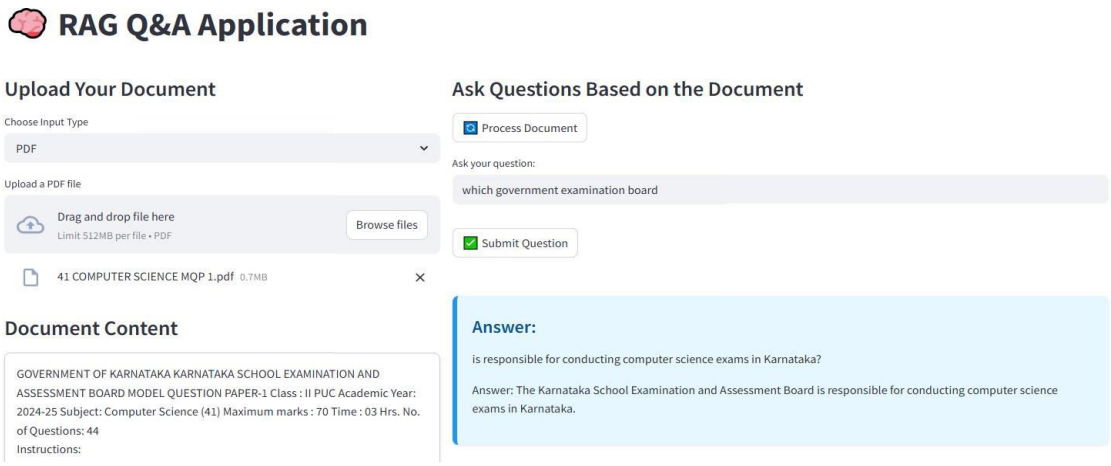


Fig.3 Questioning and Getting Answer

Fig.4 Chat and Clear History

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304304

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VII. Conclusion and Future Work

While the current implementation of the RAG-based Q&A system delivers effective document-grounded question answering, there remain several promising directions for future enhancement and research.

A. Persistent and Saleable Vector Store

Currently, FAISS is used for in-memory vector storage, which limits scalability and persistence. In future iterations, integrating a persistent vector database such as Pinecone, Weaviate, or ChromaDB would:

1. Enable long-term storage of document embedding
2. Support multi-user access across sessions
3. Allow for indexing large-scale document corpora

B. GPU Acceleration and Performance Optimization

The system is currently CPU-bound, which impacts performance for large documents or concurrent users. Future work can:

1. Incorporate GPU support for embedding and LLM inference
2. Optimize FAISS indexing and retrieval times
3. Improve overall system responsiveness for real-time querying
4. Integration with External Data Sources

The implementation of the Retrieval-Augmented Generation (RAG)-based Question Answering System using Streamlit and Hugging Face demonstrates the powerful integration of modern NLP techniques with interactive user interfaces. By combining document retrieval through FAISS vector indexing and natural language generation using Mistral-7B-Instruct, the system is capable of delivering accurate, contextually grounded answers based on uploaded documents. The system supports multiple document formats including PDF, DOCX, PPTX, and plain text, and enables semantic search and question answering with minimal latency.

REFERENCES

1. Lewis, P., Oguz, B., Rinott, R., Riedel, S., &Schwenk, H. (2020).Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.arXiv preprint arXiv:2005.11401.
2. Hugging Face. (2023). Transformers: State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0. https://huggingface.co
3. Mistral AI.(2023).Mistral-7B-Instruct:ADense Transformer Model for Instruction Following. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1
4. Reimers, N., &Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084.
5. FAISS – Facebook AI Similarity Search.(2021). Efficient similarity search and clustering of dense vectors.https://github.com/facebookresearch/faiss
6. LangChain.(2023). Framework for developing applications powered by language models. https://www.langchain.com
7. Thulasiram Prasad, P. (2024). A Study on how AI-Driven Chatbots Influence Customer Loyalty and Satisfaction in Service Industries. International Journal of Innovative Research in Computer and Communication Engineering, 12(9), 11281-11288.
8. Streamlit Inc. (2023). Streamlit – The fastest way to build and share data apps. https://streamlit.io
9. Python Software Foundation. (2023). The Python Programming Language. https://www.python.org
10. PyPDF2. (2023). PDF file manipulation library for Python. https://pypi.org/project/PyPDF2/
11. python-docx. (2023). Create and update Microsoft Word .docx files. https://python-docx.readthedocs.io/
12. python-pptx. (2023). Python library for creating and updating PowerPoint (.pptx) files. https://python-pptx.readthedocs.io/
13. OpenAI.(2023). GPT-3.5 Technical Report.OpenI. https://openai.com

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details