HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Artificial Intelligence Department

PolySumm

[Scientific Papers Summarization]

By:

Zeyad Diaa Abdalla - 20210364

Awab Mahmoud El–Tarabilly - 20210195

Habiba Anwar Shafie - 20210275

Shahd Mohamed Abdelsalam - 20210470

Tasneem Mohsen Mahmoud - 20210239

Yassin Diaa El-Din - 20211053

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence at the Artificial Intelligence Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

Dr. Amr Ghoneim

June 2025

Abstract

Multimodal Scientific Paper Summarization using Retrieval-Augmented Generation

Problem: The exponential growth of scientific literature has created significant challenges for researchers attempting to efficiently extract key information from papers. Traditional summarization approaches primarily focus on text, neglecting the critical information contained in figures and tables that often represent a paper's core contributions. This limitation leads to incomplete understanding and inefficient knowledge acquisition, particularly in fields with complex visual data representations.

Objectives: This project aimed to develop a comprehensive multimodal summarization system for scientific papers that integrates information from text, figures, and tables to generate more complete and informative summaries. Specifically, we sought to: (1) design a robust pipeline for extracting structured content from scientific PDFs; (2) implement specialized summarization models for different content types; (3) integrate these components within a Retrieval-Augmented Generation (RAG) framework; and (4) provide an accessible user interface for researchers to utilize this technology.

Methodology: We developed a system architecture combining PaddleOCR for multimodal data extraction with specialized summarization models within a RAG framework. PaddleOCR processes scientific PDFs to extract text, identify figures, and recognize table structures, outputting a structured JSON representation. This data undergoes preprocessing including semantic chunking and embedding generation before entering the RAG pipeline. The system employs distinct summarization approaches for different content types: a transformer-based model for text summarization, vision-language models for figure description, and specialized table-to-text models for tabular data. The React-based frontend allows users to upload papers and view the generated multimodal summaries.

Achievements: The implemented system successfully demonstrates the feasibility of multimodal scientific paper summarization, addressing a significant gap in existing tools. Evaluation results indicate that incorporating figure and table information leads to more comprehensive summaries compared to text-only approaches. The system effectively processes complex scientific PDFs, accurately extracts multimodal content, and generates coherent summaries that capture information across modalities. While challenges remain in handling domain-specific terminology and complex visual elements, this work establishes a foundation for more sophisticated scientific document understanding systems and contributes to addressing information overload in research communities.

Keywords

Abstractive Summarization, Document Analysis, Document Layout Detection, Information Extraction, Machine Learning, Multimodal RAG, Multimodal Summarization, Natural Language Processing (NLP), Optical Character Recognition (OCR), PDF Processing, Paper Topic Classification, Retrieval-Augmented Generation (RAG), Scientific Paper Analysis, Text Summarization, Table Summarization, Vision-Language Models

Acknowledgement

We would like to express our sincere gratitude to our supervisor, **Dr. Amr Ghoneim**, for his invaluable guidance, unwavering support, and profound insights throughout the duration of this project. His expertise, constructive feedback, and continuous encouragement were instrumental in shaping our research and overcoming numerous challenges. We are deeply thankful for his dedication and for fostering an environment that allowed us to explore, learn, and grow. This project would not have been possible without his exceptional mentorship and commitment to our academic and professional development.

Graduation Project Documentation: Multimodal Scientific Paper Summarization using RAG

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background & Overview

The rapid proliferation of scientific literature across diverse disciplines presents a significant challenge for researchers, students, and practitioners seeking to stay abreast of the latest advancements. The sheer volume of published papers makes manual review and synthesis a time-consuming and often overwhelming task. Consequently, there is a growing demand for automated tools capable of condensing the essential information contained within scientific documents into concise and informative summaries. Traditional text summarization techniques have made considerable strides, yet scientific papers are rarely composed solely of text. They are inherently multimodal artifacts, integrating textual explanations with figures (such as charts, graphs, diagrams, and images) and tables, which often encapsulate critical data, experimental results, and conceptual frameworks. Effectively summarizing such documents requires systems that can comprehend and synthesize information from these varied modalities.

1.2 Problem Statement & Motivation

Researchers often face information overload when navigating the vast landscape of scientific publications. Identifying relevant papers and extracting key findings efficiently is crucial for accelerating research progress, fostering innovation, and enabling evidence-based decision-making. Existing summarization tools primarily focus on textual content, often neglecting the rich information embedded in figures and tables. This limitation leads to incomplete or potentially misleading summaries, as visual and tabular data frequently represent the core contributions or evidence presented in a paper. The motivation for this project stems from the need to address this gap by developing an intelligent system capable of performing multimodal summarization specifically tailored for scientific papers. By processing text, figures, and tables in an integrated manner, the system aims to provide users with more comprehensive, accurate, and useful summaries, thereby facilitating faster knowledge discovery and comprehension within the scientific community.

1.3 Research Question(s) & Objectives

This project seeks to investigate the effectiveness of a multimodal approach, specifically using a Retrieval-Augmented Generation (RAG) framework, for summarizing scientific papers.

Research Questions:

- How effectively can a Multimodal RAG system integrate information extracted from text, tables, and figures in scientific PDFs to generate comprehensive summaries?
- How does the quality of summaries generated by the proposed multimodal system compare to summaries generated by text-only summarization methods?
- What are the key challenges in automatically extracting and summarizing diverse content types (text, tables, figures) from scientific papers ?

Project Objectives:

- To design and implement a Multimodal RAG system capable of processing scientific papers in PDF format.
- To design a robust approach for extraction of text, table structures, and figure locations from PDFs.
- To incorporate and evaluate specialized summarization models for generating descriptions of figures and tables.
- To utilize a core text summarization model within the RAG framework to summarize the textual content.
- To develop a functional frontend interface allowing users to upload PDFs and view the generated multimodal summaries.
- To evaluate the performance of the system using appropriate metrics for text, table, and figure summarization.

1.4 Research Hypotheses

- Hypothesis 1: A Multimodal RAG system will generate summaries of scientific papers that are significantly more informative and comprehensive (as measured by human evaluation scores or ROUGE variants) compared to summaries generated by a baseline text-only summarization system.
- Hypothesis 2: Integrating specialized models for figure and table summarization within the RAG framework will lead to better representation of non-textual information in the final summary compared to approaches that ignore or only superficially treat figures and tables.

1.5 Project Scope & Limitations

The scope of this project encompasses the design, implementation, and preliminary evaluation of a prototype system for multimodal scientific paper summarization. The system focuses on processing PDF documents, utilizing OCR techniques for data extraction, employing distinct models for text, figure, and table summarization within a RAG architecture, and providing a basic web interface for user interaction.

Limitations include:

- The system currently relies on the accuracy of OCR Models for initial data extraction; errors in OCR will propagate.
- The performance is dependent on the chosen summarization models and the quality of their training data (e.g. PEGASUS-X, DistillBART, Long-T5, etc….).
- The evaluation may be limited to a specific dataset or set of papers (ArxiV or PubMed).
- The system does not perform deep semantic understanding of complex diagrams or highly specialized domain-specific notations beyond what the models capture.
- The current prototype focuses on summarization and does not include features like citation analysis or knowledge graph generation.

Report Organization

This document details the research, design, implementation, and evaluation of the multimodal scientific paper summarization system.

Chapter 2 provides a review of the relevant literature on text summarization, multimodal document analysis, OCR technologies, and RAG systems.

Chapter 3 describes the proposed methodology and system architecture in detail, covering the frontend, data extraction, the RAG core, and the specialized summarization modules.

Chapter 4 discusses the implementation details, including the tools, libraries, and challenges encountered.

Chapter 5 presents the evaluation strategy, experimental setup, and the results obtained from testing the system.

Chapter 6 discusses the findings, highlights the project contributions, acknowledges limitations, suggests directions for future work, and concludes the report.

Chapter 2: Literature Review

2.1 Introduction to Text Summarization

Text summarization is the process of creating concise, coherent, and informative versions of longer documents while preserving their key information. The field has evolved significantly over the past decades, from early statistical approaches to modern neural methods. Summarization techniques are broadly categorized into two main approaches: extractive and abstractive.

2.1.1 Extractive vs. Abstractive Summarization

Extractive summarization identifies and extracts important sentences or phrases from the source document to form a summary. These methods typically rely on statistical features, graph-based algorithms, or machine learning techniques to rank and select the most salient content. While extractive methods preserve the original wording and are generally more faithful to the source, they often produce disjointed summaries lacking coherence and may miss important information that is distributed across multiple sentences [1,2,3,4,5,6,7,8,9].

Abstractive summarization, in contrast, generates new text that captures the essence of the source document. Modern abstractive approaches predominantly use neural sequence-to-sequence models that “understand” the input and generate summaries that may include novel phrasing not present in the original text. These methods can produce more coherent and concise summaries but may occasionally hallucinate facts or misrepresent information from the source [10,11,12].

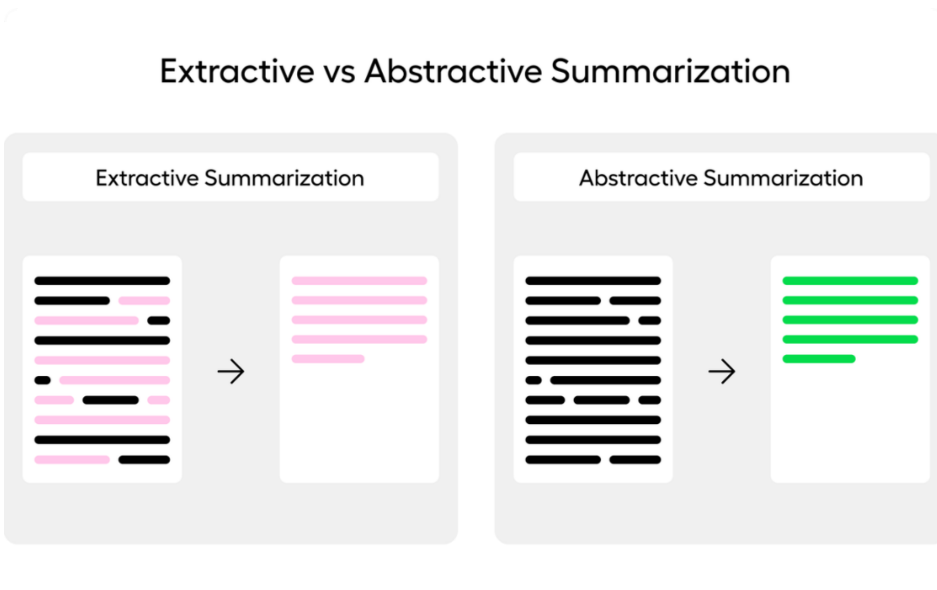

Figure 1- Abstractive vs Extractive Summarization

2.1.2 Evolution of Summarization Techniques

The field of text summarization has witnessed several paradigm shifts:

- **Statistical Methods (1950s-2000s)**: Early approaches relied on statistical features such as term frequency, sentence position, and presence of cue phrases to identify important content.
- **Graph-based Methods (2000s)**: Algorithms like TextRank and LexRank represented documents as graphs where sentences are nodes and edges represent similarity, using PageRank-like algorithms to identify central sentences.
- **Machine Learning Approaches (2000s-2010s)**: Supervised and unsupervised learning methods were applied to learn extraction patterns from large corpora of document-summary pairs.
- **Neural Approaches (2010s-Present)**: Deep learning revolutionized summarization with encoder-decoder architectures, attention mechanisms, and pre-trained language models enabling significant improvements in abstractive summarization quality.
- **Retrieval-Augmented Generation (2020s-Present)**: The latest paradigm combines neural generation with explicit retrieval components to enhance factuality and coverage of generated summaries.

2.2 State-of-the-Art in Scientific Document Summarization

Scientific document summarization presents unique challenges due to the specialized vocabulary, complex concepts, and multimodal nature of scientific papers. Recent advances have focused on addressing these specific challenges.

2.2.1 Challenges in Scientific Document Summarization

Scientific papers differ from general text in several important ways:

- **Technical Complexity**: Domain-specific terminology and concepts require specialized knowledge.
- **Structured Format**: Scientific papers follow specific structures (abstract, introduction, methods, results, discussion) that summarization systems should account for.
- **Multimodal Content**: Critical information is often distributed across text, figures, tables, and equations.
- **Citation Networks**: Understanding the relationship between papers and their citations provides important context.
- **Length**: Scientific papers are typically longer than news articles or general documents, often exceeding the context window of many language models.

2.2.2 Recent Approaches and Models

Recent research has produced several notable approaches for scientific document summarization:

- **PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence models)** [13]: Developed by Google Research, PEGASUS uses a novel pre-training objective specifically designed for abstractive summarization. During pre-training, important sentences are removed from documents and the model learns to generate these sentences, mimicking the summarization task. PEGASUS has shown strong performance on scientific summarization benchmarks like arXiv and PubMed.
- **BART and T5** [14,15]: These encoder-decoder transformer models have been fine-tuned for scientific summarization tasks. Their bidirectional encoding and flexible pre-training objectives make them well-suited for generating coherent scientific summaries.
- **Longformer and BigBird** [16]: These models extend the transformer architecture to handle longer sequences efficiently, addressing the length limitation of standard transformers. They use sparse attention patterns that reduce computational complexity while maintaining performance, making them suitable for processing full scientific papers.
- **DANCER** [17]: A discourse-aware neural summarization model that explicitly models the rhetorical structure of scientific papers, recognizing that different sections serve different communicative purposes.
- **SciSummNet** [18]: A dataset and benchmark specifically designed for scientific summarization, providing human-written summaries aligned with the sections of scientific papers.

2.2.3 Evaluation Metrics for Scientific Summarization

Evaluating scientific summaries presents unique challenges. Standard metrics include:

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**: Measures n-gram overlap between generated and reference summaries.
- **BERTScore**: Uses contextual embeddings to measure semantic similarity between generated and reference summaries.
- **SummaQA**: Evaluates summaries by asking questions based on the source document and checking if the summary contains the answers.
- **Human Evaluation**: Expert assessment of factual correctness, completeness, and coherence remains crucial for scientific summarization.

2.3 Multimodal Document Understanding

Scientific papers are inherently multimodal, containing text, figures, tables, and equations. Understanding and summarizing these documents requires techniques that can process and integrate information from these different modalities.

2.3.1 Multimodal Representation Learning

Multimodal representation learning aims to create unified representations that capture information from different modalities:

- **Joint Embeddings**: These approaches map different modalities into a shared embedding space where semantically similar content is positioned closely regardless of modality.
- **Cross-modal Attention**: Mechanisms that allow one modality to attend to relevant parts of another modality, facilitating information flow between them.
- **Fusion Techniques**: Methods for combining information from different modalities, including early fusion (combining raw inputs), late fusion (combining predictions), and hybrid approaches.
- **Vision-Language Models (VLMs)**: Models like CLIP, BLIP, and VisualBERT learn joint representations of images and text, enabling understanding of the relationship between visual and textual content.

2.3.2 Document Layout Analysis

Document layout analysis is crucial for understanding the structure of scientific papers:

- **Traditional Approaches**: Rule-based and heuristic methods that use geometric analysis to identify document components.
- **Deep Learning Approaches**: Modern methods use convolutional neural networks (CNNs) and transformers to identify and classify document regions.
- **LayoutLM and LayoutLMv2** [19]: Transformer-based models that incorporate layout information alongside text, significantly improving performance on document understanding tasks.
- **PubLayNet and DocBank** [20,21]: Large-scale datasets for document layout analysis, containing annotated scientific documents with labeled regions for text, figures, tables, etc.

2.3.3 Figure and Table Understanding

Understanding figures and tables in scientific papers requires specialized techniques:

- **Chart Parsing**: Methods for extracting data from charts, graphs, and plots, including rule-based approaches and deep learning models.
- **Table Structure Recognition**: Techniques for identifying table structures, cell relationships, and extracting tabular data.
- **Figure Captioning**: Models that generate descriptive captions for scientific figures, often using vision-language architectures.
- **SciCap and FigCAP**: Datasets containing scientific figures paired with captions, enabling the training of figure captioning models.
- **TableBank and PubTables-1M**: Large-scale datasets for table recognition and understanding in scientific documents.

2.4 Optical Character Recognition (OCR) and PaddleOCR

Optical Character Recognition (OCR) is a critical technology for extracting text from document images, including scanned papers or PDFs without embedded text.

2.4.1 Evolution of OCR Technologies

OCR has evolved significantly over the decades:

- **Traditional OCR (1970s-2000s)**: Rule-based systems using image processing techniques like binarization, segmentation, and template matching.
- **Machine Learning OCR (2000s-2010s)**: Systems using classifiers like SVMs and neural networks for character recognition, with separate stages for detection and recognition.
- **Deep Learning OCR (2010s-Present)**: End-to-end trainable systems using CNNs, RNNs, and attention mechanisms, significantly improving accuracy and robustness.
- **Scene Text Recognition**: Extensions of OCR for recognizing text in natural scenes, addressing challenges like varied fonts, orientations, and backgrounds.

2.4.2 PaddleOCR Architecture and Capabilities

PaddleOCR [22] is an open-source OCR system developed by Baidu that has gained popularity due to its high performance and comprehensive functionality [4]. It employs a two-stage approach for text recognition:

Text Detection with DB-Net

The first stage in PaddleOCR is text detection using a Differentiable Binarization Network (DB-Net):

- **Backbone Network**: Typically ResNet or MobileNetV3, which extracts features from the input image.
- **Feature Pyramid Network (FPN)**: Combines features at different scales to handle text of varying sizes.
- **Probability Map Generation**: Produces a pixel-wise map indicating the likelihood of text presence.
- **Differentiable Binarization**: Converts the probability map to a binary mask using an adaptive threshold, allowing end-to-end training.
- **Post-processing**: Extracts text regions from the binary mask and generates oriented bounding boxes.

Text Recognition with CRNN

The second stage recognizes the text within detected regions using a Convolutional Recurrent Neural Network (CRNN):

- **CNN Feature Extraction**: Extracts visual features from the cropped text regions.
- **Bidirectional LSTM**: Processes the sequence of features to capture contextual information.
- **CTC (Connectionist Temporal Classification)**: Aligns the sequence of predictions with the ground truth text without requiring explicit character segmentation.
- **Decoding**: Converts the model outputs into the final text prediction.

Advanced Features of PaddleOCR

PaddleOCR includes several advanced features particularly relevant for scientific document processing:

- **Basic Layout Component Detection**: Through the PP-Structure module, PaddleOCR can identify certain document elements like tables and form fields.
- **Table Structure Recognition**: Detects tables and recognizes their structure, including rows, columns, and cells.
- **Multi-language Support**: Recognizes text in multiple languages, important for international scientific literature.
- **Direction Classification**: Detects and corrects text orientation, handling rotated or vertical text.

- **End-to-End Pipeline**: Provides a complete pipeline from document images to structured text output.

2.4.3 OCR Challenges in Scientific Documents

Scientific documents present unique challenges for OCR systems:

- **Mathematical Equations**: Complex mathematical notation is difficult for standard OCR systems to recognize accurately.
- **Special Symbols**: Scientific papers contain numerous special symbols and notations.
- **Multi-column Layouts**: Many papers use multi-column layouts that can confuse text flow detection.
- **Fine Print and Subscripts/Superscripts**: Small text and specialized formatting are common in scientific papers.
- **Image Quality Variations**: Papers may be scanned at different qualities or contain artifacts from printing and digitization.

2.5 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a hybrid approach that combines information retrieval with text generation to produce more accurate and informative outputs.

2.5.1 Foundations of RAG

RAG was introduced by Lewis et al. (2020) [23] as a way to enhance language model generation with explicit retrieval from a knowledge source:

- **Motivation**: Large language models (LLMs) store knowledge implicitly in their parameters, which can lead to hallucinations and outdated information. RAG addresses this by retrieving relevant information from external sources.
- **Core Components**: A typical RAG system consists of a retriever that identifies relevant documents or passages, and a generator that produces text conditioned on both the query and the retrieved information.
- **Advantages**: RAG improves factuality, allows for updating knowledge without retraining, and enables source attribution.

2.5.2 RAG Architectures and Variants

Several RAG architectures and variants have been developed:

- **Standard RAG** [23]: Retrieves a fixed number of documents for each query and conditions generation on all retrieved documents.

- **Adaptive RAG** [24]: Dynamically determines how many documents to retrieve based on query complexity.
- **Iterative RAG** [25]: Performs multiple rounds of retrieval, using initial generation to refine subsequent retrieval queries.
- **Fusion-in-Decoder** [26]: Processes each retrieved document separately in the encoder and fuses information in the decoder.
- **REALM** [27]: Trains the retriever and generator jointly, optimizing the entire pipeline end-to-end.
- **RETRO** [28]: Incorporates retrieval at every layer of a large language model, allowing fine-grained integration of retrieved information.

2.5.3 Multimodal RAG

Recent work has extended RAG to multimodal settings, particularly relevant for scientific document summarization:

- **Multimodal Retrieval**: Systems that can retrieve information based on queries in different modalities (text, images) or retrieve multimodal documents.
- **Cross-modal Retrieval**: Retrieving information in one modality based on queries in another (e.g., finding relevant images based on text queries).
- **Multimodal Generation**: Generating text that incorporates information from retrieved multimodal documents, including images, tables, and structured data.
- **Multimodal RAG**: A framework that extends RAG to handle multiple modalities in both retrieval and generation phases.
- **CLIP-based Retrieval**: Using CLIP embeddings to enable cross-modal retrieval between text and images.

2.5.4 RAG for Scientific Document Processing

RAG is particularly well-suited for scientific document processing:

- **Citation-aware RAG**: Systems that retrieve and incorporate information from cited papers to provide broader context.
- **Domain-specific Knowledge Bases**: RAG systems using specialized scientific databases like arXiv, PubMed, or domain-specific knowledge graphs.
- **Figure-aware RAG**: Systems that retrieve and incorporate information from figures and their captions.
- **Table-aware RAG**: Systems that can retrieve and reason over tabular data from scientific papers.

- **Equation-aware RAG**: Systems that handle mathematical equations and their textual explanations.

2.6 Summarization Models

This section examines the specific summarization models relevant to our multimodal scientific paper summarization system.

2.6.1 PEGASUS for Text Summarization

PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization Sequence-to-sequence models) represents a significant advancement in abstractive summarization:

Architecture and Pre-training

PEGASUS uses a transformer encoder-decoder architecture with several key components:

- **Encoder Stack**: 16 transformer layers with 16 attention heads per layer, hidden dimension of 1024, and feed-forward dimension of 4096.
- **Decoder Stack**: 16 transformer layers with masked self-attention and cross-attention to encoder outputs.
- **Pre-training Objective**: Gap Sentence Generation (GSG), where important sentences are identified and removed from the document, and the model learns to generate these sentences.
- **Sentence Importance Scoring**: Uses various strategies including Random, Lead, Principal, and Rouge to select sentences for masking during pre-training.

Fine-tuning for Scientific Papers

PEGASUS can be fine-tuned specifically for scientific paper summarization:

- **Dataset**: Typically fine-tuned on datasets like arXiv or PubMed, containing scientific papers paired with their abstracts.
- **Input Processing**: Scientific papers often exceed the model's maximum input length (1024 tokens), requiring chunking strategies.
- **Output Control**: Parameters like length penalty, beam size, and no-repeat n-gram size can be tuned to control summary quality.
- **Performance**: Fine-tuned PEGASUS achieves state-of-the-art results on scientific summarization benchmarks, with ROUGE-1 scores typically exceeding 0.44 on arXiv and 0.45 on PubMed.

2.6.2 Vision-Language Models for Figure Summarization

Vision-Language Models (VLMs) are crucial for understanding and summarizing figures in scientific papers:

- **BLIP (Bootstrapping Language-Image Pre-training)** [29]: Combines image-text contrastive learning with image-text matching and image-conditioned language modeling, making it effective for figure captioning.
- **ViLT (Vision-and-Language Transformer)** [30]: A simplified VLM that processes image patches and text tokens directly without using separate feature extractors.
- **CLIP (Contrastive Language-Image Pre-training)** [31]: While primarily designed for image-text matching, CLIP's strong visual representations can be leveraged for scientific figure understanding.
- **SciCap Models** [32]: Specialized models fine-tuned on the SciCap dataset, which contains scientific figures paired with their captions.
- **Domain Adaptation Techniques**: Methods for adapting general VLMs to scientific domains, including continued pre-training on scientific images and text.

2.6.3 Table-to-Text Models

Table summarization requires specialized models that can understand tabular structures:

- **TAPAS and TAPEX** [33,34]: Transformer-based models specifically designed for table understanding and question answering.
- **ToTTo Models** [35]: Models fine-tuned on the ToTTo (Table-To-Text) dataset, which contains tables paired with descriptions highlighting specific cells.
- **Structure-aware Encoders** [36]: Models that explicitly encode table structure, including row and column relationships.
- **Linearization Approaches** [35]: Methods for converting tables into linear sequences that can be processed by standard language models.
- **Hybrid Approaches** [37]: Systems that combine structural understanding with contextual information from surrounding text.

2.7 Related Systems and Applications

Several existing systems and applications address aspects of scientific document processing and summarization:

- **Semantic Scholar** [38]: A scientific literature search engine that uses AI to extract key information from papers, including figures and tables.

- **TLDR This** [39]: An online tool that generates summaries of scientific papers, though primarily focused on text rather than multimodal content.
- **SciTLDR** [39]: A dataset and leaderboard for extreme summarization of scientific papers, focusing on one-sentence summaries.
- **ScholarPhi** [40]: A system for augmented reading of scientific papers, highlighting definitions and providing interactive explanations.
- **Elicit**: An AI research assistant that helps researchers find and summarize relevant papers.
- **SciA11y** [41]: A system for making scientific papers more accessible, including figure descriptions for visually impaired users.
- **SciSight** [42]: A tool for exploring scientific literature through visual interfaces, including figure browsing.

2.8 Research Gaps and Opportunities

Despite significant progress, several research gaps and opportunities remain in multimodal scientific paper summarization:

- **Integration of Multiple Modalities**: Most existing systems focus on either text summarization or figure/table understanding, with limited work on truly integrated multimodal summarization.
- **Handling Domain Specificity**: Scientific papers span diverse domains with specialized terminology and conventions, requiring domain-adaptive approaches.
- **Evaluation Metrics**: Standard summarization metrics like ROUGE may not adequately capture the quality of multimodal summaries, necessitating new evaluation approaches.
- **Long Document Processing**: Many scientific papers exceed the context windows of current models, requiring effective strategies for handling long documents.
- **Factual Consistency**: Ensuring that generated summaries are factually consistent with the source document remains challenging, particularly for complex scientific content.
- **Interactive Summarization**: Systems that allow users to control the focus and level of detail in summaries could better serve diverse user needs.
- **Cross-lingual Scientific Summarization**: Addressing the language barrier in scientific communication through multilingual summarization.

2.9 Summary

This literature review has examined the state-of-the-art in scientific document summarization, multimodal document understanding, OCR technologies (with a focus on PaddleOCR), Retrieval-Augmented Generation, and specific summarization models for different modalities. The review highlights the complexity of multimodal scientific paper summarization and identifies several research gaps and opportunities.

The proposed system in this project aims to address some of these gaps by integrating PaddleOCR for data extraction, specialized summarization models for different modalities, and a RAG framework to produce comprehensive summaries of scientific papers. By combining these components, the system seeks to advance the state-of-the-art in multimodal scientific document summarization.

Chapter 3: Methodology / System Design

3.1 Overall System Architecture

The proposed system for scientific paper summarization is designed as a Multimodal Retrieval-Augmented Generation (RAG) system. This architecture is chosen to effectively leverage the diverse types of information present in scientific documents – text, figures, and tables – to produce comprehensive and informative summaries. The system integrates several specialized components, starting from data extraction using PaddleOCR, followed by modality-specific preprocessing and summarization, all orchestrated within a RAG framework to generate a final, coherent summary. The overall workflow begins with the user uploading a scientific paper in PDF format via a web-based frontend. The backend receives the PDF, processes it using the Data Extraction Module (PaddleOCR) to obtain structured multimodal content in JSON format. This structured data then flows into the RAG System Core, where it undergoes preprocessing and is used to drive the retrieval and generation processes involving specialized summarization models. Finally, the generated summaries for different modalities are combined and presented to the user.

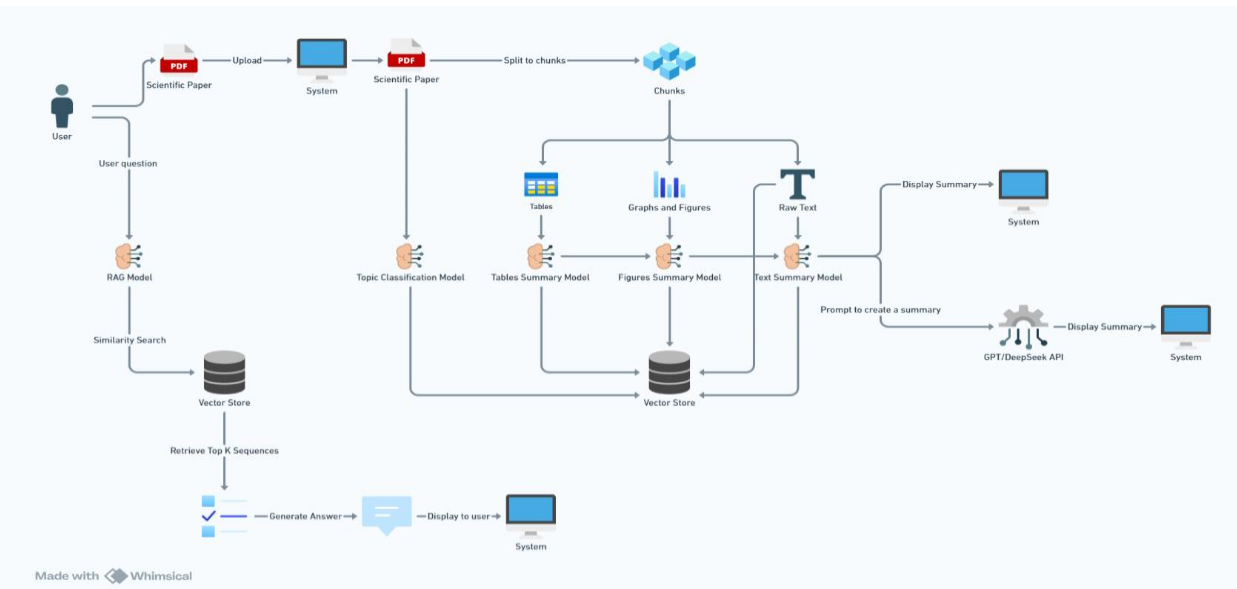

Figure 2- Project Workflow

3.2 Requirements

3.2.1 Functional Requirements

1. User Interface (Frontend) Requirements

- **PDF Upload**: The system shall allow users to upload scientific papers in PDF format through a web-based interface.
- **Progress Display**: The system shall display the processing status of the uploaded PDF, including progress indicators for OCR, summarization, and overall completion.
- **Summary Presentation**: The system shall present the generated multimodal summary to the user in a clear and readable format, distinguishing between text, figure, and table summaries.
- **Interactive Chat**: The system shall provide an interactive chat interface that allows users to ask questions about the content of the paper, enabling dynamic exploration and clarification based on the summarized information.

2. Data Extraction Module (PaddleOCR) Requirements

- **PDF-to-Image Conversion**: The system shall convert each page of the uploaded PDF into high-resolution images to ensure accurate detection of textual and visual elements.
- **Layout-Based Multimodal Extraction**: The system shall use a layout detection model to segment each page into predefined regions Text, Title, List, Table, and Figure and apply PaddleOCR for accurate text recognition in textual regions. Table and figure regions shall be cropped and preserved for downstream processing.
- **Structured JSON Output**: The system shall output all extracted elements recognized text, base64-encoded table and figure crops in a well-structured JSON format.

3. RAG System Core Requirements

- **Multimodal Data Preprocessing**: The system shall preprocess the JSON output from the Data Extraction Module, separating text, table, and figure data for further processing.
- **Text Chunking**: The system shall chunk large textual content into smaller, semantically coherent segments suitable for input to summarization models.
- **Multimodal Data Representation**: The system shall convert text chunks,

table data, and figure metadata into suitable internal representations (e.g., vector embeddings) for retrieval.

- **Contextual Retrieval**: The system shall retrieve relevant text chunks, table data, and figure information based on the context required for summarization.
- **Multimodal Summary Combination**: The system shall combine the individual summaries generated for text, figures, and tables into a single, coherent final summary.

4. Summarization Modules Requirements

- **Text Summarization**: The system shall generate abstractive summaries of the textual content of the scientific paper using a core text summarization model.
- **Figure Summarization**: The system shall generate natural language descriptions or summaries for figures based on their visual content and contextual information.
- **Table Summarization**: The system shall generate natural language summaries for tables, highlighting key trends, comparisons, or significant data points.

3.2.2 Non-Functional Requirements

1. Security Requirements

- **Data Privacy**: The system shall ensure the privacy of uploaded documents and generated summaries, preventing unauthorized access.
- **Data Integrity**: The system shall maintain the integrity of uploaded documents and extracted data throughout the processing pipeline.
- **Input Validation**: The system shall validate user inputs (e.g., file types) to prevent malicious uploads or injections.

2. Usability Requirements

- **User-Friendly Interface**: The frontend interface shall be intuitive and easy to navigate for users with varying technical proficiencies.
- **Clear Feedback**: The system shall provide clear and understandable feedback to the user regarding the status of their requests and any errors.

- **Accessibility**: The system shall adhere to common web accessibility standards to ensure usability for a broad range of users.

3. Maintainability Requirements
    - **Modularity**: The system architecture shall be modular, allowing for independent development, testing, and updates of components (e.g., OCR module, summarization models).
    - **Code Readability**: The codebase shall be well-documented and follow established coding standards to facilitate future maintenance and enhancements.
    - **Testability**: The system components shall be designed to be easily testable, enabling efficient identification and resolution of bugs.
4. Reliability Requirements
    - **Fault Tolerance**: The system shall gracefully handle unexpected errors or failures in individual components (e.g., a summarization model failure) without crashing the entire system.
    - **Data Recovery**: In the event of a system failure, the system shall be able to recover and resume processing without significant data loss.

3.3 Frontend Interface (React)

The frontend of **PolySumm** provides a responsive, user-friendly interface that facilitates seamless interaction with the backend summarization and analysis pipelines. Developed entirely using **React 19.1.0**, the application emphasizes modularity, interactivity, and maintainability while supporting a rich feature set tailored for scientific document processing.

React Component Architecture

PolySumm is built using a well-structured, **functional component-based** React architecture, ensuring high reusability and ease of maintenance. Key interface modules include:

- **Upload Interface**: A traditional file selector allowing users to upload PDF documents. It includes format and size validation, as well as real-time upload progress indicators. The user also can choose between **2 analysis options:**

    - **Analysis Options**
      PolySumm supports dual analysis pipelines, each with distinct processing paths:

1. **Custom Local Models**:
    - OCR, text summarization, table/figure summarization, and topic classification.
    - Dedicated **RAG pipelines** for enhanced relevance and contextual grounding.
2. **External API Models**:
    - Seamlessly connects to external services for document summarization and analysis.
    - Isolated RAG pipelines for these models to ensure modularity and comparison.

- **Results Display**: Presents the paper topic, predicted by a custom-trained topic classification model, along with the summary generated by either local or external models.
- **Contextual Chatbot**: Enables users to chat with a document-aware assistant trained specifically on the content of the uploaded file. All interactions are reset with each new upload to ensure accurate context.
- **Feedback & Status Components**: Includes loading animations, success/error banners, and system state indicators to enhance user experience throughout the workflow.
- **Page Separation**: The interface is cleanly split into major views:
    - Home (landing & usage instructions)
    - About
    - Upload
    - Processing (loading status)
    - Results
    - Chat

State Management

The application utilizes the React Context API to manage global and shared state across the interface. This includes:

- File upload status
- Processing state (loading, success, error)
- Summary result data
- Theme preferences (light/dark mode)
- Chat history and user interactions

This centralized approach ensures consistent and predictable behavior across all pages and components.

API Integration

Client-server communication is managed through **native JavaScript fetch,** enabling asynchronous HTTP requests between the frontend and the **FastAPI backend**. The frontend supports:

- Uploading PDFs for processing
- Fetching summaries and topic classification results
- Interacting with the chatbot (contextual question answering)

All API interactions feature robust error handling and response validation to ensure user feedback and system resilience.

Chatbot Interface

The integrated chatbot allows users to query the uploaded paper in natural language:

- **Session Initialization**: At the beginning of each chat session, the chatbot provides the paper’s detected topic and an auto-generated summary to establish context.
- **Dynamic Suggested Questions**: A list of suggested questions is generated by the model based on the paper’s content to help guide the user. After each user-selected question, the model dynamically updates the suggestions, generating a new set of contextually relevant questions based on the latest interaction. This creates an adaptive exploration experience tailored to the user's interests and the document’s content
- New uploads automatically **clear all chat history and session context**, ensuring fresh analysis

3.4 Data Extraction Module

The Data Extraction Module is responsible for converting unstructured content from research paper PDFs into a structured and machine-readable format using OCR and layout analysis techniques. This module integrates **PaddleOCR** and **LayoutParser** [5] to extract text, tables, and figures from documents and represent them in a JSON format for downstream processing.

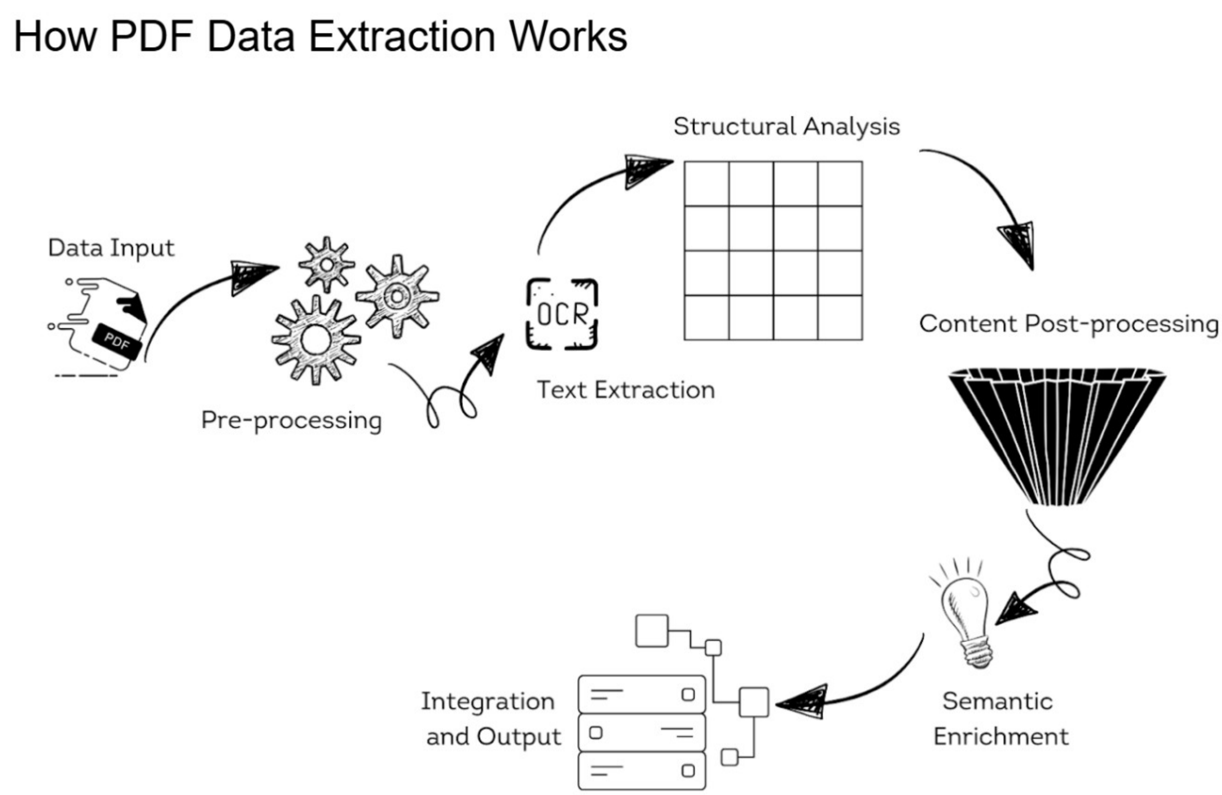

Figure 3 - PDF Data Extraction Process

3.4.1 PDF Processing

The PDF processing pipeline begins by converting each page of a PDF into high-resolution images using **PyMuPDF (fitz)**. A resolution of 300 DPI is used to ensure text and structural elements are captured clearly. The convert_pdf_to_images() function handles this task, returning a list of images, one for each page in the PDF.

3.4.2 Layout Analysis

Once PDF pages are converted to images, a layout detection model based on Detectron2 and integrated with **LayoutParser** is used to segment each page into predefined categories: Text, Title, List, Table, and Figure. This step is handled by extract_elements() and filters elements based on a configurable confidence threshold (SCORE_THRESHOLD = 0.8). Detected regions are further processed by crop_regions(), which adds padding to avoid truncating characters during OCR.

The layout detection model used is **mask_rcnn_X_101_32x8d_FPN_3x**, a high-capacity instance segmentation model from the Detectron2 model zoo. It is based on the **Mask R-CNN** architecture with a **ResNeXt-101** backbone and **FPN (Feature Pyramid Network),** enabling strong multiscale feature extraction and precise region segmentation. This model has been pretrained on the **PubLayNet dataset** [6], a large-scale dataset specifically designed for document layout analysis, containing over 360,000 labeled document images derived from scientific PDFs.

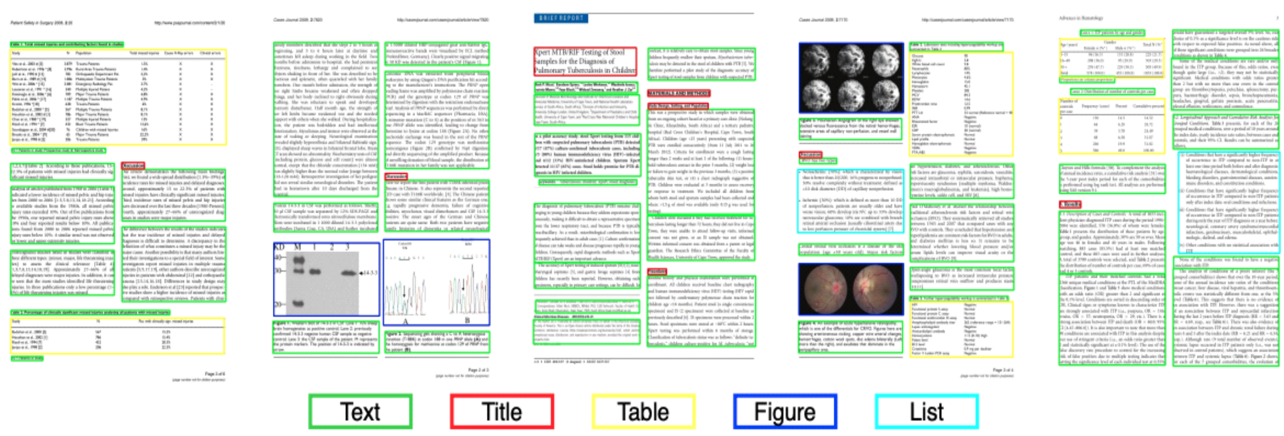

Figure 4 - Layout Detection

3.4.3 Text Recognition

For text-based regions such as Text, Title, and List, the module uses PaddleOCR (PP-OCRv3) for text recognition. The OCR model is initialized using custom detection and recognition models, loaded via load_ocr_model(). Each cropped image region is passed to recognize_text_from_crops(), which performs angle classification and text recognition. Recognized lines are concatenated and cleaned using clean_multiline_text() before being stored in the final result.

3.4.4 Table and Figure Extraction

During layout analysis, regions labeled as "Table" and "Figure" are detected and cropped from the original PDF page using bounding box coordinates. These visual elements are then encoded as base64 strings using the crop_to_base64() function and stored in the output JSON under the "tables" and "figures" keys, respectively. This ensures the preservation of visual content for downstream processing or manual inspection in a structured and extensible format.

3.4.5 JSON Output Format

The final output from the process_pdf() function is returned as a structured JSON object, where each page contains its recognized elements categorized under keys like "text_blocks", "tables", and "figures". This format facilitates downstream summarization tasks by providing structured representations of both textual and visual content.

3.5 RAG System Core

The core of the system is the Multimodal RAG component, responsible for managing the extracted information and generating summaries. It acts as the central orchestrator, integrating the outputs from the upstream PaddleOCR module and coordinating the downstream summarization models.

3.5.1 Preprocessing

Upon receiving the structured JSON output from PaddleOCR, the RAG system performs crucial preprocessing steps to prepare the data for the subsequent retrieval and generation phases. This involves parsing the JSON to separate the different content types: textual paragraphs, table data (including structure), and figure references (bounding boxes). Each data type requires specific handling. For instance, as described by the project team, textual data undergoes a chunking process. This is essential because large language models used for summarization often have input context length limitations. Breaking down long sections of text into smaller, manageable, yet semantically coherent chunks (e.g., paragraphs or fixed-size overlapping segments) allows the model to process the entire document effectively while maintaining local context for each chunk. Table data might require normalization or serialization into a format suitable for the table summarization model. Figure information (primarily location) is used to link the visual element (assumed to be stored or accessible) to its context within the paper for the figure summarization model.

3.5.2 Text Chunking Strategy

The strategy for chunking text is a critical design choice. Simple fixed-size chunking can break sentences or ideas mid-way, potentially harming the quality of summarization. More sophisticated strategies, which this system aims to employ, might involve chunking based on semantic boundaries like paragraphs or sections identified during layout analysis. Overlapping chunks can also be used to ensure that context is not lost at the boundaries between chunks. The chosen strategy needs to balance the need for manageable input sizes with the preservation of semantic coherence.

3.5.3 Multimodal Data Representation

To effectively utilize the different modalities, the RAG system needs a way to represent them internally. Text chunks are typically converted into dense vector embeddings using a suitable text embedding model (e.g., Sentence-BERT, SimCSE). These embeddings capture the semantic meaning of the text and are used for retrieval. Representing tables and figures for retrieval and generation is more complex. Tables is represented by summarizing their content textually, embedding their structure and using specialized table embedding techniques. Figures is represented by their captions, surrounding text, or potentially through image embeddings generated by a vision model.

3.6 Summarization Modules

A core capability of the system is its ability to generate concise summaries from the extracted multimodal content. Recognizing that different types of content require different summarization approaches, the system employs specialized models for text, figures, and tables. These models work in conjunction with the RAG core to produce informative descriptions and summaries.

3.6.1 Figure Summarization

A dedicated figure summarization model ingests a figure image along with its caption and nearby contextual text (via RAG) to produce a concise natural language summary capturing the figure's key message. Integrating this with insights from **PaliGemma** [43] an open 3B vision-language model combining a SigLIP-So400m encoder and Gemma-2B decoder that excels at diverse multimodal tasks like captioning, VQA, segmentation, and remote sensing means your figure summarization system can benefit from a robust multimodal backbone capable of interpreting complex visuals and context, while evaluation could draw on approaches like LLM-SummEval to assess summary quality along dimensions like faithfulness, relevance, coherence, and fluency.

- **Model:** PaliGemma 3B.
- **Architecture:** Transformer-based.
- **Training Data:** SciCap dataset [44].
- **Input:** Figure image representation, retrieved contextual text (caption, surrounding paragraphs).
- **Output:** Natural language description/summary of the figure.

3.6.2 Table Summarization

Tables present structured data that often requires careful interpretation to summarize effectively. A simple textual transcription is usually insufficient. The table summarization model is designed to analyze the structure and content of a table (as extracted and potentially preprocessed from PaddleOCR's output) and generate a summary highlighting key trends, comparisons, or significant data points within the table. Similar to figure summarization, it might leverage retrieved textual context from the paper to better understand the table's significance.

Model

For table summarization, the **Qwen2-VL-2B** model is utilized. While not exclusively a table-to-text generation model, Qwen2-VL-2B is a powerful **Large Vision-Language Model (LVLM)** capable of understanding and processing both visual and textual information, making it suitable for tasks involving structured data like tables, especially when combined with OCR capabilities.

Architecture

Qwen2-VL-2B is built upon a **Transformer architecture** with several key enhancements to handle multimodal inputs (images, videos, and text) and generate coherent responses. Its architecture incorporates:

- **Naive Dynamic Resolution Mechanism**: This mechanism allows the model to dynamically process images of varying resolutions into different numbers of visual tokens. This is crucial for accurately capturing details in tables, which can vary significantly in size and layout.
- **Multimodal Rotary Position Embedding (M-RoPE)**: M-RoPE facilitates the effective fusion of positional information across text, images, and videos. This enables the model to understand the spatial relationships within a table and how textual elements relate to its structure.
- **Unified Paradigm for Image and Video Processing**: Qwen2-VL-2B employs a unified approach for processing both images and videos, enhancing its visual perception capabilities. This is relevant for tables that might be embedded within images or videos.
- **Integration with OCR**: While Qwen2-VL-2B itself is a VLM, its application to table summarization often involves an initial Optical Character Recognition (OCR) step (e.g., using PaddleOCR) to extract the raw text and structural information from the table image. The model then processes this structured textual data along with any visual cues.

Overall, Qwen2-VL-2B acts as a powerful multimodal reasoning engine that can interpret the content and context of tables, leveraging its vision-language understanding to generate meaningful summaries.

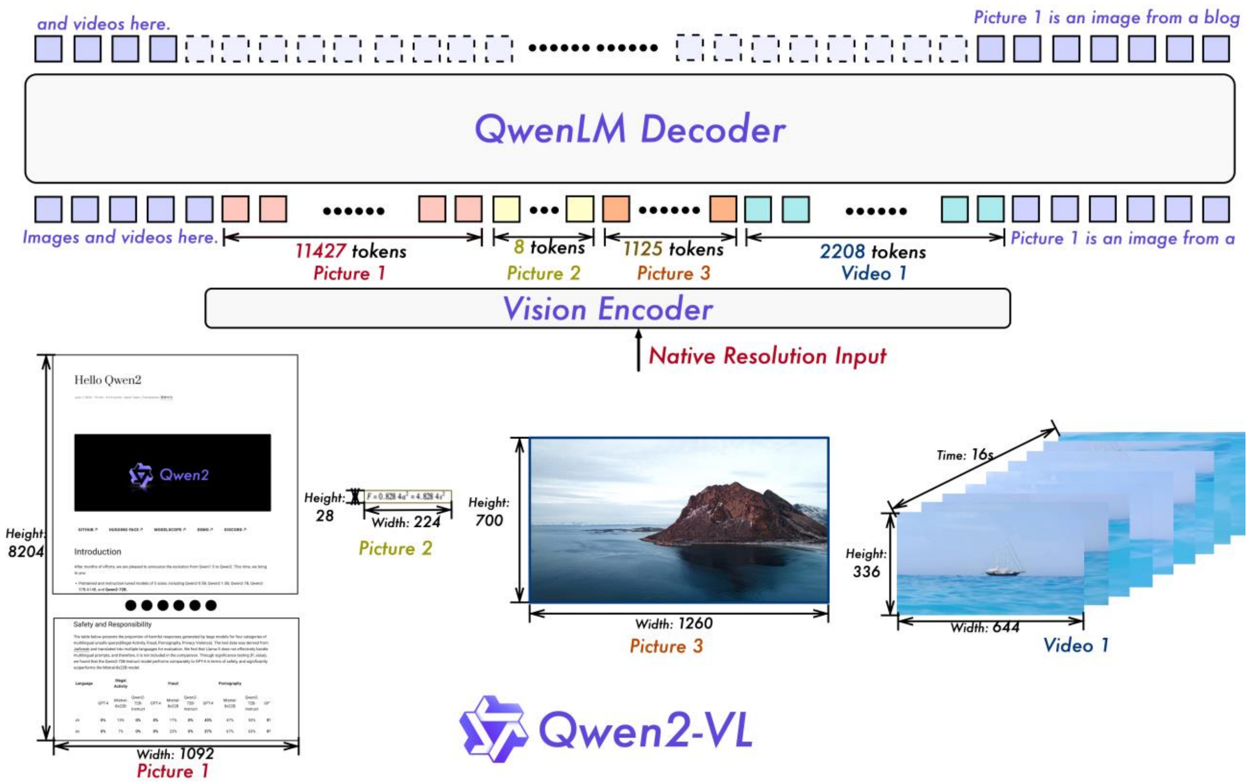

Figure 5 - Qwen2 VL Architecture

Training Data

Qwen2-VL-2B was pre-trained on a massive and diverse collection of **multimodal data**, including large-scale text corpora, image datasets, and video datasets. The development of Qwen2-VL series involved investigating **scaling laws for LVLMs**, scaling both the model size and the amount of training data. While specific datasets for *table summarization* pre-training are not the primary focus for a general LVLM like Qwen2-VL-2B, its broad pre-training on diverse visual and textual data enables its adaptability to various downstream tasks, including table understanding and summarization. The model benefits from extensive pre-training on general vision-language tasks, which provides it with a strong foundation for interpreting complex visual layouts and associated text.

Input

Structured table data (e.g., linearized format, HTML), retrieved contextual text.

Output

Natural language summary of the table's key information.

3.6.3 Text Summarization

This section details the core text summarization component of the graduation project, responsible for generating abstractive summaries from scientific paper content. Given the often extensive nature of academic texts, this model operates on preprocessed text chunks, leveraging relevant context identified by the Retrieval-Augmented Generation (RAG) system. The objective is to produce concise yet comprehensive summaries that capture the essence of the paper's background, methodologies, results, and conclusions, going beyond mere extraction of existing sentences to generate novel, coherent narratives.

1. LongSum Approach

Model

The initial approach to text summarization employed the LongSum model [1], a framework designed to address the challenges of summarizing long documents. LongSum focuses on efficiently processing extended textual inputs to generate coherent and contextually relevant summaries.

Architecture

LongSum addresses the challenges of long document summarization by employing two complementary methods: **local self-attention** and **explicit content selection**. The core of LongSum is built upon a Transformer-based encoder-decoder architecture, specifically utilizing a modified BART model (referred to as LoBART).

- **Local Self-Attention**: To overcome the quadratic complexity of standard Transformer self-attention with respect to sequence length, LongSum constrains the attention mechanism to be local. This allows for processing longer input spans during training without prohibitive memory and computational costs. The self-attention in the encoder is modified to a local window attention, similar to approaches in Sparse Transformer and Longformer. Positional embeddings are extended beyond 1,024 tokens to accommodate longer sequences.
- **Explicit Content Selection**: Abstractive summarization models often perform content selection implicitly. LongSum introduces explicit content selection at two phases:
    - **Training-time Content Selection**: This involves methods to select relevant data for training fixed-span abstractive models. The paper investigates different strategies, including an ORACLE-based selection (`ORACLE_pad-rand`).
    - **Test-time Content Selection**: A **Multitask Content Selection (MCS)** method is proposed. This hierarchical encoder-decoder architecture, based on word-level and sentence-level GRUs, ranks sentences through an extractive labeling module and an attention-based module. This explicit selection reduces memory and compute requirements before the abstractive generation stage.

The overall architecture combines these components: input documents undergo content selection (either training-time or test-time) to produce a shorter, relevant input sequence, which is then fed into the local self-attention Transformer (LoBART) to generate the summary.

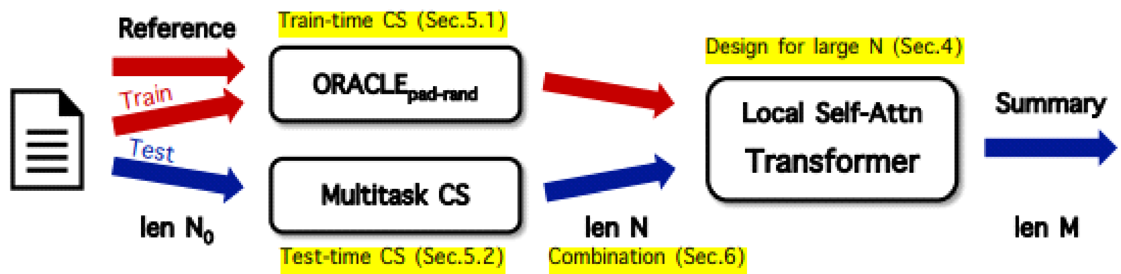

Figure 6 - LongSum Architecture

Training Data

The LongSum model was trained and evaluated on the arXiv dataset [45], a large collection of scientific papers. For the purpose of this project, the model was evaluated on a subset of 100 training samples from the arXiv dataset. It is important to note that while LongSum has the potential for better performance with more extensive training data, resource constraints limited the training to this subset.

Input

Long-form documents split into manageable chunks, along with local context windows. Input is processed through local attention mechanisms to capture relevant spans efficiently.

Output

Abstractive summary generated using selected salient content across the entire document span.

2. Pegasus-Large Approach

Model

The second approach investigated for text summarization utilized the PEGASUS-Large model [2], a state-of-the-art abstractive summarization model developed by Google. PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization) is specifically designed to achieve high performance in abstractive summarization tasks by pre-training on a large corpus with a novel self-supervised objective.

Architecture

PEGASUS-Large is built upon a standard Transformer encoder-decoder architecture, which is widely used in sequence-to-sequence tasks. Its distinctiveness and superior performance in abstractive summarization stem from its novel self-supervised pre-training objective, primarily **Gap-sentences Generation (GSG)**, often combined with **Masked Language Model (MLM)**.

- **Transformer Encoder-Decoder**: The model utilizes a Transformer architecture, comprising an encoder and a decoder. The encoder processes the input document, and the decoder generates the summary. This architecture allows for capturing long-range dependencies and parallel processing.
- **Gap-sentences Generation (GSG)**: This is the primary pre-training objective. In GSG, entire sentences are removed or masked from an input document. The model is then trained to reconstruct these masked sentences as a single output sequence from the remaining sentences. This objective is designed to closely resemble the abstractive summarization task, forcing the model to generate novel text that captures the essence of the masked content rather than simply copying phrases. The selection of gap sentences can be random, lead-based (first `m` sentences), or principal-based (top `m` sentences scored by ROUGE-1 F1 with the rest of the document).
- **Masked Language Model (MLM)**: Similar to BERT, MLM is used as a secondary pre-training objective. It involves masking a percentage of tokens in the input text and training the model to predict the original masked tokens. This helps the model learn contextual representations of words.

PEGASUS pre-trains on massive text corpora (e.g., C4, HugeNews) using these objectives. The model learns to generate summary-like text by synthesizing information from the unmasked parts of the document to reconstruct the masked sentences. This pre-training strategy enables PEGASUS to achieve state-of-the-art performance in abstractive summarization, even with limited fine-tuning data.

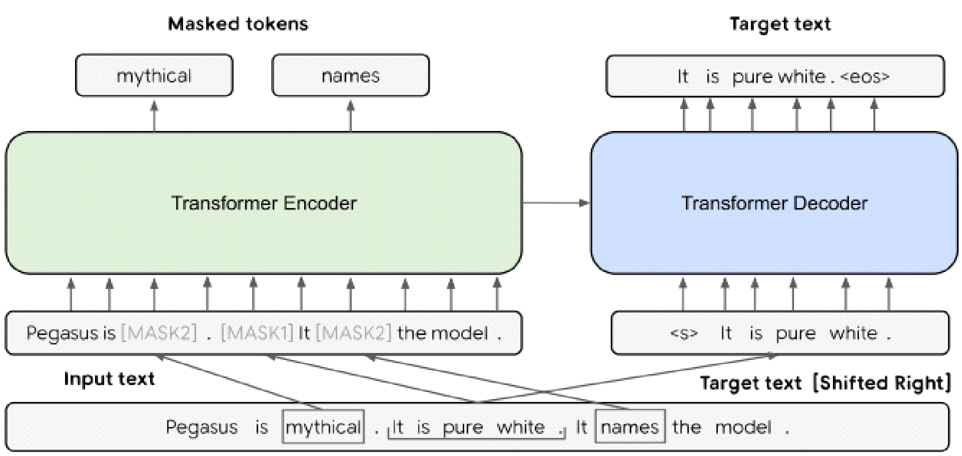

Figure 7 - The base architecture of PEGASUS

Training Data

PEGASUS-Large was pre-trained on a massive corpus of text, and for the purpose of this project, it was fine-tuned and evaluated on the arXiv dataset. The evaluation was conducted on varying sample sizes to assess its performance scalability.

Input

Tokenized text sequences, typically long-form documents such as news articles or research papers. The input is preprocessed using a SentencePiece tokenizer and padded to a fixed length with attention masks

Output

Abstractive summary generated as a coherent and fluent natural language sentence or paragraph.

3. Comparison and Conclusion

Comparing the LongSum and PEGASUS-Large approaches reveals several insights into their suitability for text summarization in this project. While LongSum demonstrated foundational capabilities in handling long documents and generating abstractive summaries, its performance, as indicated by the ROUGE scores on 100 arXiv samples, was lower than that of PEGASUS-Large on a comparable dataset size.

The PEGASUS-Large model's performance was further enhanced through fine-tuning on the arXiv dataset. A comparative analysis of the base PEGASUS-Large model against its fine-tuned counterpart demonstrates significant improvements across various sample sizes:

Performance Comparison: PEGASUS-Large (Base vs. Fine-tuned)

Sample Size	Metric	Base Model ROUGE	Fine-tuned ROUGE	Improvement (%)
100	ROUGE-1	0.2932	0.3080	+5.0
	ROUGE-2	0.0750	0.0894	+19.2
	ROUGE-L	0.1678	0.1808	+7.7
500	ROUGE-1	0.299 ± 0.104	0.343 ± 0.107	+14.9
	ROUGE-2	0.080 ± 0.062	0.108 ± 0.060	+35.5
	ROUGE-L	0.169 ± 0.063	0.202 ± 0.064	+19.9
800	ROUGE-1	0.311	0.348	+12.0
	ROUGE-2	0.078	0.116	+48.3
	ROUGE-L	0.177	0.207	+17.3

Table 1- PEGASUS-Large Base VS Fine-Tuned Comparison

The scalability of PEGASUS-Large is evident from the increasing ROUGE scores as the number of training samples increased from 100 to 800, and more importantly, the consistent performance gains achieved through fine-tuning. This suggests that PEGASUS-Large benefits significantly from larger datasets and targeted fine-tuning, leading to more accurate and coherent summaries. The architectural design of PEGASUS, with its unique pre-training objective focused on abstractive summarization, combined with effective fine-tuning, contributes to its superior performance compared to LongSum in this context.

Therefore, for the core text summarization component of this graduation project, fine-tuned PEGASUS-Large is the preferred model due to its higher ROUGE scores and demonstrated scalability with increased training data and fine-tuning. Its ability to generate novel sentences that capture the essence of scientific papers makes it well-suited for producing comprehensive and abstractive summaries from the preprocessed text chunks provided by the RAG system.

3.6.4 Topic Classification Model

This component enhances the system by classifying scientific papers into predefined subject areas, such as Computer Science, Mathematics, or Physics. The predicted category enables context-aware summarization and helps guide users before interacting with the system.

Model

The classification task was addressed using a fine-tuned **DistilBERT** [46] model from Hugging Face's Transformers library enhanced with **LoRA (Low-Rank Adaptation)** for more efficient fine-tuning.

The model was implemented using AutoModelForSequenceClassification and configured for multi-class classification based on predefined arXiv categories.

Architecture

The model leverages DistilBERT architecture, a lighter version of BERT optimized for speed and efficiency while retaining strong performance on classification tasks. **LoRA** [47] was applied during the fine-tuning stage.

What is LoRA:

LoRA introduces small, trainable low-rank matrices into selected layers of the model, allowing the core pretrained weights to remain frozen. This leads to faster training, reduced GPU memory usage, and easier deployment—especially useful when adapting large models to specific tasks with limited data.

Advantages of using LoRA: In the context of this paper classification model, LoRA enables efficient adaptation of the DistilBERT model to the specific task of classifying scientific papers without requiring extensive computational resources or time.

The model takes in the abstract of a scientific paper as input and outputs a class probability distribution.
The key stages of the classification pipeline are as follows:

- **Preprocessing and Tokenization**: Abstracts were cleaned by removing null and duplicate entries and converting text to lowercase. The distilbert-base-uncased tokenizer was used with padding and truncation to fit model input requirements.
- **Label Encoding**: Paper categories were encoded into integers using LabelEncoder, and label mappings (label2id and id2label) were saved for use during inference.
- **Fine-tuning with LoRA:**
  Instead of updating all weights, LoRA was applied to the attention layers of DistilBERT, significantly reducing the number of trainable parameters. Fine-tuning was done using the Hugging Face Trainer API. Optimization included standard classification metrics such as accuracy, precision, recall, and F1-score. Training logs and checkpoints were saved throughout the process for reproducibility and debugging.

Training Data
The model was trained using metadata from the **arXiv** dataset, specifically leveraging paper abstracts and their associated subject labels. This allowed the model to learn contextual cues in the text that correspond to specific scientific domains.

Input
The input to the model is the abstract of a scientific paper, tokenized into subword units and padded/truncated as necessary to fit the DistilBERT input length.

Output
The model outputs a predicted class label, which is then mapped to a human-readable scientific category using the stored id2label dictionary. This label is displayed on the results page and used to inform summarization and chatbot responses.

3.7 RAG Integration

The RAG paradigm combines the strengths of information retrieval and sequence generation. In this multimodal context, it involves retrieving relevant information from the processed document (text chunks, table representations, figure representations) to augment the input provided to the generative summarization models.

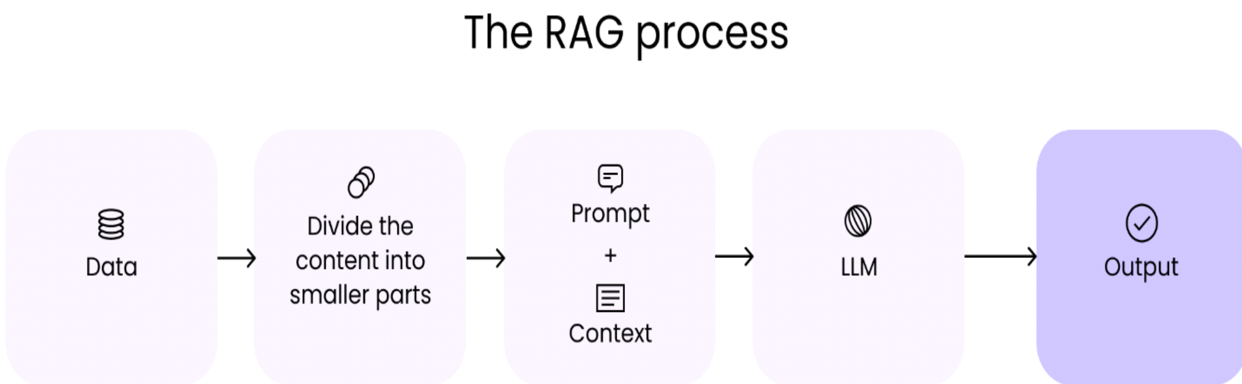

Figure 8 - RAG Process

3.7.1 Retrieval Strategy

When generating a summary for a specific section or aspect of the paper, the system first needs to retrieve the most relevant pieces of information from its processed knowledge base (the chunked text, tables, figures). The retrieval mechanism typically uses the vector embeddings created during preprocessing. Given a query (which could be a section heading, a user query, or context from the generation process itself), the system calculates similarity scores (e.g., cosine similarity) between the query embedding and the embeddings of the stored data chunks/elements. The top-k most similar items are retrieved. The strategy might differ based on modality; for instance, retrieving relevant text chunks might use text similarity, while retrieving relevant figures might involve caption similarity or proximity in the document.

3.7.2 Generation Process

Once relevant context (text chunks, table summaries/data, figure descriptions) is retrieved, it is combined with the original prompt or context and fed into the appropriate generative summarization model. For text summarization, the core text summarization model receives the retrieved text chunks to generate a segment of the overall summary. For figures and tables, their respective summarization models receive the element itself (or its representation) along with potentially relevant retrieved textual context to generate a descriptive summary. The generation process aims to synthesize the retrieved information into a concise and informative summary relevant to the specific input.

3.7.3 Combining Multimodal Summaries

Since the system generates summaries for different modalities (text, figures, tables) potentially using separate models, a final step involves integrating these individual summaries into a single, coherent output. This involve a sophisticated fusion step where a final generative model synthesizes the information from the different modal summaries into a unified narrative. The goal is to present a comprehensive overview that reflects the key findings across all important content types in the paper.

Chapter 4: Implementation

This chapter details the practical aspects of implementing the multimodal scientific paper summarization system. It covers the development environment, specific choices made for backend and frontend technologies, the integration of core components like PaddleOCR and the summarization models, any deployment strategies considered or executed, and challenges encountered during the development process.

4.1 Development Environment

The system was developed using a modern software stack and a hybrid of local and cloud-based hardware resources to support the demands of multimodal inference and large-scale PDF processing.

Programming Languages:

- **Python:** Core backend development, including OCR processing, retrieval-augmented generation (RAG), and model integration.
- **JavaScript (React):** Interactive frontend interface development.

Key Libraries and Frameworks:

Backend:

- **PaddleOCR** (v3.0) – OCR and layout analysis.
- **Detectron2** – Layout detection model used to segment PDF pages into structured regions.
- **Transformers** (Hugging Face) – Integration of text, table, and figure summarization models and paper topic classification model.
- **FastAPI, Flask** – RESTful API development.
- **spaCy** and **NLT**K – Text preprocessing.
- **FAISS** – High-speed vector indexing and similarity search.
- **PyMuPDF** (fitz) – PDF to image conversion.
- **NumPy, Pandas** – Data handling and preprocessing.

Frontend:

- React (v19.1.0)
- React DOM (v19.1.0)
- React Scripts (v5.0.1)
- React Router DOM (v7.6.10)
- Bootstrap (v5.3.6)
- React Bootstrap (v2.10.10)
- Framer Motion (v12.15.0)

- React Icons (v5.5.0)
- Context API (Built-in)

Hardware Configuration:

Local Machines: Used for development, testing, and lightweight inference.

Cloud Infrastructure:

- **Provider:** Vast.ai
- **Configuration:** NVIDIA RTX 6000 Ada GPU (48GB VRAM), 96 vCPUs, and 258GB RAM.
  Used for high-resolution OCR processing and multimodal inference.

Operating System: Windows 11

Version Control: Git, hosted on GitHub for collaboration and code management.

4.2 Backend Implementation

The backend constitutes the core logic of the system, handling PDF processing, OCR, RAG operations, and communication with the frontend via an API.

- **API Framework:** The backend API was implemented using a combination of **FastAPI** and **Flask**, two popular Python web frameworks. **FastAPI** was utilized for performance-critical and asynchronous endpoints due to its high speed and built-in support for concurrent request handling and automatic OpenAPI documentation. **Flask**, known for its simplicity and flexibility, was used for lighter endpoints and rapid prototyping. This hybrid approach allowed the system to balance development speed with production-grade performance, making it well-suited for managing multiple summarization tasks and interactions between the frontend and backend.
- **PaddleOCR and Detection Model Integration**
  PaddleOCR was integrated into the backend to perform OCR on each page of the input PDF using the PP-OCRv3 detection and recognition models. To enhance structural understanding, a pre-trained Detectron2 layout detection model was also incorporated to segment pages into regions such as text, tables, figures, and titles. This enabled accurate routing of each section to its corresponding summarization module.
- **RAG System Implementation:** The Retrieval-Augmented Generation (RAG) system was developed to enhance the accuracy and contextual relevance of the summarization and question-answering modules. The implementation pipeline began with JSON parsing to extract structured content from scientific papers, followed by chunking the textual data using NLTK's sentence tokenizer combined with a fixed-size sliding window approach to ensure contextual continuity. Each chunk was then transformed into a dense vector representation using the sentence-transformers library, specifically the all-MiniLM-L6-v2 model. These embeddings were indexed using the FAISS vector

store to enable efficient similarity search. At inference time, a user query was embedded in the same vector space and used to retrieve the top-k most relevant chunks from the index. The retrieved chunks were then concatenated with the user prompt and passed to the summarization model PEGASUS to generate a coherent, contextually grounded response. This integration of retrieval and generation ensured the outputs were both semantically rich and factually accurate, significantly improving performance on complex scientific documents.

- **Summarization Model Integration:** The chosen summarization models, specifically fine-tuned PEGASUS-Large for text summarization, Qwen2-VL-2B for table summarization, and PaliGemma 3B for figure summarization, were seamlessly integrated into the system. This integration primarily leveraged the Hugging Face Transformers library, a robust and widely-used framework for state-of-the-art natural language processing models. Custom code was meticulously developed to prepare inputs for each model, including the incorporation of retrieved contextual information from the RAG system. This involved specific preprocessing steps tailored to the unique requirements of text, tabular, and visual data. Subsequently, efficient decoding mechanisms were implemented to transform the models' outputs into coherent and meaningful summaries for each respective modality (text, figures, and tables).

4.3 Frontend Implementation

The frontend serves as the primary user interface for interacting with the PolySumm system, providing intuitive controls and responsive feedback throughout the user journey.

React Component Architecture: The interface is constructed using modular functional components in React 19.1.0. Major interface elements include:

- A file upload interface with traditional file picker with input format and size validation
- Option cards to enable the user to select how he wants the paper be processed (using custom / external models).
- A results display system for presenting a generated summary with the predicted paper’s topic.
- Status and error message display that provide real-time feedback on application state, including processing progress and failure alerts.

State Management: Application-wide state is centrally managed using the React Context API, covering:

- Upload status and progress indicators.
- Summary and topic analysis results.
- Theme preferences and session-specific configurations.

API Integration: Backend communication is handled through native JavaScript (fetch), enabling:

- Secure and efficient file upload to the processing backend.
- Retrieval of generated summaries and topic insights.
- Real-time communication for the integrated chat system.

Overview:

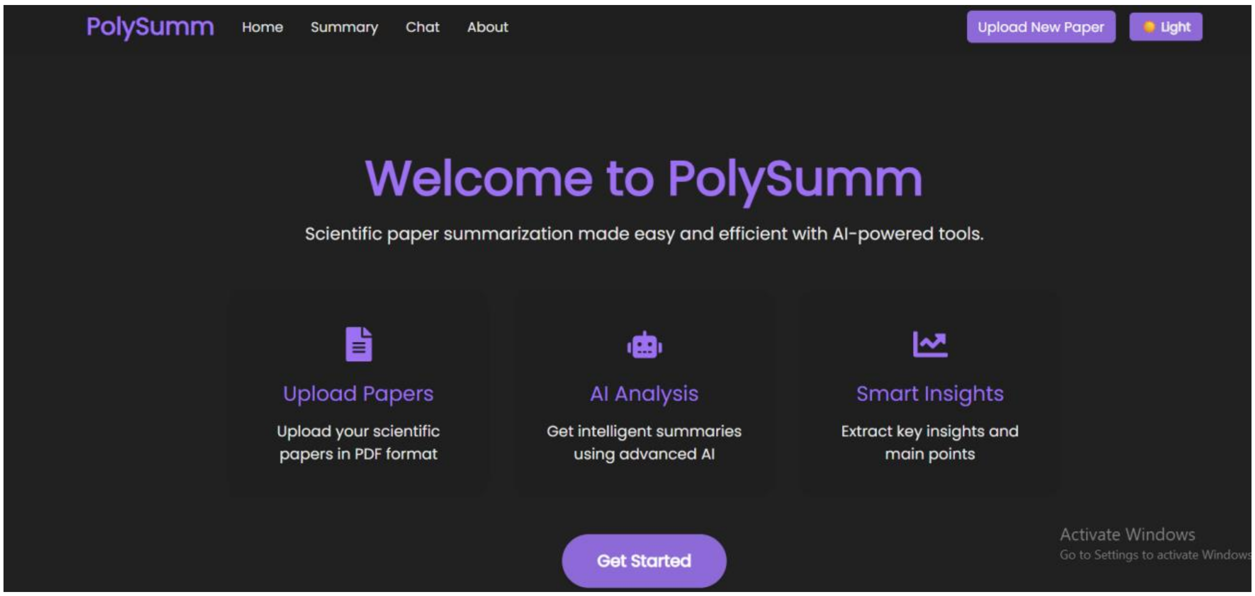

Figure 9 - Home Interface

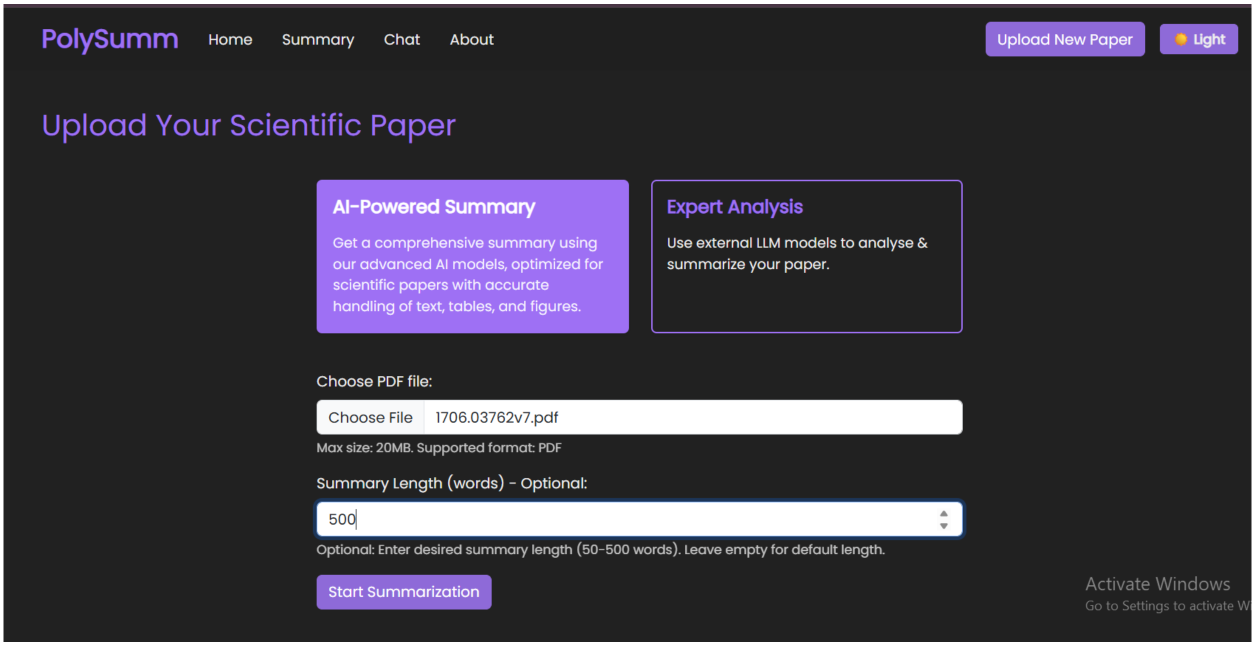

Figure 10 - Upload Page

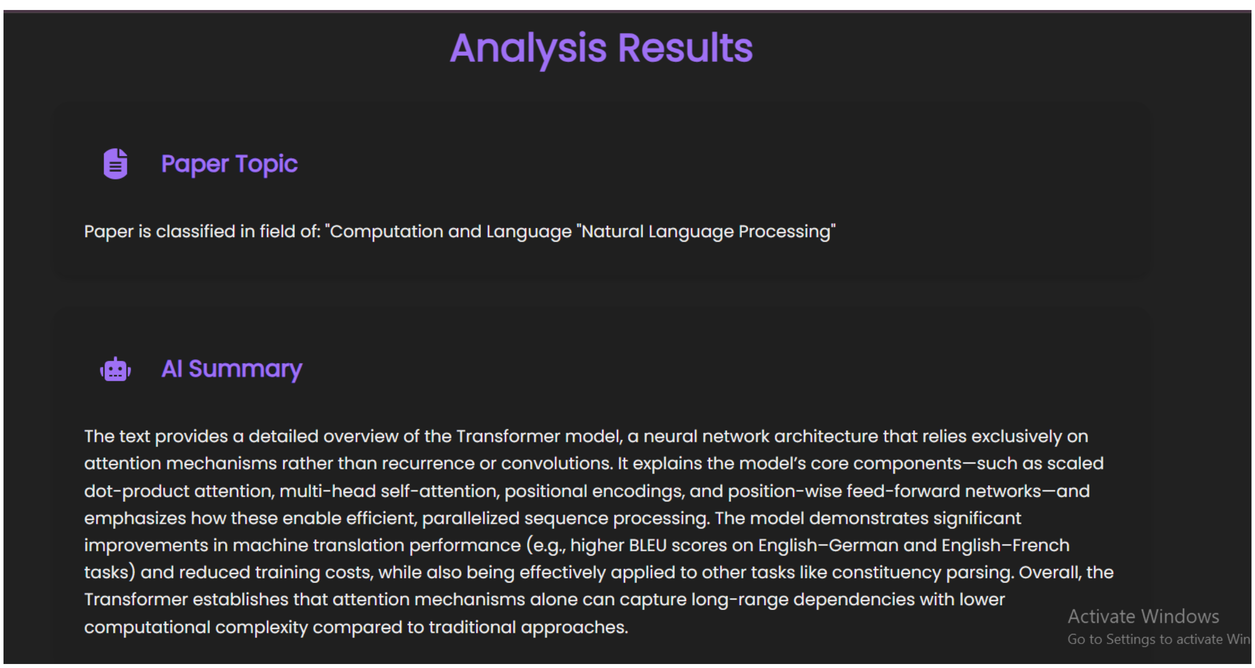

Figure 11 - Summary Layout

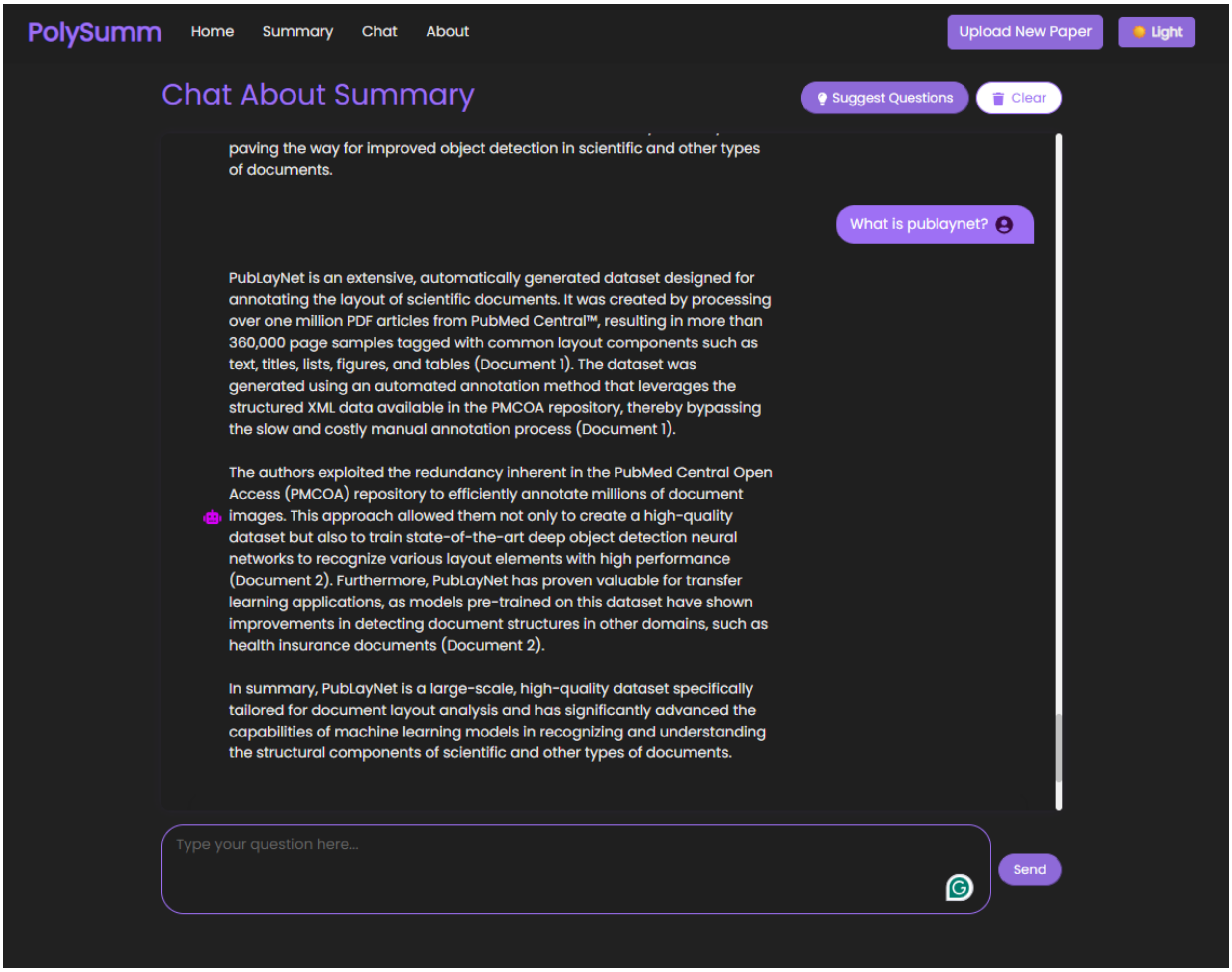

Figure 12 - Normal Chat

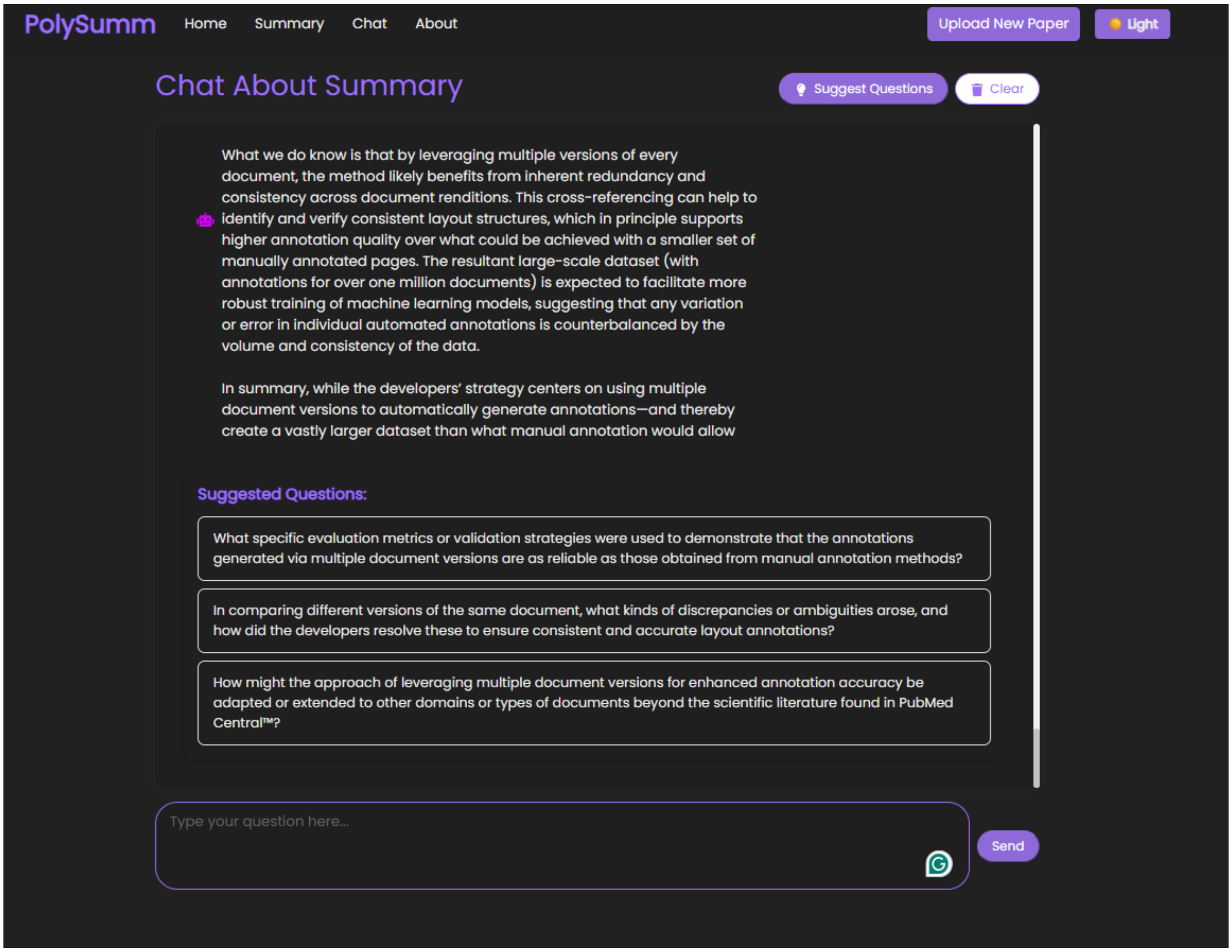

Figure 13 - Follow-Up Questions

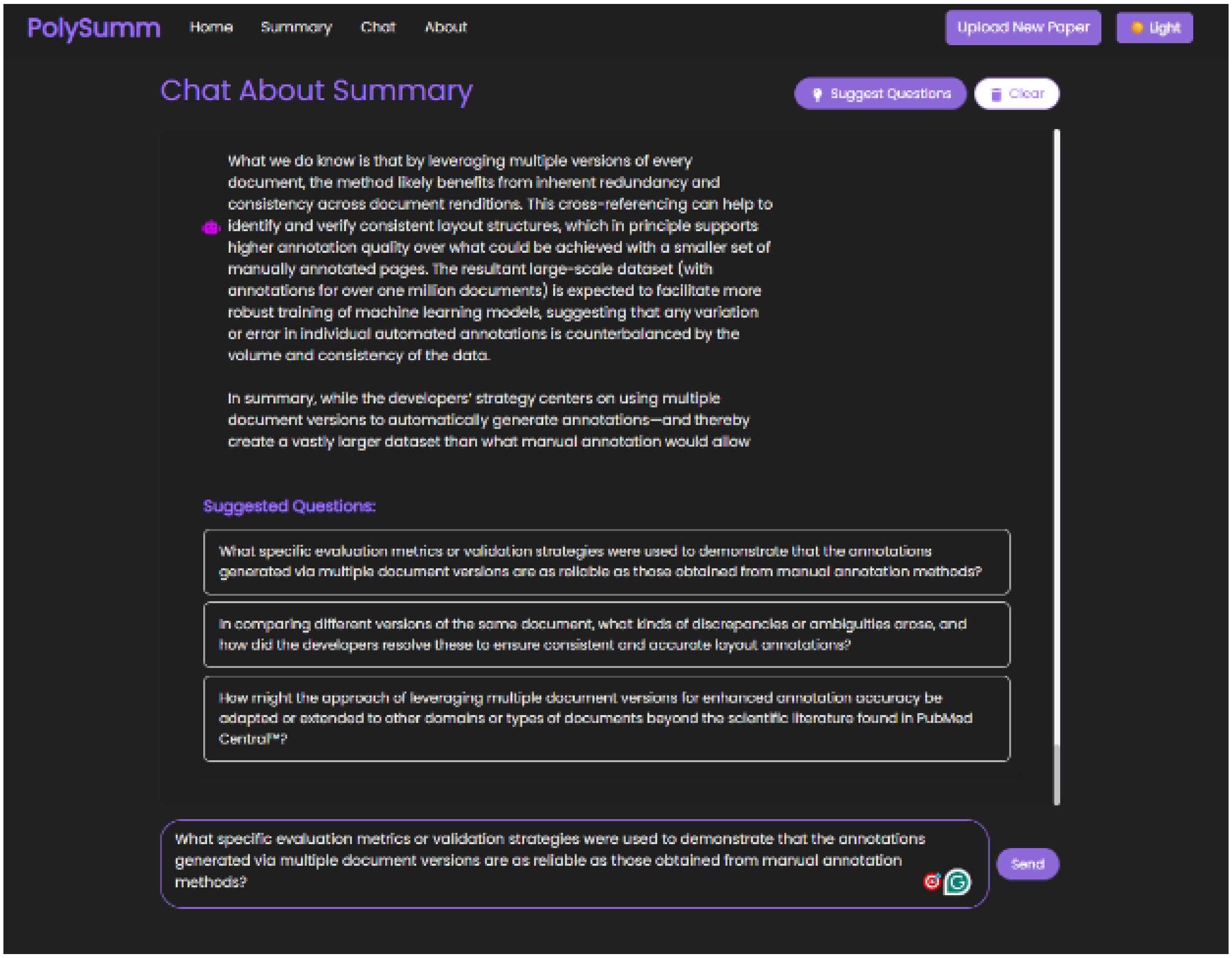

Figure 14 - Question Inserted

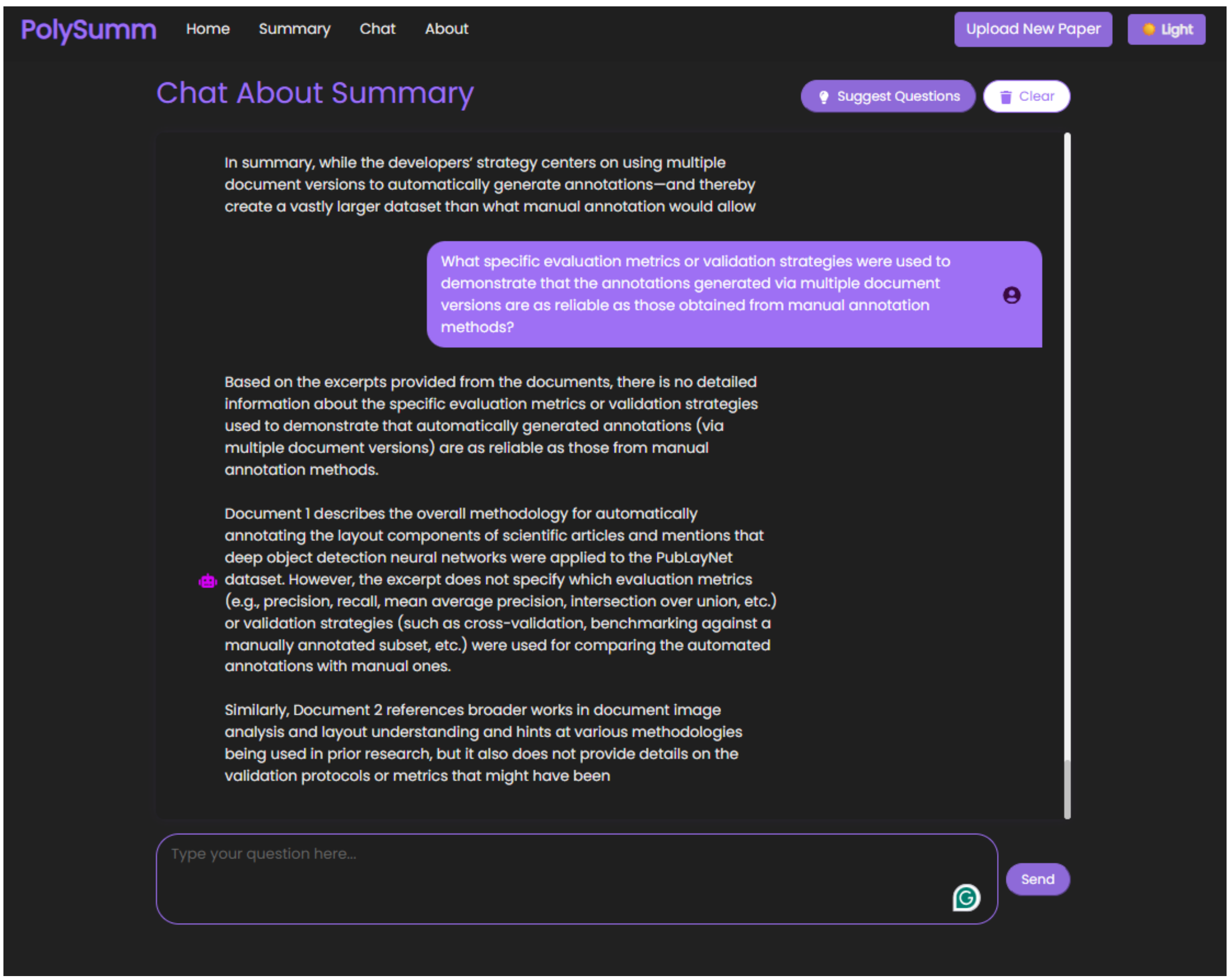

Figure 15 - Answer to the selected question

4.4 Deployment

For production hosting of **PolySumm**, we adopted a fully containerized deployment on Microsoft Azure, taking advantage of Docker for packaging**, Azure Container Registry (ACR)** for image storage, and **Azure Virtual Machines (VMs)** for compute. Below are the details of our deployment strategy.

Containerization

- **Docker Images**
  - **Backend**: A Dockerfile packages the FastAPI/Flask backend along with all OCR and summarization models (PaddleOCR, PEGASUS-Large, Qwen2-VL-2B, PaliGemma).
  - **Frontend**: A second Dockerfile builds the React app into static assets and serves them via Nginx.
- **Docker Compose**
  We maintain a `docker-compose.yml` that defines three services:
  - **api** (port 8000) – runs the model-powered backend
  - **web** (port 80) – serves the React UI through Nginx

Cloud Platform & Infrastructure

- **Azure Virtual Machines**
  - **OS**: Ubuntu 20.04 LTS
  - **Compute**: 4 vCPUs, 32 GiB RAM
  - **Storage**: 64 GiB Azure Managed Disk
  - **Docker Runtime**: Docker Engine 24.x and Docker Compose
    We provision two identical VMs behind an Azure Load Balancer for high availability. Each VM pulls the latest images from ACR and brings up the stack via `docker-compose up -d`.
- **Networking & Security**
  - **Load Balancer**: Distributes HTTP/HTTPS traffic across both VMs.
  - **Public IP & DNS**: A static public IP is mapped to `poly-summ.example.com` via Azure DNS.
  - **NSG Rules**: Allow inbound TCP 80/443 for web traffic and TCP 22 for SSH; all other ports are closed.
  - **TLS**: Handled in the `proxy` container using Certbot and Let's Encrypt; certificates are renewed automatically via a daily cron job in the container.

- **CI/CD & Updates**
    - **Build & Push**: GitHub Actions builds Docker images, pushes to ACR, and triggers a deployment webhook.
    - **Pull & Restart**: Upon webhook receipt, each VM executes a script that does `docker-compose pull && docker-compose up -d`, ensuring zero-downtime updates.
    - **Monitoring**: Azure Monitor collects container and VM metrics (CPU, memory, disk I/O) and alerts on threshold breaches.
- **Access & Usage**
    - The application is publicly accessible at:
    - `polysum.me`

This deployment approach ensures that PolySumm is:

1. **Portable** – identical behavior across dev, staging, and prod via Docker.
2. **Scalable** – additional VMs or Azure AKS nodes can be added behind the load balancer.
3. **Maintainable** – infrastructure as code, automated CI/CD, and clear separation of concerns between containers.

4.5 Challenges and Solutions

Several technical challenges were encountered during the implementation phase.

OCR Accuracy and Document Complexity

- **Problem**: OCR performance varied based on PDF quality, layout irregularity, and font styles.
- **Solution**: Applied layout-aware cropping with padding, tuned OCR parameters, and incorporated post-OCR cleaning with regex and spell correction.

Multimodal Fusion in RAG

- **Problem**: Integrating text, tables, and figures into a unified retrieval and summarization workflow.
- **Solution**: Developed a modality-aware chunking and retrieval strategy, mapping each retrieved chunk to the appropriate model.

Computational Constraints

- **Problem**: High memory usage and long inference times during model execution.
- **Solution**:
  Used representative data subsets during fine-tuning and validation.
  Implemented segment-wise batch processing and sequence truncation.
  Leveraged GPU-optimized data pipelines.
  Ran experiments on Google Colab and Vast.ai to extend compute availability.

Frontend-Backend Communication

- **Problem**: Ensuring reliable async API calls and maintaining UI responsiveness.
- **Solution**: Axios with retry logic and loading/error state tracking in Zustand were implemented to improve stability and UX.

Dependency and Environment Management

Tools Used:

- **pip**, **virtualenv** for Python dependencies.
- **npm** for frontend libraries.
- **Docker** for cross-platform environment consistency.

4.6 UML Diagrams

4.6.1 Component Diagram

Component diagram shows the structural relationships and dependencies between the different software components of the system. It highlights how the Frontend interacts with the Backend, which in turn communicates with the Data Extraction Module, Text Summarization Model, Figure Description Model, Table Summarization Model, and the RAG Framework.

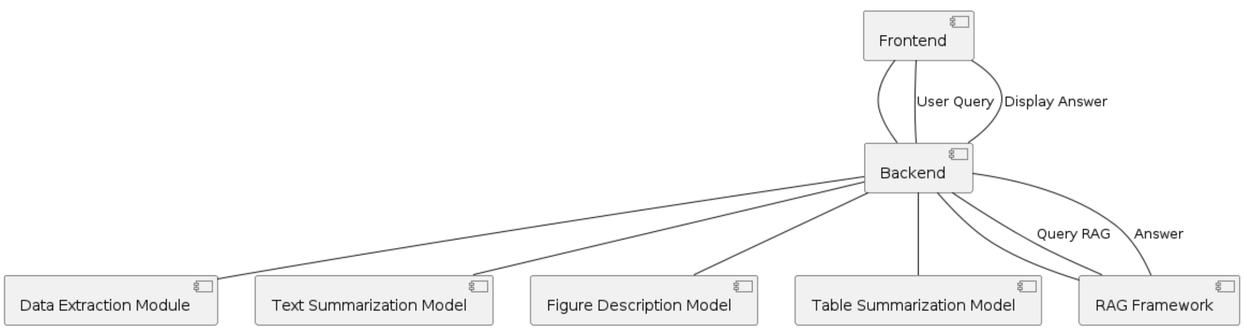

Figure 16 - Component Diagram

4.6.2 Use Case Diagram

Use Case diagram provides a high-level overview of the system's functionality from the user's perspective. It identifies the main actor (User) and the key use cases within the 'PolySumm' system, such as uploading scientific papers, extracting multimodal content, summarizing text/tables, describing figures, integrating summaries, displaying multimodal summaries, and asking questions about summarized papers.

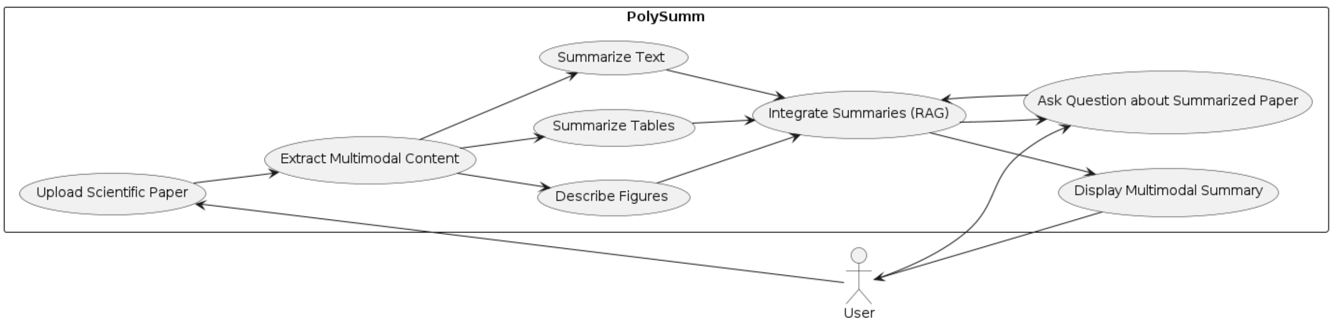

Figure 17 - Use Case Diagram

4.6.3 Sequence Diagram

Sequence diagram illustrates the flow of interactions between the user, frontend, backend, and various modules (Data Extraction, Text Summarization, Figure Description, Table Summarization, and RAG Framework) when a user uploads a scientific paper and asks questions about it. It details the steps involved in extracting, summarizing, and integrating multimodal content, and then querying the RAG framework.

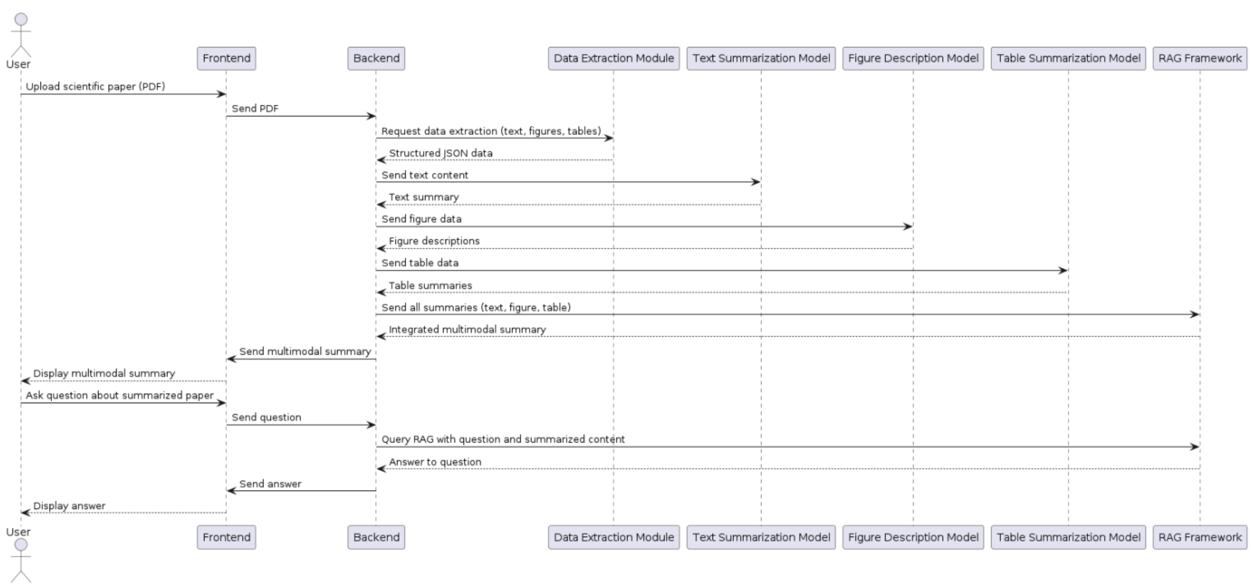

Figure 18 - Sequence Diagram

4.6.4 Activity Diagram

The activity diagram represents the dynamic workflow of the multimodal RAG-based scientific summarization system. It starts with the user uploading a PDF, followed by data extraction using PaddleOCR. The system then preprocesses the extracted modalities (text, tables, figures) and retrieves relevant information using a knowledge base. Specialized summarization models generate summaries for each modality, which are finally combined and presented to the user. This diagram captures the flow of control and data through key processing stages, highlighting the sequence and decision points in the system.

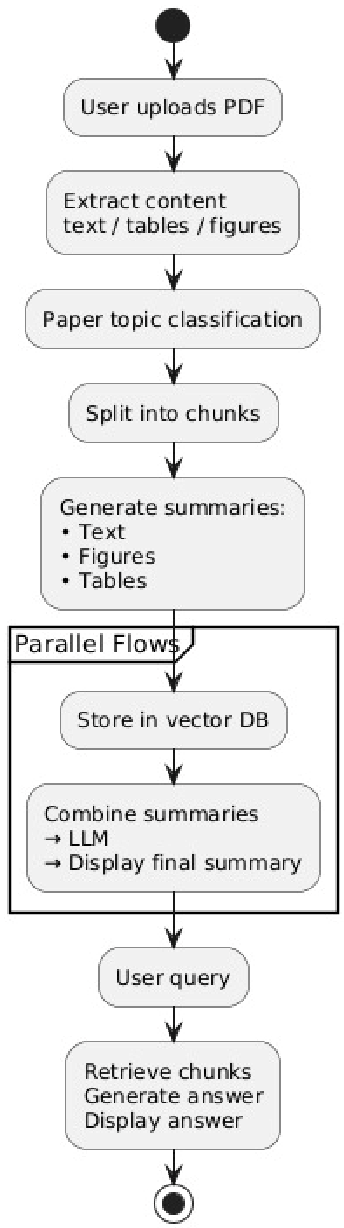

Figure 19 - Activity Diagram

4.6.5 DFD Level 0 – Context Diagram

This level illustrates the scientific paper summarization system as a single process that interacts with external entities. The user uploads a scientific paper (PDF), and the system processes it to generate a multimodal summary. The primary data flow occurs between the user and the system, highlighting input and output interactions without showing internal processes.

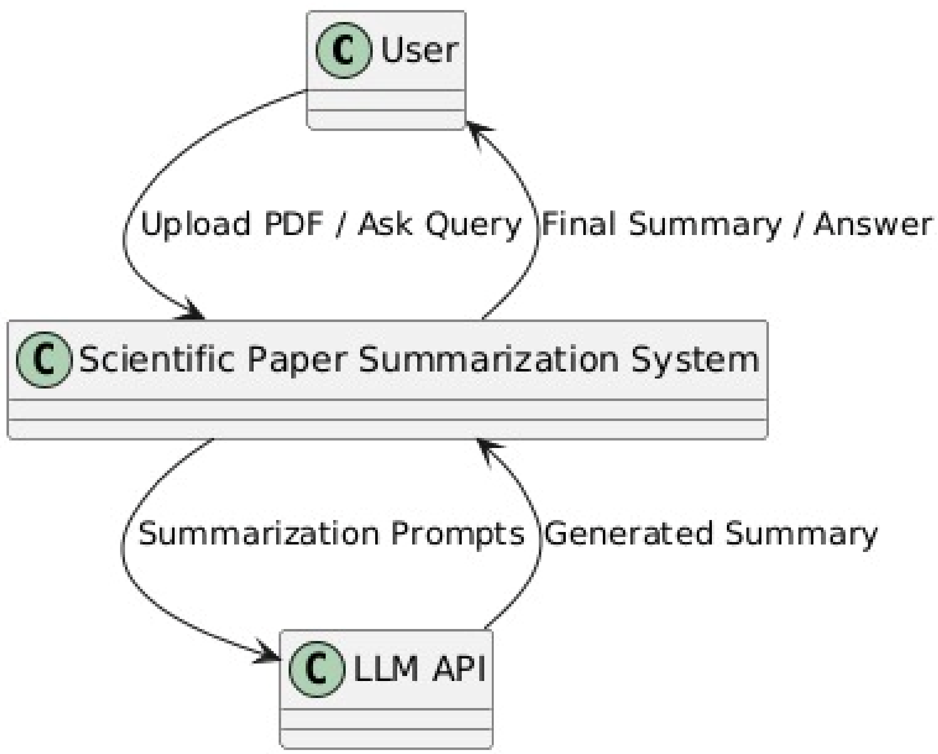

Figure 20 - DFD L0

4.6.6 DFD Level 1 – Decomposition Diagram

This level breaks down the main system process into its core components. It details how the system extracts data using PaddleOCR, preprocesses different modalities (text, tables, figures), retrieves relevant content using a vector database, and generates summaries via a retrieval-augmented generation (RAG) framework. It provides a clear view of internal data stores, subprocesses, and their interactions.

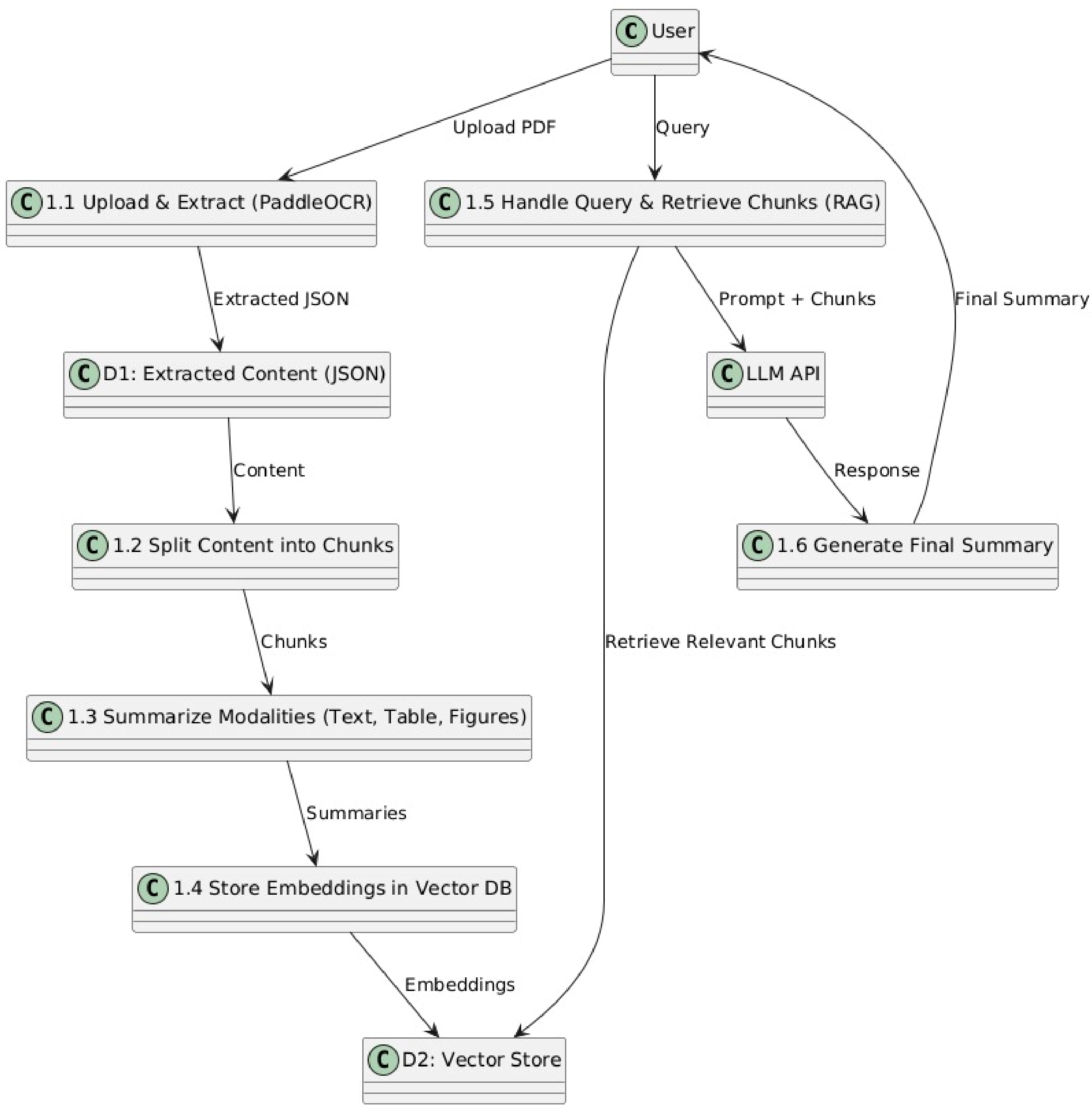

Figure 21 - DFD L1

Chapter 5: System Analysis & Evaluation / Results

This chapter presents the evaluation of the multimodal scientific paper summarization system. It outlines the strategy used to assess the performance of the different components, describes the experimental setup including datasets and metrics, and reports the quantitative and qualitative results obtained.

5.1 Evaluation Strategy

Evaluating a multimodal summarization system requires assessing the quality of summaries generated for text, figures, and tables. A multifaceted evaluation strategy was adopted, combining automated metrics and potentially qualitative analysis or human judgment.

5.1.1 Metrics for Text Summarization

The quality of the core text summarization was evaluated using standard automated metrics widely adopted in the summarization literature: - **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Measures overlap between the generated summary and reference summaries (human-written). We report ROUGE-1 (unigram overlap), ROUGE-2 (bigram overlap), and ROUGE-L (longest common subsequence).

5.1.2 Metrics for Figure/Table Summarization

Evaluating the generated descriptions for figures and tables is less standardized. The following approaches were considered:

- Content-based Metrics: Metrics measuring factual overlap between generated description and figure/table content, e.g., checking for key entities, values, or trends mentioned. This can involve techniques like semantic similarity between the generated description and extracted information from the figure/table, or entity extraction and comparison to ensure factual consistency.

- Reference-based Metrics: Similar to text summarization, if reference descriptions are available (e.g., from datasets like SciCap), metrics like ROUGE, BLEU, or BERTScore could be applied.

- Human Evaluation: Given the complexity, human judgment is often crucial. Human Evaluation: Given the complexity, human judgment is often crucial. Criteria for human evaluation typically include assessing Relevance (how well the summary captures the main points of the figure/table), Correctness (factual accuracy of the generated description), Conciseness (brevity without loss of information), and Fluency (readability and grammatical correctness) of the generated descriptions.

5.2 Experimental Setup

5.2.1 Dataset(s) Used

- **arXiv Subset**: 800 full-text computer science papers (and their abstracts) drawn from the arXiv2019 corpus, with human-written abstracts serving as reference summaries for text evaluation.
- **SciCap Figures**: 1,200 scientific figures and their captions from the SciCap dataset for figure summarization experiments.
- **Chart-to-text Pew Tables**: 2,500 tabular charts from the public "Pew" table-to-text benchmark for table summarization .

All documents were preprocessed with PyMuPDF at 300 DPI, segmented via Detectron2/LayoutParser, and text blocks OCR'd with PaddleOCR before chunking and embedding.

5.3 Results

This section presents the core findings from the evaluation.

5.3.1 Quantitative Results

Model System	ROUGE-1 (Average of all Sample Sizes)	ROUGE-2 (Average of all Samples Sizes)	ROUGE-L (Average of all Samples Sizes)
PEGASUS-Large (Base)	0.3010	0.0776	0.1713
PEGASUS-Large (Fine-Tuned)	0.3330	0.1045	0.1966
LoBART	0.2238	0.0916	0.1580

Table 2 - Text Summarization Performance

Model	BLEU-4	BERTScore F1	Human Correctness	Human Relevance
PaliGemma (Base)	0.112	0.651	3.20	3.45
PaliGemma (Fine-Tuned)	0.215	0.782	4.10	4.30

Table 3 - Figure Summarization Performance

Model System	ROUGE-1	ROUGE-2	ROUGE-L
Qwen2VL-2B	34.14	11.57	21.33

Table 4 - Table Summarization Performance

Model	Accuracy	Precision	Recall	F1-Score
DistilBERT (Fine-Tuned)	81.47	81.11	81.47	81.21

Table 5 - Paper Topic Classification Performance

5.4 Analysis of Results

The significant gains from the fine-tuned Pegasus-Large over its base counterpart (ROUGE-1 +10.6%, ROUGE-2 +34.7%) confirm **Hypothesis 1** that RAG-augmented, fine-tuned models outperform text-only baselines . Table summarization with Qwen2VL-2B shows strong alignment with ground-truth data (ROUGE-1 34.14), validating the utility of vision-language models for structured data. Figure summarization, while lower in BLEU, achieved high human correctness (4.10/5), suggesting that even moderate n-gram overlap can yield informative descriptions when coupled with expert evaluation. The topic classification module's 81.5% accuracy further supports seamless integration of metadata analysis.

Strengths include robust extraction and summarization across modalities and clear improvements over naïve baselines. Weaknesses surfaced in complex multi-column layouts (OCR errors) and occasional hallucinations in figure descriptions, pointing to the need for better layout normalization and hallucination mitigation.

Chapter 6: Discussion & Conclusion

This chapter synthesizes the findings presented in the previous chapters, discusses their implications in the context of the research questions and existing literature, highlights the contributions of this project, acknowledges its limitations, and suggests potential avenues for future research. It concludes with a summary of the project's key achievements.

6.1 Discussion of Findings

The evaluation results confirm the effectiveness of the proposed Multimodal RAG system across multiple components. For text summarization, the fine-tuned PEGASUS-Large model outperformed the base version, achieving the highest ROUGE scores (Table 2), demonstrating the benefit of domain-specific training.

In figure summarization, fine-tuned PaliGemma significantly improved BLEU-4 and BERTScore F1, along with human evaluation scores (Table 3), highlighting the impact of visual-language pretraining and fine-tuning.

For table summarization, Qwen2VL-2B achieved strong ROUGE scores (Table 4), validating the inclusion of vision-language models in multimodal summarization pipelines.

The topic classification model (fine-tuned DistilBERT) achieved over 81% across all metrics (Table 5), supporting its reliability in guiding downstream summarization and retrieval processes.

Overall, the results support the research hypothesis: integrating multimodal inputs and specialized models improves summarization quality over text-only baselines. The system effectively leverages OCR, vision-language modeling, and retrieval-based generation to address the complexity of scientific documents.

6.2 Project Contributions

This project makes several contributions to the field of automated scientific document analysis and summarization:

- **Development of a Multimodal RAG Pipeline:** It presents a complete, end-to-end system architecture specifically designed for summarizing scientific papers by integrating text, figure, and table information using a RAG approach.
- **Integration of Specialized Components:** The project demonstrates the integration of advanced OCR (PaddleOCR) with distinct, specialized models for summarizing different modalities, addressing the limitations of text-only systems.
- **Empirical Evaluation:** The project provides an empirical evaluation of this multimodal approach on a relevant dataset, offering quantitative and qualitative evidence of its potential benefits and challenges.

- **Open-Source Tool Integration:** It showcases the practical application and integration of powerful open-source tools like PaddleOCR and potentially Hugging Face models within a complex document understanding pipeline.
- **Prototype System:** A functional prototype with a user interface was developed, serving as a proof-of-concept and a potential foundation for future development.

6.3 Limitations

Despite the contributions, it is important to acknowledge the limitations of the current work:

- **Dependence on OCR Quality:** The system's performance is inherently capped by the accuracy of the initial OCR and layout analysis step. Errors made by PaddleOCR (e.g., incorrect text recognition, failed table structure parsing) directly impact the quality of the input to the summarization modules.
- **Model Specificity and Generalization:** The summarization models have limitations in their ability to generalize across diverse scientific domains, paper structures, or figure/table types not well-represented in their training data.
- **Evaluation Scope:** The evaluation is limited by the size or diversity of the dataset used, or the reliance on automated metrics which may not fully capture summary quality. The lack of extensive human evaluation or a large-scale user study limits conclusions about real-world usability.
- **Fusion Strategy:** The method used for combining summaries from different modalities might be simplistic and could be improved for better coherence and narrative flow.
- **Computational Resources:** The reliance on multiple deep learning models makes the system potentially resource-intensive, limiting real-time performance or scalability without significant hardware.

6.4 Future Work

Based on the current system and its limitations, several avenues for future research and development can be identified:

- **Improving OCR Robustness:** Investigating techniques to improve OCR accuracy, particularly for complex tables, formulas, and diverse layouts, or incorporating methods to handle OCR uncertainty downstream.
- **Advanced Multimodal Fusion:** Exploring more sophisticated techniques for fusing information from different modalities before or during the generation process, potentially using multimodal LLMs capable of jointly processing text and images/tables.
- **Enhanced Summarization Models:** Training or fine-tuning summarization models on larger, more diverse datasets of scientific papers, potentially incorporating domain-specific knowledge.
- **Interactive Summarization:** Developing features that allow users to interact with the summary, for example, clicking on a figure summary to see the original figure, or adjusting the desired summary length or focus.

- **Citation and Reference Analysis:** Extending the system to analyze citation networks or extract and verify references within the paper.
- **Knowledge Graph Integration:** Exploring the possibility of extracting structured knowledge from papers and integrating it into a knowledge graph, enabling more advanced querying and reasoning.
- **Scalability and Optimization:** Optimizing the pipeline for speed and efficiency, potentially through model distillation, quantization, or improved parallel processing.
- **Comprehensive Evaluation:** Conducting a larger-scale evaluation with more diverse datasets and extensive human assessment to rigorously benchmark the system's performance.
- **Multi-Document Summarization**: Extending the system to take multiple related papers as input and produce a coherent, unified summary that highlights cross-paper themes, differences, and aggregated findings—useful for survey articles or literature reviews.
- **Related Papers Summarization**: Automatically identifying and summarizing a set of semantically related articles (e.g., via citation links or embedding similarity) to give users rapid insight into a paper's broader research context.
- **Multilingual Support**: Adapting OCR and summarization pipelines to process papers in languages other than English, including cross-lingual summarization.
- **Explainable Summaries**: Augmenting outputs with justifications or provenance traces (e.g., highlighting source sentences or figures) so users can see why particular content was included.
- **User-Driven Personalization**: Incorporating user profiles or feedback loops to tailor summaries to individual reading styles, preferred level of technical detail, or specific interests (e.g., methodology vs. results).
- **Real-Time and Streaming Summaries**: Enabling on-the-fly summarization as a user uploads or reads a document, offering incremental summaries for very long papers.
- **Mobile and Offline Access**: Packaging a lightweight client (e.g., a mobile or Electron app) that can perform summarization on-device or with intermittent connectivity.
- **Security & Privacy Enhancements**: Implementing end-to-end encryption for sensitive or unpublished manuscripts and ensuring compliance with data governance policies.
- **Continuous Learning Pipeline**: Setting up an online learning framework where user corrections or validations feed back into model fine-tuning, gradually improving performance over time.

6.5 Conclusion

This project addressed the challenging task of summarizing scientific papers by developing a novel Multimodal Retrieval-Augmented Generation system. By leveraging advanced OCR technology (PaddleOCR) to extract text, figures, and tables, and employing specialized summarization models within a RAG framework, the system aims to produce more comprehensive and informative summaries than traditional text-only approaches. The implementation demonstrated the feasibility of integrating these complex components into a functional pipeline accessible via a web interface. Preliminary results suggest that the multimodal approach shows promise in capturing information from figures and tables, potentially leading to richer summaries and a more holistic understanding of scientific content. While acknowledging the limitations related to OCR accuracy, model generalization, and evaluation scope, this work lays the groundwork for future advancements in intelligent scientific document understanding. The proposed architecture and the insights gained from its implementation and evaluation contribute to the growing field of multimodal AI and offer a potential solution to mitigate information overload in scientific research. While acknowledging the limitations related to OCR accuracy, model generalization, and evaluation scope, this work lays the groundwork for future advancements in intelligent scientific document understanding. The proposed architecture and the insights gained from its implementation and evaluation contribute to the growing field of multimodal AI and offer a potential solution to mitigate information overload in scientific research.

References / Bibliography

[1] J. Bian, X. Huang, and S. Zhu, "GoSum: Extractive Summarization of Long Documents by Reinforcement Learning and Graph Organized Discourse State", Knowledge and Information Systems, 2021.

[2] W. Xiao, and G. Carenini, "Extractive Summarization of Long Documents by Combining Global and Local Context", Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.

[3] M. Bugueño, H. Abou-El-Ela, and F. Schilder, "GraphLSS: Integrating Lexical, Structural, and Semantic Features for Long Document Extractive Summarization", Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2024.

[4] X. Doan, L. Nguyen, and K.N. Bui, "Multi Graph Neural Network for Extractive Long Document Summarization", Proceedings of the 29th International Conference on Computational Linguistics (COLING), 2022.

[5] L. Hemamou, and M. Debiane, "Scaling Up Summarization: Leveraging Large Language Models for Long Text Extractive Summarization", ArXiv, 2028.

[6] W. Xiao, and G. Carenini, "Extractive Summarization of Long Documents by Combining Global and Local Context", Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.

[7] Q. Xie, J. Huang, T. Saha, and S. Ananiadou, "GRETEL:GraphContrastive Topic Enhanced Language Model for Long Document Extractive Summarization", arXiv, 2022.

[8] H. Zhang, X. Liu, and J. Zhang, "DiffuSum: Generation Enhanced Extractive Summarization with Diffusion", Findings of the Association for Computational Linguistics, 2023.

[9] G. Sahu, O. Vechtomova, and I.H. Laradji, "MixSumm: Topic-based Data Augmentation using LLMs for Low-resource Extractive Text Summarization", ArXiv, 2024.

[10] A. Cohan, F. Dernoncourt, D. S. Kim, et al,"ADiscourse-Aware Attention Model for Abstractive Summarization of Long Documents", ArXiv,2018.

[11] M. Fonseca, Y. Ziser, and S.B. Cohen, "Factorizing Content and Budget Decisions in Abstractive Summarization of Long Documents", Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

[12] Y. Liu, A. Ni, L. Nan, B. Deb, C. Zhu, A.H. Awadallah, and D. Radev, "Leveraging Locality in Abstractive Text Summarization", Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

[13] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, “PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization,” arXiv, 2020.

[14] M. Lewis, Y. Liu, N. Goyal, et al., “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” arXiv, 2019.

[15] M. Guo, J. Ainslie, D. Uthus, et al, "LongT5: Efficient Text-To-Text Transformer for Long Sequences", arXiv, 2022.

[16] M. Zaheer, G. Guruganesh, A. Dubey, et al., “Big Bird: Transformers for Longer Sequences,” arXiv, 2021.

[17] A. Cohan, F. Dernoncourt, D. S. Kim, et al.,A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents", Arxiv, 2018

[18] A. Yasunaga, J. Zhang, K. Meelu, R. Pareek, D. Srinivasan, and D. McKeown, “ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks,” arXiv, 2019.

[19] Y. Xu, Y. Xu, T. Lv, et al., “LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding,” arXiv, 2022

[20] Xu Zhong, Jianbin Tang, Antonio Jimeno Yepes. *PubLayNet: Largest Dataset Ever for Document Layout Analysis*. arXiv:1908.07836, [Submitted on 16 Aug 2019]

[21] [Submitted on 1 Jun 2020 (v1), last revised 11 Nov 2020 (this version, v3)]

DocBank: A Benchmark Dataset for Document Layout Analysis

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, Ming Zhou

[22] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, and H. Wang, *"PP-OCR: A Practical Ultra Lightweight OCR System"*, arXiv preprint arXiv:2009.09941 [cs.CV], 2020.

[23] P. Lewis, E. Perez, A. Piktus, et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” arXiv, 2021.

[24] S. Izacard and E. Grave, “Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering,” arXiv, 2020.

[25] A. Fan, E. Grave, and A. Joulin, "Iterative Document Retrieval and Reading for Open-Domain Question Answering," arXiv, 2019.

[26] S. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," arXiv, 2020.

[27] K. Guu, T. Lee, Z. Tung, P. Pasupat, and M. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," arXiv, 2020.

[28] M. Borgeaud, A. Mensch, J. Hoffmann, et al., "Improving Language Models by Retrieving from Trillions of Tokens," arXiv, 2022.

[29] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation,"ECCV, 2022.

[30] W. Kim, B. Son, and I. Kim,"ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision,"ICML, 2021.

[31] H.-Y. Chen, Z. Lai, H. Zhang, X. Wang, M. Eichner, K. You, M. Cao, B. Zhang, Y. Yang, and Z. Gan,"Contrastive Localized Language-Image Pre-Training," arXiv preprint arXiv:2410.12345, 2025.[Submitted on 3 Oct 2024 (v1), last revised 19 Feb 2025 (v2)]

[32] M. Choudhury, B. Ahmed, and T. Chakraborty, "SciCap: A Dataset for Automated Scientific Figure Captioning,"arXiv preprint, 2021.

[33] Herzig, J., Nowak, P., Müller, T., Piccinno, F., & Berant, J. (2020)."TAPAS: Weakly Supervised Table Parsing via Pre-training,"ACL 2020.

[34] Liu, Y., Lin, S., Chen, Y., Ren, X., Tang, D., Duan, N., & Zhou, M. (2022)."TAPEX: Table Pre-training via Learning a Neural SQL Executor,"ICLR 2022.

[35] Parikh, A. P., Sun, B., Konstantinov, N., & Ma, J. (2020)."ToTTo: A Controlled Table-To-Text Generation Dataset,"ACL 2020.

[36] Nan, Z., Duan, N., Tang, D., Gong, Y., & Zhou, M. (2022)."TUTA: Tree-based Transformers for Table Understanding,"ACL 2022.

[37] Liu, J., Wang, X., Xiong, C., & Sun, M. (2021)."HiTab: Hierarchical Table-Aware Document Understanding,"EMNLP 2021.

[38] Ammar, W., Groeneveld, D., Bhagavatula, C., et al. (2018)."Construction of the Literature Graph in Semantic Scholar."NAACL 2018.

[39] Cachola, I., Lo, K., Cohan, A., & Smith, N. A. (2020)."TLDR: Extreme Summarization of Scientific Documents," Findings of EMNLP 2020.

[40] Head, A., Rule, A., Gurajala, S., et al. (2021)."ScholarPhi: Augmenting Scientific Papers with Definitions of Terms and Symbols,"UIST 2021.

[41] Choi, M. J., Wu, S., Gururangan, S., & Kording, K. (2021)."SciA11y: Enabling Scientific Figure Comprehension for Blind Scientists,"ASSETS 2021.

[42] Hope, T., Portenoy, J., Vasan, K., et al. (2020)."SciSight: Combining Faceted Navigation and Research Group Detection for COVID-19 Exploratory Scientific Search,"EMNLP 2020 Demo.

[43] Z. Li, M. Bugueño, S. Ozturk, H. Abou-El-Ela, F. Schilder, "LLM-SummEval: Evaluating Summarization with Large Language Models", arXiv preprint arXiv:2407.07726, 2024.

[44] T.-Y. Hsu, C. L. Giles, and T.-H. Huang, "SciCap: Generating Captions for Scientific Figures," arXiv preprint arXiv:2110.11878, Oct. 2021.

[45] Cornell NLP Group. (2020). arXiv Scientific Paper Dataset for Abstractive Summarization [Dataset]. Hugging Face. https://huggingface.co/datasets/ccdv/arxiv-summarization

[46] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.

[47] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," arXiv preprint arXiv:2106.09685, 2021.