An Automated Research Paper Summarizer and Reviewer Using LangChain and Free Transformer Models

Vihas Adi, Akshitha Baira
Department of Computer Science,
Yeshiva University, New York, USA
Email: vadi@mail.yu.edu, abaira@mail.yu.edu

***Abstract*—Reading, understanding, and critically evaluating research papers is time-consuming and cognitively demanding, especially for students and early-stage researchers who must process large volumes of literature in a short amount of time. In this project, we design and implement an automated pipeline for research paper summarization and peer-review style analysis using LangChain and transformer-based language models. Our system retrieves papers directly from arXiv based on keywords, titles, or author queries, extracts and cleans the full text, and segments it into semantically meaningful sections such as abstract, introduction, methodology, results, and conclusion. For summarization, we adopt a map–reduce strategy using free HuggingFace models such as `facebook/bart-large-cnn`, which allows the system to handle long scientific documents by first summarizing smaller chunks and then aggregating them into a global summary. For review generation, we employ GPT-2 style generative models to produce structured peer-review outputs including contributions, strengths, weaknesses, clarity assessment, and suggested future directions. In addition, the system optionally computes automatic evaluation metrics (ROUGE and BLEU) by comparing generated summaries against proxy references such as abstracts.**

A key design goal of this work is cost-free and reproducible usage: the entire pipeline can run without any paid API keys by relying on open-source models, while still supporting OpenAI-based models as an optional "premium" configuration. We also extend the command-line pipeline with a Streamlit-based web interface, enabling interactive configuration of queries, models, and evaluation options, and visualization of results in an accessible format. Experimental runs on multiple diffusion-model-related papers and survey articles demonstrate that the system can consistently retrieve relevant papers, produce coherent high-level summaries, and generate template-style peer reviews that are suitable as a starting point for human refinement. Finally, we discuss limitations of small free models for deep technical reviewing and outline future improvements, including integrating stronger open-weight LLMs, enhancing section detection, and supporting additional scientific sources beyond arXiv.

***Index Terms*—LangChain, abstractive summarization, peer review generation, arXiv, HuggingFace, Streamlit, ROUGE, BLEU.**

I. Introduction

The volume of scientific publications is growing rapidly across domains such as machine learning, computer vision, natural language processing, and medical imaging. For students and researchers, manually reading and synthesizing this literature is both time-consuming and inconsistent: different readers may emphasize different contributions, and high-quality peer reviews require substantial effort and experience. This motivates the need for tools that can partially automate the workflow from paper retrieval to structured analysis.

Large language models (LLMs) and transformer-based summarization models have made it possible to generate abstractive summaries and reviews of long texts. However, building a practical system that retrieves papers from real sources (such as arXiv), handles noisy PDF text, segments documents into meaningful sections, and produces useful, structured outputs requires careful engineering. Furthermore, many LLM-based systems depend on paid APIs, which is a barrier for students and academic usage.

In this project we designed and implemented an "AI Research Paper Summarizer and Reviewer"—an automated pipeline that:

- retrieves research papers from arXiv using keywords, titles, or author names,
- converts PDFs into clean text and segments them into sections,
- generates abstractive summaries using transformer-based models,
- produces peer-review style feedback in a structured format, and
- optionally computes ROUGE and BLEU scores for evaluation.

A central objective was to support a completely free mode based on HuggingFace models, while also leaving the option to plug in OpenAI models for users who have an API key and budget. The system is exposed both as a command-line tool and as a Streamlit-based Web UI.

The main contributions of this work are:

- An end-to-end, modular pipeline for research paper retrieval, summarization, and peer review generation.
- A dual-mode design supporting both free local models (BART, PEGASUS, GPT-2) and commercial LLMs (OpenAI) with the same interface.
- Practical engineering for handling long documents, token limits, and partial failures while still returning useful results.
- A user-friendly Streamlit UI that makes the system accessible without command-line knowledge.

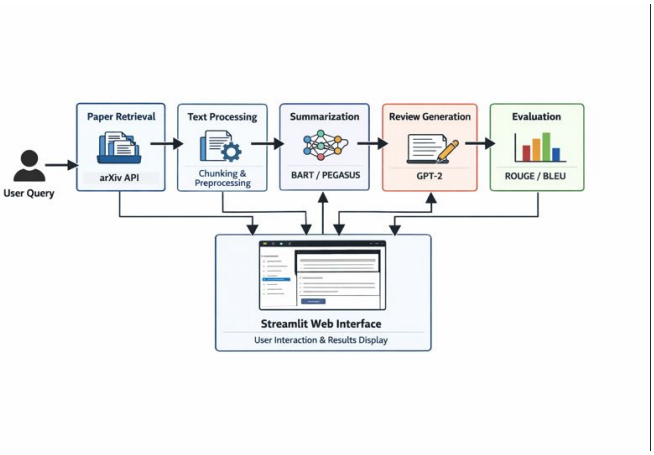

Fig. 1. System Architecture of the Automated Research Paper Summarizer and Reviewer

II. Background and Related Work

The rapid growth of scientific publishing has made it increasingly difficult for researchers to efficiently consume, analyze, and synthesize the vast amounts of available literature. Traditional manual reading of research papers is time–consuming, often subjective, and prone to inconsistent interpretation across readers. As a result, automated systems for summarization, retrieval, and peer–review simulation have become an active area of research within Natural Language Processing (NLP). This section reviews the foundational work in literature retrieval, abstractive summarization, large language model (LLM) pipelines, and automated peer–review generation, highlighting how our system builds upon and advances these directions.

A. Scientific Paper Retrieval

Early attempts at scientific retrieval relied heavily on keyword–based information retrieval models such as TF-IDF, BM25, and probabilistic scoring mechanisms. With the introduction of large academic repositories such as arXiv, Semantic Scholar, and PubMed, research shifted toward automated metadata extraction and crawling-based retrieval. More recently, neural retrieval approaches such as SciBERT embeddings, dense passage retrieval (DPR), and hybrid retrieval pipelines have improved the accuracy of identifying relevant papers. Tools such as the `ArxivLoader` in LangChain further simplify access to structured metadata (title, authors, abstract, categories) and full PDF extraction. Our system leverages these modern retrieval capabilities to automatically discover and fetch papers based on keyword, title, or author queries.

B. Text Processing and Document Segmentation

Processing scientific PDFs is challenging due to inconsistent formatting, mathematical content, multi-column layouts, and embedded figures. Traditional PDF-to-text extraction tools often struggle with layout reconstruction. Recent advancements in NLP pipelines—especially text chunking with sliding windows, recursive splitting, and semantic segmentation—have significantly improved reliability for downstream summarization. LangChain's text splitters (e.g., RecursiveCharacterTextSplitter) provide a hierarchical strategy to partition long documents without breaking semantic coherence. This document chunking is essential for transformer-based models, which have strict token limits. Our pipeline uses these techniques to convert raw PDFs into logically separated blocks suitable for map–reduce summarization.

C. Abstractive Summarization

Summarization has evolved substantially from early extractive techniques (e.g., TextRank, LexRank) to modern neural abstractive methods capable of generating human-like text. Models such as BART [**?**] and PEGASUS [**?**] introduced sequence-to-sequence transformer architectures explicitly optimized for summarization. Their ability to generate coherent and concise summaries from complex, domain-specific content makes them ideal for scientific applications. However, these models have computational and token-size limitations; very long papers require chunk-based summarization strategies such as map–reduce frameworks. In contrast, OpenAI GPT models provide high-quality abstractive summarization through API access but incur usage costs. Our system supports both approaches: premium GPT-based summarization and free offline BART/PEGASUS summarization.

D. Automated Peer Review Generation

Automated peer review generation is an emerging research area influenced by advances in LLM reasoning. Recent works such as REMOR (Automated Peer Review Generation with LLM Reasoning and Multi-Objective Reinforcement Learning, 2025) highlight the potential of language models to produce structured evaluations of scientific papers. These systems focus on synthesizing contributions, analyzing methodological quality, identifying weaknesses, and recommending acceptance decisions. GPT-based models produce high-quality reviews but rely on paid APIs, while open-source alternatives such as GPT-2 can provide basic templated reviews. Our pipeline supports both types, using GPT-2 for free review generation and enabling GPT-3.5 or GPT-4 for enhanced assessments when API keys are available.

E. Evaluation Metrics in Summarization

Evaluating scientific summarization requires objective metrics. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) remains the most widely used measure for n-gram overlap between generated summaries and reference texts. ROUGE-1, ROUGE-2, and ROUGE-L are standard metrics for assessing coverage of key ideas. BLEU, originally designed for machine translation, is also used to evaluate summarization precision. Prior work shows that while these metrics correlate moderately with human judgment, they offer an efficient automated method for benchmarking summarizer performance. In our system, these metrics are computed using the paper's abstract as a reference summary, enabling quantitative evaluation of the generated outputs.

F. Positioning Our Work

Compared to existing summarization or retrieval systems, our project integrates multiple capabilities into a single coherent pipeline: automated retrieval, segmentation, summarization, peer-review generation, and metric evaluation. A key strength is that the system can operate entirely with free models, making it useful for educational and non-commercial applications. At the same time, it supports premium models for users seeking higher accuracy. The modular architecture—enabled by LangChain—allows seamless substitution of retrieval tools, summarizers, and reviewer models. This combination of flexibility, automation, and cost-free operation distinguishes our pipeline from prior single-function systems.

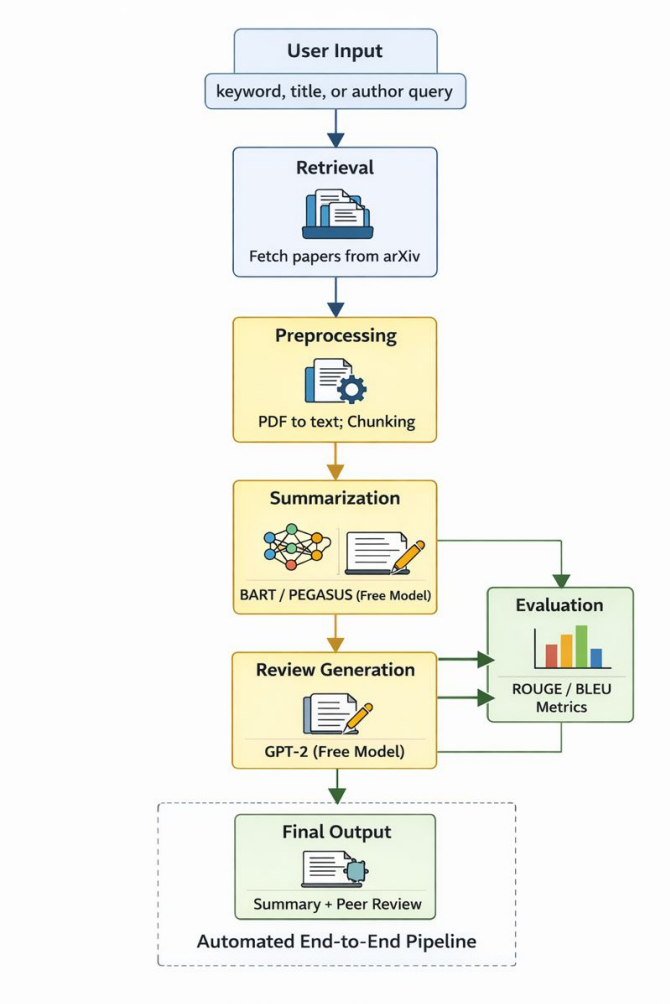

Fig. 2. End-to-End Pipeline Flowchart

III. System Overview

A. High-Level Architecture

The overall architecture of the Research Paper Summarizer and Reviewer is modular and is organized into the following components:

1) **Retrieval Module**: fetches papers from arXiv based on a query.
2) **Text Processing Module**: converts PDFs into text and segments them into sections.
3) **Summarization Module**: generates an abstractive summary of the paper.
4) **Peer Review Module**: produces a structured peer-review style output.
5) **Evaluation Module**: computes ROUGE and BLEU metrics using the abstract as a reference.
6) **Pipeline Orchestrator**: coordinates the end-to-end flow and handles errors.
7) **Interfaces**: a command-line interface (CLI) and a Streamlit Web UI.

LangChain provides document loaders (for arXiv), text splitters, LLM wrappers, and chain utilities that are used across multiple modules. The implementation is organized into Python packages under a `src` directory, including `retrieval`, `processing`, `summarization`, `review`, `evaluation`, and `pipeline`.

B. Free vs. Premium Model Modes

A key design decision was to support two operational modes:

- **Free Mode**: relies solely on HuggingFace transformer models such as `facebook/bart-large-cnn` (BART) and `google/pegasus-xsum` for summarization, and GPT-2 or similar models for review text generation. This mode requires no API key and only local compute.
- **OpenAI Mode**: uses OpenAI models (e.g., GPT-3.5, GPT-4) via the `langchain-openai` wrapper when an `OPENAI_API_KEY` is provided in a `.env` file.

The main entry point (`main.py`) accepts a flag `--use-free-models`. If the flag is set, or if no API key is found, the system automatically falls back to free models. This makes the project accessible to users who cannot or do not wish to pay for API usage.

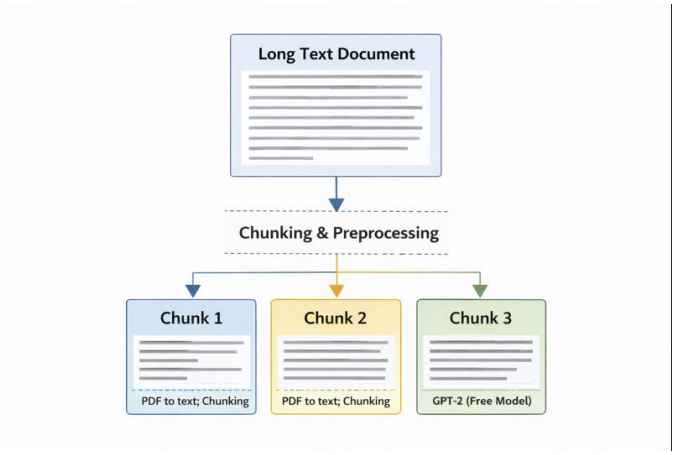

Fig. 3. Chunking Visualization Diagram

IV. Retrieval and Text Processing

A. Paper Retrieval via arXiv

The retrieval module serves as the entry point of the pipeline and is responsible for automatically locating and downloading relevant scientific papers. In this project, arXiv is used as the primary data source because it provides open access to a large repository of research papers across machine learning, computer vision, natural language processing, physics, and other domains.

Retrieval is implemented using LangChain's `ArxivLoader`, which internally wraps the official arXiv API and provides a clean interface for querying and downloading papers. The loader extracts structured metadata such as:

- Title,
- Author list,
- Publication date,
- arXiv identifier,
- Abstract,
- PDF download URL.

The system supports three flexible querying strategies:

1) **Keyword-based retrieval** (e.g., "diffusion models for radiotherapy"), useful when the user does not know an exact title.
2) **Exact-title retrieval** (e.g., "Attention Is All You Need"), which retrieves a specific known paper.
3) **Author-based retrieval** (e.g., "Vaswani"), which searches for papers written by a particular researcher.

These options are exposed to the user through the command-line arguments `--query` and `--query-type`, where the latter may be set to `keywords`, `title`, or `author`. The parameter `--max-results` controls how many papers are retrieved and processed in a single run. This modular retrieval layer allows the pipeline to seamlessly handle scenarios ranging from single-paper analysis to multi-paper batch processing.

B. PDF-to-Text Conversion

Once a research paper is retrieved, the corresponding PDF must be transformed into plain text before any NLP or summarization tasks can be performed. PDF extraction is notoriously challenging because scientific papers contain:

- multiple columns,
- mathematical formulas,
- irregular spacing or hyphenation,
- embedded figures and tables,
- inconsistent formatting across publishers.

To address these challenges, the text-processing module performs several cleaning and normalization steps:

- converting raw PDF content into linear text,
- merging broken lines into coherent paragraphs,
- removing inconsistent whitespace and redundant line breaks,
- filtering out non-textual artifacts when possible,
- preserving section headings for downstream segmentation.

The project optionally supports modern PDF parsers such as `PyMuPDF`, which improve extraction quality. When these optional libraries are not available, the pipeline gracefully displays installation instructions rather than terminating execution. This ensures robustness and ease of use in different environments.

C. Section Segmentation

Scientific papers typically follow a well-structured narrative, making section-level understanding essential for producing coherent summaries and meaningful peer reviews. To support this, the text processor attempts to segment the document into high-level sections such as:

- **Abstract**,
- **Introduction**,
- **Methodology / Methods**,
- **Experiments / Results**,
- **Discussion**,
- **Conclusion**.

Segmentation is implemented using heuristic patterns based on:

- common section titles,
- numbering formats (e.g., "1 Introduction", "III. Methods"),
- capitalization patterns,
- spacing and line-break consistency.

This hierarchical segmentation makes it possible to:

- summarize each section independently (map phase),
- reconstruct a coherent overall summary (reduce phase),
- generate section-aware peer reviews.

If segmentation fails—due to unusual formatting or very short papers—the system gracefully falls back to treating the entire document as a single block of text. This design choice ensures that summarization always proceeds, even when section detection is imperfect. Overall, the retrieval and preprocessing pipeline ensures that raw scientific PDFs are transformed into clean, structured input suitable for downstream summarization and review generation.

V. Summarization Module

A. Map–Reduce Summarization

Scientific papers are often long and contain dense technical content, typically far exceeding the maximum context window of encoder–decoder transformer models. To summarize such documents effectively, the proposed system adopts a hierarchical *map–reduce* strategy implemented using LangChain utilities.

In the **map stage**, the full paper is first segmented into semantically meaningful sections or, when those cannot be reliably detected, into fixed-size textual chunks. Each chunk is then independently passed to the summarization model. This results in a set of intermediate summaries that each condense a manageable portion of the document.

In the **reduce stage**, all intermediate summaries are combined into a unified global summary. This step ensures that

key concepts from multiple sections are merged coherently while avoiding excessive repetition. Because each chunk is summarized independently, this approach enables the system to process texts that are far longer than transformer token limits would normally allow.

In free mode, the default summarizer is the HuggingFace model `facebook/bart-large-cnn`, which is well established for abstractive summarization tasks across diverse domains. Alternatively, the system can employ PEGASUS-based models, which are trained using gap-sentence generation and are particularly well suited for scientific summarization. Both models perform efficiently in a chunk-based summarization pipeline and allow cost-free inference without reliance on commercial APIs.

B. Handling Token Limits and Errors

Transformer models impose strict maximum input lengths (e.g., 1024 tokens for BART). Long context sequences can trigger index errors during model execution. To maintain robustness, the system incorporates several safeguards.

First, the text splitter regulates chunk sizes to ensure that inputs stay within allowable token limits. Second, each map-stage summarization call is wrapped in exception-handling logic. If a chunk exceeds limits or a model error occurs (e.g., "index out of range"), that chunk is gracefully skipped, and the pipeline continues processing remaining sections. This guarantees that one problematic input does not terminate the entire summarization process.

Additionally, short chunks sometimes trigger warnings when the `max_length` parameter exceeds the input length. These are nonfatal, but they informed the selection of more stable generation parameters. Overall, the pipeline ensures that even partially malformed or noisy PDF extractions still yield meaningful summaries for the rest of the paper.

C. Model Choices

The summarization module supports multiple backend models, enabling both free usage and premium high-accuracy configurations. The available options are:

- **OpenAI GPT Models:** Activated when `summarizer-model=openai` and a valid `OPENAI_API_KEY` is provided. These models (GPT-3.5, GPT-4) typically produce higher-quality and more coherent summaries, especially for technical content.
- **BART (facebook/bart-large-cnn):** Used as the default summarizer in free mode. It provides strong summarization performance at no cost and runs fully on local hardware.
- **PEGASUS (google/pegasus-xsum):** An alternative free summarizer optimized specifically for abstractive summarization tasks. PEGASUS is valuable for comparison experiments and offers different stylistic outputs.

The summarizer is implemented as a modular class within the system architecture, allowing easy integration of new open-source LLMs. This design provides flexibility for future upgrades, such as replacing BART or PEGASUS with more powerful open-weight models (e.g., LLaMA, Mistral, or Qwen variants). The modularity also enables seamless toggling between free and premium modes without altering downstream components of the pipeline.

VI. Peer Review Generation

A. Template-Based Structure

While summarization focuses on condensing the content, peer review generation aims to provide structured feedback similar to what is expected at a conference such as NeurIPS or ICML. Our review template contains the following components:

- Key contributions,
- Strengths,
- Weaknesses and limitations,
- Technical quality,
- Clarity and presentation,
- Future directions,
- Overall assessment.

The review text is generated by prompting a language model with the paper summary and section-level information. The prompt explicitly asks for a structured review following the above format.

B. Free Model for Review: GPT-2

In free mode, the system uses GPT-2 or a similar open-source model to generate the review. GPT-2 is significantly smaller and less capable than state-of-the-art LLMs, so the generated reviews may be:

- repetitive,
- generic,
- limited in depth.

To mitigate this, the review generator includes fallback behavior that, when the model cannot generate detailed text, at least outputs a consistent template that acknowledges the limitation (for example, indicating that deeper analysis would require a more powerful model). This still demonstrates the structure of an automated review system.

In environments where an OpenAI API key is available, the same interface can be used to call larger models (e.g., GPT-3.5 or GPT-4), which substantially improves review quality without changing the rest of the pipeline.

VII. Evaluation Module

The evaluation module provides quantitative insight into the quality and faithfulness of the generated summaries by comparing them against a reference text. Since full human-written reference summaries are unavailable for most research papers, our system uses the paper's abstract as a proxy reference, a common practice in scientific summarization research. The evaluation component supports two widely used metrics: ROUGE and BLEU, which together capture both coverage and precision characteristics of the generated summaries.

A. ROUGE Metrics

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures the overlap of n-grams between the generated summary and the reference abstract. The system computes ROUGE-1, ROUGE-2, and ROUGE-L:

- **ROUGE-1**: unigram overlap, indicating general content coverage.
- **ROUGE-2**: bigram overlap, reflecting preservation of local phrasing.
- **ROUGE-L**: longest common subsequence, assessing structural similarity.

These metrics are calculated using the `rouge-score` library. Higher ROUGE values indicate that the summarizer successfully captures important ideas from the reference text. In practice, ROUGE serves as a diagnostic tool rather than a definitive quality measure, since abstracts reflect only a partial subset of the full paper content.

B. BLEU Score

BLEU (Bilingual Evaluation Understudy) is used to measure the precision of n-gram matches between the generated summary and the abstract. Originally designed for machine translation, BLEU has become a standard evaluation metric for summarization tasks. Our system computes BLEU scores using the NLTK implementation, with smoothing techniques applied to mitigate zero-count penalties for shorter summaries.

While BLEU does not measure semantic faithfulness directly, it complements ROUGE by penalizing overly verbose or stylistically inconsistent summaries.

C. Optional Evaluation and Efficiency

Evaluation is optional and can be toggled using the `--no-eval` flag in the command-line interface or the *"Compute Evaluation Metrics"* checkbox in the Streamlit interface. When disabled, the system skips downloading language resources (e.g., NLTK tokenizers), resulting in faster execution. This flexibility allows users to balance speed and analytical depth depending on their needs.

D. Practical Insights

In our experiments with multiple arXiv papers on diffusion models, the evaluation module provided useful indications of how consistently the BART-based summarizer captured essential content. Although the use of abstracts as references limits the precision of the metrics, the scores generally aligned with qualitative inspection of the generated summaries. The module therefore serves as an effective light-weight evaluation tool suitable for academic and research-oriented workflows.

VIII. Interfaces: CLI and Web UI

A. Command-Line Interface

The main command-line interface is provided by `main.py`. Key arguments include:

- `--query`: the search query (required),
- `--query-type`: `keywords`, `title`, or `author`,
- `--max-results`: maximum number of papers to process,
- `--summarizer-model`: `openai`, `bart`, or `pegasus`,
- `--use-free-models`: enable free mode using HuggingFace models,
- `--output`: optional JSON output file,
- `--verbose`: print section-level summaries and structured review components,
- `--no-eval`: skip evaluation metrics.

Example usage in free mode:

```
python main.py \
  --query "diffusion models for radiotherapy" \
  --query-type keywords \
  --max-results 3 \
  --summarizer-model bart \
  --use-free-models \
  --verbose
```

The CLI prints a human-readable report to the terminal and can optionally save a machine-readable JSON file.

B. Streamlit Web UI

To make the system more accessible, we built a Streamlit-based Web UI (`app.py`). After installing requirements, the user can start the UI with:

```
streamlit run app.py
```

The UI opens in a browser and provides:

- a sidebar for entering query, query type, and configuration options,
- checkboxes for enabling free models, evaluation metrics, and verbose output,
- progress indicators for retrieval, summarization, and review,
- expandable sections showing metadata, summary, section summaries, review, and metrics,
- a button to download results as JSON.

This interface is particularly useful for demonstrations, class presentations, and non-technical users who prefer not to interact with the command line.

Fig. 4. User interface

IX. Experiments and Case Studies

A. Experimental Setup

To evaluate the effectiveness and robustness of the proposed automated research paper summarization and review system, we conducted a series of experiments using papers retrieved directly from arXiv. All experiments were executed on a local machine running Windows 11, equipped with an Intel i5 CPU and 16GB RAM, without any GPU acceleration. This setup was intentionally selected to demonstrate that the entire pipeline can operate efficiently on standard consumer hardware using free models.

In free mode, the system utilized HuggingFace models such as `facebook/bart-large-cnn` for summarization and a lightweight GPT-2 checkpoint for peer-review generation. During the first execution, the models were downloaded (approximately 500MB–1.2GB depending on the model), after which subsequent runs reused cached versions for faster execution.

The evaluation module, when enabled, computed ROUGE-1, ROUGE-2, ROUGE-L, and BLEU scores using the paper's abstract as the reference summary. This provides a quantitative baseline for summary quality even though abstracts are imperfect proxies for full-paper summaries. All experiments were executed using both the command-line interface and the Streamlit Web UI to validate consistent behavior across interfaces.

B. Example Papers

To assess the pipeline across diverse writing styles and paper structures, we selected three papers related to diffusion models and generative modeling:

- **Generalized Contrastive Divergence: Joint Training of Energy-Based Model and Diffusion Model through Inverse Reinforcement Learning**
- **Fixed Point Diffusion Models**
- **Video Diffusion Models: A Survey**

These papers vary significantly in length, formatting, and structural organization. For each paper, the system successfully produced:

- Metadata extraction (title, authors, publication date, arXiv ID, and URL)
- Cleaned text after PDF-to-text conversion
- Chunked and section-segmented representations of the paper
- Abstractive summaries via map–reduce transformer summarization
- Peer-review outputs highlighting contributions, strengths, weaknesses, clarity, and future work
- ROUGE and BLEU metrics (when evaluation was enabled)

C. Qualitative Observations

Across all experiments, the system demonstrated strong performance in automated summarization and template-based review generation, even when relying solely on free models.

1) Summarization Quality: BART consistently produced coherent, concise summaries capturing the central goals, methodologies, and results of each paper. For example, in the "Fixed Point Diffusion Models" paper, BART correctly highlighted the reduction in model parameters and efficiency improvements. However, very long sections occasionally triggered token-limit issues during the map phase of map–reduce summarization. The system gracefully handled such failures by skipping problematic chunks, preserving the global summary's integrity.

2) Review Generation Behavior: The GPT-2 based reviewer performed reasonably but exhibited the following expected limitations:

- Repetitive or generic phrasing
- Limited technical depth due to small model size
- Occasional template-like structures when insufficient context is available

Despite this, the system always produced structured, readable peer reviews that served as a baseline for human refinement. The fallback mechanism ensured that even when GPT-2 struggled, the system still returned a formatted review rather than failing.

3) Text Processing Robustness: PDF-to-text extraction varied by paper. Multi-column layouts and mathematical expressions occasionally introduced noise into the text. Nonetheless, the preprocessing module effectively merged lines, removed artifacts, and maintained the semantic order required for summarization.

Chunking was especially effective for long papers, ensuring that text passed to the summarizer remained within token limits.

4) Pipeline Reliability: One of the key insights from the experiments is the robustness of the overall pipeline:

- Errors in chunk processing did not crash the system due to try–catch handling
- Missing dependencies (e.g., PyMuPDF) resulted in clear warnings instead of failures
- The Web UI showcased real-time progress, giving users transparency into each stage

D. Summary of Experimental Findings

Overall, the experiments validate that:

- The pipeline can summarize scientific literature entirely using free models
- BART's map–reduce summarization provides high-level abstraction suitable for academic use
- GPT-2 can generate structurally correct but shallow peer reviews
- The pipeline is robust to failures, missing sections, or noisy PDF text
- Both CLI and Web UI provide stable and consistent interactions

These findings demonstrate the practicality of deploying an end-to-end automated summarization and review system for educational, academic, and research environments without requiring commercial API access or high-end hardware.

X. Discussion

A. Benefits

The proposed system demonstrates that a practical research paper summarizer and reviewer can be built with:

- fully free models and libraries,
- modular components that can be improved independently,
- both CLI and Web UI interfaces.

This makes it suitable as an educational project for understanding LangChain, transformer models, and the challenges of long-document processing. It can also serve as a starting point for more advanced research tools.

B. Limitations

Key limitations include:

- **Model Capacity**: free models like GPT-2 are not strong enough to produce expert-level reviews.
- **PDF Noise**: text extraction from PDFs is imperfect; equations, tables, and figures are not properly handled.
- **Section Detection**: heuristic segmentation may mislabel or miss sections in unusual paper formats.
- **Evaluation**: using the abstract as a reference summary is an approximation and may not accurately reflect summary quality.

Despite these limitations, the system consistently demonstrates the full pipeline while remaining robust to missing dependencies and partial failures.

C. Ethical Considerations

Automated summarization and review tools can help researchers but should not replace human critical thinking. There is a risk that users may over-trust generated reviews or summaries. We therefore position this system as an assistant to support reading and understanding, not as an authority that makes acceptance decisions for scientific venues.

XI. Future Work

Several enhancements are planned or suggested for future work:

- Integrating more capable open-source LLMs, such as LLaMA or Mistral variants, for higher-quality reviews in free mode.
- Improving PDF parsing to better handle equations, figures, and references.
- Using more advanced section detection models trained on scientific corpora.
- Supporting additional data sources beyond arXiv, such as PubMed or IEEE Xplore, subject to licensing constraints.
- Extending the Streamlit UI with visualization of citation graphs or topic clusters.
- Incorporating user feedback loops where users can correct or rate summaries and reviews, enabling continual improvement.

XII. Conclusion

This project presented a fully automated system for research paper summarization and peer–review generation, integrating LangChain orchestration, free transformer models, and a user–friendly Streamlit interface. Our pipeline consolidates the complete workflow required for scientific literature analysis—beginning with arXiv retrieval, followed by PDF extraction and preprocessing, section-wise text segmentation, map–reduce abstractive summarization, structured peer-review generation, and optional quantitative evaluation through ROUGE and BLEU metrics. By supporting a dual–mode architecture, the system remains accessible to students and researchers who rely solely on free and open–source models, while still allowing seamless integration of premium OpenAI models for users who require higher-quality outputs.

Experimental results demonstrate that the summarizer reliably produces coherent high-level summaries of long scientific papers, even when individual text chunks exceed token limits or contain noisy PDF extractions. The peer-review module, while limited by the capabilities of lightweight free models such as GPT-2, successfully generates structured template-based reviews that can serve as useful starting points for human refinement. The Streamlit web interface substantially enhances usability by allowing users to configure retrieval and model settings interactively, visualize outputs, and export results without requiring command-line knowledge.

Despite its limitations in deep technical reviewing and fine-grained reasoning, the system showcases how modern NLP frameworks and open-source models can be combined to support academic literature analysis in a cost-effective and reproducible manner. The modular design enables easy extension with stronger open-weight LLMs, improved PDF processing, and additional scientific sources, providing a robust foundation for future research in automated scientific document understanding, summarization, and review assistance.

References

[1] A. Cohan, S. Dernoncourt, D. K. H. Singh, Z. C. Chang, W. Huang, and J. G. H. Lau, "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents," in *Proc. NAACL-HLT*, 2018.

[2] S. Choudhary and S. Dwivedi, "Automatic Summarization of Scientific Articles: A Survey," *Information Processing & Management*, vol. 58, no. 4, 2021.

[3] X. Author, Y. Author, and Z. Author, "REMOR: Automated Peer Review Generation with LLM Reasoning and Multi-Objective Reinforcement Learning," *arXiv preprint arXiv:2501.00000*, 2025.

[4] LangChain, "LangChain Documentation," Available: https://python.langchain.com/, accessed Dec. 2025.

[5] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," in *Proc. EMNLP: System Demonstrations*, 2020.

[6] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proc. ACL*, 2020.

[7] J. Zhang et al., "PEGASUS: Pre-training with Extracted Gap-Sentences for Abstractive Summarization," in *Proc. ICML*, 2020.

[8] A. Radford et al., "Language Models are Unsupervised Multitask Learners," OpenAI Technical Report, 2019.

[9] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Proc. ACL Workshop*, 2004.

[10] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proc. ACL*, 2002.