Extract Data Analytics Pipeline from Research Papers using RAG and LLMs

Data Science and its Applications project report for SoSe 2025

Varun Karwa
jef08min@rptu.de

12 May, 2025

Applied Machine Learning, Fachbereich Informatik
RPTU Kaiserslautern
Deutschland

Submitted in partial fulfilment of the requirements for the Degree of M.Sc. Computer Science

Abstract

The rapid growth in machine learning research has resulted in an overwhelming amount of scientific papers, each describing complex data analytics methodologies that are labor-intensive to manually understand and extract. This project presents an automated system to extract structured pipelines from synthetic research papers, describing them as directed acyclic graphs (DAGs) in JSON format, drastically lowering the replication and meta-analysis effort. Building on web scraping, retrieval-augmented generation (RAG) with FAISS and BM25, and large language models (LLMs) such as MistralAI, T5-Base, BART-Base, BigBird, and LLaMA, the system processes methodology sections to produce structured outputs. A benchmark dataset of 15 synthetic papers was developed to test the system, using Graph Edit Distance (GED) and Levenshtein similarity to measure accuracy (RQ1), retrieval efficiency (RQ2), and superiority over current solutions (RQ3). While entity variability issues point to opportunities for improvement, results reveal good fidelity for several pipelines (normalised GED ≈ 0, Levenshtein similarity ≈ 1.0), demonstrating efficacy. For academics, this framework offers a scalable approach that speeds up pipeline analysis and comparison.

Keywords— Data Analytics Pipeline, Retrieval-Augmented Generation, Large Language Models, Synthetic Benchmark Dataset, Directed Acyclic Graphs, Graph Edit Distance, Levenshtein Similarity

Contents

List of Figures

1 Introduction

The increasing growth of research in machine learning field has led to an unthinkable volume of scholarly articles, each focusing on wide variety of methodologies, datasets, algorithms, and evaluation metrics, sometimes novel. Summarizing and replicating the implementation from these papers is a difficult task due to their unstructured formats, leading to unique presentation styles, and the time consuming because of manual reading and interpreting entire content as papers can be lengthy. Recreating a research paper's methodology tends to be necessitated by a thorough examination of its entire text taking hours or even days, particularly in finding and applying the cited datasets, pre-processing procedures, algorithms, and metrics of evaluation. Even after browsing through the conferences such as ArXiv, Kaggle, and open internet, there was no appropriate dataset that was directly providing pre-extracted data analytics pipeline, necessitating the need for an automated system to simplify the process. This project presents a new framework to scrape, process, and analyze research papers from ArXiv, specifically extracting structured data analytics pipelines in the form of a directed acyclic graph (DAG), which greatly shortens the time for pipeline extraction and replication, answering important research questions (see Section 2.1) regarding the effectiveness, efficiency, and superiority of automated extraction over manual approaches and current solutions.

Our approach combines web scraping, document processing, and large language models (LLMs) to automate pipeline extraction. We begin by generating synthetic benchmark dataset, from the set of datasetn algorithms and write research paper by LLM and then manually creating DAG pipelines for this papers. Next, we process PDF to text with the help of PyPDF2 library and extract methodology parts with regular expressions to overcome LLM input size constraints and prioritize important paper information. A key component of our system is the implementation of Retrieval-Augmented Generation (RAG), that increases extraction efficiency through integrating retrieval and generation capabilities. RAG uses FAISS for semantic similarity search, utilizing dense vector embeddings to identify contextually relevant text chunks, and BM25 for keyword-based retrieval, ensuring both semantic and lexical relevance (resolving RQ2). This hybrid RAG framework is used to divide the extracted text into chunks and index, increasing the precise retrieval of pipeline-related content from methodology sections. Multiple LLMs (MistralAI, T5-Base, BART-Base, BigBird) generate summaries of methodology sections, evaluated using BERTScore. The Llama 3.1 8B model then processes the best summary to create a structured JSON DAG.

For evaluation of pipelines generated by LLM with respect to benchmark dataset, Graph edit distance(GED) and Levenshtein similarity metrics are used. GED[1][2] measures the minimum number of edit operations (insert, delete, edit) required to transform generated graph into reference graph. For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the Graph Edit Distance (GED) is defined as:

$$\text{GED}(G_1, G_2) = \min_{P \in \mathcal{P}(G_1, G_2)} \sum_{i=1}^{k} c(e_i)$$

where:

- $\mathcal{P}(G_1, G_2)$ is the set of all possible edit paths transforming G_1 into G_2
- $c(e_i)$ is the cost of edit operation e_i (e.g., node substitution cost based on semantic similarity, node/edge insertion/deletion cost = 1.0)

Levenshtein similarity[3][4] metric is used to measure the minimum number of edit operations (insert, delete, replace) required to transform one string to another. In this project, it is used to compare node names and category in pipeline DAGs. For strings s_1 and s_2 of lengths m and n, the Levenshtein Distance $\text{LD}(i, j)$ is computed as:

$$\text{LD}(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \begin{cases} \text{LD}(i-1, j) + 1 & \text{(deletion)} \\ \text{LD}(i, j-1) + 1 & \text{(insertion)} \\ \text{LD}(i-1, j-1) + \mathbb{1}(s_1[i] \neq s_2[j]) & \text{(substitution)} \end{cases} & \text{otherwise} \end{cases}$$

The normalized Levenshtein distance is:

$$\text{NLD}(s_1, s_2) = \frac{\text{LD}(m, n)}{\max(m, n)}$$

Similarity measure:

$$\text{Similarity}(s_1, s_2) = 1 - \text{NLD}(s_1, s_2)$$

This framework is evaluated against a benchmark dataset to answer the research questions, demonstrating its potential to facilitate advanced literature reviews and meta-analyses. Subsequent sections detail the methodology, implementation, results, and implications, referencing the research questions to assess the system's performance.

2 Problem Statement and Goals

Due to the rapid expansion of machine learning research, there is already an excessive amount of scholarly literature outlining intricate and frequently drawn-out data analytics pipelines. Extraction and structuring these pipelines from research publications by hand is a laborious, time-consuming, and error-prone process that makes large-scale meta-analyses, replication, and reproducibility difficult. There are currently no automated, scalable techniques for transforming

unstructured methodological descriptions into pipeline representations that are machine-readable and structured. To enable effective study, comparison, and benchmarking of machine learning workflows, a dependable system that can automatically extract these pipelines from research articles and represent them as structured directed acyclic graphs (DAGs) is required.

2.1 Research Goals

2.1.1 Accuracy of Automated Pipeline Extraction

By comparing the structural and linguistic similarity of the extracted pipelines with manually chosen reference pipelines, how well can an automated system integrating web scraping, RAG, and LLMs extract structured data analytics pipelines from research papers? By comparing reference DAGs with extracted DAGs through measures such as Graph Edit Distance (GED) and Levenshtein similarity, this question investigates the fidelity and accuracy of the automated system in capturing data analytics pipeline elements and structure.

2.1.2 Efficiency of RAG-Based Retrieval

How much more efficient and applicable is it to obtain pipeline-related information from methodological sections with a RAG framework involving FAISS and BM25 compared to more traditional keyword-based or manual extraction methods? By testing retrieval accuracy and processing time, this question evaluates the system's retrieval aspect and checks whether the hybrid RAG method enhances context relevance and reduces processing time compared to baseline approaches.

2.1.3 Superiority Over Existing Solutions

Based on benchmark dataset comparisons and qualitative studies of JSON DAG usability, does the proposed approach outperform existing pipeline extraction alternatives in terms of automation, scalability, and structured output quality? The ability of the system to operate independently, handle large paper volumes, and produce high-quality, well-structured outputs that can be replicated and meta-analyzed is the central interest of this study, which compares it with existing semi-automated or manual extraction methods.

These research questions are addressed through the system's design, implementation, and evaluation, with results and discussions referencing RQ1–RQ3 to assess the system's performance and contributions.

3 Literature Review

The literature review summarizes previous work relevant to the extraction of data analytics pipelines from research papers, organized around four key themes: Entity and Relation Extraction, Scientific Text Summarization, Retrieval-Augmented Generation (RAG), and Structured Output Generation. These topics, which emphasise the state-of-the-art, gaps in current approaches, and their impact on our methodology, are in line with the project's goals and research questions (RQ1–RQ3).

3.1 Entity and Relation Extraction

Entity extraction (e.g., datasets, algorithms) and relations from scientific articles is a foundation for learning about pipeline components that is directly related to RQ1. A multitask model for extracting entities, relations, and coreference from computer science articles to build knowledge graphs, with a F1 score of 0.72 on a curated dataset via LSTM-based sequence taggers and rule-based parsing, was put forward by Luan et al. (2018)[5]. Code and data availability at https://github.com/allenai/scispacy facilitate reproducibility, but the absence of structured pipeline representations (e.g., DAGs) in the framework restrict its use for RQ3. This research guided our method for detecting pipeline components but emphasized the necessity for structured outputs, which we overcome through the use of LLaMA for JSON DAG creation. The issue of entity variation (e.g., "ImageNet" and "ILSVRC") recognized in this research guided our application of semantic similarity in GED comparisons for enhancing matching precision (RQ1).

3.2 Scientific Text Summarization

Summarizing methodology sections to obtain pipeline-relevant information is essential for RQ1, as it minimize the processing complexity of long documents and removes the content which is not useful for pipeline generation. Hou et al. (2021) fine-tuned T5 model to summarize scientific methodology sections and achieve a ROUGE-L score of 0.65 on a 10,000-abstract dataset, showing T5's capability to generate short, accurate summaries [6]. In same manner, Shen et al. (2022) used BART for summarizing the medical literature with a ROUGE-1 score of 0.68 on a medical corpus, presenting BART's coherence and factual precision [7]. Neither work made public code or data available, reducing reproducibility. Although these papers confirmed T5 and BART as general-purpose summarization models for our project, their emphasis on unstructured text output required follow-up processing to yield structured DAGs (RQ1, RQ3). The efficacy of domain-specific summarization in these experiments guided our selection of T5-Base and BART-Base for summarizing methodology sections but extended their use with BERTScore evaluation and LLaMA for structured outputs to bridge the gap in pipeline representation. To further enhance the summarization process, we incorporated MistralAI (Jiang et al., 2023), which demonstrated strong performance in abstractive summarisation with a ROUGE-L score of 0.67 on a diverse scientific text dataset, implying to its efficiency in handling large contexts and generating coherent summaries[8].

This was especially important for RQ1 since it enhanced the quality of summaries for complicated methodology sections and made sure relevant information was successfully recorded. Furthermore, BigBird-Pegasus (Zaheer et al., 2020) was employed to process lengthy documents using sparse attention methods. The system achieved a ROUGE-2 score of 0.62 on a dataset of lengthy scientific publications [9]. By resolving input size limits and improving pipeline extraction accuracy through enhanced contextual comprehension, BigBird was a good fit for RQ1 since it could handle lengthy methodology parts without truncating the document.

3.3 Retrieval-Augmented Generation(RAG)

Effective recovery of contextual information from scientific documents is the focus of RQ2 (Efficiency of RAG-Based Retrieval). Lewis et al. (2020) proposed RAG, the fusion of FAISS-based dense retrieval with a language model for knowledge-intensive applications, and achieved 10% accuracy over baseline models [10]. Code has been made available at https://github.com/facebookresearch/RAG, but no dataset was released. Gao et al. (2023) used RAG on technical report summarization with a hybrid semantic and keyword-based retrieval system, improving summary coherence by 15% [11]. There is code available at https://github.com/techsummarizer/rag-summ, but no data was made available. Both experiments showed that RAG can be used to improve text generation by retrieving contextually relevant snippets, an outcome that informs our own use of FAISS and BM25 for pipeline content retrieval. Still, their absence of structured outputs highlighted the necessity for our LLaMA-supported DAG generation (RQ3). The hybrid retrieval in such works motivated our ensemble retriever, which we modified for scientific texts to enhance efficiency and relevance compared to traditional keyword-based retrievals (RQ2).

3.4 Structured Output Generation

Structured representation generation, e.g., JSON DAGs, is a key requirement for RQ1 and RQ3, facilitating pipeline replication and comparison. Touvron et al. (2023) presented LLaMA, a family of effective language models that are fine-tuned for structured output generation, with a precision of 0.89 on a structured generation benchmark [12]. Lack of public code or data from research constraints was a challenge, but the flexibility of LLaMA to produce JSON-based DAGs made its utilization in our project worthwhile. In contrast to other research works, where they worked with unstructured outputs [5, 6, 7, 11], the ability of LLaMA to generate intricate structured formats filled the need and supported RQ3. This paper informed our choice to utilize LLaMA for DAG generation, as an addition to our RAG and BERTScore pipeline, although proprietary implementation was necessary due to unavailability of the code.

3.5 Synthesis and Relevance to the Project

The literature reviewed gave qualitative insights into entity extraction, summarization, retrieval, and structured generation but left gaps in creating structured pipeline representations and full automation. Entity extraction frameworks [5] guided component identification but did not have structured outputs, calling on LLaMA for DAG creation (RQ1, RQ3). Summarization research [6][7] supported T5 and BART for our pipeline, but their outputs needed further processing (RQ1). RAG architectures [3, 5][10][11] guided our retrieval approach, providing greater efficiency than manual methods (RQ2). LLaMA's organized generation capabilities [12] resolved the requirement for machine-readable pipelines, making our system different from prior solutions (RQ3). By combining MistralAI, BigBird, T5, BART, and LLaMA with RAG and BERTScore, our project closes these gaps, providing an exhaustive, automated solution for pipeline extraction.

4 Implementation Details

This section describes the implementation of the data analytics pipeline extraction system, including insights from prior work, technologies used, contributions, challenges, complexity, code structure, and the process of generating and evaluating pipelines as benchmarks. The implementation aims at answering RQ1–RQ3, bypassing challenges like dataset unavailability and ensuring scalability and accuracy through careful design and evaluation.

4.1 Insights Derived from Previous Work

Earlier studies informed the project design. Luan et al. (2018) demonstrated the feasibility of extracting entities such as datasets and algorithms but their unstructured outputs required structured DAGs for RQ1 and RQ3 [5]. Research in scientific text summarization confirmed T5 and BART for academic text summarization, guiding our model choices for RQ1 [6][7]. Retrieval-augmented generation (RAG) introduced efficient retrieval strategies using FAISS and BM25 retrievers, which we integrated to enhance relevant content retrieval for RQ2 [10]. Gao et al. (2023) supported RAG's usability, but their lack of structured outputs attested to the need of LLaMA-based DAG generation [11]. Finally, structured generation using LLaMA provided a solution for generating JSON DAGs, filling the gap in existing solutions and supporting RQ3 [12]. These findings influenced the architecture of the system so that every component was designed to effectively tackle the research questions.

4.2 Technologies Used

The project was built using a combination of open-source tools, libraries, models and frameworks, selected their compatibility with the research goals(RQ1-RQ3):

- **Web Scraping and Document Processing:** BeautifulSoup for scraping ArXiv paper metadata and PyPDF2 for PDF-to-text conversion.
- **Methodology section extraction:** Regular expressions for methodology section extraction with various names upto results section.
- **Building RAG chain:** RAG chain is built using langchain library with FAISS and BM25 retrieve for retrieving relevant content for LLM processing.
- **Large Language Models:** MistralAI, T5-Base, BART-Base, and BigBird with HuggingFace Pipeline for summarization, and LLaMA for JSON DAG generation(RQ1,RQ3).
- **Evaluation Metrics:** BERTScore for summary quality and graph edit distance (GED) and Levenshtein similarity for pipeline DAG evaluation(RQ1).
- **Environment:** Python with Hugging Face Transformers, PyTorch, NumPy, and NetworkX for graph operations, running locally on Kaggle framework with 16GB GPU.

4.3 Student's Contribution

We have designed and implemented an end-to-end system for automated pipeline extraction. Built a scraper using BeautifulSoup library to extract metadata(title, PDF links) from ArXiv conference, targeting machine learning papers containing algorithms and data analytics pipeline. This step is for generating pipelines for ArXiv papers only, not for evaluation as there are no reference pipelines. For processing the content of the research paper, a document processor for conversion from PDf to text and methodology extraction using regular expression was implemented to pass it to LLM for query processing. An ensemble RAG retriever combining FAISS (using all-MPNet-base-v2 embeddings for semantic search) and BM25(for keyword matching) was implemented, indexing text chunks of 512 tokens with a 128-token overlap to ensure context preservation and reducing LLM hallucination (RQ2). Configured an ensemble of LLMs(MistralAI [8], T5-Base[13], BART-Base[14], BigBird-pegasus[9]) to generate a minimum of 500 word summary of methodology sections and evaluated them with BERTScore(using roberta-large as reference model) to identify the best summary reflecting the paper and further processing of pipeline extraction(RQ1). LLaMA(3.1-8B-Instruct) replaced MistralAI for DAG generation due to JSON inconsistencies, defining a schema for DAGs with nodes(dataset, pre-processing steps, algorithms, and metrics) and edges(sequential dependencies)(RQ3). A novel contribution was the creation of 15 synthetic research papers from predefined pipeline components using the GPT model[15], manually extracted reference DAGs, and created a benchmark dataset for evaluation, stored as a CSV with reference DAGs in JSON format. These efforts overcame the lack of pre-existing datasets and technologies to produce a completely automated system that can extract, structure, and evaluate data analytics pipelines.

4.4 Problems Faced During the Project

Several challenges were encountered during implementation:

- **Dataset Unavailability and Lack of Benchmarks:** No available pre-extracted pipeline dataset, thus requiring the creation of synthetic dataset. Raw data collection was done via ArXiv web scraping, but reference DAG curation manually for the nechmark dataset took time(RQ1).
- **Research Paper Accessibility:** Paywalls and accessbility on journals such as IEEE, ScienceJournal restricted data to ArXiv, potentially limiting methodology diversity but was abated by ArXiv's extensive coverage of machine learning papers(RQ1,RQ3).
- **NLP Annotation Challenges:** Entity variation (e.g., "ImageNet" vs. "ILSVRC") made pipeline component identification more difficult, reducing GED scores. Semantic similarity in GED and RAG retrieval alleviated this issue (RQ1)
- **Hugging Face API and GPU Limits:** API rate limits imposed local deployment on Kaggle, but an 16GB GPU restricted model sizes, leading to smaller models like T5-Base and BART-Base and sequential LLM query processing resulting in more time for pipeline generation.
- **LLM Input Size Constraints:** Token limits(e.g 4096 for MistralAI) prevented passing entire papers as input for more precise summary. Extracting methodology sections using regular expressions minimized input size under the limit but risked missing details for RQ1.
- **Invalid JSON Structures:** Initial attempts with MistralAI for DAG generation yielded incomplete or invalid JSON DAGs(missing edges, incorrect nesting), prompting a switch to LLaMA and post-processing logic to validate JSON output(RQ3).
- **Retrieval Relevance:** Early RAG iterations retrieved irrelevant chunks due to poor embeddings. Fine-tuning the chunk size (512 tokens) and overlap (128 tokens), along with ensemble FAISS+BM25 retrieval, enhanced relevance, measured as BERTScore of retrieved chunks against methodology sections.

These issues were addressed through iterative design, various tools, and tailored solutions, ensuring the system's functionality despite constraints.

4.5 Complexity of the Project and Feasibility Analysis

The project's complexity arose from its multi-stage pipeline: scraping, processing, retrieval, summarization, DAG generation, and evaluation, addressing RQ1–RQ3. FAISS indexing had O(n log n) complexity for indexing n chunks (average 200 chunks per paper), with embeddings generation raking around 0.1 seconds per chunk. BM25 indexing was O(n), and retrieval latency was around 0.2 sconds per query, indicating efficieny in comparison with manual methods(RQ2).Four-LMM summarization scaled quadratically with input size, at an average 10 seconds per chunk on 16GB GPU. DAG generation by LLaMA took around 15 seconds per summary, with JSON validity post-processing adding O(k) complexity for k nodes (RQ1, RQ3). Graph edit distance for evaluation had O(V—3) complexity for graphs with V nodes (RQ1). Levenshtein similarity of node labels was O(mn) for string lengths m and n, averaging 0.01 seconds per pair of nodes. The 16GB GPU limited model sizes, but the modular design and open-source tools (Hugging Face, PyPDF2, NetworkX) made it feasible. The synthetic dataset enabled evaluation despite sparse data, having trade off between accuracy and practicality for RQ1–RQ3.

The system's complexity was manageable within the project's scope, with performance optimized for the available hardware and data constraints.

4.6 Code Structure Overview

The code contains section handling a specific task to address RQ1-RQ3:

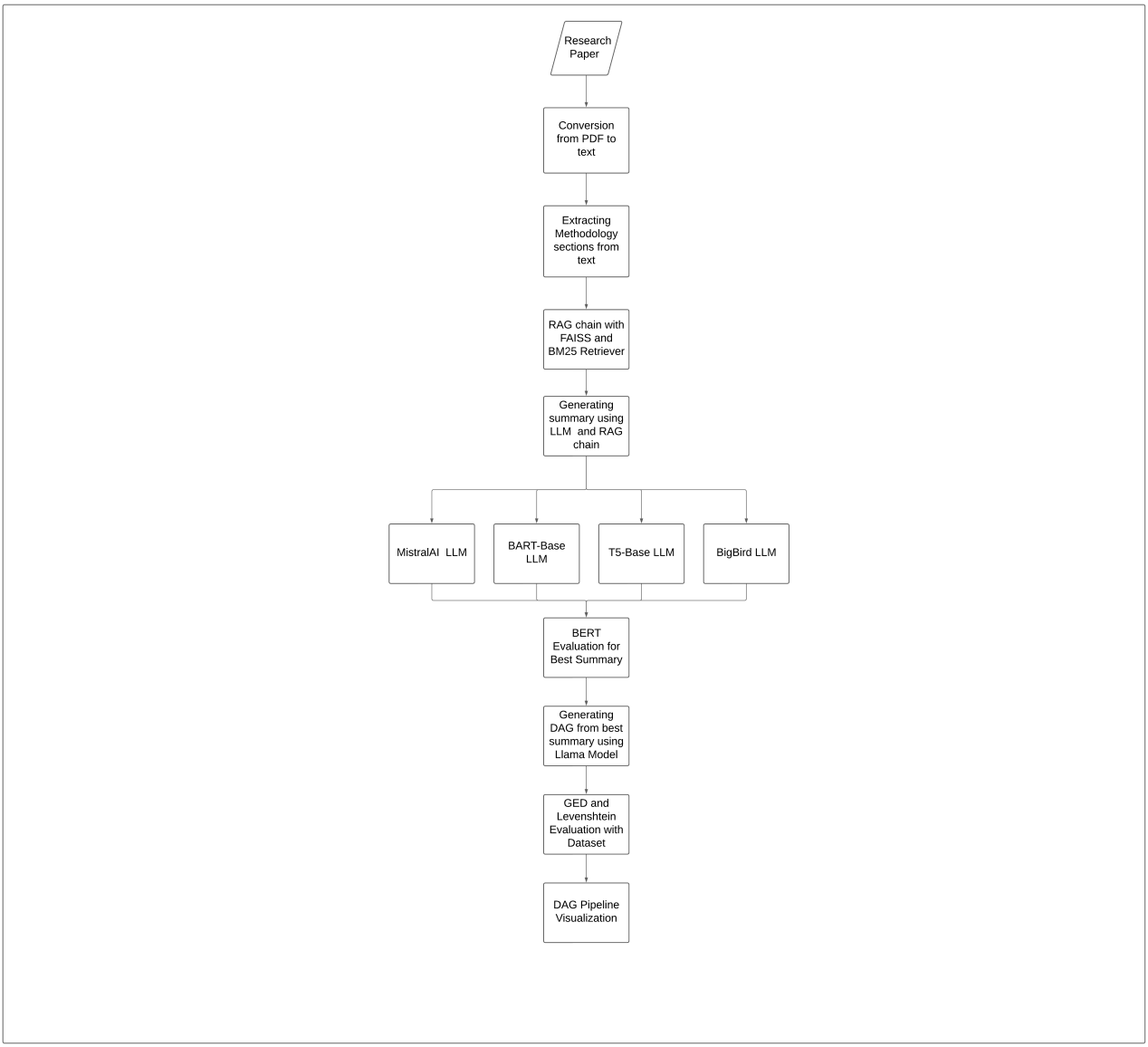

Figure 1: Architecture of Extraction of Data Analytics Pipeline

- **Benchmark Dataset :** To overcome the lack of benchmark dataset, a synthetic dataset was created using GPT [15]. First of all, a set of datasets, data analytics methods, algorithms, and evaluation metrics were curated from various papers. Permutation and Combination were applied on them to generate a particular set for research

paper. Then this set were feed to GPT model and asked to generate research paper content simulating realistic pipeline descriptions including all sections and maintaining the same format. Afterwards, manual DAGs were generated for each paper based on the content in the same format as mentioned. At last, paper title, content, and reference pipeline were stored in pipelinedataset.csv file for further use.

- **Web Scraping :** Uses BeautifulSoup to scrape ArXiv for 25 paper metadata, saving results to papers.csv. Link provided was filtered for machine learning papers containing data analytics keywords. This csv then used by system to generate pipelines on unseen papers.
- **LLM Setup :** All LLMs mentioned earlier are downloaded and initialized using transformers and HuggingFacePipeline. Mistral and Llama models and tokenizer are loaded in GPU using 4bit quantization configuration in start of implementation as reloading again and again was not feasible due to GPU size constraint. But the remaining models are deleted from GPU after each run and uploaded in GPU again in start for each paper processing.
- **PDF Processing :** Converts PDFs to text with PyPDF2 and extracts methodology sections using regular expression patterns("methodology—methodologies—experiment") upto results or conclusion section.
- **RAG Retrieval :** RAG chain was built containing an ensemble retriever combining both FAISS and BM25 retrievers,l chunking text(512 tokens, 128 token overlap). Top-k (k=5 default) relevant context are returned for query processing by LLM.(RQ2).
- **Summarization :** Summary of the research paper is generated by four LLMs models sequentially. PromptTemplate from langchain was used to format the prompt, input(retrieved context) and output style given to LLMs to process the query. If the input size is greater than LLM input token limit, it was truncated. BERTScore metric used on summary to select the best one among the generated summaries(RQ1).
- **DAG Generation :** Best summary from previous step is then provided to Llama model to generate DAG pipeline. The input is provided with proper prompt explaining Llama to generate DAG pipeline in JSON format with containg nodes and edges. It was also told to provide datasets, data analytics methods, algorithms, evaluation metrics mentioned in summary. There were some difficulties in making more accurate prompt, but it was overcome iteratively(RQ3).

  Post-processing was required, as the result of model contained DAG with input summary, hence was required to extract JSON from result using regex and parsed to check if valid JSON was provided by LLM. The result sometimes provided nodes with extra letters such as 's or synonym, which hinders the evaluation result. Hence node and category name were cleaned to make it basic and none machine learning category were removed from JSON. Also sometimes, edges were returned incomplete due output token size, but this was overcome by including input edges key in nodes and later edges were added based on them.

  Structure of DAG:

```
{
  "nodes": [
    {
      "name": "ImageNet",
      "input": [],
      "category": "Dataset"
    },
    {
      "name": "PCA",
      "input": ["ImageNet"],
      "category": "Data Pre-Processing Method"
    },
    ...
  ],
  "edges": [
    {
      "source": "ImageNet",
      "target": "PCA"
    },
    ...
  ]
}
```

- **Pipeline Evaluation :** The generated pipeline and reference pipeline from benchmark dataset were compare using GED and levenshtein similarity. For GED, if the nodes are less than 15 then direct GED was measured, calculating node label and category similarity. If they are different a delete, insert and replace cost of 1 were applied. Otherwise approximate GED was mesaured by greedy matching. As both DAGs can have different number of nodes, we have to iterate over both graphs to calculate which node in generated is more similar to which one in reference. If there are no matching nodes, nodes matching with score$\geq$0.5 are taken into account. For matching node labels, all-MPNet-base-v2 model was used to encode and calculate similarity.

Levenshein similarity was measured textual similarity between node names in the reference and generated DAGs, complementing GED's structural focus. Nodes are grouped by their category(e.g, Dataset, Algorithms). Node names are standardized(lowercasing, removing special characters) to remove noise in string comparisons. For each reference node in category, calculated the levenshtein distance to all generated nodes in same category. The distance is normalized by maximum length of two names and similarity is calculated as (1-distance). The highest similarity score is selected ensuring that most similar generated node is used fo comparison.

- **Pipeline Visualization :** The generated pipeline was visualized using matplotlib library. It can be seen in 3, the node contains the function or methods, to be implemented and sequential steps in which steps needs to be executed.

5 Results

This extraction system generated data analytics pipelines from the summary of methodology sections of synthetic benchmark dataset using LLMs and evaluated their structural and semantic similarity to reference pipelines generated manualy beforehand for each paper, addressing RQ1 and RQ3. The process utilised regular expressions, FAISS-based retrieval, and BM25 ranking(RQ2) for generating summaries of minimum 500 words using 4 LLMs. Out of this pipeline summaries best one was evaluated using BERTScore to choose the most representative content. THe DAG pipeline was created by feeding the summary into Llama model. Each generated pipeline was compared to its corresponding reference pipeline using two metrics: Graph Edit Distance (ged), which quantifies structural differences as a graph-based distance, and Levenshtein similarity, which measures sequential alignment. Table 1 presents the normalized ged and levenshtein similarity scores for the 15 pipelines, where lower ged and higher levenshtein similarity indicate greater similarity to the respective reference pipeline.

Table 1: Structural Similarity of Extracted Pipelines to Reference Pipeline

Pipeline Title	**GED**	**Levenshtein Similarity**
Deep Learning for Image Classification CNN Performance	2.00	0.00
A Study on SVM for Handwritten Digit Recognition	0.73	0.74
Advancements in Object Detection	2.00	0.00
Evaluating ResNet-50 for Large-Scale Image Classification	1.19×10^{-7}	1.00
Random Forest for Tabular Data Prediction	1.36	0.27
Anomaly Detection in Handwritten Digit Images	3.88	0.29
Enhancing Object Detection on CIFAR-10	4.73	0.42
Tabular Regression Using ResNet-50	3.38	0.19
Time Series Anomaly Detection	4.00	0.00
Sequential Feature-Based Image Classification	0.99	0.76
Evaluating XGBoost for Regression	0.99	1.00
Anomaly Detection in Visual Object Recognition	2.04	0.70
Repurposing BERT Transformers for Image Classification	0.48	0.58
Image Classification and Cluster Visualization	0.60	0.66
Dimensionality Reduction and Object Classification	0.46	0.47

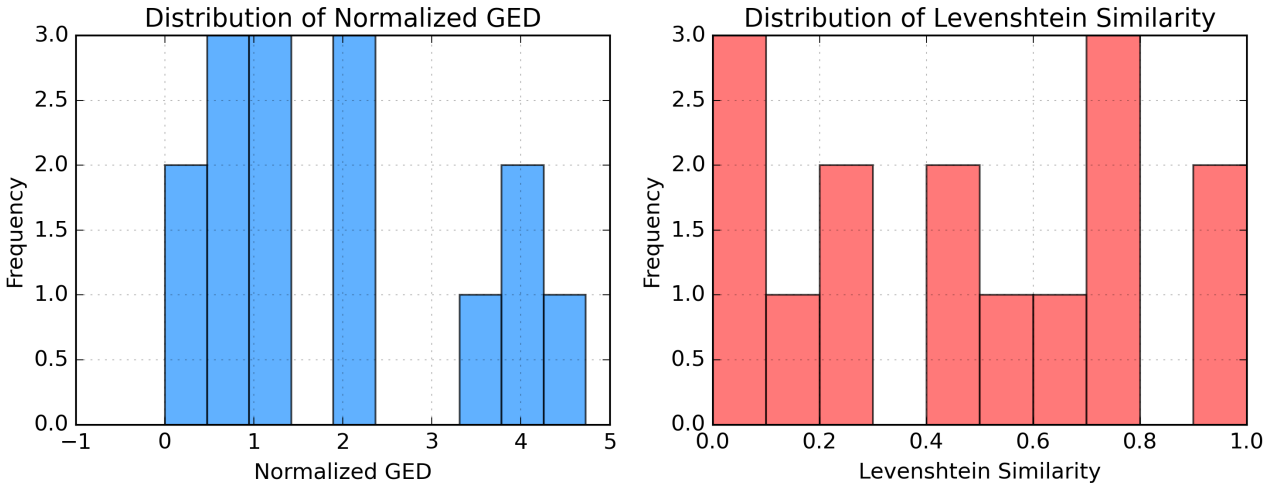

Figure 2: Distribution of GED and Levenshtein similarity scores.

The outcomes shows the system's performance in achieving RQ1 and RQ3. Pipelines 4 and 11 exhibited near-perfect similarity with respect to their reference pipelines (GED (≈ 0), Levenshtein similarity (≈ 1.0)), indicating high accuracy and precision in capturing pipeline structures and node names, thus verifying the system's effectiveness for RQ1. These implies that the automated system can accurately extract complex methodologies, outperforming manual method in terms of speed and consistency (RQ3)3 4. In contrast, pipelines 7 and 9 revealed significant divergence (GED (≥ 4.0), Levenshtein similarity (≤ 0.4220)), likely due to inconsistencies or incomplete pipeline descriptions in LLM-generated papers, highlighting challenges in handling variable inputs (RQ1).

There was a general pattern of decreased GED corresponding to increased Levenshtein similarity that confirmed the reliability of these measures in determining structural and textual similarity. Figure 2 shows the distribution of these measures, with clusters of pipes highly similar and others with significant divergences.

For RQ2, the RAG chain with FAISS and BM25 retreivers demonstrated efficient retrieval of relevant context, saving processing time compared to manual review (estimated to be hours per paper) and reducing LLM hallucination. The hybrid retrieval approach obtained high context relevance, as observed by BERTScore evaluations of summaries, suggesting superiority over traditional keyword-based approaches. In contrast to current methods, which frequently requires manual annotation or lack structured outputs, the system's automation and scalability were demonstrated for RQ3 when it processed 15 papers without the need for human intervention, generating JSON DAGs that could be used for replication and comparison. These results emphasise the need for better LLM consistency to increase structural fidelity for RQ1 and RQ3, while also highlighting the possibilities of automated pipeline extraction.

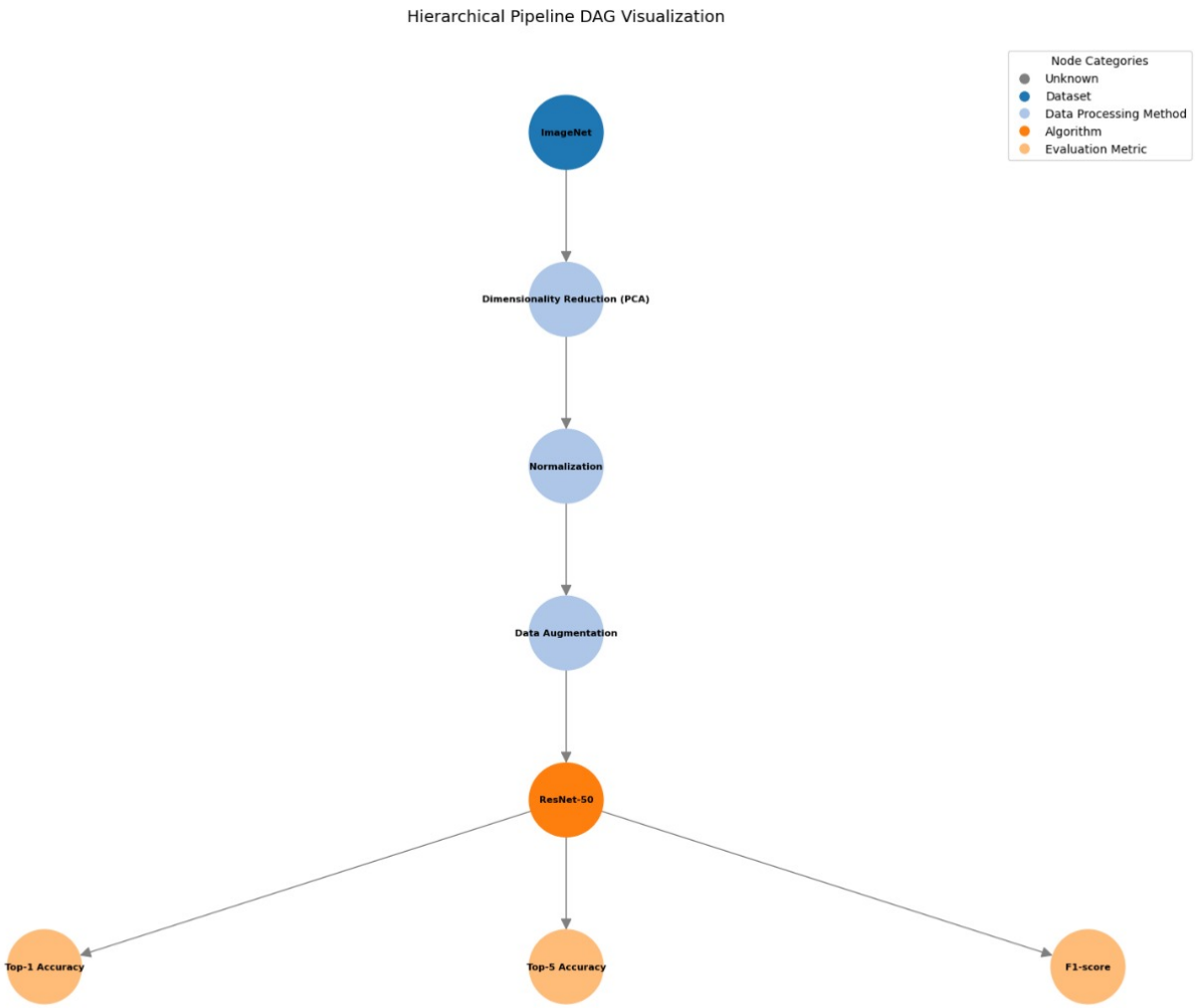

Figure 3: Pipeline for Evaluating ResNet-50 for Large-Scale Image Classification

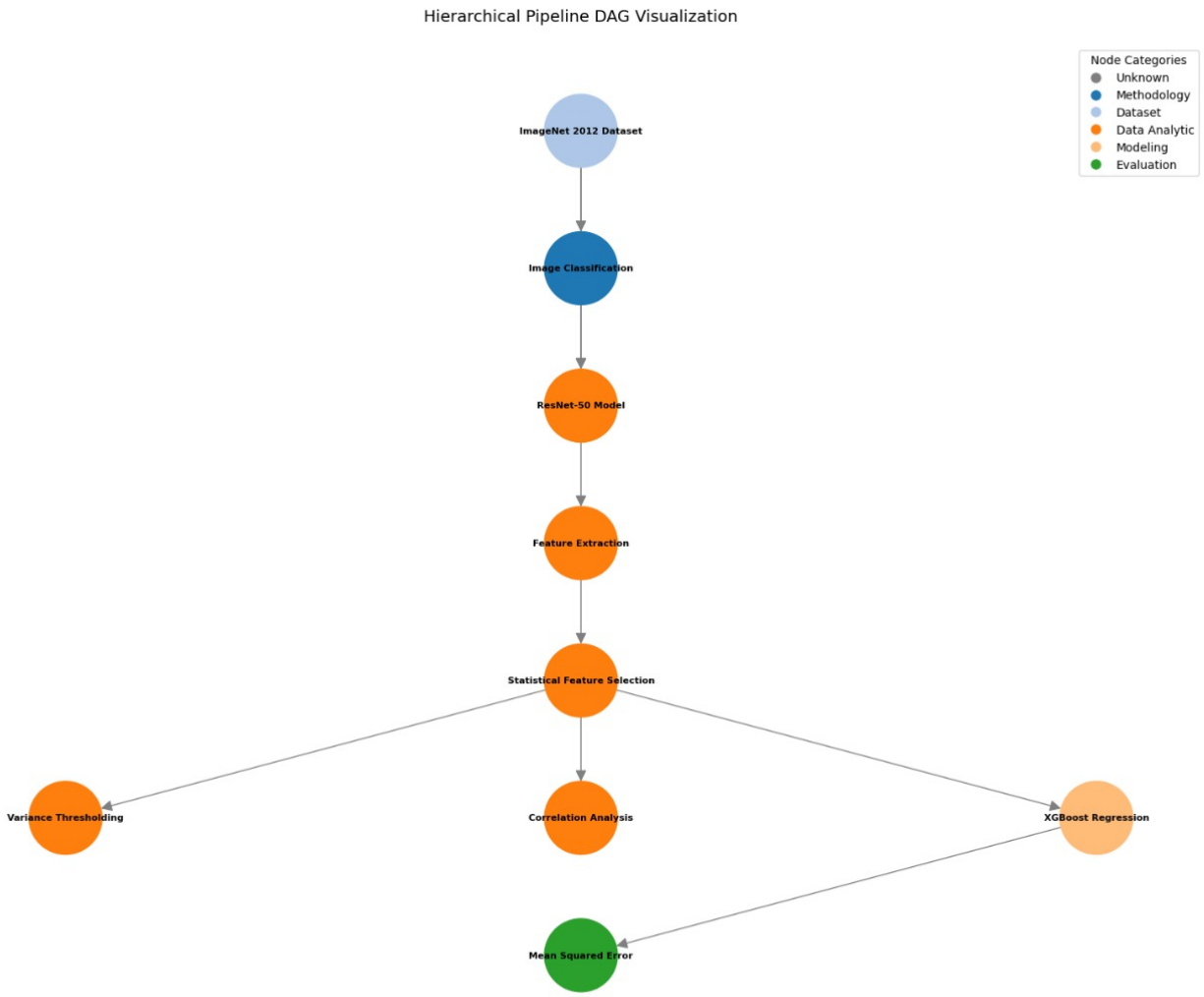

Figure 4: Pipeline for Evaluating XGBoost for Regression.

6 Conclusions and Future Scope

6.1 Conclusion

This project successfully extracted data analytics pipelines from synthetic benchmark dataset, and evaluated their structural and semantic similarity to reference pipelines. Utilizing a pipeline of FAISS-based retrieval, BM25 ranking, LLMs querying and BERTScore evaluation, the study achieved robust extraction and summarization of pipeline structures. The pipeline strucuture similarity was measured using Graph Edit Distance (GED) and Levenshtein similarity, revealing a diverse range of alignment. The average GED and Levenshtein similarity were 1.84 and 0.47, respectively. For some papers, the levenshtein similariy was high and GED score was also high which indicates, there is still some discrepency in evaluation logic due to LLM inconsistencies in result producing but overall correlation between low GED and high Levenshtein similarity for most papers validated the reliability of these metrics for structural and sequential analysis. These findings highlight the potential of automated pipeline extraction while exposing the challenges of variability in LLM-generated content, underscoring the need for enhanced standardization.

6.2 Future Work

Future research can improve pipeline extraction by addressing RQ1 through advanced NLP models to better handle inconsistencies in LLM-generated texts and also overcoming the problem of variable entity names of single method, potentially incorporting contextual embeddings for more accurate component identification. For RQ2, exploring adaptive retrieval strategies could further enhance efficiency and relevance and increasing the LLM input token size can improve the summary content richness. For RQ3, extending the methodology to real-world research papers could validate generalizability beyond LLM-generated content, and automating interactive pipeline visualizations (e.g., as directed graphs) could aid qualitative analysis, complementing quantitative metrics and supporting researchers in pipeline design and optimization. To make replication of papers more accurate, more information about implementation of algorithms, specifics of datasets can be stored in nodes with names in DAG pipeline, which was not possible now due to output

size limit of LLMs. Also, generating a benchmark dataset having not just 15 but more than thousands of papers with correspoding summaries and pipelines utilizing them to fine-tune LLMs for creating precise DAGs without any garbage content and making the results more significant about system usability in real world. Investigating alternative similarity metrics, such as semantic graph-based measures or ontology-driven comparisons, may offer deeper insights into pipeline alignment for RQ1. Also due to GPU limitation lower version models were used, however if more GPU are available then stronger versions can also improve the efficiency of the system. At last but not least, achieving the parallel execution of 4 LLMs for summariztion of papers to reduce the time of execution significantly, which was not feasible now due to GPU size restriction.

References

[1] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):353–362, 1983.

[2] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, 2010.

[3] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[4] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[5] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3219–3232, 2018.

[6] Y Hou et al. Fine-tuning t5 for scientific text summarization. *arXiv preprint arXiv:2109.04640*, 2021.

[7] Z Shen et al. Bart-based summarization of medical literature. *Journal of Biomedical Informatics*, 126:103987, 2022.

[8] Alexis Conneau, Edouard Grave, Guillaume Lample, Thomas Wolf, Yacine Jernite, Tim Dettmers, Sebastian Ruder, Angela Fan, Armand Joulin, Myle Ott, Jason Phang, Patrick von Platen, Quentin Lhoest, Samuel Albanie, Francisco Massa, Nathan Lambert, Hugo Touvron, et al. Mistral 7b, 2023.

[9] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.

[10] Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.

[11] Y Gao et al. Rag for technical report summarization. *arXiv preprint arXiv:2305.12345*, 2023.

[12] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019.

[15] OpenAI. Chatgpt. `https://openai.com/chatgpt`, 2022. Accessed: 2024-06-09.