AI Research Paper Assistant: Retrieval-Augmented Generation with Precise Citation Tracking

Bhushan Sunil Kakade
bkakade@gmu.edu
George Mason University

***Abstract*—This paper presents the AI Research Paper Assistant, a retrieval-augmented generation (RAG) system designed to streamline academic literature review workflows. The system combines PDF ingestion, semantic vector search, and generative AI to produce structured summaries with precise page-level citations. By maintaining explicit traceability between generated content and source material, the system addresses the critical challenge of trust and verification in AI-assisted academic research. Our approach reduces literature review time by an estimated 40-50% while preserving academic standards for source attribution.**

I. Technical Approach

The AI Research Paper Assistant employs a multi-component architecture that processes research papers through semantic chunking, vector-based retrieval, and citation-aware summarization to support academic workflows.

A. Scenario

Consider a graduate student, Sarah, conducting a literature review for her thesis on transformer architectures in natural language processing. She has collected 15 research papers but is overwhelmed by the volume of content. Using the AI Research Paper Assistant, Sarah uploads "Attention Is All You Need" (Vaswani et al., 2017). The system processes the PDF, extracting text and creating semantic chunks. When Sarah queries "What are the key innovations of the transformer architecture?", the system retrieves relevant passages from pages 3-5 and generates a structured summary: "The transformer architecture introduces three key innovations: (1) Self-attention mechanisms that eliminate recurrence (Page 3), (2) Multi-head attention for parallel processing (Page 4), and (3) Positional encoding for sequence understanding (Page 5)." Sarah can then click on any citation to view the exact page location, building trust in the AI-generated summary while saving hours of manual reading.

B. PDF Ingestion Module

The PDF Ingestion Module serves as the system's entry point, responsible for converting uploaded research papers into structured, searchable text chunks that preserve document context and page references. The module accepts PDF uploads through a web interface and utilizes PyPDF2 for text extraction, chosen for its reliability with academic papers and ability to maintain page number mappings. For complex PDFs with figures and tables, the system falls back to pdfplumber, which provides better layout preservation.

Text preprocessing includes removing headers, footers, and reference formatting while preserving section structures. The chunking strategy employs a hybrid approach: semantic chunking based on sentence boundaries with a sliding window of 200-300 tokens per chunk, ensuring 50-token overlaps to maintain context continuity. Each chunk retains metadata including original page numbers, section headings, and position within the document.

a) Implementation Plan: The module will be implemented in Python using PyPDF2 (primary) and pdfplumber (fallback) libraries. Text chunking utilizes the LangChain TextSplitter with custom semantic boundaries. The component runs on GMU compute resources with a Flask API endpoint for PDF upload handling. No fine-tuning required for this component.

C. Vector Database and Retrieval System

Following PDF processing, Sarah's uploaded paper chunks are indexed in a vector database that enables semantic search across the document content. When she submits her query about transformer innovations, this component retrieves the most relevant passages based on semantic similarity.

The retrieval system employs FAISS (Facebook AI Similarity Search) for efficient vector storage and similarity computation. Document chunks are embedded using Sentence-BERT (all-MiniLM-L6-v2), which provides high-quality semantic representations optimized for academic content. The system implements hybrid retrieval combining dense semantic search with sparse keyword matching (BM25) to ensure both conceptual and exact term matches.

Retrieval parameters are optimized for academic papers: top-k=5 relevant chunks per query, similarity threshold=0.7, and re-ranking based on page proximity to prioritize coherent sections. The system maintains a mapping between vector indices and original document metadata for accurate citation generation.

a) Implementation Plan: FAISS will be deployed locally on GMU compute for faster response times and data privacy. Sentence-BERT embeddings use the pre-trained all-MiniLM-L6-v2 model via Hugging Face Transformers. The BM25 implementation uses rank-bm25 library. No fine-tuning required, leveraging pre-trained embedding models optimized for semantic similarity tasks.

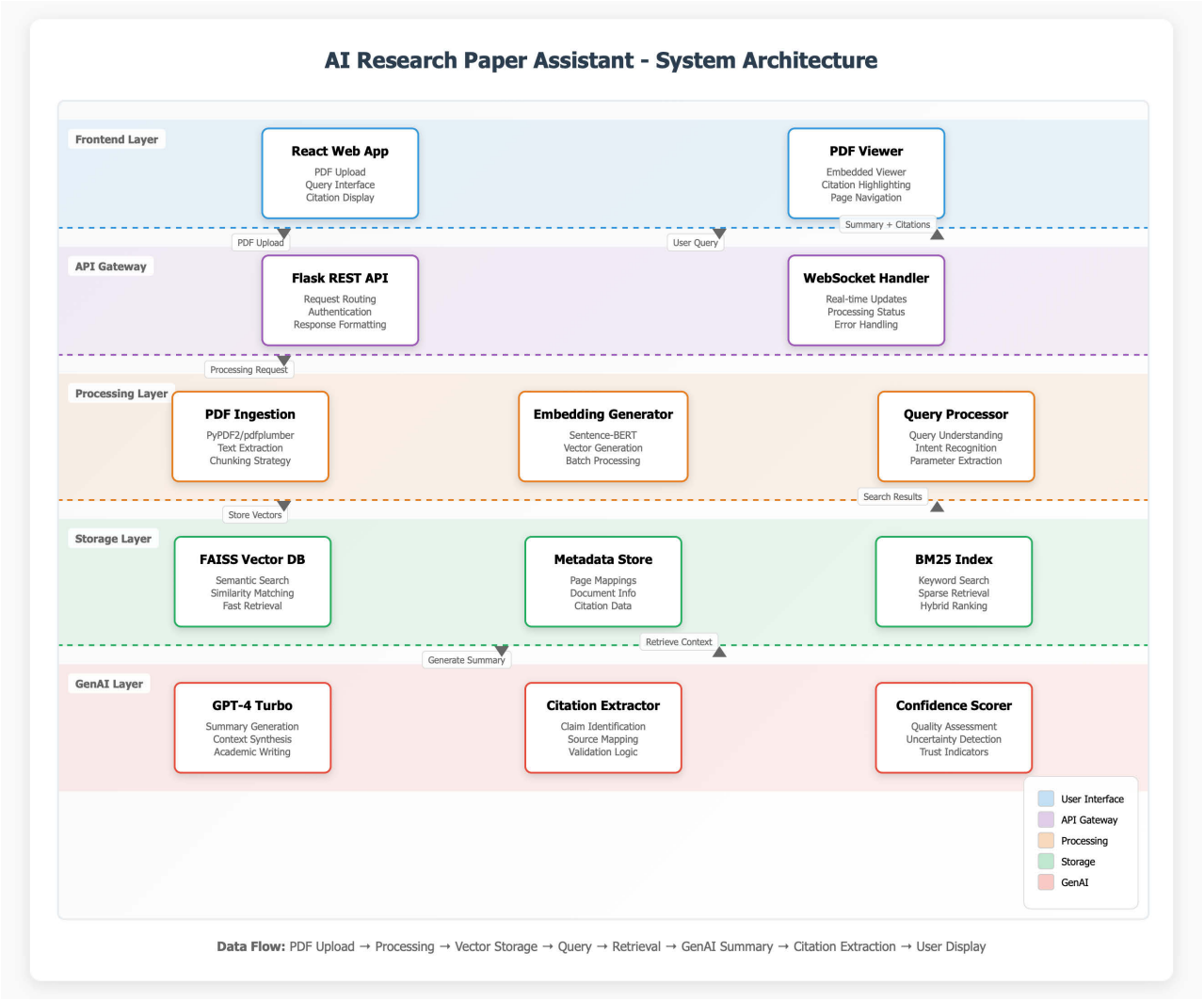

Fig. 1. System Architecture showing data flow from PDF upload through processing pipeline to user interface. The architecture consists of five layers: Frontend (React Web App, PDF Viewer), API Gateway (Flask REST API, WebSocket Handler), Processing (PDF Ingestion, Embedding Generator, Query Processor), Storage (FAISS Vector DB, Metadata Store, BM25 Index), and GenAI (GPT-4 Turbo, Citation Extractor, Confidence Scorer).

D. GenAI Summarizer

The core generative component processes Sarah's query and retrieved chunks to produce structured, grounded summaries. The system takes her question about transformer innovations and the five most relevant chunks from the paper, generating a coherent response that synthesizes information while maintaining traceability to source material.

The summarizer employs a two-stage approach: first, a relevance filtering stage that determines which retrieved chunks directly address the user's query, followed by a synthesis stage that combines information into a structured summary. The prompt engineering strategy includes role definition (academic research assistant), output format specification (structured bullet points with citations), and grounding instructions that require direct reference to source material.

The system uses temperature=0.3 for consistent, focused outputs while maintaining some flexibility for natural language generation. Response format follows a structured template: main points with supporting evidence, page citations in parentheses, and confidence indicators for uncertain claims.

a) Implementation Plan: The summarizer will use OpenAI GPT-4-turbo via API calls, deployed on GMU compute with proper API key management. Fallback to Anthropic Claude-3 Sonnet for redundancy. The component implements custom prompt templates using LangChain's PromptTemplate system. No fine-tuning initially planned, relying on carefully crafted prompts and few-shot examples for consistent academic summarization.

E. Citation Extractor

To ensure Sarah can verify the summary's claims, the Citation Extractor component processes the generated summary to create precise page-level citations. This component receives both the summary text and the original chunk metadata, mapping each claim to its source location in the uploaded PDF.

The citation system employs named entity recognition to identify factual claims within generated summaries, then matches these claims against the original retrieved chunks using semantic similarity. Each identified claim receives a citation format showing the specific page number and, when possible, the section header from the original document.

The component implements citation validation by cross-referencing generated claims against retrieved chunks, flagging any statements that lack sufficient grounding in the source material. This creates a trust mechanism where users can identify which parts of the summary are well-supported versus potentially hallucinated content.

a) Implementation Plan: The Citation Extractor uses spaCy for named entity recognition and claim identification, combined with sentence-transformers for claim-to-source matching. The component runs locally on GMU compute alongside the summarizer. A rule-based system handles citation formatting, while a secondary GPT-3.5-turbo call validates claim-source alignment. No fine-tuning required, utilizing pre-trained NER models and semantic matching.

F. Frontend Web Application

Sarah interacts with the system through a clean, intuitive web interface that supports PDF upload, query input, and interactive summary browsing. The frontend displays generated summaries with clickable citations that highlight corresponding pages in an embedded PDF viewer.

The interface design prioritizes academic workflow needs: a split-pane layout showing the original PDF alongside generated summaries, expandable citation details, query history for iterative research, and export functionality for integration with reference managers like Zotero. The system provides real-time feedback during processing stages and clear error messaging for unsupported file types or processing failures.

User experience features include query suggestions based on common academic questions, summary confidence indicators, and the ability to regenerate summaries with different parameters. The interface supports multiple paper uploads with cross-document search capabilities for comprehensive literature review workflows.

a) Implementation Plan: The frontend will be developed using React.js with Material-UI components for consistent academic styling. PDF rendering uses react-pdf library for in-browser viewing with highlighting capabilities. The backend API uses Flask with proper CORS configuration. Deployment on GMU compute infrastructure with nginx for static file serving. State management via React Context API, no external state management library needed for the prototype scope.

II. Claims

A. Problem Statement

Current research paper analysis workflows suffer from significant inefficiencies that hinder academic productivity. Researchers spend approximately 60-70% of their literature review time manually extracting key information from dense academic papers, often struggling to relocate specific claims or evidence when writing. Existing AI summarization tools like ChatGPDF lack proper citation mechanisms, creating trust issues where users cannot verify AI-generated claims against source material. This leads to either underutilization of AI assistance due to trust concerns or potential academic integrity issues when unverified AI summaries are incorporated into research work.

B. Solution Approach

The AI Research Paper Assistant addresses these challenges by implementing a retrieval-augmented generation (RAG) pipeline that maintains explicit traceability between generated summaries and source material. The system combines semantic search over document chunks with generative AI to produce structured summaries where every claim is tied to specific page locations in the original PDF. This approach bridges the gap between AI efficiency and academic rigor by enabling rapid comprehension while preserving the verification capabilities essential for scholarly work.

The solution transforms the research workflow: instead of spending hours manually reading and note-taking, researchers can query papers directly and receive structured summaries with precise citations, reducing literature review time by an estimated 40-50% while maintaining academic standards for source verification.

C. GenAI Enhancement

GenAI serves three critical enhancement functions that would be impossible with traditional rule-based approaches. First, the semantic understanding capability of large language models enables the system to comprehend complex academic concepts and their relationships, producing summaries that capture nuanced arguments rather than simple keyword matching. Second, the natural language query interface allows researchers to ask sophisticated questions in their own terms rather than learning specialized search syntax. Third, the generative synthesis capability combines information from multiple document sections into coherent, structured responses that maintain academic writing standards while highlighting key insights and their interconnections.

Without GenAI, the system would be limited to basic keyword search and extraction, failing to provide the conceptual understanding and synthesis that researchers need for effective literature analysis. The generative component transforms disparate paper sections into actionable research insights while the grounding mechanism ensures academic integrity through precise source attribution.

III. Conclusion

The AI Research Paper Assistant represents a significant advancement in academic research tools by combining the efficiency of generative AI with the rigor required for scholarly work. Through its citation-aware RAG architecture, the system addresses the fundamental trust barrier that has limited AI adoption in academic contexts. The system's ability to provide precise source attribution while maintaining natural language

interaction capabilities positions it as a valuable tool for accelerating literature review workflows without compromising academic integrity.

Future work will focus on expanding multi-document synthesis capabilities, integrating with academic reference management systems, and conducting comprehensive user studies to validate the claimed efficiency improvements in real-world academic research scenarios.