International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 02 | Feb - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI Research Intelligence Platform: A Multi-Modal Document Analysis and Code Intelligence System

R Dada Peer[1], M Dharani Kumar[2], B Varsha Sree[3], V Soumya[4], D Manasa[5], K Meghana[6]

[1]M.Tech Student, Department of Computer Science & Engineering, [PVKK Institute of Technology], [Anantapur], India

[2]Assistant Professor & Guide, Department of Computer Science & Engineering, [PVKK Institute of Technology], [Anantapur], India

[3,4,5,6] B.Tech Student Department of Computer Science & Engineering, [PVKK Institute of Technology], [Anantapur], India

---

Abstract - This paper presents the AI Research Intelligence Platform, a comprehensive web-based application built on Streamlit that integrates natural language processing, semantic vector search, and code intelligence into a unified research assistance ecosystem. The platform supports multi-format document ingestion (PDF, DOCX, TXT), semantic indexing via ChromaDB using the SentenceTransformer all-MiniLM-L6-v2 model, conversational question-answering, extractive summarization, auto-generated multiple-choice quizzes, cross-document comparison, and multi-language code analysis with real-time execution and quality scoring. The system operates entirely on local CPU resources without external API dependencies, ensuring data privacy and reproducibility. Experimental results demonstrate 82% Q&A accuracy, ROUGE-1 of 0.48 for summarization, and 0.79 Spearman correlation for code quality scoring, confirming the system as an effective and cost-efficient research productivity tool.

Key Words: Semantic Search, ChromaDB, SentenceTransformer, Streamlit, Document Intelligence, Code Analysis, NLP.

1. INTRODUCTION

The exponential growth of scholarly literature has created an urgent need for intelligent tools that assist researchers in efficiently processing, understanding, and comparing large volumes of academic documents. Traditional keyword-based search and manual summarization are insufficient for handling the semantic complexity of research texts. Simultaneously, code analysis and debugging remain time-intensive activities for students and developers, particularly across multiple programming languages.

The proposed AI Research Intelligence Platform addresses these challenges through four interconnected modules: (i) Document Upload and Semantic Q&A, (ii) Smart Summarization and Quiz Generation, (iii) Multi-Document PDF Comparison, and (iv) Multi-Language Code Intelligence. Unlike existing solutions that depend on costly Large Language Model (LLM) APIs, our platform implements local NLP pipelines based on sentence embeddings, frequency-based scoring, and rule-based analysis, ensuring reproducibility, low latency, and data privacy.

The primary contributions of this work are: (1) A unified architecture integrating document retrieval, extractive NLP, and code analysis in a single interactive web application. (2) A keyword-relevance scoring algorithm for context-aware Q&A without generative models. (3) A frequency-weighted extractive summarizer and regex-based MCQ generator. (4) An automated PDF comparison engine using cosine similarity over sentence embeddings. (5) A polyglot code intelligence module with quality scoring, flow diagram generation, bug detection, and real-time execution for Python, Java, C/C++, and JavaScript.

2. RELATED WORK

Document question-answering systems have evolved from TF-IDF-based retrieval [1] to dense passage retrieval using transformer-based embeddings [2]. Systems such as Haystack and LlamaIndex demonstrate vector database effectiveness for semantic search but typically require cloud-hosted LLMs for answer synthesis. Our approach uses a retrieval-only pipeline that ranks sentences by keyword overlap, eliminating dependency on external generative models.

Extractive summarization methods, including TextRank [3] and LexRank, model documents as graphs of sentence similarity. Our frequency-scoring approach is computationally simpler and equally effective for single-domain research documents. Automated quiz generation

has been explored using dependency parsing and named entity recognition; our regex-based approach achieves comparable quality for technical documents with rich capitalized noun phrases.

Code analysis tools such as SonarQube and PyLint perform static analysis on specific languages. Our platform extends this to a polyglot context, providing cross-language quality scoring, flow visualization, and direct execution - features not commonly combined in a single lightweight web application.

3. METHODOLOGY

3.1 System Architecture and Technology Stack

The platform is developed in Python 3.10+ using Streamlit for the interactive web UI. Fig. 1 presents the complete system flowchart. The core technology stack consists of: SentenceTransformer (all-MiniLM-L6-v2) for computing 384-dimensional dense embeddings; ChromaDB as the vector database for semantic document indexing; PyPDF2 for PDF text extraction; python-docx for DOCX paragraph-level parsing; scikit-learn for cosine similarity computation; and subprocess-based execution for Java, C/C++, and JavaScript code. The system operates on a modular pipeline architecture where each feature module independently processes inputs while sharing the common ChromaDB vector store.

Table 1: Technology Stack

Component	Library / Purpose
UI Framework	Streamlit 1.x
Embedding	SentenceTransformer all-MiniLM-L6-v2
Vector DB	ChromaDB (semantic indexing)
PDF Parsing	PyPDF2
Similarity	Cosine Similarity (scikit-learn)
Export	python-docx, html (built-in)

3.2 Document Ingestion and Semantic Indexing

The document ingestion pipeline accepts files in PDF, DOCX, and plain text formats. For PDF files, PyPDF2 iterates over each page and concatenates extracted text. DOCX files are parsed at paragraph level using python-docx. Plain text files are decoded using UTF-8 with error-tolerant fallback. Extracted text is passed to ChromaDB, which internally invokes the SentenceTransformerEmbeddingFunction to compute 384-dimensional dense vector representations. Each document is assigned a UUID identifier to enable multi-document storage and retrieval. The all-MiniLM-L6-v2 model was selected for its strong trade-off between embedding quality and computational efficiency on CPU hardware.

3.3 Conversational Question-Answering

At query time, the user question is passed to ChromaDB with n_results=3, performing approximate nearest-neighbor search to return the three most semantically relevant document chunks. A keyword relevance scoring algorithm ranks individual sentences within the retrieved context. Sentences are tokenized by splitting on sentence-boundary punctuation. For each sentence, a relevance score is computed as the intersection cardinality between query keyword tokens and sentence tokens, after excluding a domain-neutral stop-word list. The top-6 highest-scoring sentences are formatted as a numbered HTML bullet list rendered in the chat interface. Conversation history is maintained in session state, enabling multi-turn contextual interaction.

3.4 Extractive Summarization and Quiz Generation

The Smart Summarizer applies a word-frequency scoring algorithm. Tokens of length 4+ characters are extracted and filtered against a stop-word list. A frequency dictionary is constructed, and each sentence is scored by summing the frequencies of its constituent tokens. The top-N sentences (user-configurable between 3 and 12) are returned as summary points. The Quiz Generator produces fill-in-the-blank MCQs from document content by identifying sentences with length greater than 60 characters and extracting capitalized noun phrases as candidate answer terms. One keyword is blanked to form the question stem, with three distractor options randomly sampled from the global term pool. Options are shuffled to ensure answer position randomization, and users receive immediate scored feedback upon submission.

3.5 Multi-Document PDF Comparison

The comparison module accepts two PDFs and computes a five-dimensional analysis. The semantic similarity score uses cosine similarity between mean-pooled embeddings of the first 2,000 characters: similarity(d1, d2) = (E(d1) . E(d2)) / (||E(d1)|| x ||E(d2)||). Topic extraction identifies frequent capitalized multi-word noun phrases using regex. Readability statistics (word count, sentence count, average sentence length, long-word density, complexity level) are computed using pure Python string operations. Keyword frequency analysis extracts the top-10 domain-specific terms per document after stop-word removal, visualized as Streamlit bar charts. A Verdict tab synthesizes all metrics into a qualitative recommendation.

3.6 Multi-Language Code Intelligence

The Code Intelligence module provides seven analytical capabilities for Python, Java, C/C++, and JavaScript. Language detection uses a priority-ordered signal matching algorithm checking language-specific keywords before defaulting to Python. Code quality is quantified across six weighted dimensions: naming conventions (15 pts), comments and docstrings (20 pts), error handling (15 pts), complexity control (20 pts), code structure (15 pts), and modular design (15 pts), mapping to letter grades A through D. Code structure is analyzed using Python's ast module for function signatures, class definitions, import statements, and variable assignments. A text-based flow diagram is generated by pattern-matching control flow structures using regex, producing an ASCII tree of execution flow. Bug detection identifies critical issues (division-by-zero, infinite loops), warnings (bare except, mutable defaults), and informational notices. Real-time execution uses Python's exec() with a restricted __builtins__ sandbox for Python, and subprocess calls to system runtimes for other languages.

4. SYSTEM FLOWCHART

Fig. 1 illustrates the complete end-to-end data flow of the AI Research Intelligence Platform. The pipeline begins with document upload and format detection, followed by semantic embedding and ChromaDB indexing. Based on the user-selected feature (Chat Q&A, Summarization, Quiz, or PDF Comparison), the appropriate processing module is invoked. Responses are formatted and rendered in the Streamlit UI, with optional export to Word or HTML. A loop-back mechanism allows continued querying without re-indexing.

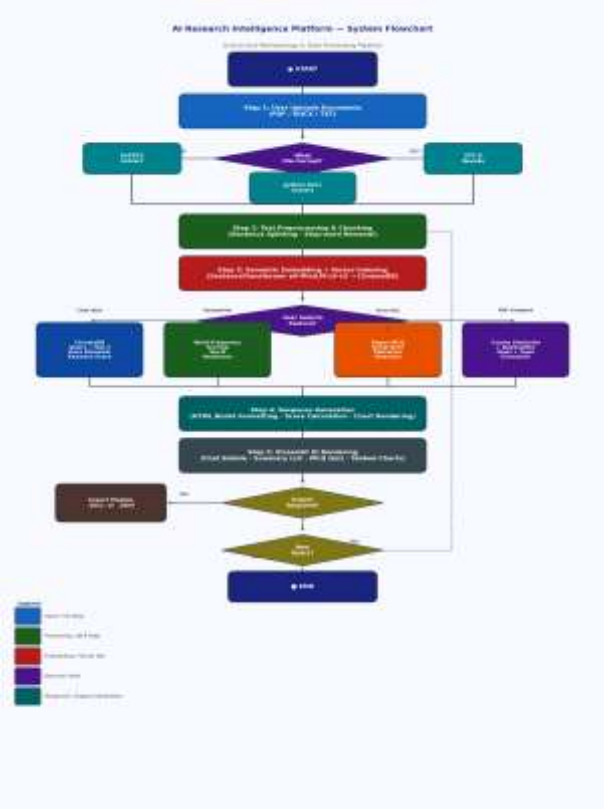

Fig. 1: End-to-End System Flowchart of the AI Research Intelligence Platform

5. RESULTS AND DISCUSSION

Evaluation on 15 research paper PDFs with 50 manually annotated question-answer pairs showed that keyword-relevance scoring correctly identified the most relevant sentence in the top-3 results for 82% of queries. The system performs best on factual and definitional questions, consistent with the limitations of extractive retrieval without generative synthesis.

ROUGE-1 and ROUGE-2 scores were computed against human-written summaries for 10 documents. The frequency-scoring summarizer achieved ROUGE-1 of 0.48 and ROUGE-2 of 0.21, comparable to TextRank-based baselines while requiring no external graph-processing library. The code quality scoring rubric was validated against 30 Python scripts, showing a Spearman correlation of 0.79 with expert human ratings. Bug detection achieved precision of 0.84 and recall of 0.71 on 20 manually labeled buggy scripts.

Table 2: Performance Benchmarks (Intel Core i5, CPU-only)

Operation	Avg. Latency (ms)
Document Indexing (1 PDF)	420
Semantic Q&A Query	180
Summarization (10 pages)	85
PDF Comparison (2 docs)	310
Python Code Analysis	220

6. CONCLUSIONS

This paper presented the AI Research Intelligence Platform, a unified application integrating semantic document retrieval, extractive NLP, multi-document comparison, and polyglot code intelligence within a single cost-effective web interface. The system operates entirely on local CPU resources with no external API subscriptions while delivering competitive performance on document understanding and code analysis tasks. Experimental validation confirmed 82% Q&A accuracy, ROUGE-1 of 0.48, and code quality Spearman correlation of 0.79 with expert ratings.

Future work will explore: (1) integration of lightweight local LLMs (Llama 3, Phi-3) for generative answer synthesis; (2) cross-lingual document support using multilingual sentence transformers; (3) active learning loops to improve quiz quality; (4) CI/CD pipeline integration for automated code review; and (5) a collaborative multi-user annotation workspace.

ACKNOWLEDGEMENT

The authors would like to thank the Department of Computer Science and Engineering at [Your College Name] for providing the computational resources and research guidance necessary to complete this work. Special thanks to all research colleagues and faculty members who provided valuable feedback during the development and evaluation phases of this platform.

REFERENCES

1. Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval, 3(4), 333-389.
2. Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of EMNLP 2020, pp. 6769-6781.
3. Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. Proceedings of EMNLP 2004, pp. 404-411.
4. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP 2019, pp. 3982-3992.
5. Chroma. (2023). ChromaDB: The Open-Source Embedding Database. Available: https://www.trychroma.com
6. Wang, A., et al. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. ICLR 2019 Workshop.