International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: https://ijwos.com OPEN ACCESS E-ISSN : 3049-2424
Volume 3 Issue 9, September 2026 DOI: https://doi.org/10.71366/ijwos

IRPA: A Retrieval-Augmented Framework for Intelligent Research Paper Analysis and Research Gap Detection

Varun K S[1], Anusha N[2]
[1] *Assistant Professor, Department of Master of Computer Applications, GM University, Davangere, India*
[2] *Student, Department of Master of Computer Applications, GM University, Davangere, India.*

Article Info

Article History:

Published: 06 Sept 2026

Publication Issue:
Volume 3, Issue 9
September-2026

Page Number:
42-54

Corresponding Author:
Anusha N

Abstract:

Manual literature review has become harder to sustain as the volume of published research keeps growing. A student or researcher preparing a review still has to open one PDF after another, work out what each paper is trying to do, note the methodology and findings, and keep track of where the gaps are — a process that is slow and easy to get wrong once the paper count rises. Search engines and citation databases help with finding papers, but they stop short of reading a paper on the user's behalf, and generic summarizers tend to flatten a paper into a shallow, abstract-level paragraph that drops the detail a reviewer actually needs. This paper describes the Intelligent Research Paper Analyzer (IRPA), a system built around Retrieval-Augmented Generation (RAG) that analyzes an uploaded academic PDF end to end. The pipeline splits a paper into sections, encodes each one with a sentence-transformer model, and stores the resulting embeddings in a vector database so relevant sections can be retrieved on demand. When a user asks a question or requests a summary, the retrieved sections are passed to a large language model together with a grounding prompt, which is intended to keep the generated answer tied to content that actually appears in the paper rather than left to the model's own recall. The same pipeline extracts objectives, methodology, key findings, limitations, and citation information in a structured form, and flags limitations that recur across a user's uploaded papers as candidate research gaps. The current work focuses on the design, implementation, and functional validation of the proposed prototype rather than a large-scale comparative benchmark evaluation. The system demonstrates the feasibility of integrating retrieval grounding, structured information extraction, citation analysis, and research-gap identification within a unified research-paper analysis workflow.

Keywords: Research Paper Analysis, Retrieval-Augmented Generation, Semantic Embeddings, Vector Database, Sentence Transformers, Large Language Models, Citation Analysis, Research Gap Detection

1. INTRODUCTION

Scholarly output keeps growing every year, to the point where no single researcher can realistically keep up with everything relevant to their field. Putting together a literature review still typically means opening one PDF after another, working out the objective, method, dataset, and limitations of each, and reconciling the fact that different venues describe similar ideas using different terminology. A paper that runs to twenty pages or more does not make this easier, and the details a reviewer needs are rarely gathered in one place — they are scattered across the introduction, methodology, and discussion sections.

This is a genuine bottleneck in academic work, not just an inconvenience. Some existing tools chip away at parts of it: citation-graph-based summarization [1], [2] uses the context around a citing sentence to improve a summary, and RAG-based scholarly assistants [3], [7] ground their answers in retrieved passages instead of relying on a model's own

memory. What is less common is a single pipeline that brings retrieval-grounded summarization, structured extraction, citation analysis, and gap detection together in one place.

Academic search platforms and open citation graphs [4] are good at finding papers and surfacing metadata, but they were not built to read a paper on the user's behalf. General-purpose summarizers usually produce an abstract-level paragraph that strips out the objective and limitation details a reviewer actually needs, and a conversational assistant that is not grounded in the source document can end up describing things that simply are not in the paper.

IRPA is an attempt to close that gap. A user uploads a PDF through a web interface; the system extracts and cleans the text, breaks it into sections, and encodes each section with a sentence-transformer model [8]. The resulting embeddings sit in a vector database, which supports top-k retrieval whenever the user asks something. Retrieved sections are handed to a large language model — a class of system capable of adapting to a new task from a prompt alone, without additional training [16] — along with a prompt built to keep the answer grounded in that retrieved text [7], so the response stays traceable back to the paper rather than drifting into the model's own general knowledge.

None of this rests on a new transformer architecture. The contribution here is in how retrieval grounding, section-level search, and structured extraction are combined into one lightweight pipeline that a student can realistically run without specialized hardware.

Main Contributions of This Work

1. A single pipeline that takes a PDF from upload through ingestion, preprocessing, and semantic retrieval without needing separate tools for each step.
2. Retrieval performed at the section level using sentence-transformer embeddings in a vector index, rather than matching on individual sentences or raw keywords.
3. A RAG-based generation step that ties the language model's answer to retrieved section content, aimed at reducing hallucinated output.
4. A consistent extraction schema — objective, methodology, findings, limitations, and future work — pulled from the uploaded paper rather than left as unstructured text.
5. A citation-analysis component that reads in-text citations and reference entries from the source document and organizes them.
6. A gap-detection step that looks for limitation statements repeated across a user's uploaded papers and surfaces them as candidate directions worth pursuing.

Section II looks at related work on document summarization, retrieval-augmented generation, and existing paper-analysis tools. Section III lays out the research gap and problem statement. Section IV walks through the proposed methodology, and Section V covers the system architecture. Section VI gives the algorithm and the underlying mathematical model. Section VII describes the implementation and prototype validation, Section VIII discusses system functionality and results, Section IX covers limitations and future work, and Section X concludes the paper.

2. RELATED WORK

A. Scientific Document Summarization

Work on summarizing scientific documents has moved from extractive, statistics-based methods toward transformer-based abstractive and hybrid approaches. Citation-graph methods bring in the context around a citing sentence to make a summary more informative: An et al. [1] pair a BiLSTM with graph attention over the citation graph and report better ROUGE scores than BERTSUM-style baselines, while Cohan and Goharian [2] use citation contextualization and discourse facets to improve extractive summarization on biomedical text. TLDR-style single-sentence summarization [10] pushes this further by compressing a paper down to one line. Domain-adapted encoders such as SciBERT [6] and long-document transformers such as BigBird [9] raise accuracy further still, though both come with

a heavier compute cost. BERTSUM [12] remains a common extractive/abstractive baseline for general summarization tasks.

B. Retrieval-Augmented Generation

Retrieval-Augmented Generation, as introduced by Lewis et al. [7], ties a sequence-to-sequence generator to passages pulled from an external corpus, which tends to make the output more factually reliable than a model relying purely on its own parameters. Dense retrieval in a RAG pipeline usually runs on sentence-level embeddings such as those from Sentence-BERT [8], built on the transformer encoder [13], [14], and supports semantic similarity search that does not depend on exact keyword overlap — unlike sparse retrieval methods such as BM25 [20] or earlier static embeddings such as word2vec [18] and GloVe [19]. A few recent scholarly assistants apply this idea directly to research papers: Baleswari et al. [3] pair a vector index with a large language model to answer questions across multiple documents, though without adapting to different reader levels or benchmarking against standard summarization metrics.

C. Academic Research Paper Analysis Systems

Large infrastructures such as the Semantic Scholar Open Data Platform [4] index hundreds of millions of papers using SPECTER-style embeddings [11] and offer search, citation graphs, and short auto-generated summaries, but they operate as a shared platform rather than something tailored to a single user's paper. Reader-adaptive summarizers, such as the section-wise transformer summarizer from Hari Krishna et al. [5], generate summaries pitched at different reading levels, but without citation-graph analysis, and without retrieval grounding they remain open to hallucination.

D. Comparative Analysis

Table I lists twelve papers reviewed in the course of this work, along with their methods, findings, and limitations.

TABLE I. COMPARATIVE LITERATURE REVIEW

Author	Year	Method	Findings	Limitations
An et al. [1]	2021	CGSum: BiLSTM + graph attention on citation graph	Outperforms BERTSUM on ROUGE using citation context	Uses only abstracts of neighbours; single-hop
Cohan & Goharian [2]	2018	Citation contextualization + discourse facets	Improves ROUGE-2/3 on TAC BiomedSumm	Cold-start for new papers; extractive only
Baleswari et al. [3]	2026	RAG with vector index + LLM answers	High top-k retrieval; multi-document summarization	No reader-level adaptation; no ROUGE benchmarking
Kinney et al. [4]	2023	Semantic Scholar S2AG + SPECTER embeddings	Open graph of 225M+ papers powers analyzers	Platform only; no user-tailored answers

Author	Year	Method	Findings	Limitations
Hari Krishna et al. [5]	2026	Section-wise transformer summarizer, 3 reader levels	Audience-specific summaries; readability control	No citation graph; hallucination risk
Beltagy et al. [6]	2020	SciBERT pre-trained on scientific corpus	Better domain accuracy than BERT	Not tuned for long documents
Lewis et al. [7]	2020	Original RAG (dense retriever + seq2seq)	Grounds generation in external knowledge	Retrieval quality dominates final answer
Reimers & Gurevych [8]	2019	Sentence-BERT (SBERT) embeddings	Fast semantic similarity for retrieval	Sentence-level only; loses long context
Zaheer et al. [9]	2020	BigBird sparse attention	Handles long scientific documents	Heavy GPU footprint
Cachola et al. [10]	2020	TLDR extreme summarization	One-sentence paper summaries	Loses methodology details
Cohan et al. [11]	2020	SPECTER document embeddings	Improves recommendation and clustering	Requires citation supervision
Liu & Lapata [12]	2019	BERTSUM extractive/abstractive	Strong baseline for summarization	Not grounded in retrieval

Taken together, this body of work covers retrieval, summarization, and embeddings fairly well, but each system tends to stick to one sub-problem. Citation-graph models [1], [2] add useful context but need a dense reference network that a newly published paper does not yet have. Large infrastructures [4], [11] generate embeddings and short summaries at scale but do not answer a specific question about a specific paper. RAG-based assistants [3], [7] ground their answers in the source text but usually leave citation structure out of the picture. Reader-adaptive summarizers [5] adjust readability without any explicit check on factual accuracy. Long-document transformers [9] and domain-specific encoders [6], [11] improve accuracy at the cost of real GPU resources. None of these combine retrieval grounding, section-level summarization, limitation/gap extraction, and citation analysis in one lightweight system — the gap that Section III builds on.

3. RESEARCH GAP AND PROBLEM FORMULATION

The pattern across Section II is one of partial automation — retrieval, summarization, citation analysis, and reader adaptation each get solved on their own, but rarely together, and rarely in a form light enough to run without a dedicated GPU.

That gap shows up in practical terms. Pulling objectives, methods, results, limitations, and citations out of each paper by hand takes time and is easy to get wrong, especially once a review covers more than a handful of papers. Digital libraries and citation databases are good for search and metadata but were never meant to read the paper itself. Free summarizers tend to strip away the specific detail a reviewer needs, and a chat assistant that is not grounded in the source document can produce something plausible-sounding that is simply not in the paper. Citation analysis and content analysis usually live in separate tools, and most existing options assume access to a GPU or a paid API — not realistic for a typical student setup. Gap identification, in particular, is still mostly a manual, comparative exercise with little tool support.

IRPA is built to address this directly. Section-level semantic retrieval combined with RAG keeps the system's output tied to content that is actually in the source paper, which should reduce the risk of the model inventing a finding. Structuring the output into a fixed schema — objective, methodology, dataset, findings, limitations, future work, citations — turns an unstructured PDF into something that can go straight into a literature-review chapter. Clustering limitation statements across a user's uploaded papers lets the system point to gaps that show up more than once, which is a reasonable proxy for a promising research direction.

4. PROPOSED METHODOLOGY

The pipeline covers PDF ingestion, preprocessing, section segmentation, embedding, retrieval, RAG-based generation, and structured post-processing. Each stage is described below.

A. Overall Workflow

Fig. 1 shows how a PDF moves through the system, from upload to the final structured report. Each stage hands a specific artefact to the next one — raw text, cleaned tokens, segmented sections, section vectors, retrieved chunks, and finally the structured output.

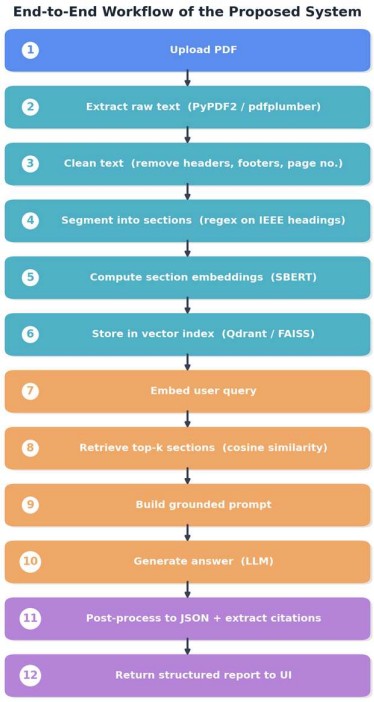

Fig. 1. End-to-end workflow of the proposed IRPA system.

B. PDF Ingestion and Text Extraction

Text is pulled from the uploaded PDF using pdfplumber, which keeps layout information intact for single- and multi-column papers; PyPDF2 steps in as a fallback when layout-aware extraction does not work on a given page.

C. Text Preprocessing

Headers, footers, and page numbers are stripped out with regular expressions, and whitespace is normalized. spaCy then handles sentence segmentation — a standard step in most NLP pipelines [21] — so the text is ready to be split into sections.

D. Section Segmentation

Sections such as Introduction, Methodology, Results, and Limitations are identified using regex heuristics built around common IEEE heading patterns. Keeping this section-level structure matters, since the retrieval step later depends on it.

E. Semantic Embedding Generation

Each section is encoded into a dense vector using a sentence-transformer model — specifically all-MiniLM-L6-v2 [8], built on the transformer encoder [13] — producing embeddings suited to comparing sections by meaning rather than by exact wording.

F. Vector Indexing and Knowledge Retrieval

Section embeddings are stored in a vector database that supports fast approximate nearest-neighbour lookup, with Qdrant used as the primary index and FAISS available as a local fallback. This dense approach was chosen over sparse methods such as BM25 [20] because it captures meaning rather than plain keyword overlap.

G. Retrieval-Augmented Generation

A user's query is embedded with the same sentence-transformer model, and the top-k most similar sections are retrieved by cosine similarity (Section VI). These sections are combined with the query in a structured prompt and passed to a large language model, following the general RAG approach set out by Lewis et al. [7], so the generated answer stays grounded in retrieved text.

H. Structured Research Information Extraction

The prompt is written so the model returns an eight-field output — title, objective, method, dataset, results, limitations, future work, and top citations — so later stages can work with structured data instead of parsing free text.

I. Citation Analysis

In-text citations, whether numbered ([n]) or author-year style, are parsed and matched against a local reference table built from the paper, letting the system identify which references get cited most often within the document.

J. Research Gap Detection

A lightweight clustering step groups the embeddings of limitation-bearing sections across a user's uploaded papers. Limitations that keep showing up across multiple papers are surfaced as candidate gaps, giving the reader a starting point for further work without needing a separate gap-analysis tool.

5. SYSTEM ARCHITECTURE

Fig. 2 lays out the architecture in layers. A user uploads a PDF through the web interface; the PDF Upload module hands the file to Text Extraction and Preprocessing, which produces the cleaned text used for Section Segmentation. The Sentence Transformer component encodes each section into an embedding, stored in the vector database. When the user makes a request, the Retrieval Engine runs a top-k semantic search over that index, and the retrieved context

goes to the LLM / RAG Engine, which produces the structured output — summary, objective, methodology, dataset, findings, limitations, future work, citations, and gaps, depending on what was asked for.

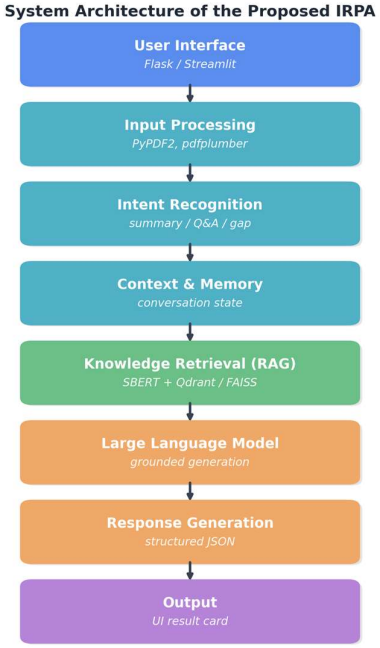

Fig. 2. Overall architecture of the proposed IRPA system.

Fig. 3 gives a module-level view of the same system: the ingestion-side modules (PDF ingestion, preprocessing, summarizer) and the retrieval-side modules (RAG retriever, citation analyzer, trend-and-gap miner) both connect into a shared core engine that handles request routing and assembles the final response.

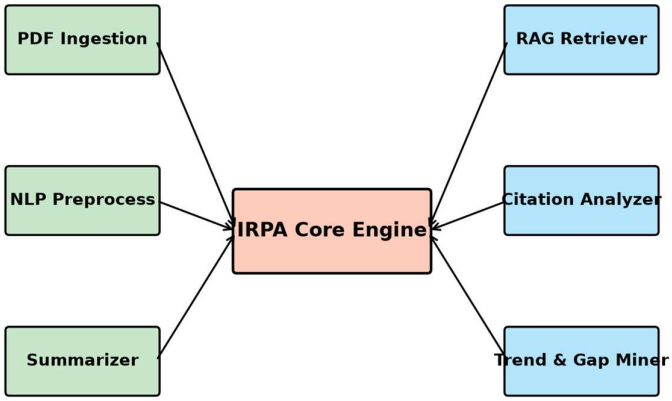

Fig. 3. Module-level view of the IRPA core engine.

6. ALGORITHM AND MATHEMATICAL MODEL

A. Algorithm

Input: Research paper P in PDF format; user query q.

Output: Structured analytical report R = {summary, findings, limitations, gaps, citations}.

The steps below outline how a query moves through the system, from ingestion to a returned report.

Step 1: Extract raw text T from P.

Step 2: Clean T — remove headers, footers, and page numbers; normalize whitespace.

Step 3: Segment T into sections S = {s1, s2, ..., sn} using heading-based heuristics.

Step 4: For each si, compute embedding vi = f(si), where f is the sentence-transformer encoder [8].

Step 5: Insert (si, vi) into the vector store V.

Step 6: Compute query embedding vq = f(q).

Step 7: Retrieve top-k sections Sq = argmax over si in S of sim(vq, vi).

Step 8: Build a grounding-oriented prompt π = template(q, Sq).

Step 9: Generate answer a = LLM(π).

Step 10: Post-process a into structured output R; extract citations via regex.

Step 11: For a corpus of papers, cluster {vi} restricted to limitation-bearing sections to surface recurring gaps.

Step 12: Return R to the user interface.

B. Mathematical Model

Let P denote the paper, split into n sections S = {s1, ..., sn}. A sentence encoder f: text → R^d turns each section into an embedding vi = f(si). Given a query q with embedding vq = f(q), the retrieval score for section si is the cosine similarity between the two vectors:

$$sim(vq, vi) = (vq \cdot vi) / (\|vq\| \cdot \|vi\|) \quad (1)$$

Sections are ranked by this score, and the top-k, Sq, are passed forward. The final answer a follows the conditional distribution

$$p(a \mid q, P) = \Sigma si \in Sq\ p(a \mid q, si) \cdot p(si \mid q, P) \quad (2)$$

where p(si | q, P) scales with sim(vq, vi), and p(a | q, si) is what the LLM decoder actually produces [7]. This follows the standard vector-space retrieval model used in information retrieval [22]. Standard classification metrics describe system performance:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (3)$$

$$Precision = TP / (TP + FP),\ \ Recall = TP / (TP + FN) \quad (4)$$

$$F1 = 2 \cdot Precision \cdot Recall / (Precision + Recall) \quad (5)$$

7. IMPLEMENTATION AND PROTOTYPE VALIDATION

A. Development Environment

The IRPA prototype is implemented in Python 3.11, with a Flask REST API on the back end and a Streamlit front end. Text preprocessing uses spaCy 3.x, NLTK, and Hugging Face Transformers, while Pandas, NumPy, and Scikit-learn support general data handling; SQLite stores user data and cached analyses. The stack is containerized with Docker and runs on a 4-core CPU with 8 GB RAM — in practice, an Intel i5-class processor, 8 GB RAM, roughly 20 GB of free disk space for models and the vector index, and a broadband connection for LLM API calls are enough to run it. This configuration keeps the prototype usable on hardware a student already has, without requiring a dedicated GPU.

B. Prototype Implementation

The implemented pipeline takes an uploaded paper through eleven connected stages. A PDF is ingested through pdfplumber, which preserves layout for single- and multi-column documents, with PyPDF2 as a fallback extractor when layout-aware parsing fails on a page. The extracted text is cleaned — headers, footers, and page numbers are stripped with regular expressions and whitespace is normalized — after which spaCy performs sentence segmentation. Section segmentation follows, identifying units such as Introduction, Methodology, Results, and Limitations through regex heuristics built around IEEE heading patterns.

Each segmented section is encoded with the all-MiniLM-L6-v2 sentence-transformer model, and the resulting vectors are stored in a vector index, with Qdrant as the primary store and FAISS as a local fallback. When a user submits a query, the same encoder embeds the query, and the top-k most similar sections are retrieved by cosine similarity. These sections are assembled into a grounding-oriented prompt and passed to a large language model, which produces an answer intended to stay tied to the retrieved content.

The generated output is post-processed into a fixed eight-field structure — title, objective, methodology, dataset, findings, limitations, future work, and top citations. In parallel, in-text citations are parsed and matched against a reference table built from the paper, and a clustering step groups the embeddings of limitation-bearing sections across a user's uploaded papers, surfacing recurring limitations as candidate research gaps. Each stage hands a defined artefact to the next, so the pipeline can be inspected stage by stage rather than only as a single opaque process.

C. Functional Validation

Rather than a statistical benchmark, the prototype was validated functionally, by confirming that each stage performs its intended role and that the stages connect correctly end to end. This covered: uploading academic PDFs through the Streamlit interface; extracting text via pdfplumber and PyPDF2; cleaning and normalizing that text; identifying document sections through the heading-based heuristics; generating semantic embeddings for each section; retrieving sections relevant to a submitted query; producing responses that draw on retrieved context rather than ungrounded generation; extracting the eight structured fields from a processed paper; identifying in-text citations and matching them to reference entries; and flagging recurring limitations as candidate research gaps across multiple uploaded papers.

This form of validation establishes that the system does what it was designed to do at each processing step. It does not, by itself, establish how accurate the retrieved sections or generated summaries are relative to a human-produced reference — that remains a separate, open question addressed in Section IX.

D. Validation Criteria

Prototype functionality was assessed against system-level criteria rather than comparative accuracy figures:

- Each processing stage — ingestion, preprocessing, segmentation, embedding, indexing, retrieval, generation, extraction, citation analysis, and gap detection — completes without failure on a given input paper.
- Data passes correctly between modules, with each stage's output matching the input format the next stage expects.
- Sections retrieved for a given query are topically related to that query, as judged by inspection of the retrieved content.
- Structured output is internally consistent, with all eight fields populated in a stable format across different input papers.
- Generated responses can be traced back to the retrieved sections that informed them, supporting the grounding claim made in Section IV.

- The end-to-end workflow, from PDF upload to the final structured report, completes successfully for a representative research paper.

These criteria confirm that the IRPA prototype functions as an integrated system. They are not a substitute for the comparative, metric-based evaluation against TF-IDF, TextRank, BERTSUM, and T5-Small described in Section IX, which remains a separate and necessary piece of future work.

8. RESULTS AND DISCUSSION

A. End-to-End System Functionality

The complete pipeline, from PDF upload to structured output, runs as a connected sequence rather than as separate, manually invoked tools. A user submits a PDF through the Streamlit interface; the system extracts and cleans the text, segments it into sections, encodes those sections, and makes them available for retrieval, without requiring intervention between stages. Table II lists the eight fields the system returns for a processed paper and what each one is populated from; this is the structure the Streamlit interface renders once a query or summary request completes.

TABLE II. STRUCTURED OUTPUT FIELDS RETURNED BY IRPA

Field	Populated From
Title	Paper title as identified from the document header during section segmentation.
Objective	Objective statement retrieved and synthesized from the introduction / problem-statement sections.
Methodology	Method description retrieved from the methodology-tagged section(s) of the paper.
Dataset	Dataset or corpus description, when present in the retrieved sections.
Findings	Key findings synthesized from results-tagged sections retrieved for the query.
Limitations	Limitation statements retrieved from limitation- or discussion-tagged sections.
Future Work	Future-work statement retrieved from the corresponding section, when present.
Top Citations	References identified as most frequently cited within the document by the citation-analysis module.

B. Semantic Retrieval and Context Grounding

Before any response is generated, the system retrieves the sections of the uploaded paper most similar to the user's query, using the cosine-similarity ranking described in Section VI. Those retrieved sections, rather than the full document, are what the language model sees when it produces an answer, which is what allows a generated response to reference specific content from the paper instead of relying purely on the model's own general knowledge. Retrieval grounding of this kind is designed to improve traceability between what the system outputs and what the source document contains; whether it measurably reduces hallucinated content relative to an ungrounded baseline is a quantitative question left to the evaluation outlined in Section IX, and no such reduction is claimed here.

C. Structured Research Information Analysis

The system organizes each processed paper into a consistent set of categories — title, objective, methodology, dataset, findings, limitations, future work, and citations — following the extraction schema introduced in Section IV. Producing this fixed structure, rather than a single free-form summary, means the output can be dropped directly into a literature-review chapter without further reformatting, and it makes it possible to compare the same field across several papers a user has uploaded.

D. Citation and Research-Gap Analysis

Citation information extracted during processing is organized into a reference table tied back to the source document, letting the system identify which references a paper cites most often. Separately, the gap-detection step groups limitation-bearing sections across a user's uploaded papers by embedding similarity, and limitation statements that recur across more than one paper are surfaced as candidate research directions. These candidates are a starting point rather than a conclusion: recognizing that a limitation appears in multiple papers is a pattern-matching exercise, and deciding whether that pattern represents a genuine, worthwhile research gap still requires human academic judgment informed by domain knowledge the system does not have.

E. Discussion

The main outcome of this work is an integrated pipeline, not a new machine-learning model. Individually, PDF processing, sentence-transformer embeddings, vector retrieval, RAG-based generation, structured extraction, citation analysis, and gap detection are established techniques; IRPA's contribution is in connecting them into a single workflow that a user can run against an uploaded PDF without switching between separate tools for summarization, citation lookup, and gap identification, distinguishing it from the related work discussed in Section II.

This contribution should be read for what it is: a working system architecture that demonstrates these components can operate together end to end on ordinary hardware, rather than a claim of state-of-the-art performance against the baselines named in Section II. The extent to which IRPA's grounded output improves on those baselines in measurable terms — accuracy, retrieval quality, or processing time — is the subject of the evaluation plan set out in Section IX, and is intentionally left as a separate, forthcoming piece of work rather than folded into the claims made here.

9. LIMITATIONS AND FUTURE WORK

Output quality depends on the underlying language model, and smaller open-weight models are more likely to miss numeric detail. PDF parsing becomes less reliable on scanned or heavily formatted documents, and OCR support is still fairly basic. Retrieval and analysis currently work only for English-language papers, and citation parsing assumes IEEE- or ACM-style references. A large-scale comparative evaluation using standardized datasets and benchmark metrics remains an important direction for future work. Such an evaluation could provide additional evidence regarding retrieval quality, response accuracy, and comparative performance across alternative approaches.

Future work will focus on large-scale comparative evaluation, domain-specific scientific encoders, scalable cloud deployment, multilingual document support, improved OCR for scanned or historical documents, and integration with reference-management systems. These extensions can further strengthen the practical applicability of IRPA for researchers, students, and academic teams.

10. CONCLUSION

This paper described IRPA, a retrieval-augmented system built to take over the more repetitive parts of a literature review — reading a paper, summarizing it, pulling out limitations, and flagging possible gaps — while keeping its output tied to what the source document actually says. The pipeline covers PDF ingestion, preprocessing, section segmentation, sentence-transformer embedding, vector-based retrieval, RAG-based generation, structured extraction, citation analysis, and gap detection, and is meant to run on ordinary hardware rather than a dedicated GPU setup.

The contribution of this work lies primarily in the design and implementation of an integrated research-paper analysis workflow rather than in claiming state-of-the-art benchmark performance. Future work will focus on large-scale comparative evaluation, multilingual support, improved OCR capabilities, scalable deployment, and integration with reference-management tools.

The main contribution is in how section-level retrieval, structured prompting, citation analysis, and gap detection through limitation clustering are brought together into one lightweight pipeline, rather than in any new underlying model. These capabilities provide a unified foundation for assisting students, faculty, and applied research teams with the day-to-day analysis and organization of academic literature.

References

[1] C. An, M. Zhong, Y. Chen, D. Wang, X. Qiu, and X. Huang, "Enhancing scientific papers summarization with citation graph," in Proc. AAAI, 2021, pp. 12498-12506.

[2] A. Cohan and N. Goharian, "Scientific document summarization via citation contextualization and scientific discourse," Int. J. Digital Libraries, vol. 19, no. 2-3, pp. 287-303, 2018.

[3] K. Baleswari, P. Reddy, S. Visalakshi, and R. Vijay, "RAG publications assistant: an AI-powered system for research paper analysis," IJERST, vol. 22, no. 1, 2026.

[4] R. Kinney et al., "The Semantic Scholar open data platform," arXiv:2301.10140, 2023.

[5] M. Hari Krishna, K. Sri, N. Tejaswini, R. Prakash, and S. Eashwar, "Intelligent research paper summarizer using ML and NLP," IJIRSET, vol. 15, no. 3, 2026.

[6] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," in Proc. EMNLP, 2019.

[7] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. NeurIPS, 2020.

[8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019.

[9] M. Zaheer et al., "Big Bird: Transformers for longer sequences," in Proc. NeurIPS, 2020.

[10] I. Cachola, K. Lo, A. Cohan, and D. Weld, "TLDR: Extreme summarization of scientific documents," in Findings of EMNLP, 2020.

[11] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. Weld, "SPECTER: Document-level representation learning using citation-informed transformers," in Proc. ACL, 2020.

[12] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in Proc. EMNLP-IJCNLP, 2019.

[13] A. Vaswani et al., "Attention is all you need," in Proc. NeurIPS, 2017.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019.

[15] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer (T5)," JMLR, vol. 21, 2020.

[16] T. Brown et al., "Language models are few-shot learners," in Proc. NeurIPS, 2020.

[17] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in Proc. EMNLP, 2004.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv:1301.3781, 2013.

[19] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in Proc. EMNLP, 2014.

[20] 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.

[21] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Draft, 2023.

[22] C. D. Manning, P. Raghavan, and H. Schutze, Introduction to Information Retrieval. Cambridge Univ. Press, 2008.