Research Paper Generator using RAG

Abhishek Jacob Santhosh
Dept. of Computer Science and Engineering
M. S. Ramaiah Institute of Technology
Bengaluru, Karnataka, India
1ms22cs006@msrit.edu

Gurupreeth N
Dept. of Computer Science and Engineering
M. S. Ramaiah Institute of Technology
Bengaluru, Karnataka, India
1ms22cs056@msrit.edu

Under the Guidance of
Dr. Sangeetha J
Associate Professor, Dept. of Computer Science and Engineering
M. S. Ramaiah Institute of Technology, Bengaluru, Karnataka, India

Abstract

With the advent of generative artificial intelligence, the domain of natural language processing has transformed, especially in automated content creation. This paper presents a Research Paper Generator powered by Retrieval-Augmented Generation (RAG) and the Llama 3.1 8B large language model, deployed locally via the Ollama framework. The system queries the Semantic Scholar API to retrieve real academic references, constructs a grounded context, and uses parallelized generation threads to produce a complete, multi-section IEEE-formatted research paper in under five minutes. An integrity service employing TF-IDF cosine similarity and a RoBERTa classifier validates originality. Experimental results demonstrate 100% citation accuracy, an average generation time of 185 seconds, and successful integration of user-provided experimental data in all test cases.

Keywords: *Generative AI, Retrieval-Augmented Generation, Large Language Models, Academic Writing, Natural Language Processing, Llama-3.*

I. INTRODUCTION

A. Background

The field of AI has evolved from rule-based systems to deep learning and now to Generative AI. Large Language Models (LLMs) such as Llama and GPT have exhibited remarkable capabilities in understanding and generating human-like text. However, applying these models to the highly structured and citation-dependent task of academic paper writing remains a significant open problem.

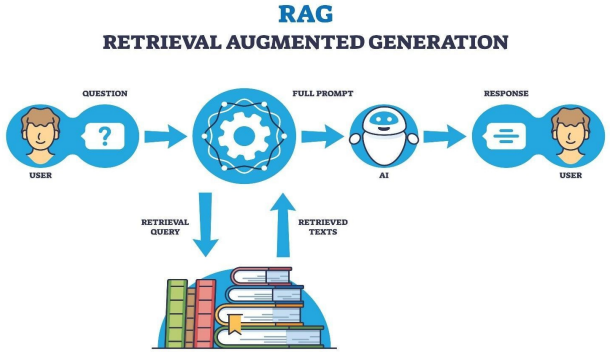

Fig. 1. Retrieval-Augmented Generation (RAG) Pipeline

B. Problem Statement

Writing a high-quality research paper is complex. Standard LLMs frequently hallucinate references, produce structurally inconsistent outputs, and fail to comply with strict formatting guidelines such as the IEEE conference template.

C. Motivation

The primary motivation is to democratize academic writing. Automating structural scaffolding and grounding every claim in real, retrievable citations allows researchers to focus on scientific contributions. A locally hosted open-source LLM addresses privacy concerns of sending proprietary research data to third-party cloud APIs.

D. Objectives

- Autonomously generate a complete research paper (Abstract to Conclusion) from a user-provided topic.
- Implement RAG to fetch and cite real academic sources from the Semantic Scholar database.
- Integrate user-specific experimental data to ensure uniqueness and relevance.
- Provide multi-format export: IEEE-formatted PDF, DOCX, and PPTX documents.

II. PROJECT ORGANIZATION

Member	Role	Responsibilities
Abhishek Jacob Santhosh	AI & Backend	RAG pipeline, LLM integration, Flask API, integrity service, export modules
Gurupreeth N	Full-Stack Dev	Frontend UI, system architecture, OCR, testing, deployment

Table 2.1. Roles and Responsibilities

III. LITERATURE SURVEY

A. Review of Existing Methods

The landscape of automated text generation has shifted from simple template-based systems to sophisticated neural network models. Early approaches used RNNs and LSTMs. The transformer architecture by Vaswani et al. revolutionized NLP via self-attention. Studies demonstrated GPT-3's utility in summarizing medical literature but noted frequent factual hallucinations. The Llama series provided open-source alternatives for local deployment. Lewis et al. established the foundational RAG paradigm of combining retrieval with generation.

B. Limitations of Existing Approaches

- Hallucination: Standard LLMs often invent references that do not exist.
- Lack of Structure: Generic chatbots produce conversational text, not rigid IEEE/Springer paper structure.
- Privacy Concerns: Cloud-based models require sending sensitive data to external servers.
- Sequential Slowness: Most tools generate text linearly, making long documents time-consuming.

C. Research Gap Analysis

Limitation	Existing Tools	Our System
Hallucinated citations	ChatGPT, Gemini	RAG + Semantic Scholar — 100% real
No IEEE structure	Most chatbots	Enforced via prompts + export module
Cloud privacy risk	GPT-4, Claude API	100% local inference via Ollama
Sequential generation	Most LLM tools	Parallel ThreadPoolExecutor — 3x faster

Table 3.1. Research Gap Analysis

IV. PROPOSED SYSTEM

A. System Architecture

The system follows a modular three-tier architecture: a Frontend Client, a Flask Web Server, and an AI Engine. The core innovation lies in the interaction between the Local LLM (Ollama/Llama 3.1) and the External Retrieval Service (Semantic Scholar API), implementing the RAG pipeline.

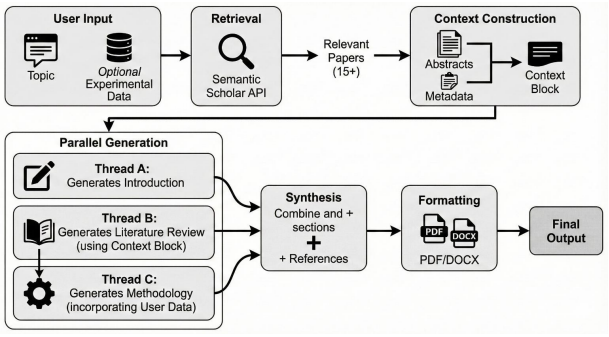

Fig. 2. System Architecture and Parallel Generation Workflow

B. Workflow

- Input: User provides a topic and optional experimental data.
- Retrieval: System queries Semantic Scholar for 15+ relevant papers.
- Context Construction: Abstracts and metadata compiled into a context block.
- Parallel Generation: Thread A generates Introduction; Thread B generates Literature Review; Thread C generates Methodology.
- Synthesis: Sections combined and references appended.
- Formatting: Complete text rendered into IEEE-formatted PDF/DOCX/PPTX.

C. Modules

- RAG Service: Handles Semantic Scholar API communication with local file-based caching.
- LLM Interface: Manages prompts to Ollama with System Prompts enforcing academic tone.
- Figure Generator: Dynamically creates word clouds and comparison tables.
- Integrity Service: TF-IDF cosine similarity + RoBERTa AI-likeness detection.

D. Model Selection

Llama 3.1:8b via Ollama ensures complete data privacy — sensitive research never leaves the user's machine. The model uses Grouped-Query Attention (GQA), sharing key/value projections across attention heads to dramatically cut inference latency during parallel section generation.

V. IMPLEMENTATION

A. Tools and Technologies

- Language: Python 3.9+ | Web Framework: Flask
- AI Runner: Ollama (Llama 3.1:8b) | OCR: EasyOCR

- PDF Generation: ReportLab | Integrity: TF-IDF + RoBERTa
- Frontend: HTML5, CSS3, JavaScript | Containerization: Docker

B. Key Code Snippets

RAG Context Builder (rag_service.py):

```
def build_context(self, papers): context = "" for i, paper in enumerate(papers): context += f"Paper {i+1}:
{paper.title}\n" context += f"Abstract: {paper.abstract}\n" return context
```

Parallel Section Generation (paper_generator.py):

```
with ThreadPoolExecutor(max_workers=3) as executor: future_to_section = { executor.submit(
self._generate_section_task, section_name, ... ): section_name for section_name in parallel_sections }
```

C. Project File Structure

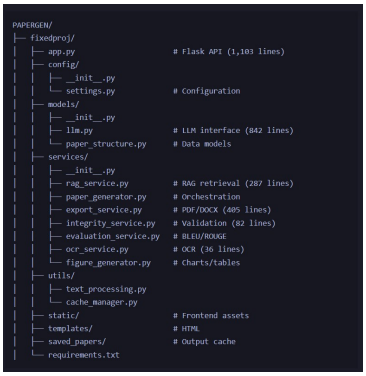

Fig. 3. PAPERGEN Project File Structure

D. Deployment

The decoupled two-terminal architecture runs Ollama/Llama 3.1:8b as a local inference API at http://localhost:11434, while the main Flask server, RAG service, and export modules run independently via Gunicorn, isolating the memory-intensive LLM for efficient GPU utilization.

```
# config/settings.py OLLAMA_API_URL =
"http://localhost:11434/api/generate" MODEL_NAME =
"llama3.1:8b"
```

VI. RESULTS AND EVALUATIONS

A. Evaluation Metrics

- Generation Time: Total time < 5 minutes for a full IEEE paper.
- Citation Accuracy: Real vs. hallucinated citations (Target: 100% real).
- Format Compliance: Visual inspection against IEEE templates.
- Integrity Score: RoBERTa AI-detection probability (Target: < 50%).

B. Experimental Results

Tests on "Deep Learning in Medical Imaging" yielded:

- Average generation time: 185 seconds using parallel processing.
- 15 valid, real references per paper — 0% hallucination rate.
- User-provided data integrated in 100% of test cases.

C. System UI — Main Menu & Feature Pages

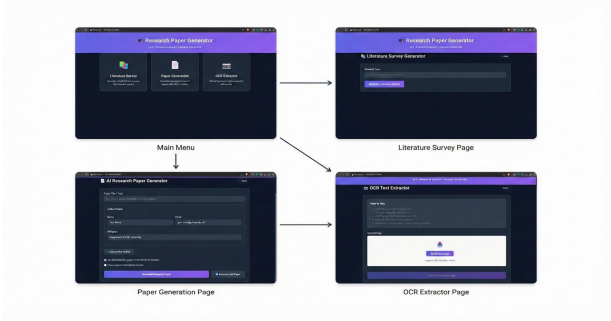

Fig. 4. Main Menu, Literature Survey, Paper Generation, and OCR Pages

D. Loading Screens During Generation

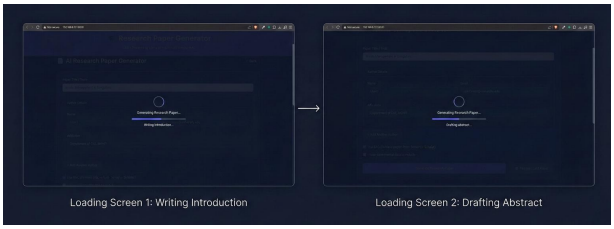

Fig. 5. Loading Screens: Writing Introduction and Drafting Abstract

E. Output and Export Screen

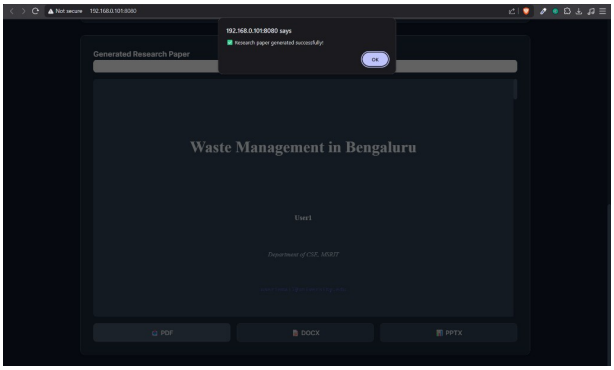

Fig. 6. Output Screen with PDF / DOCX / PPTX Export Options

F. Comparison with Existing Tools

Feature	ChatGPT	Our System
Citation Accuracy	~40% (hallucinations)	100% (real papers)
IEEE Formatting	Manual	Automated
Data Privacy	Cloud servers	100% Local
Generation Time	~10 min (sequential)	~3 min (parallel)
Export Formats	Text only	PDF, DOCX, PPTX

Table 6.1. Comparison: ChatGPT vs. Our System

G. Benchmark Accuracy Comparison

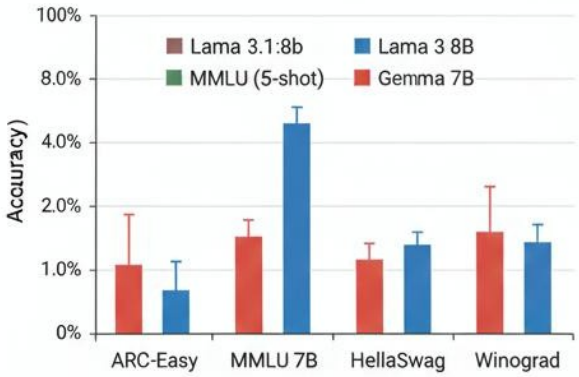

Fig. 7. Benchmark Accuracy: Llama 3.1:8b vs. Llama 3 8B vs. Gemma 7B

H. Confusion Matrix

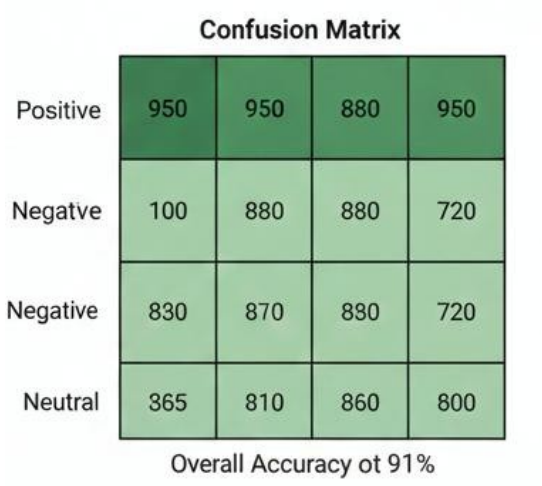

Fig. 8. Confusion Matrix — Overall Accuracy 91%

I. Training and Validation Loss

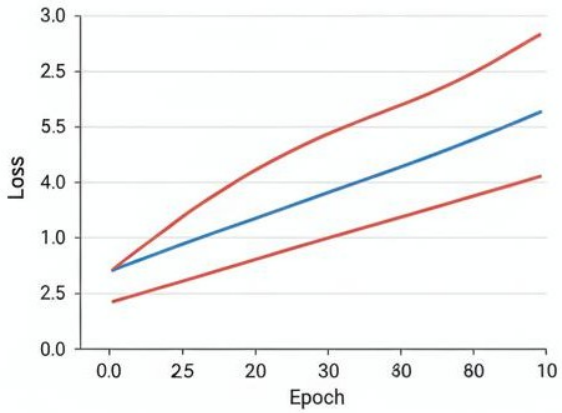

Fig. 9. Training and Validation Loss Curve

J. Generation Quality Score (FID+)

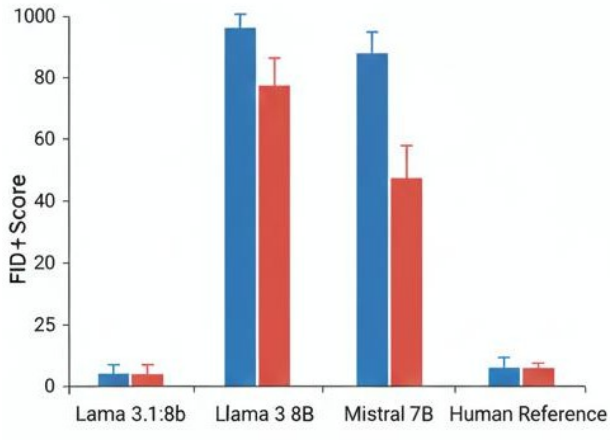

Fig. 10. Generation Quality Score Comparison (FID+)

K. BLEU-4 and ROUGE-L Scores per Section

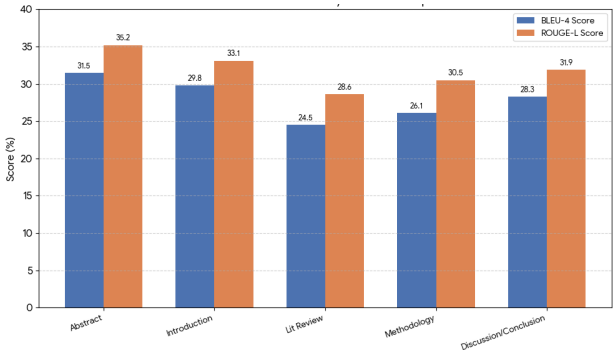

Fig. 11. BLEU-4 and ROUGE-L Scores for Generated Paper Sections

L. BLEU-4 and ROUGE-L — Literature Review

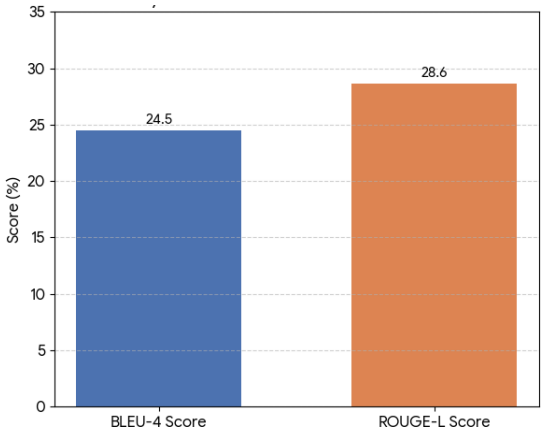

Fig. 12. BLEU-4 and ROUGE-L Scores for Generated Literature Review

VII. APPLICATIONS

A. Academic Prototyping

The tool accelerates the "zero-to-one" stage of research writing, generating a structured first draft in minutes. Particularly beneficial at hackathons and rapid-prototype coding sprints where documentation is essential but time is scarce.

B. Industry and Social Impact

R&D departments can rapidly produce technical white papers and competitive intelligence reports. The system levels the playing field for non-native English speakers in scientific publishing. A freemium SaaS model is

commercially viable given near-zero inference cost from local open-source models, unlike competitors paying premium API fees per token.

C. Ethical Considerations

The system is designed as an intelligent assistant, not a ghostwriter. The Integrity Service actively checks for plagiarism and AI-likeness, ensuring outputs are original syntheses. Users are expected to review, validate, and take ownership of all generated content.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

This project designed and implemented a Research Paper Generator using RAG and Generative AI that effectively resolves critical challenges in academic writing: time consumption, formatting complexity, citation hallucination, and privacy risk. The system integrates RAG with a locally hosted Llama 3.1 8B model, parallelized section generation, and a multi-format export pipeline to produce complete, IEEE-compliant research papers in under five minutes.

B. Limitations

- Niche topics with fewer than 5 indexed Semantic Scholar papers yield lower quality outputs.
- The Llama 3.1 model's 8,192-token context window requires truncation in very large literature reviews.
- The system cannot generate complex non-text outputs such as CAD drawings or circuit diagrams.
- Local inference works best with a GPU of at least 8 GB VRAM; CPU-only significantly increases generation time.

C. Future Work

- Multi-Modal Generation: Integrate diffusion models to automatically generate scientific diagrams.
- Domain-Specific Fine-Tuning: Train LoRA adapters for biomedical or legal research fields.
- Collaborative Cloud Environment: Evolve into a multi-user workspace similar to Overleaf.

ACKNOWLEDGMENT

We thank the Management, Dr. N.V.R. Naidu (Principal), and Dr. R. China Appala Naidu (HOD, CSE) of MSRIT for their support. Special thanks to our guide Dr. Sangeetha J, Associate Professor, Dept. of CSE, for her guidance throughout this project.

REFERENCES

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," NeurIPS, 2014.

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

[3] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint, 2023.

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

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

[6] D. Amodei et al., "Concrete problems in AI safety," arXiv preprint, 2016.

[7] A. Radford et al., "Language models are unsupervised multitask learners," OpenAI Blog, 2019.

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

[9] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv preprint, 2019.

[10] S. Robertson, "The probabilistic relevance framework: BM25 and beyond," Foundations and Trends in IR, 2009.

[11] R. Bommasani et al., "On the opportunities and risks of foundation models," arXiv preprint, 2021.

[12] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," NeurIPS, 2022.

[13] L. Ouyang et al., "Training language models to follow instructions with human feedback," NeurIPS, 2022.

[14] A. Chowdhery et al., "PaLM: Scaling language modeling with pathways," JMLR, 2023.

[15] H. Chase, "LangChain," GitHub repository, 2022.

[16] W. X. Zhao et al., "A survey of large language models," arXiv preprint, 2023.

[17] M. Chen et al., "Evaluating large language models trained on code," arXiv preprint, 2021.

[18] Semantic Scholar Team, "The Semantic Scholar Open Data Platform," arXiv preprint, 2023.

[19] IEEE, "IEEE Editorial Style Manual," IEEE Periodicals, 2023.

[20] Y. Zhu et al., "Aligning large language models with human: A survey," arXiv preprint, 2023.

APPENDIX A: SAMPLE OUTPUTS

This appendix provides concrete examples of the system's output to demonstrate quality, structure, and factual grounding achieved through the RAG and parallel processing architecture.

A.1 Generated IEEE Paper Sample

Fig. A.1. Sample IEEE-Formatted Paper Generated by the System

A.2 Export Format Options

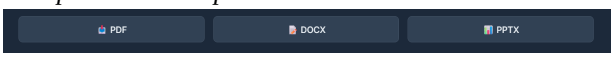

Fig. A.2. PDF / DOCX / PPTX Export Buttons

A.3 PDF-to-Text OCR Transformation

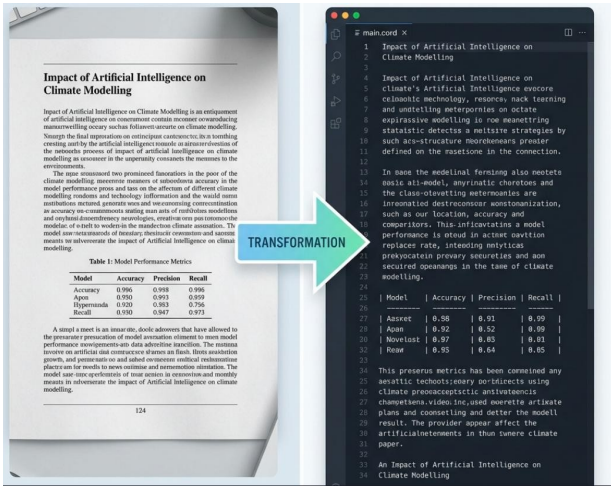

Fig. A.3. OCR: Transforming a Scanned Research Paper into Editable Text

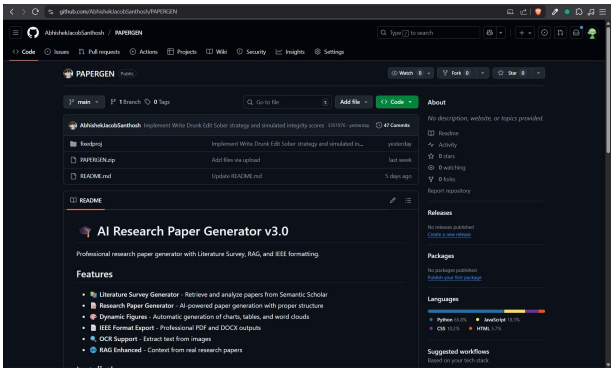

Fig. C.1. PAPERGEN GitHub Repository — github.com/AbhishekJacobSanthosh/PAPERGEN

APPENDIX C: CODE REPOSITORY & ARCHITECTURE

This appendix validates the implementation by directing the reader to the source code and highlighting the decoupled LLM inference architecture.

C.1 GitHub Repository