Ho Chi Minh city - University of Science
Faculty of Information Technology

NATURAL LANGUAGE PROCESSING
APPLICATIONS

A13 - Academic Paper SummarAIzer

LECTURER

- Nguyễn Hồng Bửu Long
- Lương An Vinh

PREPARED BY

- 22127056 - Châu Vĩnh Đạt
- 22127220 - Nguyễn Anh Kiệt
- 22127275 - Trần Anh Minh
- 22127280 - Đoàn Đặng Phương Nam

Ho Chi Minh city, 2025

Contents

1) Project members

ID	Name	Role	Job description
22127056	Châu Vĩnh Đạt	Developer	• Implemented user interface for the application. • Implemented backend server to handle the APIs. • Designed the database for the product.
22127220	Nguyễn Anh Kiệt	Project Leader	• Controlled the workflow of the team. • Designed RAG + LLM system for contextual QNA. • Implemented citation tracking.
22127275	Trần Anh Minh	Developer	• Collected and processed data. • Trained and evaluated summarization models. • Deployed the product locally using Docker
22127280	Đoàn Đặng Phương Nam	Developer + Tester	• Researched related technologies and methods for the project. • Supported team members on debugging. • Tested the products.

2) Abstract

The exponential growth of academic literature poses significant challenges for researchers, students, and professionals in efficiently extracting relevant insights from complex research papers. This project introduces an intelligent Academic Paper Summarizer, a web-based system designed to automatically summarize lengthy scientific documents and enable contextual question-answering. Leveraging a fine-tuned Longformer Encoder-Decoder (LED) model for summarization and a Retrieval-Augmented Generation (RAG) architecture powered by Gemini 2.0 Flash and Neo4j graph database for question-answering, the system achieves robust performance on structured datasets like SciSummNet, with notable improvements in ROUGE, BERTScore, BLEURT, and METEOR metrics. By integrating Level 1 LLMOps principles, including automated pipelines, prompt guarding, and scalable deployment, the system ensures efficiency, safety, and maintainability. Despite challenges in generalizing to diverse corpora like arXiv and limitations in lexical fluency, the project demonstrates the transformative potential of combining advanced NLP techniques and LLMOps to democratize access to scientific knowledge. Future work aims to enhance model robustness, advance LLMOps maturity, and expand system functionality for broader research applications.

Keywords: Academic Paper Summarization, Natural Language Processing, Large Language Models, Retrieval-Augmented Generation, LLMOps, Longformer Encoder-Decoder, Semantic Chunking, Contextual Question-Answering, Neo4j Graph Database, Gemini 2.0 Flash, ROUGE, BERTScore, BLEURT, METEOR, SciSummNet, arXiv, Prompt Guarding, Fine-Tuning, Scalability, Knowledge Democratization.

3) Introduction

3.1) Problem Statements

With the exponential growth of academic research across diverse disciplines, researchers, students, and professionals face an overwhelming challenge in keeping up with the latest findings. Traditional search engines and digital libraries offer access to papers, but they lack the ability to provide quick, contextual understanding of a document's content. Users often spend significant time reading, interpreting, and synthesizing multiple lengthy papers just to extract relevant insights.

There is a pressing need for an intelligent system that can:

- Automatically summarize long and complex research papers.
- Enable interactive question-answering on specific content within the paper.
- Provide contextual understanding tailored to the user's queries.

This system should not only reduce the time and cognitive effort needed to digest academic literature but also democratize access to scientific knowledge by making it more accessible and searchable through natural language

3.2) Motivation

Research workflows today are overwhelmed by the volume and complexity of academic literature. Researchers, students, and professionals often spend hours manually reading through papers just to grasp key findings or retrieve specific insights. This is inefficient, especially in fast-evolving fields like machine learning, medicine, or climate science.

Recent advancements in Large Language Models (LLMs) have unlocked the potential for machines to understand, summarize, and reason over long-form text—including research papers. However, leveraging LLMs effectively in production systems is not trivial. This is where LLMOps (Large Language Model Operations) becomes crucial.

LLMOps provides the infrastructure, tooling, and practices required to:

- Efficiently deploy and scale LLMs for inference over large documents.
- Continuously monitor performance and accuracy in user-facing environments.
- Fine-tune models and prompts using real-world user feedback and domain-specific corpora.
- Manage lifecycle, versioning, and reproducibility of LLMs and RAG pipelines.

By integrating LLMOps into the system design, we ensure that the AI-driven summarization and contextual Q&A pipeline is not only powerful, but also:

- Scalable for institutional or enterprise use.
- Adaptable to different domains and document types.
- Maintainable and improvable over time with minimal friction.

In essence, this project is motivated by the need to combine intelligent research understanding with operational excellence, delivering a seamless, robust tool that transforms how users interact with complex academic content.[1]

3.3) LLMOps Principles

LLMOps (Large Language Model Operations) extends traditional MLOps to support the full lifecycle of LLM-powered applications. It enables reliable, secure, and scalable deployment of large language models with optimized cost, performance, and governance.

According to Google Cloud's LLMOps framework, LLMOps are divided into three levels:

- *Level 0 - Manual LLMOps*: This initial stage involves manual processes for most tasks related to developing, deploying, and managing Large Language Models (LLMs).
- *Level 1 - Automate ML pipeline*: This level introduces automation for the core machine learning pipeline, specifically for training and validating the LLM.

[1]You can find the full project source code and documentation at: *https://github.com/Gitnut11/Academic-Paper-SummarAIzer*

- *Level 2 - Full LLMOps automation*: This represents the most mature level, characterized by end-to-end automation of the entire LLM lifecycle, integrating CI/CD/CT practices.

This project is designed around Level 1 maturity, which represents the transition from manual experimentation to a more structured and repeatable operational pipeline. Key characteristics of Level 1 include:

- Automation of the Core Pipeline: The model inference, document chunking, embedding, and retrieval steps are automated and modularized.
- Prompt Templating & Evaluation: Prompt design is standardized across use cases and evaluated iteratively.
- Model Deployment: Inference is served through a scalable, containerized backend using Docker.
- Vector Store Integration: Embeddings are stored in a vector database such as Neo4J Aura to support efficient and accurate retrieval.
- Basic Monitoring: Logging of system behavior (for example, request latency, token count, response quality) is set up to facilitate debugging and improvement.

This level offers the right balance between performance and development agility for a research-stage system. As the project evolves, it can be progressively upgraded to Level 2 maturity by integrating CI/CD for prompt and model updates, feedback-driven fine-tuning, and advanced governance workflows.

4) Literature Review

4.1) Related Work

In recent years, significant progress has been made in using Large Language Models (LLMs) for document understanding, summarization, and question-answering tasks. Notable research contributions include:

- BART (Lewis et al., 2019) and T5 (Raffel et al., 2020): These transformer-based encoder-decoder architectures demonstrated strong performance on summarization and text generation benchmarks, laying the foundation for modern abstractive summarizers.
- GPT-3 / GPT-4 (Brown et al., 2020; OpenAI, 2023): These models showcased the power of few-shot and zero-shot learning, capable of performing summarization and Q&A with minimal fine-tuning. Their generalization capabilities sparked the rise of prompt engineering.
- Long-form Document QA Systems (for example, REPLUG, FiD, RAG): These systems utilize a retrieval-augmented generation pipeline to address the context window limitations of LLMs and to ground answers in external documents.
- Academic-focused Tools:
  - SciQA, PaperQA, and ChatPDF offer document-specific question-answering interfaces using off-the-shelf LLMs with retrieval.
  - Semantic Scholar API and arXiv summarization tools apply NLP pipelines to index and generate summaries of academic papers.

While these systems are promising, they often lack full lifecycle support—prompt tuning is ad hoc, monitoring is minimal, and scalability remains a challenge. Most notably, these systems rarely account for governance, versioning, or observability, which are critical for trust in academic or enterprise settings.

4.2) Necessity of LLMOps

As NLP systems increasingly rely on large-scale foundation models, the development and deployment process has shifted from training models to orchestrating and maintaining complex systems that incorporate LLMs. This shift makes LLMOps essential for several reasons:

- Scalability: LLMs require significant compute for inference. LLMOps enables efficient use of hardware via quantization, caching, and batching strategies, ensuring scalability even with limited resources.

- Prompt & Model Management: In LLM-powered applications, prompt design is a form of programming. Managing prompt versions, performing A/B testing, and evaluating effectiveness require systematic tooling, which LLMOps provides.
- Observability & Reliability: Traditional NLP systems rarely monitored hallucinations, token overuse, or latency bottlenecks. LLMOps introduces observability pipelines that track these metrics, enabling teams to detect and respond to system degradation.
- Security & Privacy: As LLMs can inadvertently leak sensitive data or be exploited via prompt injection, LLMOps introduces layers of control including input sanitization, PII detection, and sandboxed execution environments.
- Continuous Improvement: Unlike static models, LLM applications can improve over time through feedback. LLMOps supports human-in-the-loop (HITL) pipelines, collecting user signals to fine-tune prompts or models (for example, via Reinforcement Learning from Human Feedback).
- Reproducibility & Governance: In academic and enterprise use cases, reproducibility and version traceability are critical. LLMOps brings in experiment tracking, prompt versioning, and deployment rollbacks—key for model governance and audits.

In conclusion, as LLM applications grow in sophistication and reach, LLMOps is not optional—it is foundational for building robust, ethical, and production-grade NLP systems.

5) Methodology

5.1) Large Language Model and NLP Techniques

In this project scale, we integrated multiple LLM and NLP techniques to handle the complex task of scientific document summarization and contextual question-answering.

- LED-based Summarization: We utilized a Longformer Encoder-Decoder (LED) model fine-tuned specifically for scientific summarization. LED's sparse attention mechanism enables it to effectively process documents up to 16,384 tokens, making it ideal for handling lengthy scientific papers.
- Retrieval-Augmented Generation (RAG) for Contextual Q&A: For answering the question based on the uploaded documents, we employed a RAG architecture. A semantic retriever, backend by Gemini embeddings and a Neo4j graph database, retrieves the most relevant chunks. The answers are then generated, based on the retrieved context, through a LLM, Gemini 2.0 Flash, ensuring factual consistency and relevance.
- Prompt Guarding using Gemini 2.0: To enhance the safety and robustness of the system, a Prompt Guard mechanism was added. Before any question is processed, the Gemini model evaluates the question to detect and filter out potentially unsafe, inappropriate, or harmful inputs, ensuring compliance with ethical AI standards.

In addition to leveraging powerful LLMs, several fundamental and advanced NLP techniques were integrated in to the system to improve both summarization and question-answering:

- Semantic Chunking: Scientific documents often exceed the input size limit, even for the long-document models like LED. To address this, a semantic chunking mechanism is employed. Documents are split at meaningful sentence boundaries, preserving the natural flow of information. Nevertheless, overlapping regions between chunks were also introduced to avoid information loss at chunk edges.
- Hierarchical Summarization: For extremely long documents, a two-stage summarization process was applied. First, individual chunks are summarized separately. Then the partial summaries were concatenated and passed through the model again to produce a final and coherent document summary.
- Entity Extraction: To enhance retrieval performance in the RAG system, named entities were automatically extracted from document chunks using spaCy `en_core_web_sm` model. Extracted entities were stored in a graph database (Neo4j) and used retrieval to expand the context beyond purely semantic similarity.

- Embedding Generation: Every text chunk and user query was embedded into a high-dimensional vector space using Gemini embedding models. These embeddings were critical for semantic retrieval, allowing the system to find the most contextually relevant chunks even when there was no exact keyword match. The embedding generation was handled via LangChain integration for streamlined processing.

5.2) Dataset Collection

We employed two scientific datasets: *SciSummNet*[1] and arXiv[2]:

- SciSummNet is a high-quality benchmark specifically curated for summarization scientific papers within the field of computational linguistics. It provides concise, human-annotated summaries emphasizing research contributions.
- arXiv, in contrast, contains a large-scale collection of scientific articles from diverse fields where abstract serve as approximate summaries. While arXiv provides broader coverage, its summarization style varies more in consistency and structure compared to SciSummNet.

To highlight their differences:

Dataset	**Domain**	**Size**	**Summary Type**	**Annotation Quality**
SciSummNet	Computational Linguistics	$\approx 1,000$	Human-annotated summaries	High
arXiv	Multidisciplinary Research	$> 200,000$	Author-written abstracts	Moderate

For fine-tuning model, only the SciSummNet dataset was utilized, and during the evaluation phase, the models were tested on both the SciSummNet testset and 100 articles drawn from the arXiv corpus.

This dual evaluation approach enabled us to assess both the model's in-domain performance and its ability to generalize to unseen, out-of-domain scientific texts.

5.3) Model Selection, Fine-tuning Strategy, and Performance Optimization

5.3.1) Model Selections

Two models were selected to address the summarization and contextual question-answering tasks:

- For summarization, we used the pretrained **allenai/led-base-16384** model.[3] This model, capable of processing input sequences up to 16,384 tokens via sparse attention, was fine-tuned on the SciSummNet dataset using parameter-efficient techniques. The final fine-tuned model was hosted publicly on Hugging Face Hub.[4]
- For contextual Q&A, we employed RAG with Gemini 2.0 Flash standing as the LLM core. Moreover, the Neo4j graph database is also utilized, which served as the knowledge backbone for retrieval. We constructed a structured graph-based representation of our documents, chunking them into semantically meaningful units and storing them as nodes with rich metadata. Relationships between these chunks—such as topic similarity, document hierarchy, or semantic links—were encoded as edges, enabling more context-aware traversal and retrieval.

This combination ensured that long scientific documents could be summarized effectively and that downstream question answering remained fast, accurate, and contextual relevant.

[1]Available at: *https://cs.stanford.edu/~myasu/projects/scisumm_net/*
[2]Available at: *https://huggingface.co/datasets/armanc/scientific_papers*
[3]Available at: *https://huggingface.co/allenai/led-base-16384*
[4]Available at: *https://huggingface.co/Mels22/led-scisummnet*

5.3.2) Fine-tune Strategy

The summarization model was fine-tuned on the SciSummNet dataset using a parameter-efficient approach designed for both memory efficiency and high performance:

- We applied 4-bit quantization (NF4 type with bfloat16 compute) using BitsAndBytes to reduce GPU memory usage while maintaining numerical stability.
- PEFT (Parameter-Efficient Fine-Tuning) with LoRA (Low-Rank Adaption) and DoRA (Direction and Rotation Adjustment) was used to adapt a small subset of parameters (`q_proj`, `k_proj`, `v_proj`, `out_proj`) while keeping the rest of the model frozen, enabling efficient fine-tuning.
- Training used the `paged_adamw_8bit` optimizer with a cosine learning rate schedule and a 10% warmup ratio, with Mixed-precision training (bf16) enabled to further accelerate training.

5.3.3) Prompt Guarding Strategy

To ensure the safety and appropriateness of the inputs entering the question-answering system, a Prompt Guard mechanism was implemented:

- Each user-submitted question was first passed through Gemini 2.0 to evaluate its safety.
- Questions deemed unsafe, inappropriate, or harmful were flagged or filtered out before entering the retrieval and answer-generation pipeline.
- This step ensured that the downstream RAG (Retrieval-Augmented Generation) process remained aligned with ethical AI usage standards and prevented the generation of responses based on problematic queries.

5.3.4) Citations Strategy

For tracking the citations of the uploaded paper, first the system will extract the bibliography/reference section of the paper. Then we will prompt on Gemini API to perform the following actions:

- Extracting the existing IDs from any Arxiv Papers/Journals in the provided cites.
- Generating the URLs to the Arxiv Papers/Journals based on the extracted IDs.
- Checking whether the titles and authors from the generated URLs are matching with the cites from the uploaded paper.
- Returning the list of URLs to valid citations.

5.3.5) Performance Optimization

- Chunk-wise Processing: Documents are processed in manageable semantic chunks to fit within the LED model's 8192-token limit, using a 512-token overlap strategy to preserve contextual continuity across chunks.
- Safe Batching: Inference is performed using dynamic batch sizing (`batch_size=4` by default), which helps balance throughput while avoiding out-of-memory errors, especially on consumer-grade GPUs.
- Memory Management: Explicit garbage collection (`gc.collect()`) and CUDA cache clearing (`torch.cuda.empty_cache()`) are used before and after major operations to optimize GPU memory utilization and avoid accumulation of inactive memory.
- PEFT (Parameter-Efficient Fine-Tuning): The model is fine-tuned using LoRA via the PEFT framework. This significantly reduces the number of trainable parameters and runtime memory load during inference, enabling high-quality summarization on large models like LED with minimal computational overhead.

5.4) Evaluation Metrics

To evaluate the quality of our generated summaries, we utilize a suite of both lexical and semantic metrics. These are chosen to capture different aspects of summarization quality — such as surface overlap, semantic fidelity, fluency, and contextual similarity. Below is an overview of the key metrics employed:

5.4.1) ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE is a set of metrics designed to evaluate the quality of summaries by comparing them to reference summaries via n-gram overlap. It primarily measures recall, i.e., how much of the reference content is covered by the candidate summary. ROUGE is effective for identifying literal similarity and is widely used in traditional summarization benchmarks.

- ROUGE-1: Measures unigram (single word) overlap between generated and reference summaries. It evaluates basic content coverage.
- ROUGE-2: Measures bigram (two-word sequence) overlap, providing a proxy for fluency and phrase-level matching.
- ROUGE-L / ROUGE-Lsum: Calculates the longest common subsequence (LCS), capturing sentence-level structure and summary coherence. ROUGE-Lsum is a version adapted for summarization tasks.

Why use ROUGE?

ROUGE has been a longstanding standard in summarization benchmarks. It quantifies how much of the reference content appears in the prediction, offering a direct measure of recall and surface-level similarity. However, it doesn't account for meaning or paraphrasing.

5.4.2) BERTScore

BERTScore leverages contextual word embeddings from pre-trained transformer models (for example, BERT) to assess semantic similarity between generated and reference summaries. Instead of relying on exact word matching, it measures how contextually aligned two sentences are on a token-by-token basis. This makes it ideal for evaluating abstractive summaries where paraphrasing is common.

Why use BERTScore?

BERTScore addresses a major limitation of ROUGE: it can recognize paraphrased or semantically equivalent sentences, even if they share few words. It is particularly important in tasks where exact wording is less critical than meaning — such as scientific or abstractive summarization.

5.4.3) BLEURT

BLEURT (Bilingual Evaluation Understudy with Representations from Transformers) is a learned evaluation metric trained on human judgment data. It incorporates fluency, grammar, factual correctness, and contextual relevance using a fine-tuned transformer. BLEURT is sensitive to subtle quality differences and correlates well with human preferences.

Why use BLEURT?

BLEURT provides a more human-aligned score, sensitive to nuances in language quality, factual correctness, and coherence. It's particularly valuable when evaluating outputs that go beyond basic extractive summaries.

5.4.4) METEOR

Originally developed for machine translation, METEOR (Metric for Evaluation of Translation with Explicit ORdering) measures similarity using exact word matches, stemming, and synonym mapping. It considers both precision and recall, and applies penalties for fragmented or disordered matches. It is more flexible than ROUGE when handling paraphrasing or linguistic variation.

Why use METEOR?

METEOR captures semantic similarity at a finer linguistic level and is more forgiving of lexical variation than ROUGE. It is often more consistent with human judgment, especially in high-level, abstractive generation tasks.

5.5) LLMOps Pipeline

- Explanation of LLMOps pipeline (deployment, monitoring, feedback loop, etc.).

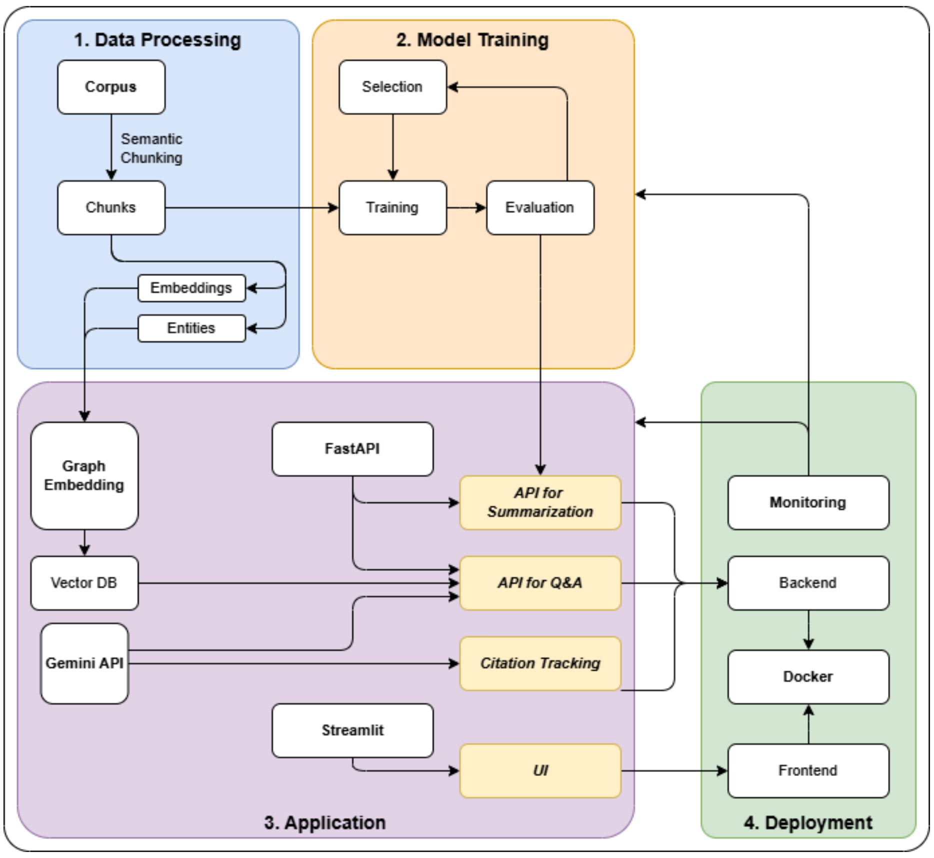

Figure 2:

The LLMOps workflow is structured into four main stages: Data Processing, Model Training, Application, and Deployment.

Data Processing:

- The process begins with a Corpus of data.
- This corpus undergoes Semantic Chunking to break down the text into smaller, meaningful Chunks.
- From these chunks, Embeddings (vector representations) and Entities (named entities like people, places, ...) are extracted.

Model Training:

- This stage involves Selection of appropriate models.
- The selected models undergo Training using the processed data.
- After training, the models are subjected to Evaluation to assess their performance. There's a feedback loop from Evaluation back to Training, indicating an iterative process of refinement.

Application:

- The processed data (embeddings and entities) feeds into this stage. Specifically, chunks and their embeddings are used for Graph Embedding and stored in a Vector DB.
- The Gemini API is also utilized, likely for its large language model capabilities (for example, generation, advanced understanding).
- This stage powers several application functionalities through distinct APIs:
- API for Summarization: Leverages models and data to generate summaries. This is shown to be accessible via FastAPI.
- API for Q&A: Uses the Vector DB (for retrieving relevant context) and potentially the Gemini API to answer questions.
- Citation Tracking: Uses the LLM capabilities of the Gemini API to process and track citations.
- A User Interface (UI) is developed using Streamlit, which interacts with these backend APIs (FastAPI for summarization, and implicitly others) to provide services to the end-user.

Deployment:

- The application (UI and backend APIs) moves into the deployment stage.
- The Frontend (Streamlit UI) and Backend (FastAPI services and other API logic) are managed here.
- Docker is used for containerization, ensuring consistency across environments and simplifying deployment.
- Monitoring is a crucial part of this stage, observing the deployed application for performance, errors, and potentially data drift or model degradation. The output of Monitoring feeds back into the Model Training stage (specifically to Evaluation), creating a loop for continuous improvement and potential retraining or model updates based on live performance.

The diagram illustrates key LLMOps principles:

- Automation: Implied in data processing pipelines, API-driven services, and potentially in the deployment process with Docker.
- Modularity: Different components like data processing, model training, various APIs, and UI are distinct.
- Reproducibility & Versioning: While not explicitly detailed, the use of Docker and distinct model training/evaluation stages hints at capabilities for versioning and ensuring reproducible deployments.
- Monitoring & Feedback Loops: A clear feedback loop exists from Monitoring in the Deployment stage back to Model Evaluation, enabling continuous learning and improvement of the models based on real-world performance. Another loop exists within Model Training between Evaluation and Training.

6) Implementation

6.1) System Overview

This system is a web-based application that allows users to upload PDF documents, automatically generate summaries, and interact with the content through a chatbot. Built using Streamlit for the front-end and SQLite for data management, the system ensures a seamless flow from user registration to intelligent document interaction. It is designed for simplicity, modularity, and ease of use, making it suitable for educational, research, and productivity applications.

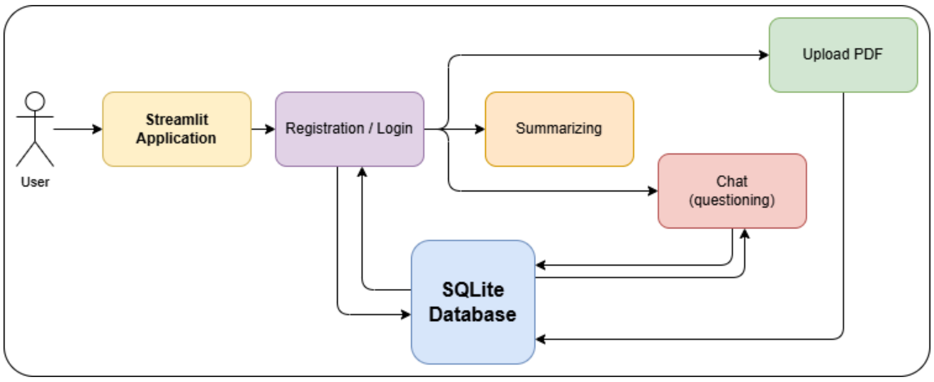

Figure 3: The Application Simplification Workflow

1. **Streamlit Application**: Acts as the user interface of the system, enabling user interaction through a browser. It provides forms for login/registration, PDF uploading, viewing summaries, and using the chatbot.
2. **Registration / Login**: Handles user authentication. Users need to create an account or log in to access the system. User credentials, session data such as history and uploaded PDF are stored within the SQLite database.
3. **Upload PDF:** Allows authenticated users to upload PDF documents. These files are then passed to the summarization module for processing.

4. **Summarizing:** Processes the uploaded PDF to generate a concise textual summary. This module let model to summarize the text (PDF) and stores the results in the database.
5. **Chat (Questioning):** By default, the chat is considered a process of asking the system upon the uploaded PDF as context. The system will extract relevant information to the question before sending it to a LLM to handle the answering.

6.2) Summarization Process

Using our fine-tuned LED model, the input text will go through operation of outlining.

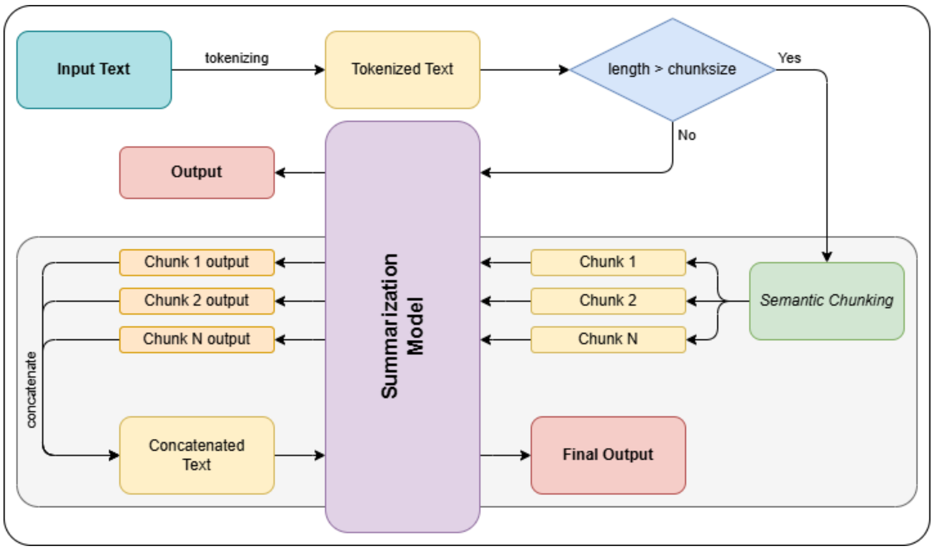

Figure 4: The Summarization Process in Details

1. The input text **is tokenized** by a tokenizer that relate to the model itself. Then based on the number of occurring tokens, the model will handle to input accordingly.
2. If the number of tokens is **within the range of a chunk size** (current set as 8192), as a Longformer model, it can handle and produce the output summarization directly.
3. But if the tokens **exceed the defined amount**, the input will be further chunked into smaller chunks. By chunking semantically, the sentences are grouped in a way that preserves their contextual meaning, ensuring that each chunk remains coherent and informative.
4. Each resulting chunk is then **passed independently** through the Summarization Model. Afterward, these partial outputs are concatenated in order, forming a representation of a unified summary of the full document. Finally, the concatenate summary is further refined by undergoing another summarization stage to produce the Final Output.

This approach enables efficient processing of long documents by overcoming the token limit constraints of transformer model while preserving the semantic integrity of the content.

6.3) Q&A Process

The system's QnA functionality is based on a Retrieval-Augmented Generation (RAG) framework. It consists of two main stages: the RAG Building Stage, where documents are processed and indexed, and the Answering Stage, where user questions are answered using retrieved context.

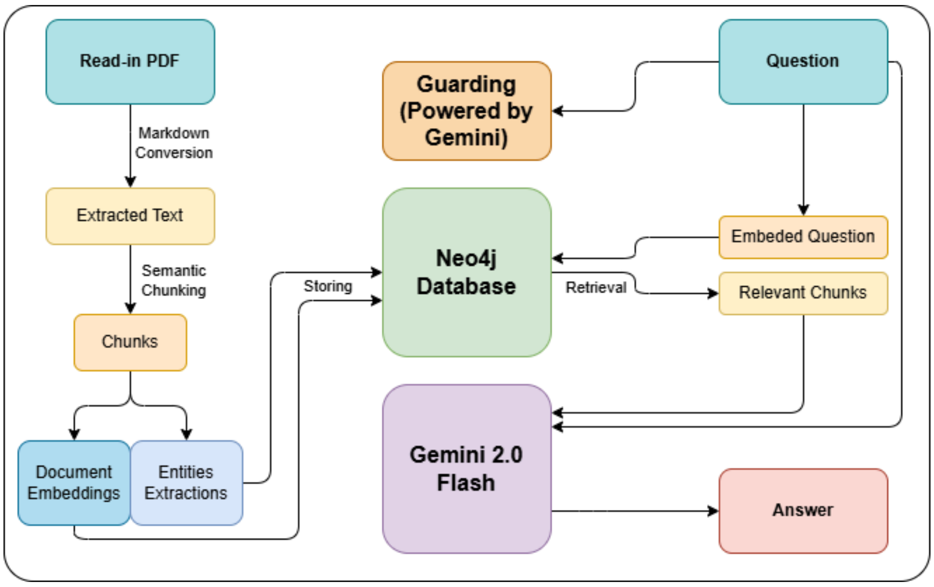

Figure 5: The End-to-end Question-Answering Process

6.3.1) RAG Building Stage (Preprocessing & Indexing)

This stage prepares the knowledge base by reading, processing, and storing document information for later retrieval:

- Read-in PDF: The uploaded PDF is read and converted into markdown format to preserve its structure.
- Extracted Text: The text is extracted and cleaned, ready for further semantic processing.
- Semantic Chunking: The text is broken down into semantically meaningful chunks that serve as retrieval units during the answering stage.
- Embeddings & Entities Extraction:
  - Document Embeddings: Each chunk is encoded into a vector representation using a language model. These embeddings are used for similarity search.
  - Entities Extractions: Named entities (for example, people, places, dates) are identified to enrich the knowledge graph.
- Neo4j Database Storage: All chunks, embeddings, and associated metadata (entities, relationships) are stored in a Neo4j graph database. This setup allows both vector-based and graph-based retrieval of context.

6.3.2) Answering Stage (Retrieval & Generation)

This stage processes user queries and generates answers using the indexed document knowledge:

- User Question Input: The user asks a natural language question through the interface.
- Guarding (Powered by Gemini): A lightweight Gemini model verifies that the question is relevant, safe, and aligned with the document context before proceeding.
- Embedding & Retrieval:
  - The question is embedded into a vector form.
  - Using vector similarity and graph relations, relevant chunks are retrieved from the Neo4j database.
- Gemini 2.0 Flash: The embedded question and the most relevant document chunks are passed to Gemini 2.0 Flash, which generates a coherent and accurate answer based on the provided context.
- Answer Output: The final answer is presented to the user in the chat interface, enabling real-time interactive exploration of the uploaded document.

6.3.3) References Extraction

To support academic use cases, the system includes an AI-assisted citation extraction and formatting module. This process automatically identifies, cleans, and presents references from the uploaded document in a structured way.

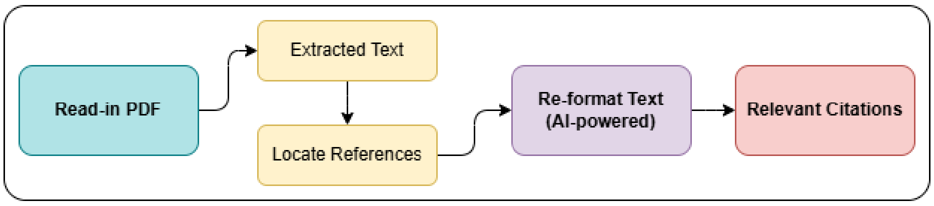

Figure 6: The flow to take out the references in the including paper

Process Steps:

1. Read-in PDF: The document is ingested in its original format, and raw content is extracted.
2. Extracted Text: The full body of text is extracted from the PDF, including references typically found at the end of academic papers.
3. Locate References: A rule-based and AI-assisted engine scans the extracted text to identify the reference section. It then isolates each reference entry for further processing.
4. Re-format Text (AI-powered): Using natural language models, the located references are parsed, cleaned, and converted into a standardized citation format (for example, APA, IEEE, or user-specified style).
5. Relevant Citations Output: The final result is a list of structured, machine-readable citations that can be reused for summaries, QnA responses, or bibliography generation.

7) Evaluation

7.1) Summarization Models Comparison

Model	Training (on GPU)		Inference (in seconds)	
	Num Epochs	Total Time Cost	With GPU	Without GPU
LED	6	≈ 7 hours	12.83	58.76
Long T5	3	≈ 7 hours	16.17	199.59

*Note:

- The model was trained on Kaggle GPU (T4) environment.
- The inference process was performed on a group of paper that contains about 7,000 words (translating into over 10,000 tokens each).
- Inference time was measured on Windows 11 environment with GTX 1650ti (mobile) as GPU and AMD R7 4800h CPU (2.9GHz).

Despite LED undergoing twice the number of epochs, it trains in the same time due to its sparser and more efficient architecture (based on Longformer). During inference, LED significantly outperforms LongT5 — especially on CPU, where LongT5 is very slow due to its decoder-heavy, full-attention design.

Although both LED and LongT5 are capable models for long-document summarization, LED proves to be the more practical choice in real-world scenarios. Despite LongT5's potential benefits—such as its strong performance on generative language tasks and encoder-decoder architecture—it suffers from high computational overhead, especially during inference. This is largely due to its reliance on full attention mechanisms, which scale poorly with long input sequences.

In contrast, LED is designed specifically for long documents using a sparse attention mechanism, making it significantly more efficient in both training and inference time. Its lower latency, especially on CPU-based systems, makes it more suitable for deployment in environments with limited hardware resources.

Therefore, while LongT5 may offer advantages in model expressiveness, LED offers a better trade-off between performance and efficiency for long-document summarization tasks.

All in all, we ultimately chose to use LED, as it offers a more balanced solution in terms of efficiency, scalability, and practicality for real-world summarization tasks involving long documents.

7.2) Summarization Metrics

Metric	**SciSummNet**		**arXiv**	
	Original LED	**Fine-tuned LED**	**Original LED**	**Fine-tuned LED**
ROUGE-1	0.3261	0.6797	0.2875	0.3575
ROUGE-2	0.0935	0.5885	0.0772	0.1045
ROUGE-L	0.1618	0.6343	0.1443	0.1787
ROUGE-Lsum	0.2556	0.6705	0.2145	0.3143
BERTScore Presision	0.8053	0.9288	0.7898	0.8237
BERTScore Recall	0.8443	0.9272	0.8300	0.8392
BERTScore F1	0.8242	0.9275	0.8092	0.8311
BLEURT	−0.8591	0.3810	−0.9282	−0.7313
METEOR	0.2978	0.5714	0.2592	0.2786

Across both SciSummNet and arXiv, the fine-tuned LED model shows consistent improvements over the original, as seen in all ROUGE, BERTScore, BLEURT, and METEOR metrics. This upward trend confirms that the model has learned to generate more relevant and coherent summaries after fine-tuning.

Focusing on the fine-tuned LED, we observe that on SciSummNet, it achieves particularly high ROUGE-1 (0.6797), ROUGE-Lsum (0.6705), and BERTScore F1 (0.9275), indicating strong lexical recall and semantic alignment with reference summaries. BLEURT also rises sharply to 0.3810, reflecting improved fluency and factual consistency. These scores suggest that the model has effectively captured the concise, targeted nature of scientific summaries in SciSummNet and is producing outputs that are both informative and well-structured.

On arXiv, although the scores improve (for example, ROUGE-1 at 0.3575, BERTScore F1 at 0.8311), the absolute values are lower. This reflects the greater difficulty of summarizing long, complex papers with multiple themes and contributions. While the model does demonstrate better semantic understanding (evidenced by high BERTScores), the relatively modest BLEURT (−0.7313) and METEOR (0.2786) scores imply that the generated summaries may still struggle with fluency, faithfulness, or structural coherence.

In summary, the fine-tuned model is highly effective for structured and well-scoped datasets like SciSummNet, but remains limited when applied to broader, more heterogeneous corpora such as arXiv. Further fine-tuning or model adaptation may be necessary for better generalization in such open-ended domains.

7.3) Contextual Question-Answering benchmark

The Gemini-2.0-Flash model was evaluated on the QuAC (Question Answering in Context) dataset[1] to measure its performance in contextual question answering tasks.

Benchmark	**Values**
ROUGE-1	0.2243

[1]From *https://quac.ai/*

ROUGE-2	0.1454
ROUGE-L	0.1999
BERTScore Presision	0.8532
BERTScore Recall	0.8641
BERTScore F1	0.8578
BLEURT	0.3771
METEOR	0.2631

Semantic Understanding (BERTScore):
The BERTScore metrics (F1: 0.8578) are strong, showing that Gemini-2.0-Flash excels at capturing the semantic meaning of the reference answers. This suggests the model understands the context and generates answers that align well with the intended meaning, even if the exact words differ.

Lexical and Structural Similarity (ROUGE and METEOR):
The ROUGE scores (ROUGE-1: 0.2243, ROUGE-2: 0.1454, ROUGE-L: 0.1999) and METEOR score (0.2631) are low. These metrics focus on exact word matches, word order, and synonyms. The low scores indicate that the model's answers often use different wording or sentence structures compared to the reference, even though the meaning is preserved (as shown by BERTScore).

Human-Like Quality (BLEURT):
The BLEURT score of 0.3771 is moderate to low. BLEURT is trained to mimic human judgments of text quality, and this score suggests that while the model captures meaning well (per BERTScore), its answers may lack fluency, coherence, or other qualities that align with human preferences.

Comparison Across Metrics:
The high BERTScore (F1: 0.8578) contrasts with the lower ROUGE, METEOR, and BLEURT scores. This indicates that Gemini-2.0-Flash prioritizes semantic accuracy over lexical fidelity. It likely paraphrases answers in its own style, which preserves meaning but reduces exact word overlap and may affect perceived quality (BLEURT).

ROUGE-2 (0.1454) being the lowest among ROUGE scores suggests particular difficulty in maintaining exact bigram sequences, pointing to issues with precise phrasing or multi-word expressions.

The METEOR score (0.2631) being higher than ROUGE-1 (0.2243) is expected, as METEOR accounts for synonyms and stemming, providing a more flexible evaluation of lexical similarity.

8) Ethical and Privacy Considerations

To ensure responsible AI usage, SummarAIzer incorporates a **prompt safety filtering mechanism**. This uses Gemini to classify user queries as either "safe" or "unsafe" based on a predefined prompt template. Queries deemed unsafe — including those that are offensive, harmful, off-topic, or violating academic integrity (e.g., "write this paper for me") — are blocked automatically before reaching the model.

For **data privacy**, uploaded academic documents **are stored locally** on the host system to enable downstream processing, caching, and performance improvements. These files are not transmitted to external servers unless explicitly used by external APIs (e.g., Gemini), and no long-term cloud storage is involved. All access to locally stored data remains within the application's container or environment, ensuring controlled handling and traceability.

By integrating prompt guarding and secure local storage, SummarAIzer is designed with ethical AI practices and user privacy in mind.

9) Installation and Usage Guideline

For running our application locally, please follow these steps.

1. Clone the repository

```
git clone https://github.com/Gitnut11/Academic-Paper-SummarAIzer.git
cd Academic-Paper-SummarAIzer
```

2. Configure environment

Create a `.env` file with the following:

```
NEO4J_URI="bolt://neo4j:7687"
NEO4J_USERNAME="neo4j"
NEO4J_PASSWORD="your-password"
GEMINI_API_KEY="your-gemini-api-key"
```

You can get your *Google Gemini API here*.

3. Launch Services with Docker

```
docker-compose up --build
```

Services will be accessible at:

- Backend: *http://localhost:8000*
- Frontend: *http://localhost:8501*
- Neo4j Browser: *http://localhost:7474*
- Prometheus Dashboard: *http://localhost:9090*

For using our core application functionalities, here is the link to the *demo video*.

10) Challenges & Limitations

During the development of the Academic Paper Summarizer, several technical challenges were encountered, and the resulting system, while robust, possesses certain limitations.

10.1) Technical challenges faced during development

- Model Performance and Computational Efficiency: Selecting an appropriate model involved balancing performance with computational resources. While the LED model was chosen for its efficiency in handling long sequences, especially compared to alternatives like LongT5 which is very slow on CPU, its inference, particularly for extensive documents requiring chunking, necessitated careful memory management techniques such as chunk-wise processing, safe batching, manual garbage collection, and CUDA cache clearing.
- Semantic Integrity in Document Processing: Techniques like semantic chunking were employed to manage document lengths exceeding model input limits. Ensuring that these chunks, even with overlapping regions, maintained full semantic integrity and avoided information loss at boundaries was a complex task. Similarly, the hierarchical summarization process, where summaries of chunks are further summarized, added a layer of complexity in preserving coherence and the core message of the original document.
- Prompt Engineering and System Safety: Crafting effective prompts for the various LLM interactions, including summarization, Q&A, and the citation strategy, required iterative refinement. Implementing the Prompt Guard mechanism using Gemini 2.0 to filter unsafe or inappropriate inputs was crucial for ethical AI deployment but also added to the development overhead.

10.2) Limitations of the model or system

- Summarization Quality on Heterogeneous Texts: The fine-tuned LED model, despite its strong performance on SciSummNet, exhibited limitations when summarizing papers from the arXiv dataset. Metrics such as BLEURT and METEOR indicated that summaries for these diverse texts might sometimes lack fluency, faithfulness, or optimal structural coherence.
- LLMOps Maturity: The project was implemented based on Level 1 LLMOps maturity. While this provided a structured operational pipeline, it lacks the full automation characteristic of Level 2, such as prompt and model updates or advanced feedback-driven fine-tuning mechanisms.
- Evaluation Scope and Metric Reliance: The evaluation relied on a suite of standard NLP metrics. While informative, these metrics may not fully capture all nuances of summarization quality or user satisfaction. For instance, ROUGE primarily measures lexical overlap and doesn't fully account for paraphrasing or semantic equivalence.

11) Future Work

To build upon the current system and address its limitations, several avenues for future work are proposed:

Enhancing Model Robustness and Generalization:

- Conduct further fine-tuning of the LED summarization model using a more extensive and diverse corpus of scientific papers, potentially incorporating techniques to improve performance on heterogeneous texts like those found in arXiv.
- Explore and evaluate newer large language models and architectures for both summarization and Q&A tasks, which may offer improved performance, better generalization, or greater efficiency.

Advancing LLMOps Capabilities:

- Transition the system towards Level 2 LLMOps maturity by implementing comprehensive CI/CD (Continuous Integration/Continuous Deployment) and CT (Continuous Training) pipelines. This would automate the updating of models and prompts.
- Integrate feedback loops where user interactions and ratings are systematically collected and used for further fine-tuning of models and prompts (for example, through Reinforcement Learning from Human Feedback).
- Develop more sophisticated monitoring dashboards to track a wider array of metrics, including potential model hallucination rates, token consumption patterns, and detailed response quality indicators.

Expanding System Functionality and User Experience:

- Extend support to include a broader range of input document formats beyond PDF.
- Introduce advanced features such as comparative summarization across multiple documents or identifying conflicting information.
- Further leverage the Neo4j graph database for more complex query capabilities and knowledge discovery from the interconnected entities within documents.

Refining Evaluation Methodologies:

- Supplement automated metrics with more extensive human evaluation to gain deeper insights into the perceived quality, utility, and trustworthiness of the generated summaries and answers.
- Research and potentially develop custom evaluation metrics tailored specifically to the nuances of scientific document understanding and summarization.

Strengthening Ethical AI Safeguards:

- Continuously refine and expand the Prompt Guard mechanism to address emerging safety concerns.
- Explore advanced techniques for bias detection and mitigation within the LLMs and investigate methods for more granular PII (Personally Identifiable Information) redaction.

12) Conclusion

This project successfully developed an intelligent system, the Academic Paper Summarizer, designed to automatically summarize complex research papers and facilitate interactive, contextual question-answering. The work underscores the transformative potential of Large Language Models (LLMs) and advanced Natural Language Processing (NLP) techniques in making scientific knowledge more accessible and manageable.

Key findings indicate the effectiveness of the fine-tuned Longformer Encoder-Decoder (LED) model, publicly available as Mels22/led-scisummnet, for the task of scientific paper summarization, particularly on structured datasets like SciSummNet where it achieved significant improvements over its base version across multiple metrics including ROUGE, BERTScore, BLEURT, and METEOR. The chosen LED model also demonstrated superior efficiency compared to alternatives like LongT5, making it a practical choice for this application. Furthermore, the implementation of a Retrieval-Augmented Generation (RAG) architecture, utilizing Gemini embeddings, a Neo4j graph database for knowledge storage, and Gemini 2.0 Flash for answer generation, proved to be a viable approach for providing contextually relevant answers to user queries about the document content. The Q&A system excelled in semantic understanding, though opportunities for improving lexical similarity and fluency were identified.

The adoption of LLMOps principles, even at Level 1 maturity, was instrumental in creating a more structured, scalable, and maintainable system. The project successfully integrated various NLP techniques, including semantic chunking, hierarchical summarization, entity extraction for populating the knowledge graph, and prompt guarding for enhanced safety.

While challenges in model generalization to highly diverse corpora and balancing semantic accuracy with output fluency in Q&A were noted, the overall system represents a significant step towards alleviating the information overload faced by researchers, students, and professionals. By reducing the time and cognitive effort required to digest academic literature, this project contributes to the democratization of scientific knowledge and offers a foundation for future enhancements in AI-driven research tools.

13) References

- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877-1901.
  Note: Referenced in the report for GPT-3 (Brown et al., 2020).
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., … & Zettlemoyer, L. (2019). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.
  Note: Cited in the report for BART (Lewis et al., 2019).
- OpenAI. (2023). GPT-4 technical report. arXiv preprint arXiv:2303.08774.
  Note: Referenced in the report for GPT-4 (OpenAI, 2023).
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., … & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1-67.
  Note: Cited in the report for T5 (Raffel et al., 2020).
- Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.
  Note: Implied reference for the Longformer Encoder-Decoder (LED) model used in the project.
- Yasunaga, M., Leskovec, J., & Liang, P. (2021). SciSummNet: A large-scale dataset for scientific paper summarization. arXiv preprint arXiv:2104.03934.
  Note: Implied reference for the SciSummNet dataset used for fine-tuning and evaluation.

- Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., & Goharian, N. (2018). A discourse-aware attention model for abstractive summarization of long documents. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 615-621.
  Note: Implied reference for related work on long-document summarization techniques.
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., … & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.
  Note: Implied reference for the RAG (Retrieval-Augmented Generation) framework used in the Q&A system.
- Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). spaCy: Industrial-strength natural language processing in Python. Zenodo. *https://doi.org/10.5281/zenodo.1212303*
  Note: Implied reference for the spaCy library used for entity extraction.
- Google Cloud. (2023). LLMOps: Operationalizing large language models. Retrieved from *https://cloud.google.com/architecture/llmops*
  Note: Implied reference for the Google Cloud LLMOps framework cited in the report.
- Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, 74-81.
  Note: Implied reference for the ROUGE evaluation metric.
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating text generation with BERT. International Conference on Learning Representations (ICLR).
  Note: Implied reference for the BERTScore evaluation metric.
- Sellam, T., Das, D., & Parikh, A. P. (2020). BLEURT: Learning robust metrics for text generation. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 7881-7892.
  Note: Implied reference for the BLEURT evaluation metric.
- Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, 65-72.
  Note: Implied reference for the METEOR evaluation metric.
- Choi, E., He, H., Iyyer, M., Yatskar, M., Choi, Y., Liang, P., & Zettlemoyer, L. (2018). QuAC: Question answering in context. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2174-2184.
  Note: Implied reference for the QuAC dataset used for Q&A evaluation.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., … & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.
  Note: Implied reference for the LoRA (Low-Rank Adaptation) fine-tuning technique.
- Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. arXiv preprint arXiv:2305.14314.
  Note: Implied reference for quantization techniques (e.g., 4-bit quantization) used in fine-tuning.
- Neo4j. (2023). Neo4j graph database documentation. Retrieved from *https://neo4j.com/docs/*
  Note: Implied reference for the Neo4j graph database used in the RAG system.
- LangChain. (2023). LangChain documentation. Retrieved from *https://python.langchain.com/docs/*
  Note: Implied reference for the LangChain framework used for embedding generation.
- Semantic Scholar. (2023). Semantic Scholar API documentation. Retrieved from *https://www.semanticscholar.org/product/api*
  Note: Implied reference for the Semantic Scholar API mentioned in related work.