Optimizing VisDoMRAG System: Computational Efficiency and Answer Quality Improvements for Multimodal Document Question-Answering

Candidate Number: MWRP3 [1]

MSc Data Science and Machine Learning

Internal Supervisor: Professor Cox Ingemar
Industry Supervisor: Dr. Omer Gunes, Oxtractor

Submission date: 08 September 2025

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Computational Statistics and Machine Learning / Data Science and Machine Learning / Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author. The code used in this project can be found at `https://github.com/ra-lab-coder/Optimizing-VisDoM-`

Abstract

This dissertation addresses computational efficiency limitations in the VisDoM [48] multimodal retrieval-augmented generation system for document question-answering tasks. Specifically, two optimization strategies were developed and evaluated: V1, implementing parallelization and batch processing with improved metadata recovery algorithm, and V2, adopting page-level chunking to eliminate metadata recovery processes entirely.

Through evaluation across VisDoMBench [48] datasets using the Gemini-1.5-Flash [49] LLM backbone, the results demonstrate substantial runtime improvements. Specifically, both strategies significantly reduced processing times, with V1 achieving a 97-99% reduction in textual indexing time through parallelization and V2 showing even greater efficiency gains via architectural simplification.

Furthermore, the research reveals that optimization effectiveness depends critically on document characteristics. In particular, page-level chunking proves superior for visually-oriented datasets or datasets with sparse text like SlideVQA, while document-level chunking benefits text-dense materials like SPIQA, requiring cross-page context preservation. Moreover, LLM backbones evaluation using Word Overlap F1 scores shows that Llama3-70b [29, 30] consistently outperforms Gemini-1.5-Flash [49] across both V1 and V2, with F1 scores improving from 3.6-26.89% to 5.64-53.73%.

Additionally, we conducted error analysis on the SlideVQA dataset using V2 configuration across the three LLM backbones (Gemini-1.5-Flash [49], Llama3-70b [29, 30], Pixtral-Large [31] [32]) to supplement F1 score evaluation and identify specific error types in those LLM responses.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the foundational framework for optimizing document processing within the VisDoMRAG [48] multimodal RAG pipeline. We begin by providing an overview of VisDoMRAG's architecture and its extension to support multiple LLM backbones, followed by a comparison of document-level versus page-level chunking approaches. The chapter then establishes our research motivations and objectives, outlines the experimental methodology, summarizes the key contributions, and concludes with the dissertation structure and roadmap.

1.1 Overview of VisDoM, LLM Integration, and Chunking Strategies

VisDoMRAG [48] is a multimodal retrieval-augmented generation pipeline that combines visual and textual retrieval to answer document-based queries. Operationally, it extracts text and images from PDFs, retrieves the most relevant documents from each modality, and conditions a multimodal LLM on this context to produce answers. In addition to the original implementation, we integrate two more LLM backbones to investigate whether alternative language models can improve response quality when paired with our optimized retrieval strategies.

The original VisDoMRAG [48] implementation suffers from three critical bottlenecks: sequential processing, inefficient metadata recovery, and limited integration with only three LLM backbones. Specifically, the inefficient metadata recovery stems from the document-level chunking strategy, which concatenates all pages before segmentation, discarding page boundaries in the process. This means the system loses track of which chunk belongs to which original page. Without this information, the system cannot provide the precise document traceability required by users in regulated industries for citations or compliance. To recover this lost metadata, the system performs exhaustive string matching between every chunk and every page that significantly increases indexing time. We address these limitations through two complementary optimizations. First, we introduce batch parallel processing throughout the pipeline, replacing sequential operations with concurrent execution, and optimize the metadata lookup in document chunking through efficient two-stage similarity matching. Second, we implement page-level chunking as an alternative strategy that preserves metadata by construction, eliminating the metadata recovery process entirely. This allows us to compare the performance trade-offs between document-level chunking with optimized metadata recovery versus page-level chunking with inherent metadata preservation.

1.2 Research Motivations

This dissertation addresses critical performance bottlenecks in multimodal retrieval-augmented generation pipelines for PDF question answering. Our work is motivated by three key observations:

- **Sequential Processing Bottlenecks.** The baseline VisDoMRAG [48] implementation processes all operations sequentially, like document caching, index construction, and query processing occur one at a time without parallelization. This sequential architecture results in hours of computation, particularly when processing large document collections with hundreds of queries.

- **Inefficient Metadata Recovery.** Document-level chunking concatenates all PDF pages before segmentation, discarding page boundaries. The system then performs string comparisons between chunks and pages to recover metadata (i.e., the original pages that each chunks belongs to), which is a computationally expensive operation.

- **Limited LLM Extensibility.** The VisDoM [48] pipeline integrates with only three specific LLMs. With numerous new LLMs being developed and released regularly, this limitation prevents comprehensive investigation of how different model architectures perform within the VisDoMRAG pipeline. Expanding LLM backend support is therefore essential for conducting thorough performance research and identifying the most effective model choices for VisDoMRAG.

1.3 Research Objectives

This dissertation systematically optimizes the VisDoMRAG [48] pipeline through architectural improvements and empirical evaluation. Our specific objectives are:

1. **Parallelize Pipeline Operations.** Implement batch processing and concurrent execution across document caching, index construction, and query processing. And measure runtime improvements at each pipeline stage while ensuring identical output quality.

2. **Compare Chunking Strategies.** Evaluate document-level chunking with optimized metadata recovery against page-level chunking with inherent metadata preservation. And assess the trade-offs in processing time, and answer quality using Word-Overlap F1 scores[41].

3. **Benchmark Advanced LLM Backbones.** Assess whether incorporating more advanced LLM backbones beyond the original VisDoMRAG [48] implementation can improve end-to-end answer quality performance across different datasets and optimization configurations.

4. **Supplement Quantitative Evaluation with Error Analysis.** Perform error analysis on representative experimental configurations to gain deeper insights into system performance beyond aggregate metrics and understand failure patterns across different LLM backbones.

5. **Provide Implementation Guidelines.** Synthesize findings into practical recommendations for deploying multimodal RAG systems, including optimal chunking approaches, best-performimg LLMs for different use cases.

1.4 Research Experiments

To systematically optimize the VisDoMRAG pipeline, this dissertation conducts three experiments that progressively introduce performance improvements while monitoring their impact on answer quality:

1. **Experiment 1: Baseline Performance Profiling.** We establish reference metric using the original VisDoMRAG [48] implementation on the complete VisDoMBench dataset. This baseline employs:

   - Sequential processing for all operations
   - Document-level chunking (3000 characters, 300 overlap) with exhaustive metadata recovery
   - Gemini-1.5-Flash [49] as the only LLM backbone.

   We measure component-level runtime (building visual and textual indices, document caching, and query processing) and Word-Overlap F1 scores [41] for experiments running exclusively on Gemini-1.5-Flash [49].

2. **Experiment 2 (V1): Comprehensive Parallelization with Optimized Metadata Recovery.** We optimize the baseline through systematic parallelization and the integration of two more LLM backbones:

   - Batch processing for visual and textual index construction
   - Parallel document caching with ProcessPoolExecutor
   - Concurrent query processing via ThreadPoolExecutor
   - Optimized metadata recovery using two-stage similarity matching (Word-Overlap filtering at 0.3 threshold)
   - LLM backbones: Gemini-1.5-Flash [49], Llama3-70b [29, 30], and Pixtral-Large [31] [32].

   For experiments using Gemini-1.5-Flash [49], we measure runtime reduction at each pipeline stage and verify answer quality preservation by comparing Word-Overlap F1 [41] scores against the baseline. Then we compare the Word-Overlap F1 scores across all the three LLM backbones.

3. **Experiment 3 (V2): Page-Level Chunking Strategy.** We replace document-level chunking with page-level segmentation while maintaining V1's other optimizations:

    - Page-level chunking that preserves metadata by construction
    - Elimination of metadata recovery computation
    - Sequential query processing for text index construction
    - Batched Chroma insertion (5000-chunk batches) to prevent memory overflow
    - LLM backbones: Gemini-1.5-Flash [49], Llama3-70b [29, 30], and Pixtral-Large [31] [32]

    This experiment isolates the impact of chunking strategy on both runtime and final answer quality, revealing trade-offs between metadata preservation efficiency and cross-page context availability. We use the similar evaluation process as V1.

Each experiment processes the entire VisDoMBench [48] dataset, enabling direct comparison of runtime improvements and answer quality across different optimization strategies. The progression from baseline through V1 to V2 systematically identifies which optimizations provide the greatest performance gains.

1.5 Contributions to Science

This dissertation makes four main contributions to multimodal retrieval-augmented generation systems:

- **Systematic Pipeline Parallelization.** We transform the sequential VisDoMRAG pipeline into a parallel architecture through batch processing, concurrent execution, and index persistence, achieving significant speedup without sacrificing answer quality.
- **Efficient Metadata Recovery Optimization.** We develop a two-stage similarity matching algorithm that reduces metadata recovery complexity from exhaustive comparisons to filtered matching with 0.3 Word-Overlap threshold, eliminating unnecessary sequence matching operations while maintaining metadata accuracy.
- **Comparative Analysis of Chunking Strategies.** We empirically evaluate document-level versus page-level chunking across VisDoMBench, revealing that page-level chunking eliminates metadata recovery overhead entirely while achieving comparable Word-Overlap F1 scores[41]. This finding challenges the assumption that document-level context is necessary for effective retrieval.
- **Generalizable Optimization Framework.** We validate our optimizations in three LLM back-ends and five datasets, demonstrating that performance improvements are generalizable in models and document types. All code and benchmarks are released to support reproducible research.

1.6 Thesis Structure

This thesis is organized into six chapters:

- **Chapter 1 – Introduction.** Presents the research motivation, objectives, three-experiment design, and main contributions, establishing the need for optimized multimodal RAG systems.
- **Chapter 2 – Background.** Reviews the LLM revolution, retrieval-augmented generation, multimodal architectures, the VisDoMRAG pipeline, and the VisDoMBench benchmark comprising five datasets (SPIQA, SlideVQA, SciGraphQA, PaperTab, FetaTab).
- **Chapter 3 – Approach.** Details the baseline implementation, two optimization variants (V1: comprehensive parallelization with optimized metadata recovery, V2: page-level chunking), and evaluation metrics. Describes the systematic progression from sequential to parallel architecture.
- **Chapter 4 – Evaluation.** Reports runtime and Word-Overlap F1 scores [41] for all experimental configurations. For consistency, runtime comparisons use Gemini-1.5-Flash [49] across baseline, V1, and V2 pipelines. In contrast, F1 scores are evaluated across all LLM backbones (Gemini-1.5-Flash, Llama3-70b, and Pixtral-Large). Additionally, the chapter includes error analysis of LLM responses on selected configurations that have low F1 scores to examine the nature and frequency of different error types across LLM backbones.
- **Chapter 5 – Related Work.** This chapter discusses other existing multimodal RAG frameworks. Specifically, we highlight the differences between these frameworks and VisDoM's dual-pipeline architecture, and investigate how insights from related work could enhance the VisDoM system's performance and capabilities.
- **Chapter 6 – Conclusion.** Summarizes key findings on parallelization and chunking optimizations, analyzes trade-offs between efficiency and quality, provides deployment recommendations, discusses limitations, and proposes future extensions.

Chapter 2

Background

This chapter provides the theoretical and technical foundation for optimizing multimodal retrieval-augmented generation systems. We begin by tracing the evolution of large language models from recurrent architectures to modern transformers, discussing how their impressive capabilities and inherent limitations motivated the development of retrieval-augmented approaches. Next, we explore how retrieval-augmented generation emerged as a solution to the hallucination, static knowledge, and attribution problems inherent in purely parametric models. We then examine the extension from single-modality text systems to multimodal architectures that process both textual and visual information, with particular focus on the VisDoMRAG [48] pipeline architecture. We also introduce the VisDoMBench benchmark [48] and its constituent datasets. This background establishes the technical landscape within which our pipeline optimizations operate and highlights the performance gaps our work addresses.

2.1 Evolution of Large Language Models

The development of large language models represents a fundamental shift in natural language processing, moving from a field where each task, such as translation, question-answering, and summarization, required its own specialized model to one where a single pre-trained model could handle all these tasks. This evolution revealed both impressive capabilities and fundamental limitations in purely parametric models, leading researchers to develop retrieval-augmented approaches.

2.1.1 From Recurrent Networks to Transformers

Prior to 2017, sequence modeling relied primarily on recurrent architectures. For example, Long Short-Term Memory networks (LSTMs)[16] and Gated Recurrent Units (GRUs)[9] processed text sequentially, maintaining hidden state that captured context from previous tokens. While effective for many tasks, these architectures suffered from two critical limitations: the sequential nature prevented parallelization during training, and the fixed-size hidden state forced the model to compress all precious information into a single vector, causing important details to be lost in long sequences. Subsequently, [51] introduced the Transformer architecture, which processes all words in a sequence simultaneously rather than one by one like previous models. Specifically, Transformers use 'self-attention' to directly compare every word with every other word in the input, identifying relevant connections regardless of distance. Moreover, the model runs multiple attention operations in parallel (called 'multi-head attention'), with each 'head' learning to track different types of relationships: one might focus on grammatical dependencies while another tracks semantic similarities. Because Transformers process all positions at once rather than sequentially, they train much faster than RNNs while better capturing long-distance relationships in text.

2.1.2 The GPT Paradigm and Scale

Building on the Transformer's decoder architecture, [39] introduced GPT (Generative Pre-trained Transformer), demonstrating that unsupervised pre-training on large text corpora followed by task-specific fine-tuning achieved strong performance across diverse NLP tasks. This pre-training paradigm shifted the field from training task-specific models from scratch to adapting general-purpose language models. Encouraged by these results, researchers began investigating whether the benefits would continue with larger models. Hence, GPT-2 [40] scaled this approach to 1.5 billion parameters, revealing emergent capabilities in zero-shot task transfer. Specifically, The model could perform tasks like translation and summarization without explicit training, simply by conditioning on appropriate prompts. This led researchers to wonder whether simply increasing model size, rather than improving algorithms, could be the key to better performance. This scaling hypothesis is then validated by [6] with GPT-3, scaling to 175 billion parameters trained on 300 billion tokens. GPT-3 demonstrated that sufficient scale enables in-context learning, where models adapt to new tasks using only a few examples in the prompt, without updating parameters. This few-shot learning capability suggested that during pre-training, large models don't just memorize facts but learn general patterns for how to approach new tasks. To understand these scaling phenomena more systematically, [20] formalized these observations as scaling laws, showing predictable relationships between model size, dataset size, compute budget, and performance.

2.1.3 Training Paradigms and Capabilities

Having traced the development from Transformers to GPT-3 and the emergence of scaling laws, we now examine the broader training paradigms that emerged from these breakthroughs and the capabilities they enable. Modern LLMs employ self-supervised pre-training on massive text corpora, where the training objective is to predict next tokens given preceding context. During the pre-training process, the models implicitly encode vast amounts of world knowledge into their parameters. This training approach produces models that capture syntax, semantics, factual knowledge, and reasoning patterns. Subsequently, these models demonstrate remarkable capabilities in diverse areas: generating coherent long-form text, answering questions, translating between languages, and even performing arithmetic and logical reasoning. For example, [36] demonstrated that language models can function as knowledge bases, retrieving facts by completing prompts like "The capital of France is [MASK]." However, this parametric knowledge storage introduces fundamental limitations that no amount of scaling has resolved.

2.1.4 Fundamental Limitations

Despite their impressive capabilities, LLMs suffer from several critical limitations that impact their reliability for knowledge-intensive tasks. First, they exhibit hallucination, generating plausible but factually incorrect information with high confidence. [43] demonstrated this systematically, showing that even large models frequently produce incorrect facts, particularly for less common knowledge. This hallucination problem is inherent to the autoregressive generation process, where the model must produce something plausible even when uncertain.

Second, LLMs cannot update their knowledge after training. Information encoded in parameters remains frozen at the training cutoff date. The practical impact of this limitation is severe, as [22] demonstrated significant performance degradation on facts that changed after training, highlighting the temporal brittleness of parametric knowledge. Retraining models to add new information requires millions in compute costs and risks overwriting existing knowledge.

Third, LLMs provide no mechanism for source attribution or verification. When generating factual claims, models cannot indicate which training documents support their statements, and research by [42] showed that models struggle to provide reliable citations even when explicitly prompted, often fabricating plausible but non-existent sources. This lack of provenance is particularly problematic in domains requiring accountability, such as medical or legal applications. These limitations(hallucination, static knowledge, and absent attribution) motivated the development of retrieval-augmented generation. By augmenting language models with external retrieval mechanisms, RAG systems address each limitation: retrieved documents provide factual grounding to reduce hallucination, retrieval corpora can be updated without retraining, and retrieved passages offer explicit provenance for generated claims. This paradigm shift from purely parametric to semi-parametric architectures represents the next evolution in language model development.

2.2 Retrieval-Augmented Generation(RAG)

2.2.1 RAG as a Solution to LLM Limitations

Retrieval-augmented generation emerged as a direct response to the fundamental limitations of purely parametric language models. Specifically, [23] introduced RAG by combining a pre-trained seq2seq[51] model with a pre-trained neural retriever, creating a semi-parametric architecture that addresses each core limitation of LLMs. The neural retriever is a learnable search component that uses neural networks to find relevant documents. This RAG architecture directly tackles the three fundamental problems identified earlier. In particular, for hallucination, RAG grounds generation in retrieved documents, constraining the model to information present in its corpus. And for static knowledge, the retrieval corpus can be updated instantly without model retraining. Lastly, for attribution, retrieved passages provide explicit sources for generated claims.

Having established RAG's conceptual advantages, we now examine its technical implementation. As detailed in [23], RAG models leverage two components: (i) a retriever $p_\eta(z|x)$ with parameters η that returns top-K distributions over text passages z given a query x, and (ii) a generator $p_\theta(y_i|x, z, y_{1:i-1})$ parametrized by θ that generates tokens y_i based on the context of previous tokens, the original input x, and retrieved passages z. For end-to-end training of both retriever and generator, the retrieved document is treated as an unobserved latent variable. Then

[23] proposed two variants that produce text generation probabilities ($p(y|x)$) by marginalizing over latent documents using different approaches: first, RAG-Sequence, which uses the same retrieved documents to generate the entire output sequence, and RAG-Token, which can retrieve different documents for each generated token.

2.2.2 Dense Retrieval Methods

The effectiveness of RAG systems depends critically on retrieval quality. However, traditional sparse methods like BM25[44] rely on lexical matching, missing semantic relationships between queries and relevant documents.

To address the limitation, [21] proposed retrieval with Dense Passage Retrieval (DPR), using separate BERT[11] encoders for queries and passages that project text into a shared dense embedding space where cosine similarity captures semantic relevance.

To train DPR, the method uses contrastive learning with carefully selected hard negatives. Specifically, the contrastive loss minimizes the distance between question-answer pairs while maximizing the distance between questions and negative examples. Moreover, the quality of these negative examples is crucial for effective training. In particular, using BM25-retrieved[44] passages as hard negatives proves effective because, while BM25 has lower overall performance, its retrieved passages are challenging examples that help the model learn to make fine-grained semantic distinctions, resulting in better performance than using random negatives.

Once trained, DPR provides substantial efficiency gains during inference. More precisely, the resulting dense representations enable efficient maximum inner product search (finding the most similar documents by computing dot products between query and document vectors) through approximate nearest neighbor methods, reducing latency from seconds to milliseconds.

Subsequent work refined dense retrieval training. First, [57] introduced ANCE, which periodically refreshes the negative passage pool during training to provide increasingly challenging examples. Additionally, [37] proposed RocketQA, which improves training efficiency by using passages from other questions in the batch as additional negative examples. Furthermore, they also use knowledge distillation. In this approach, a slower but more accurate cross-encoder first generates relevance scores, and these scores are then used to train the faster bi-encoder to learn similar ranking behavior. Overall, these improvements pushed dense retrieval performance significantly beyond the sparse baselines while maintaining computational efficiency.

2.2.3 Architectural Variants

The first major architectural variant addresses RAG's memory constraints. In particular, while the original RAG concatenated retrieval passages with queries, this approach faces memory constraints with multiple documents. To overcome this limitation, [17] addressed this with Fusion-in-Decoder (FiD), which encodes each retrieved passage independently with the query, then combines representations in the decoder. As a result, this design scales to 100 retrieved passages without quadratic memory growth, achieving substantial improvements on knowledge-intensive tasks.

While FiD addressed how to fuse multiple documents, another architectural variant focuses on retrieval frequency. Specifically, [5] introduced RETRO, which retrieves information at a finer granularity: every 64 tokens rather than once per document. To eliminate retrieval latency during training, RETRO pre-computes the nearest neighbors for every training chunk and stores them before training begins. At inference, the model retrieves relevant chunks for each generated chunk, allowing dynamic adaptation to changing context. Most notably, they demonstrated 'RETRO-fitting', which adds retrieval capabilities to existing pre-trained models by training only additional retrieval layers while keeping original weights frozen, avoiding expensive retraining from scratch.

Beyond these single-retrieval and continuous-retrieval variants, iterative retrieval approaches further enhanced RAG capabilities. For example, [45] proposed Iter-RetGen, alternating between generation and retrieval steps. In this approach, the model generates an initial response, identifies knowledge gaps, formulates new queries, and retrieves additional information. Subsequently, this process continues until sufficient information is gathered, particularly benefiting complex questions requiring information synthesis.

Furthermore, recent work explored adaptive retrieval. [19] introduced FLARE, which triggers retrieval only when generation probability falls below a threshold, indicating uncertainty. This selective approach reduces average latency by avoiding unnecessary retrieval for questions answerable from parametric knowledge. Building on this insight, [28] showed that retrieval benefits correlate inversely with entity popularity: common facts need no retrieval while rare information benefits significantly.

2.2.4 Limitations of Text-Only RAG

Despite these advances, current RAG systems face a fundamental limitation: they process only textual information. Meanwhiel, real-world document, particularly technical and academic materials, convey critical information through multiple modalities. Specifically, PDFs contain figures that illustrate complex processes, tables that summarize data, diagrams that show relationships, and equations that formalize concepts. As a result, text-only RAG systems must either ignore these visual elements entirely or rely on inadequate automatic descriptions, losing essential information. For example, consider a scientific paper about neural network architectures. The text might describe a model abstractly, but the architecture diagram provides immediate understanding of layer connections, data flow, and component relationships. Similarly, experimental results presented in tables and graphs contain precise numerical information that textual descriptions only approximate. Consequently, when answering questions about such documents, text-only RAG systems miss crucial evidence, leading to incomplete or incorrect responses. Moreover, this limitation becomes particularly acute in domains like scientific literature, technical documentation, financial reports, and educational materials, where visual elements are not supplementary but integral to understanding.

These observations motivated the development of multimodal RAG systems that can retrieve and reason over both textual and visual information. By extending retrieval beyond text to include images, tables, and figures, multimodal RAG promises more complete document understanding and more accurate question answering.

The following section examines how multimodal RAG architectures address these limitations by incorporating visual understanding capabilities into the retrieval-augmented generation paradigm.

2.3 Multimodal Retrieval-Augmented Generation (MRAG)

2.3.1 Emergence of Multimodal RAG Systems

The extension from text-only to multimodal RAG required fundamental architectural changes to handle diverse data types. Early attempts at multimodal document understanding focused on converting visual elements to text through OCR (Optical Character Recognition) and image captioning, then processing everything through text-only pipelines. For instance, [58] introduced LayoutLM, which uses OCR-extracted words and their bounding boxes and models layout with 2-D positional embeddings in a BERT-based architecture. Subsequently, [59] (LayoutLMv2) further integrates visual features and spatial-aware attention. Similarly, systems like TAPAS [15] converted tables to linearized text representations for question answering. However, these text-serialization approaches lose crucial information. Specifically, spatial relationships between elements disappear when a two-dimensional layout becomes a one-dimensional sequence. Moreover, visual semantics embedded in charts and diagrams cannot be captured through automatic captions. Additionally, fine-grained details in figures, such as data point distributions or architectural connections, are lost when reduced to textual descriptions.

To address these limitations, true multimodal RAG systems emerged with the development of vision-language models that jointly encode textual and visual information. Vision-language models are neural networks trained to understand both images and text simultaneously, rather than processing each modality separately. To create these models, researchers combine pre-trained vision encoders (which process images) with language models (which handle text) and train them together on large datasets of image-text pairs. Through this joint training process, they learn to align visual features with textual descriptions, enabling them to reason about relationships between what they see and what they read. In contrast to OCR-based approaches that convert images to text, vision-language models can directly interpret visual elements like charts, diagrams, and spatial layouts while maintaining their visual semantics.

For example, [24] proposed BLIP-2, which aligns visual and textual representations through a lightweight Querying Transformer, enabling unified multimodal understanding. Similarly, [26] introduced LLaVA (Large Language and Vision Assistant), which connects visual encoders to language models, enabling the system to understand and respond to instructions about visual content. For document-specific applications, [61] developed mPLUG-DocOwl, specializing in document understanding with OCR-free page comprehension. Their experiments on document VQA datasets showed substantial improvements, particularly for questions requiring diagram interpretation or data extraction from figures.

2.3.2 Architectural Approaches to Multimodal Retrieval

Multimodal RAG systems employ various strategies to combine visual and textual evidence. For instance, MuRAG [7] constructs a non-parametric multimodal memory comprising images, text, and image-text pairs. It uses a unified encoder for indexing and retrieval, and then conditions a generator on the retrieved items. Similarly, CLIP-based retrieval [38] maps images and text into a shared embedding space via contrastive learning, enabling cross-modal search. Specifically, contrastive learning trains the model by showing it millions of matching image-text pairs (like a photo of a dog with the caption "dog") and teaching it to map these pairs close together in the embedding space, while pushing apart non-matching pairs (like the dog photo with "car"). This creates a shared space where semantically similar images and text, regardless of modality, are located near each other, enabling users to search for images using text queries or find text using images.

However, these approaches may fail when documents require understanding how text and visuals are spatially connected on a page, since they process text and images as separate, unconnected elements rather than understanding their spatial relationships.

Recent work addresses this limitation through document-level multimodal encoding. [12] introduced ColPali, which processes entire PDF pages as images through a vision transformer, preserving layout and visual elements while maintaining text readability. This approach eliminates the need for separate OCR and layout analysis, instead learning to directly match queries to relevant document regions.

2.3.3 Challenges in Multimodal RAG Implementation

Multimodal RAG systems face unique challenges beyond those of text-only systems. First, preprocessing diverse documents requires multiple extraction pipelines: OCR for text in images, table detection for structured data, figure extraction for diagrams, each introducing potential errors.

Second, aligning representations across modalities remains difficult, as visual and textual encoders often produce embeddings in different semantic spaces that resist direct comparison.

Moreover, the computational overhead multiplies with multimodal processing. In particular, visual encoding through models like CLIP[38] or ColPali[12] requires substantial GPU resources, while maintaining separate indices for each modality increases memory requirements. Furthermore, query processing becomes more complex when determining which modalities to search and how to weight their contributions. For example, should a question about "experimental results" prioritize tables, figures, or textual descriptions? Ultimately, these decisions significantly impact retrieval quality.

Additionally, evaluation of multimodal RAG systems lacks standardized benchmarks. While text-only RAG can use established QA datasets, multimodal evaluation requires questions specifically designed to test cross-modal understanding. To address this need, datasets need questions that test different capabilities: visual-only questions (answerable from charts or diagrams alone), integrated questions (requiring both text and visual elements), and modality selection questions (where the system must choose whether to use text, visuals, or both).

2.3.4 VisDoMRAG: A Comprehensive Multimodal RAG System

Among recent multimodal RAG implementations, VisDoMRAG[48] represents a comprehensive approach to document-grounded visual question answering. Specifically, the system implements dual retrieval pathways: one for textual content and another for visual elements, operating on the same PDF documents.

The architecture leverages specialized models for each modality: ColQwen2(`vidore/colqwen2-v0.1`)[60] or ColPali[12] for visual retrieval and dense text embedders like *BGE 1.5*[56] for textual retrieval. Then, retrieved content from both pathways is fed into a multimodal language model that synthesizes information across modalities. This design achieves strong performance on benchmarks requiring understanding of figures, tables, and diagrams alongside text.

However, VisDoMRAG [48] inherits and amplifies the efficiency challenges of text-only RAG. Specifically, by checking its codebase, the system processes all operations sequentially(document caching, visual encoding, text extraction, retrieval, and generation) without parallelization. Moreover, the dual-pathway architecture doubles the retrieval overhead compared to text-only systems. These computational bottlenecks make the baseline implementation impractical for production deployment, motivating our optimization work. The following section examines VisDoMRAG's architecture in detail, establishing the baseline system that our experiments optimize.

2.4 VisDoM

2.4.1 VisDoM in the Context of Multimodal RAG Evolution

While previous section outlined the general evolution of multimodal RAG systems, from early text-serialization approaches through unified vision-language models, VisDoM [48] represents a specific instantiation that addresses many of the identified challenges while introducing new architectural patterns. As noted, systems like ColPali[12] and mPLUG-DocOwl[61] demonstrated the potential of processing documents as unified multimodal entities, VisDoM implements parallel modality-specific pipelines with sophisticated fusion mechanisms, offering both empirical validation and revealing new optimization opportunities.

2.4.2 VisDoMBench: Addressing the Evaluation Gap

The lack of standardized multimodal benchmarks, identified as a critical challenges in Section 2.3.3, motivated the creation of VisDoMBench. As shown in Table 2.1, existing benchmarks either focus on text-only content (L-Eval[1], LongBench[3], Marathon[62]) or lack multi-document capabilities (MPDocVQA[50], MMLONGBENCH-DOC[27]), while VisDoMBench uniquely combines multimodal content with genuine multi-document requirements. This benchmark specifically targets the multi-document setting where questions require evidence from an average of 8.4 documents and 129 pages (Table 2.2), far exceeding the scope of existing document VQA datasets[**?**]. By incorporating five diverse sources (PaperTab, FetaTab, SciGraphQA, SPIQA and SlideVQA), the benchmark ensures coverage of tables, charts and slides, testing the full range of multimodal understanding capabilities.

Table 2.1: Comparison of long-context document QA benchmarks. *Adapted from* VisDoM [48].

Benchmark	Content Type	Multi-Doc	Domain
L-Eval	Text	✗	Multi-domain
LongBench	Text	✓	Wikipedia
Marathon	Text	✗	Multi-domain
MPDocVQA	Text, Tables, Charts	✗	Multi-domain
MMLONGBENCH-DOC	Text, Tables, Charts, Slides	✗	Multi-domain
VisDoMBench	**Text, Tables, Charts, Slides**	✓	**Multi-domain**

Table 2.2: Summary of data splits included in VisDoMBench. 'Combined' means VisDoMBench combines five datasets across three domains: Wikipedia pages (PaperTab), Scientific Papers (FetaTab, SciGraphQA, SPIQA), and Presentation Decks (SlideVQA). *Adapted from* VisDoM [48].

Dataset	Domain	Content Type	Queries	Docs	Avg.D/Q	Avg.Pages/Q
PaperTab	Wikipedia	Tables, Text	377	297	10.8	113
FetaTab	Scientific Papers	Tables	350	300	7.8	124
SciGraphQA	Scientific Papers	Charts	407	319	5.9	130
SPIQA	Scientific Papers	Tables, Charts	586	117	9.5	136
SlideVQA	Presentation Decks	Slides	551	244	7.0	140
VisDoMBench	Combined	Tables, Charts, Slides, Text	2271	1277	8.4	129

The benchmark uses Word-Overlap F1 [41] as its primary evaluation metric (defined in Section 4.1). While this metric creates challenges for systems that generate semantically correct but lexically different answers, it provides a consistent evaluation framework that enables direct comparison across systems. Furthermore, this choice reflects real-world information extraction requirements where precise, literal answers are often necessary.

2.4.3 VisDoMRAG: Parallel Processing with Late Fusion

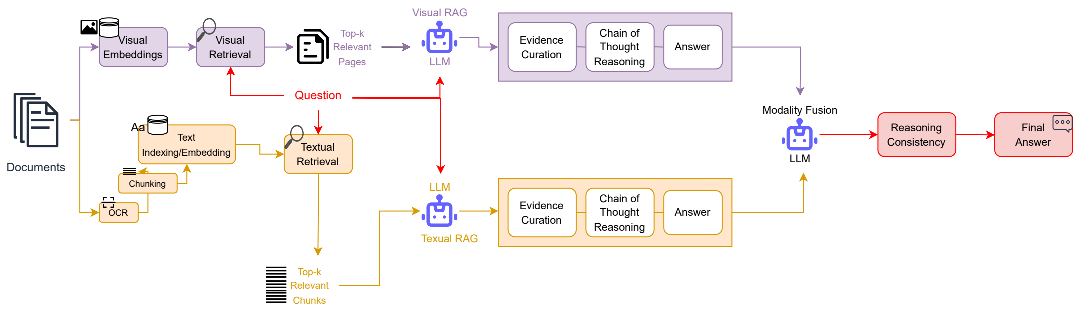


Figure 2.1: VisDoMRAG's dual-pipeline architecture with late fusion (adapted from [48])

As demonstrated in Figure 2.1, the first part of this architecture is the index construction stage. VisDoM [48] evaluateded diverse retrievers for this stage, as illustrated in Table 2.3. For textual retrieval, they tested BM25[44], MiniLM[53] , MPNet[47], and *BGE 1.5*[56], finding that while *BGE 1.5*[56] performed best overall, BM25[56] notably outperformed dense methods on SlideVQA due to the sparse, keyword-heavy nature of slide text. For visual retrieval, it used ColPali and ColQwen2 `vidore/colqwen2-v0.1`)[60], which are multi-vector retrieval models built on LLMs [12]. Specifically, their base LLMs are PaliGemma [4] and Qwen2 [60] respectively. In terms of performance, ColQwen2 `vidore/colqwen2-v0.1`)[60] outperformed ColPali across most datasets.

Following Figure 2.1, after the visual and textual retrieved content is fed into the LLMs separately, both pipelines follow identical three-stage prompting: (1) Evidence Curation, which extracts and structures only the relevant passages, tables, and figure details from retrieved documents. (2) Chain-of-Thought, which links the curated pieces into a concise step-by-step rationale

Table 2.3: Retriever performance on source-document identification at $k = 5$. Values are the percentage of queries for which the ground-truth document is identified: a query counts as correct if $\geq \lceil k/2 \rceil$ of the top-k retrieved items (pages/chunks) come from that ground-truth document. (Average is across datasets.) *Adapted from* VisDoM [48].

Retriever	PaperTab	FetaTab	SciGraphQA	SPIQA	SlideVQA	Average
BM25	65.51	84.00	72.73	88.23	**98.55**	81.80
MiniLM	65.51	88.85	91.65	61.06	0.73	61.56
MPNet	90.18	89.71	91.40	95.84	0.73	73.57
BGE1.5	96.81	94.00	90.91	**98.43**	81.85	92.40
ColPali	96.93	**97.71**	95.28	93.17	97.64	96.15
ColQwen2	**97.61**	96.86	**95.58**	96.85	97.82	**96.94**

across documents (CoT;[54]). (3) Answer generation, which produces a precise, well-justified answer conditioned on the rationale, using prompts to match the required output format. This parallel structure allows each modality to leverage its specialized strengths without compromise.

The end of this architecture (2.1) is the fusion stage, which is the key innovation of the system. Specifically, instead of the early fusion approaches common in systems like MuRAG[7], VisDoM-RAG delays integration until after each pipeline has independently reasoned over its retrieved evidence. An LLM then examines both reasoning traces, identifies consistencies and conflicts, and synthesizes the final answer. Importantly, this approach sidesteps the representation alignment problem by operating at the reasoning level rather than the embedding level.

2.4.4 Empirical Validation and Performance Patterns

VisDoMRAG's experiments validate several architectural hypotheses while revealing modality-specific strengths. The system achieves $12 - 20\%$ improvements over long-context baselines across multiple LLM backbones (Qwen2-VL[52] [2], Gemini-1.5-Flash[49], GPT-4o[34]), demonstrating that structured retrieval outperforms brute-force context processing even with modern long-context models. Additionally, ablation studies provide crucial insights:

- Late fusion substantially outperforms early fusion (50.01 vs 43.63 Word-Overlap F1[41] with GPT-4o[34]), validating the hypothesis that modality-specific reasoning should precede integration.

- The three-stage prompting structure proves essential, with each component (Evidence Curation, CoT, Reasoning Consistency) contributing 3-5 Word Overlap F1 points[41] with GPT-4o[34]).

2.4.5 Computational Challenges and Optimization Opportunities

VisDoMRAG exemplifies and amplifies the efficiency challenges identified in Section 2.3.3. The system requires three LLM calls per query, one for each modality pipeline plus fusion, tripling the inference cost of single-pipeline approaches. The dual retrieval pathways double the indexing and search overhead. Moreover, as revealed through codebase analysis, the implementation processes all operations sequentially without parallelization and rebuilds textual indices for each query session, resulting in prohibitively long execution times that render the system impractical for real-world deployment.

Beyond computational efficiency, there are additional optimization opportunities. First, VisDoM was evaluated with only three LLMs (Qwen2-VL[2, 52], Gemini-1.5-Flash[49], GPT-4o[34]), missing the potential benefits of more recent and advanced models such as llama_meta-llama-3-70b-instruct [29, 30] and Mistral/Pixtral-Large [31] [32].

Chapter 3

Approach

We aim to reduce VisDoMRAG's [48] document processing and retrieval latency without degrading answer quality, also integrate this pipeline with more LLMs. More specifically, by running the original code, we identified five bottlenecks: slow per-chunk metadata lookups during textual index construction, slow ingestion when OCR is required (for example, SlideVQA), high retrieval latency during generation and limited LLMs integration. To address these problems, we created two variants of the baseline and compared the baseline with these two variants. For instance, Variant 1 (V1) uses document-level chunking with optimized metadata lookup, persists and reuses the text index, applies parallel and batched ingestion with caching, and uses optimized textual and visual retrieval. Additionally, Variant 2 (V2) switches to page-level chunking to remove metadata lookups and builds a page-level text index. It also persists and reuses the text index, applies the same parallel and batched ingestion with caching, and uses the same retrieval. Datasets, hardware, retrievers and OCR are held fixed. The primary outcomes are wall-clock time for text and visual indexing, caching, and retrieval. The secondary outcome is token-level word overlap F1 for answer quality.

3.1 Experimental Setup

We do comparative evaluation with one baseline and two composite variants. All comparisons are within the same documents and questions. Our implementation is from the open source VisDoM-RAG codebase[48]. Core components include the OCR engine [46], the chunker, text and visual embedding models, vector database, the retriever, and the generation backbones. Experiments ran on Intel Xeon CPU @ 2.20 GHz (8 cores) with 31 GiB RAM and an NVIDIA L4 GPU (23.0 GiB VRAM) on Ubuntu 24.04.3 LTS (Noble Numbat) with a 400 GB NVMe SSD. Python 3.11.13.

For retrieval, we use ColQwen2 [60] as the visual retriever and *BGE 1.5* [56] as the textual retriever with `top_k` $k = 5$ across all the experiments.

For generation backbones, we use three LLM backbones for answer generation: Gemini-1.5-Flash [49],meta-llama/Meta-Llama-3-70B-Instruct (`llama_meta-llama-3-70b-instruct`) [29, 30], Mistral/Pixtral-Large ([31] [32]). Prompt templates are identical across backbones.

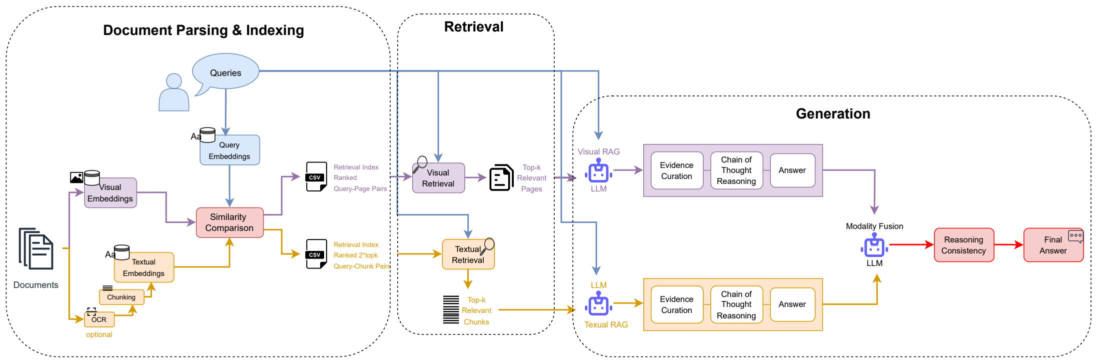


Figure 3.1: **Detailed VisDoMRAG Pipeline Architecture:** The system consists of three main phases: (1) **Document Parsing&Indexing**, where documents undergo visual and textual embedding extraction, with optional OCR [46] for pages lacking extractable text, following by similarity comparison to generate retrieval indices; (2) **Retrieval**, where queries are processed to retrieve top-k(k=5) relevant pages and chunks for both visual and textual modalities; and (3)**Generation**, involving separate generation of visual and textual responses followed by fusion to obtain the final answer. This detailed representation extends the simplified overview(Figure 2.1)with more comprehensive preprocessing steps.

3.1.1 Datasets

We use the full **VisDoMBench** benchmark[48] and include every item from each of its five constituent datasets. There are no official splits. All experiments in Sections 3.1.2, 3.1.3, and 3.1.4 use the exact same documents and questions.

The benchmark comprises five visual question-answering datasets requiring multi-page PDF comprehension: **SPIQA** (586 queries on scientific papers), **SlideVQA** (551 queries on presentation slides), **SciGraphQA** (407 queries on scientific papers), **PaperTab** (377 queries on Wikipedia tables and text), and **FetaTab** (350 queries on scientific papers). Each dataset provides PDF documents with associated questions requiring comprehension of both textual and visual elements.

3.1.2 Experiment 1: Baseline

Objective.

Establish reference timing and performance metric for the unmodified pipeline.

Configuration

The baseline employs document-level chunking with 3000-character segments and 300-character overlap, using the original `identify_document_and_page` function for metadata extraction through exhaustive sequence matching. All the processing operates sequentially without parallelization or batching. Document caching processes PDFs individually without parallel workers. For retrieval, we use retrievers stated in Section 3.1 Answer generation only employs Gemini-1.5-flash [49].

Implementation details

We adopt the orginal VisDoMRAG pipeline, the detailed pipeline is demonstrated in Figure3.1. The VisDoMRAG baseline implements a dual-pathway retrieval architecture consisting of independent visual and textual processing streams that are subsequently combined through an LLM-based fusion mechanism.

Document Parsing & Indexing

- **Visual Pathway:** The visual pathway employs ColPali or ColQwen2 models for generating visual embeddings of document pages. Since the original paper reported that the ColQwen2 visual retriever outperforms ColPali, so we choose ColQwen2 as the visual retriever for this baseline experiment. After we got the visual embeddings for each PDF page image , the implementation follows a two-stage approach: query embedding generation and multi-vector similarity scoring for ranking documents for each query.

  The query embedding generation process iterates through all queries in the dataset, transforming each question into a dense vector representation using the same ColQwen2 visual processor used for document encoding. During this process, each query undergoes preprocessing through the vision processor's query-specific pipeline, which adapts the text input for compatibility with the visual embedding space. The resulting query embeddings are computed using the ColQwen2 model, then stored in memory for immediate use and persisted to disk for future retrieval operations.

  At the second stage, the document ranking phase implements a sophisticated multi-vector scoring mechanism that addresses the challenge of matching queries against large document collections. Specifically, for each query, the system first determines the relevant document scope by extracting document identifiers from the dataset's metadata fields. For example, for the SPIQA datasets, we extract the relevant documents' identifiers from the the 'documents' column in spiqa.csv. Then we only do the multi-vector scoring between queries and the pages embeddings for their relevant documents. This relevance filtering serves to constrain the search space to documents explicitly associated with each query, improving both computational efficiency and retrieval precision. When document metadata parsing fails or no specific documents are indicated, the system defaults to searching across the entire document collection.

  Once the relevant document scope is established, the core ranking computation employs ColQwen2's multi-vector scoring function, which performs similarity calculations between the query embedding and the concatenated tensor of relevant document page embeddings. Then the ranking results undergo post-processing to generate a comprehensive retrieval index stored in CSV format, where each query and document page pair receives a similarity score. This CSV file enables subsequent top-k retrieval operations during the question answering phase.

- **Textual Pathway:** To enable efficient access to document content, we implemented a preprocessing step that caches the textual content of all PDF documents in the dataset. The system first extracts unique document identifiers, then the text content is extracted from each

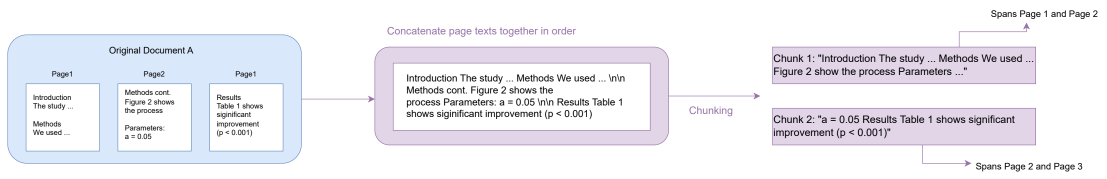

Original Document A
Page1
Introduction The study ... Methods We used ...
Page2
Methods cont. Figure 2 shows the process Parameters: a = 0.05
Page1
Results Table 1 shows siginificant improvement (p < 0.001)
Concatenate page texts together in order
Introduction The study ... Methods We used ... \n\n Methods cont. Figure 2 shows the process Parameters: a = 0.05 \n\n Results Table 1 shows siginificant improvement (p < 0.001)
Chunking
Chunk 1: "Introduction The study ... Methods We used ... Figure 2 show the process Parameters ..."
Spans Page 1 and Page 2
Chunk 2: "a = 0.05 Results Table 1 shows significant improvement (p < 0.001)"
Spans Page 2 and Page 3

Figure 3.2: **Document-level chunking:** it concatenates page texts into a single document string before segmentation. While chunks may span page boundaries within the same document (for example, a chunk covering material from Page 1 into Page 2), they never cross document boundaries.

PDF and stored in an in-memory cache. For pages containing no extractable textual content, the system falls back to OCR [46] extraction to ensure comprehensive content coverage.

This cached textual content then feeds into the indexing process, which comprises three subroutines: (i) a chunking pipeline, (ii) construction of text embeddings using *BGE 1.5* [56] (the original codebase also exposes alternatives such as dense retrievers like MiniLM[53] and MPNet[47], but we restrict ourselves to the best performing *BGE 1.5*[56] as reported in Table 2.3), and (iii) retrieval of the nearest chunks for each query to construct a textual retrieval index CSV file.

Firstly, the chunking pipeline operates at the document level. For each document d in the cache, all page texts are first merged through concatenation with newline delimiters to yield a single document string T_d. The segmentation function `split_text` then processes T_d using default parameters (chunk size = 3000 characters, overlap = 300 characters) to produce contiguous chunks that may span page boundaries within the same document but never cross into another document. As illustrated in Figure 3.2, this design preserves narrative continuity while enforcing strict per-document boundaries.

Then each generated chunk undergoes metadata enrichment through a document identification process that attempts to determine both the source document and specific page location of the text segment. This is the most time-consuming stage in the preprocessing phase. Specifically, the system employs the `identify_document_and_page` method to extract document identifiers, and available page numbers through exhaustive sequence matching against all cached document pages. This identification process presents significant computational challenges for large-scale deployments due to its $O(nmp)$ complexity, where n represents the number of chunks, m the number of documents, and p the average pages per document. More critically, this routine assigns only a single page number to each chunk, whereas our document-level chunking strategy often produces chunks that span multiple consecutive pages. As a result, the current source attribution method is only partially accurate.

After chunking, we start the embedding process. During this process, we use *BGE 1.5* as the sentence transformer embedding model, which handles the transformation of text chunks into dense vector representations that capture semantic meaning beyond surface-level lexical matching. Additionally, query processing operates through an iterative workflow where each question undergoes embedding transformation using the identical *BGE 1.5* model to ensure vector space consistency. And to store these textual embeddings, the pipeline establishes a

ChromaDB [10] client instance and creates a dedicated collection with a uniquely generated identifier to prevent naming conflicts during concurrent operations.

At the third stage, the ChromaDB collection is configured to use cosine distance as the distance metric, which represents vector space distances in the *BGE 1.5* embedding space. By evaluating the cosine distance between each query-chunk pair, the system retrieves twice the target number of relevant chunks $2 \times$ `top_k`where$(k = 5)$ for each query. Then the retrieved results of each query are saved into a CSV file with additional details. For instance, firstly, each query-chunk pair receives a similarity score by converting the distance scores from ChromaDB through $cosine\_similarity = 1 - cosinedistance$. This maintain intuitive score interpretation where higher values indicate stronger relevance. Secondly, each retrieved chunk is enriched with its corresponding document identifier and page number information from the previously constructed mapping structure, ensuring comprehensive provenance tracking for result attribution.

Retrieval

- **Visual Pathway:** For visual context processing, the ranking selection mechanism extracts the top-k$(k = 5)$ highest-scoring document pages identifiers from the CSV-format retrieval index file. Then these abstract page identifiers must be converted into actual image objects, which are suitable for multimodal language model processing. This conversion algorithm first parses composite document identifiers to extract base document names and specific page numbers, then constructs file paths to locate the corresponding PDF documents within the configured document directory. Each PDF undergoes page-by-page conversion to image format using the pdf2image library. However, since we ultimately retain only the images of the retrieved pages as visual context, so rendering every page is redundant and incurs unnecessary computational overhead.

- **Textual Pathway:** The textual context processing also retrieves top-k $(k = 5)$ relevant chunks from the CSV-formatted retrieval index document for each query. Each chunk is enriched with its metadata information (including the chunk's source PDF name and the page number).

Generation The baseline processes queries in each dataset sequentially. For each query, it also processes visual and textual responses sequentially. Specifically, it waits until the visual pathway completes processing and returns the visual response for the query, then calls the textual pathway to obtain the textual response. While Figure3.1 shows a parallel structure, this just illustrates that we're getting visual and textual responses separately. In the baseline codebase, sequential processing is employed.

- **Visual Pathway:** We provide the visual context together with a specific prompt template to the LLM backbones. This prompt implements a three-stage analytical framework for document-based question answering. It systematically progresses from Evidence Curation, where relevant elements are extracted from PDF sources, through Chain of Thought, which links individual pieces of evidence into a coherent step-by-step narrative, to Answer Synthesis, where the final response emerges from the established reasoning chain. We also incorporated template variables for question and answer specifications, by doing this, the architecture can be easily adapted to different datasets and questions without requiring code modifications. The complete prompt is provided in the appendix.

- **Textual Pathway:** Similarly, the retrieved textual contexts are processed through the same LLM backbones using a prompt template similar to that of the visual pathway, with the complete prompt specification provided in the appendix.
- **Fusion:** Once we obtain both textual and visual responses, we provide these two responses along with a specific prompt template to the same LLM backbones. The prompt implements a modality fusion mechanism that combines outputs from visual and textual question-answering pathways through structured LLM-based evaluation. It presents both candidate responses with their complete reasoning chains, including evidence, chain-of-thought processes, and final answers, for systematic analysis. It also establishes clear decision guidelines that prioritize visual evidence when both pathways demonstrate logical reasoning, while handling edge cases where one response declines to provide an answer. The prompt enforces a structured analytical framework requiring explicit evaluation of reasoning consistency, evidence validity, and cross-modal agreement before fusion. By constraining the output format to include detailed analysis, conclusion, and final answer sections, this approach ensures interpretable decision-making.

3.1.3 Experiment 2: Variant 1(V1)

Objective.

Optimise the baseline VisDoMRAG pipeline to reduce overall execution time while maintaining the quality of generated responses.

Configuration.

This configuration differs from the baseline in the following ways: First, it implements the optimized `identify_document_and_page_batch` function with word-overlap filtering at a 0.3 threshold. Second, processing operates with a batch size of 4 and utilizes 2 parallel workers by default, with query execution handled through ThreadPoolExecutor for concurrent processing. For answer generation, it employs the three backbones stated in Section 3.1 with identical prompt templates and decoding parameters.

Implementation details

The optimized pipeline retains the dual-pathway retrieval architecture from the baseline.

Document Parsing & Indexing: The first problem we address is the redundant textual index reconstruction for each query. We implement index pre-computation during initialization through the `_precompute_indices` method, which builds both visual and textual indices once at startup rather than rebuilding them for each query. This one-time computation dramatically reduces redundant processing. The indices are persisted to disk and reused across all queries, transforming an $O(n)$ operation into $O(1)$ for subsequent textual retrievals, where n is the number of queries.

- **Visual Pathway:** For visual preprocessing, Colqwen2 still serves as the visual retriever. The most significant enhancement in this optimized pipeline for visual index construction is the adoption of batch processing for both document pages and queries. While the original implementation processes each PDF page individually through the vision model, generating embeddings one at a time, the optimized version accumulates pages into batches before processing. This batching strategy dramatically reduces the overhead of model invocations and enables more efficient GPU utilization through parallel computation. The optimized version introduces dedicated helper functions `_process_visual_batch` and `_process_query_batch` that handle batch sizes defined by a configurable parameter, processing multiple images or queries simultaneously through the ColQwen2 model.

  For query processing, the original implementation iterates through the dataset row by row, generating query embeddings sequentially. The optimized version batches queries, accumulating them before processing through the vision model. This parallel query processing significantly reduces the time required for embedding generation, especially when dealing with large datasets containing hundreds of queries.

  Despite the changes introduced above, the optimized implementation maintains identical core functionality as the baseline implementation: converting PDFs to images, generating embeddings, computing multi-vector scores, and producing CSV retrieval indices. The embedding generation logic, scoring algorithm, and output format remain unchanged, ensuring that the optimized version delivers equivalent retrieval quality and output structures while achieving substantial time savings through its optimized architecture.

- **Textual Pathway:** As we mentioned above, during document caching, when OCR [46] is required for pages containing non-extractable text, processing each page sequentially takes considerable time, as in the baseline approach. Therefore, our optimization implements parallel document processing to accelerate the PDF text extraction process. The system initializes a process pool executor with a configurable number of worker processes. Each worker process handles different documents concurrently by executing the `extract_text_from_pdf_worker` function, which maintains the same extraction logic as the original method by implementing identical algorithm for attempting PyPDF2 extraction before falling back to OCR when pages contain no extractable text. Importantly, the final cache structure remains identical to what the baseline would produce. The optimized implementation incorporates the same cache persistence mechanism, serializing the document cache to disk using pickle for subsequent reuse.

  Following the optimization of the caching process, we now turn to the optimization of the indexing process. Recall that this process has three subroutines: (i) a chunking pipeline, (ii) construction of text embeddings, and (iii) retrieval of the nearest chunks for each query to construct a textual retrieval index CSV file.

  For the chunking pipeline, the optimization targets the metadata extraction process, where the original `identify_document_and_page` function processes chunks individually, performing exhaustive sequence matching between each chunk and every page in the document cache. The optimized `identify_document_and_page_batch` function introduces two key improvements. First, it processes multiple chunks in a single function call, reducing function invocation overhead and improving memory efficiency. Second, it implements a two-stage similarity computation that dramatically reduces computational cost. Specifically, the function first performs a lightweight word-overlap check using set operations, computing the intersection between chunk words and page words. Then the function proceeds to the computationally expensive SequenceMatcher calculation only if this initial overlap exceeds a threshold of 0.3. The threshold of 0.3 was chosen because if the initial overlap between the chunk words and the page words is below 30%, the clearly does not originate from that page. This filtering mechanism eliminates unnecessary sequence matching operations for clearly dissimilar text pairs.

At the next stage, the embedding stage, our optimized pipeline improves upon the original implementation by batching both insertion and query operations. For insertion, chunks are added to the Chroma collection in manageable batches, which avoids memory issues. Query processing is transformed through the `_process_st_batch` function, which executes multiple queries in a single collection query call rather than making individual database calls.

The third retrieval stage remains unchanged. Additionally, the optimized implementation maintains identical core functionality to the baseline implementation: generating embeddings, computing cosine-similarity scores, and producing CSV retrieval indices.

Retrieval The visual and textual retrieval algorithms remain largely unchanged from the baseline, with the following optimizations:

- **Visual Pathway:** We introduced the `retrieve_visual_contexts_optimized` function to optimize visual retrieval by directly accessing visual contexts from pre-computed CSV retrieval indices. The function minimizes memory usage and processing time by converting only the most relevant PDF pages for each query, utilizing the `first_page` and `last_page` parameters of `convert_from_path`.

- **Textual Pathway:** Building on pre-computed indices, the retrieval methods undergo significant streamlining. Hence in `retrieve_textual_contexts_optimized` function, we eliminate index building entirely, instead directly loading pre-computed CSV files containing retrieval results.

Generation When generating LLM responses for hundreds of queries in our dataset, we don't process responses for each query sequentially. Instead, we process them in parallel using two execution strategies. First, the `run_parallel` method processes multiple queries concurrently using ThreadPoolExecutor with configurable workers, enabling the system to handle multiple independent queries simultaneously. Seoncd, the `run_gpu_parallel` method provides an alternative execution path optimized for API-based models, processing queries in controlled batches to respect rate limits while maintaining parallelism within each batch.

When processing each query, we retain the same prompt templates and LLM backbones as the baseline for both visual, textual response generation as well as the fusion stage. The optimization here focuses on parallel context processing within individual queries. For instance, the `_query_optimized` method employs a ThreadPoolExecutor to simultaneously retrieve and process visual and textual contexts through the `_process_visual_context` and `_process_textual_context`

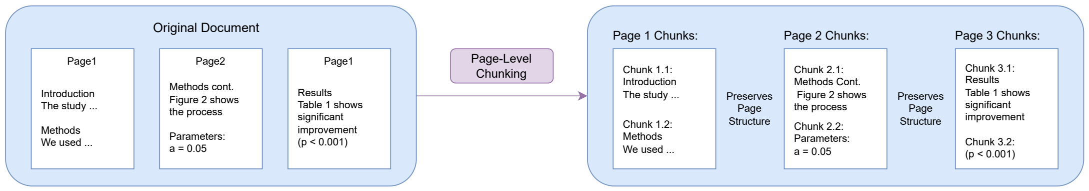


Figure 3.3: **Page-level chunking:** The chunking process maintains page boundaries by segmenting each page into separate chunks while preserving the original page-level organization.

helper methods. This concurrent execution eliminates the sequential bottleneck where visual processing would wait for textual processing to complete. Once we obtain both the visual and textual responses for the query, we call the same fusion function as the baseline.

3.1.4 Experiment 3: Variant 2(V2)

Objective. Replace document-level chunking with page-level chunking to assess the impact of chunking strategy on pipeline runtime and Word Overlap F1 performance.

Configuration. This experiment shifts to page-level chunking, eliminating the metadata extraction step entirely as chunks maintain direct page mappings through their creation context. While retaining ColQwen2 and *BGE 1.5* as retrievers, with `top_k=5`, the text index construction diverges from V1 by reverting to sequential query processing and implementing batched Chroma insertion with a 5000-chunk batch size to prevent memory overflow. Meanwhile, visual index construction, document caching, and the query execution preserve V1's parallel processing architecture. All other parameters also remain consistent with V1, including worker counts, persistence settings, and generation configurations.

Implementation details The only variant in this experiment is that our `build_text_index` function implements page-level chunking(Figure3.3, iterating through each page of each document and creating chunks directly from individual page text. This approach maintains an inherent connection between chunks and their source pages, as the chunking occurs within the page iteration loop. The metadata mapping is straightforward: each chunk knows its document ID and page number immediately because it was created from a specific page. The function directly assigns `doc_id` as the `chunk_pdf_name` and uses the loop's `page_num` variable for `pdf_page_number`, eliminating any need for post-processing metadata identification. Consequently the `identify_document_and_page_batch` function is unnecessary.

For the retrieval stage of the textual indexing part, we continue using *BGE 1.5* [56] as our retriever. While the retrieval process for each query in the `build_text_index` function remains similar to the baseline, iterating through queries sequentially and retrieving the top-k($k = 5$) nearest neighbors for each, the function improves how chunks are inserted into the Chroma vector database. Specifically, the implementation uses a fixed batch size of 5,000 chunks, processing the entire dataset in sequential batches until all chunks are inserted with their IDs. This batched approach prevents memory overflow when dealing with large document collections.

The rest of the implementation remains the same as V1.

3.2 Summary

This methodology presents a systematic approach to optimizing VisDoMRAG for PDF question answering. We establish a baseline implementation of the VisDoMRAG architecture, then introduce two optimization variants that progressively improve runtime efficiency while monitoring answer quality through word-overlap F1[41] scores.

Specifically, our experimental progression follows a clear optimization trajectory. The baseline establishes reference performance with sequential processing and exhaustive metadata recovery for chunks. Building on this foundation, Variant 1 introduces comprehensive parallelization across visual and textual pipelines, implementing batch processing, optimized metadata recovery, and concurrent query execution. Subsequently, Variant 2 explores an alternative chunking strategy, replacing document-level chunking with page-level segmentation to eliminate metadata lookup overhead while accepting the trade-off of sequential query processing during text index construction.

The evaluation framework employs consistent metrics across all experiments, using macro-averaged word-overlap F1[41] to measure answer quality and wall-clock time for runtime assessment. This enables direct comparison of the trade-offs between processing efficiency and retrieval effectiveness. Additionally, the five-dataset VisDoMBench benchmark ensures our optimizations generalize across diverse document types and question complexities.

Through this methodology, we systematically identify computational bottlenecks in VisDoMRAG pipelines and demonstrate that substantial runtime improvements are achievable without sacrificing answer quality, providing practical insights for deploying these systems at scale.

Chapter 4

Evaluation

This chapter presents a comprehensive evaluation of optimization strategies for the VisDoMRAG multimodal retrieval-augmented generation system. Specifically, through comparative experiments on the VisDoMBench dataset, we assess the impact of architectural improvements and integration with modern LLMs. For quantitative assessment, the evaluation used the Word-Overlap F1 as the primary end-to-end metric, and we also reported the wall-clock runtime for each pipeline component. Additionally, we analyze the number of chunks generated by different chunking methods across all the datasets, revealing how document characteristics influence the effectiveness of different chunking strategies. Furthermore, we'll conduct error analysis on SlideVQA across three LLM backbones (Gemini-1.5-Flash [49] , Llama3-70b [29, 30] , Pixtral-Large ([31] [32])) to understand the types of errors made by different model architectures.

4.1 Evaluation Metric

We evaluate model outputs with the SQuAD token-level (word-overlap) F1 score [41]. Unlike exact match metrics, this F1 assigns partial credit based on word overlap, making it more tolerant when predictions contain the correct tokens alongside additional content, though it remains sensitive to paraphrasing and penalizes verbose outputs through reduced precision.

To compute this metric, for each prediction-gold pair, we first normalize both strings following SQuAD v1.1 [41] conventions: lowercasing, removing punctuation and articles (a, an, the), and collapsing whitespace. After normalization, the normalized strings are tokenized into multisets that preserve token multiplicity, ensuring repeated words contribute proportionally to the overlap score.

With the normalized tokens prepared, we compute Word-Overlap F1 [41] from precision and recall based on token overlap. Let C_p and C_g denote token-count functions for the prediction and gold answer respectively, where $C(w)$ represents the count of token w. We first calculate the true positives as the overlap between prediction and gold tokens:

$$\mathrm{TP} = \sum_{w} \min\{C_p(w),\, C_g(w)\}. \tag{4.1}$$

False positives and false negatives are calculated as:

$$\mathrm{FP} = \sum_{w} C_p(w) - \mathrm{TP}, \qquad \mathrm{FN} = \sum_{w} C_g(w) - \mathrm{TP}. \tag{4.2}$$

Precision and recall are computed as:

$$\mathrm{Prec} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \qquad \mathrm{Rec} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \tag{4.3}$$

The F1 score is the harmonic mean when defined:

$$\mathrm{F1} = \begin{cases} \dfrac{2\,\mathrm{Prec}\cdot\mathrm{Rec}}{\mathrm{Prec}+\mathrm{Rec}} \times 100\%, & \text{if Prec} + \text{Rec} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

The macro-average across all valid examples weights each instance equally:

$$\overline{\mathrm{F1}} = \frac{1}{N_{\text{valid}}} \sum_{i=1}^{N_{\text{valid}}} \mathrm{F1}_i. \tag{4.5}$$

This lexical metric intentionally avoids stemming or lemmatization to maintain transparency and reproducibility. Examples with missing or empty answer fields are excluded from evaluation.

4.2 Pipeline runtime

In this section, we report the end-to-end wall-clock runtime of the pipelines using the Gemini-1.5-Flash LLM backbone on each dataset.

4.2.1 Runtime for Baseline with LLM: Gemini-1.5-Flash

The Table 4.1 reports end-to-end runtime in seconds of the baseline pipeline using Gemini-1.5-Flash across five datasets. The table presents runtime for four pipeline components (visual indexing, document caching, textual indexing, and query processing) as well as the total runtime. Most notably, according to Figure 4.1, textual indexing represents the dominant computational cost across all datasets, consuming approximately 63.7–98.3% of total runtime. Following this primary bottleneck, query processing emerges as the second largest contributor, proving particularly significant for SPIQA (33% of total runtime) and SlideVQA (23.5%). In contract, visual indexing and document caching impose minimal overhead across most datasets. However, the notable exception is SlideVQA's document caching process, which requires significantly more time due to OCR [46] processing requirements, as slide presentations lack directly extractable text content.

4.2.2 Runtime for V1 with LLM: Gemini-1.5-Flash

Table 4.2 shows the end-to-end runtime in seconds of the V1 pipeline using Gemini-1.5-Flash across five datasets, decomposed into the same components as the baseline. Meanwhile, Table 4.3 reports the runtime reduction by comparing Table 4.2 and Table 4.1.

Time-Taken (s)	visual indexing	document caching	textual indexing	queries processing time	total
SPIQA	469	116	28020	14065	42670
SlideVQA	1702	10895	62396	23009	98002
SciGraphQA	1658	2076	410448	11428	425610
PaperTab	749	105	97230	7403	105487
FetaTab	1087	1224	648356	8973	659640

Table 4.1: Time taken for each component of baseline. LLM: Gemini-1.5-Flash

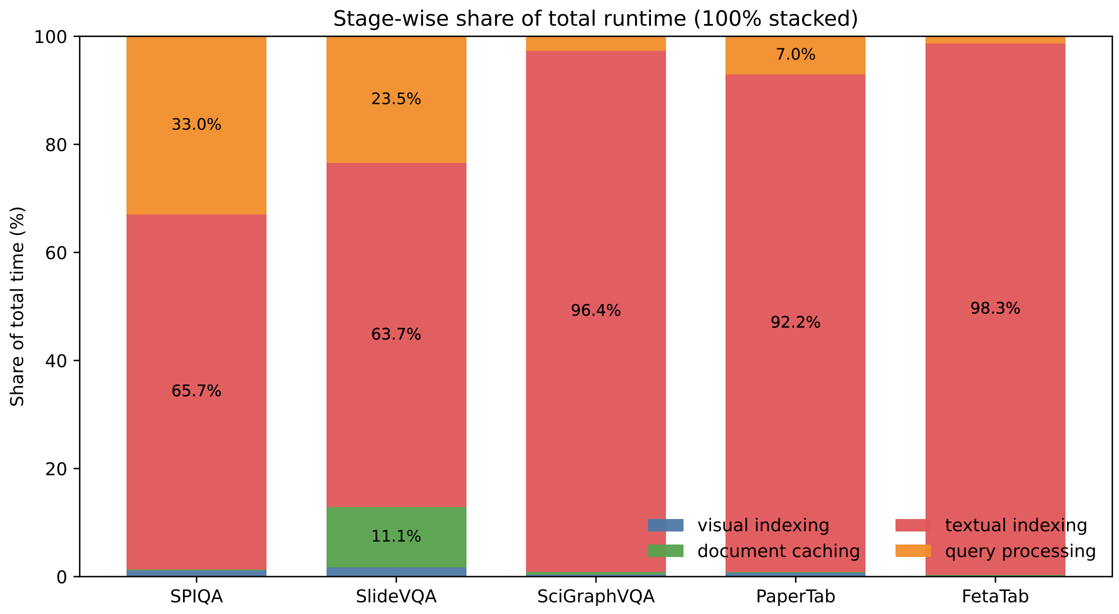


Figure 4.1: Stage-wise Runtime Distribution Across VisDoMBench Datasets for Baseline with Gemini-1.5-Flash

The analysis of these tables reveals that the V1 configuration shows dramatic performance improvements compared to the baseline across all pipeline components. Specifically, the most striking improvements occur in textual indexing, where V1 achieves reduction percentages between 97.76% and 99.68% across all datasets. These substantial gains demonstrate the effectiveness of V1's optimization strategy. More precisely, by combining an improved metadata recovery algorithm with batch and parallel processing, V1 significantly accelerates the textual indexing process.

Meanwhile, query processing follows a similar pattern of substantial optimization. Across all datasets, query processing achieves reduction percentages ranging from 84.51% to 91.33%, representing significant computational savings. Additionally, SlideVQA's document caching, which represented a major bottleneck in the baseline due to OCR [46] requirements, achieves a 71.86% reduction in processing time. These improvements indicate that V1's parallel processing architecture and concurrent execution strategies effectively address the primary computational bottlenecks identified in the baseline analysis.

Time-Taken (s)	visual indexing	document caching	textual indexing	queries processing time	total
SPIQA	228	24	470	2230	2952
SlideVQA	937	3066	200	2339	6542
SciGraphQA	834	1564	4188	991	7577
PaperTab	417	26	1578	809	2830
FetaTab	546	325	14555	909	16335

Table 4.2: Time taken for each component of V1. LLM:Gemini-1.5-Flash

Reduction Percentage (%)	visual indexing	document caching	textual indexing	queries processing time
SPIQA	51.39	79.31	98.32	84.51
SlideVQA	44.95	71.86	99.68	89.83
SciGraphQA	49.70	24.66	98.98	91.33
PaperTab	44.33	75.24	98.38	89.07
FetaTab	49.77	73.45	97.76	89.87

Table 4.3: Runtime Reduction Percentage per Component (Baseline to V1)

4.2.3 Runtime for V2 with LLM:Gemini-1.5-Flash

In the design of experiment V2, we only vary the chunking method from document-level to page-level chunking, which eliminates the entire chunks metadata recovery process. Additionally, we revert to sequential processing in the textual indexing stage, similar to the baseline. All other component structures remain the same as V1. As expected from these design choices, comparing Table 4.2 and Table 4.4, textual indexing shows dramatic further improvement in V2, while visual indexing, document caching and query processing remain relatively stable. More precisely, the runtime reduction for textual indexing ranges from $156s$ to $14436s$, while the runtime changes for visual indexing, document caching and query processing range from $1s$ to $87s$ compared to V1.

Furthermore, another observation is that the total pipeline runtime for the last three datasets (SciGraphQA, PaperTab, FataTab) in the Table 4.4 reduces significantly compared to Table 4.2 due to substantial reductions in their textual indexing stages. For example, for the FetaTab dataset, while other components show runtime variations between $2s$ and $57s$, its total pipeline runtime decreases by $14,397s$, primarily driven by the $14,436s$ reduction in textual indexing. This demonstrates that FetaTab benefits substantially from the V2 optimization strategy. In contrast, for the SlideVQA dataset, even though the runtime of its textual indexing reduces by $156s$, the total pipeline runtime for V2 increases by $5s$. This occurs because the combined increases in runtime for other components (visual indexing, document caching, and query processing) exceed the $156s$ reduction achieved in textual indexing.

Time-Taken (s)	visual indexing	document caching	textual indexing	queries processing time	total
SPIQA	315	23	37	2145	2520
SlideVQA	946	3134	44	2423	6547
SciGraphQA	822	1478	94	978	3372
PaperTab	392	26	53	791	1262
FetaTab	530	323	119	966	1938

Table 4.4: Time taken for each component of V2. LLM: Gemini-1.5-Flash

4.2.4 Summary of the Pipeline Runtime

The pipeline runtime evaluation for the baseline reveals that textual indexing dominated baseline performance, consuming 64-98% of processing time and making the system impractical with runtimes ranging from around 12 to 183 hours. Then, V1's comprehensive parallelization strategy achieved dramatic improvements across all components, with textual indexing showing 97-99% reduction and query processing improving by 84-91%, transforming processing times from hours to minutes. Moreover, V2's alternative approach of eliminating metadata recovery entirely through page-level chunking proved even more effective for textual indexing, achieving additional 78-99% runtime reduction over V1. However, when considering total pipeline performance, the benefits of optimization in V2 vary significantly across datasets. For instance, while datasets with more textual content that needs to be chunked, like FetaTab (which consists of scientific papers), show substantial total improvements, SlideVQA (whose slide pages contain much less textual content than scientific pages) experiences minimal gains.

4.3 Chunk Count Comparison by Chunking Method

The following table shows the number of chunks generated by document-level chunking and page-level chunking separately for each dataset.

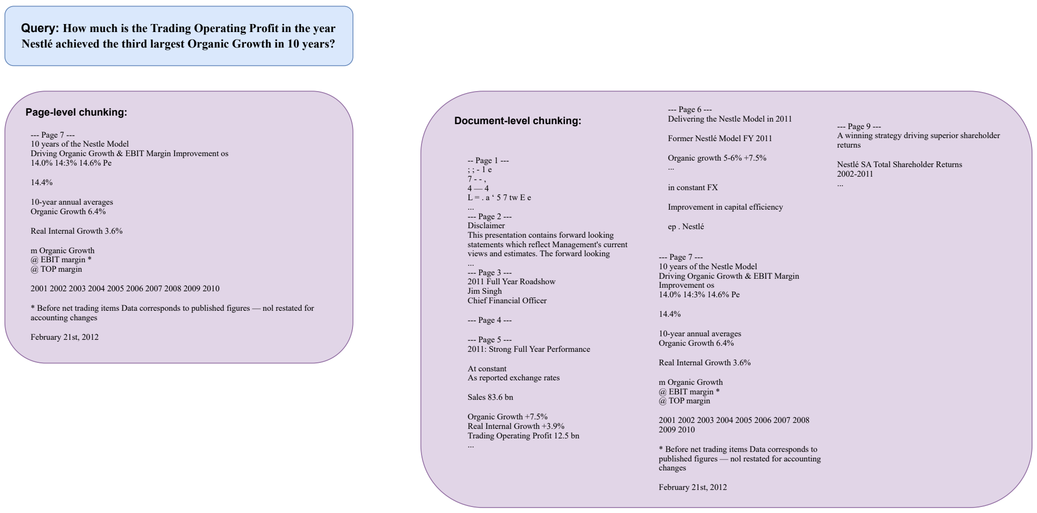
Number of Chunks	Document-Level Chunking	Page-Level Chunking
SPIQA	2305	2907
SlideVQA	1554	9774
SciGraphQA	7454	10239
PaperTab	4384	5542
FetaTab	9712	13557

Table 4.5: Chunk Generation Comparison: Document-Level vs Page-Level Chunking Strategies

According to the Table 4.5, we observed that the page-level chunking consistently produces more chunks than document-level chunking across all datasets, but the magnitude of increase varies dramatically based on document type.

Specifically, SlideVQA shows the most extreme difference, with page-level chunking generating about 6.3 times more chunks (9774 vs 1554). This massive increase occurs because of how the two strategies handle sparse text content. More precisely, in document-level chunking (see Figure 3.2), text from multiple slides gets concatenated together until it reaches 3,000 characters (the default chunk size) before creating a chunk. So if each slide only has 200-300 words, the system combines 8-10 slides into one chunk. In contrast, in page-level chunking (see Figure 3.3), since most slides probably have less than 3000 characters each, each slide page ends up creating exactly one small chunk since the page boundaries act as hard limits. For example (Figure 4.2), consider the query: 'How much is the Trading Operating Profit in the year Nestlé achieved the third largest Organic Growth in 10 years?' from SlideVQA. The chunk retrieved using page-level chunking corresponds exactly to page 7 of the slides, whereas the chunk from document-level chunking spans content from pages 1 through 9.

Furthermore, the text-dense datasets show more moderate increases. In particular, SPIQA and PaperTab exhibit relatively small differences, with page-level chunking producing only about



Query: **How much is the Trading Operating Profit in the year Nestlé achieved the third largest Organic Growth in 10 years?**

Page-level chunking:

```
--- Page 7 ---
10 years of the Nestle Model
Driving Organic Growth & EBIT Margin Improvement os
14.0% 14:3% 14.6% Pe

14.4%

10-year annual averages
Organic Growth 6.4%

Real Internal Growth 3.6%

m Organic Growth
@ EBIT margin *
@ TOP margin

2001 2002 2003 2004 2005 2006 2007 2008 2009 2010

* Before net trading items Data corresponds to published figures — nol restated for
accounting changes

February 21st, 2012
```

Document-level chunking:

```
-- Page 1 ---
; ; - 1 e
7 - - ,
4 — 4
L = . a ' 5 7 tw E e
...
--- Page 2 ---
Disclaimer
This presentation contains forward looking
statements which reflect Management's current
views and estimates. The forward looking
...
--- Page 3 ---
2011 Full Year Roadshow
Jim Singh
Chief Financial Officer

--- Page 4 ---

--- Page 5 ---
2011: Strong Full Year Performance

At constant
As reported exchange rates

Sales 83.6 bn

Organic Growth +7.5%
Real Internal Growth +3.9%
Trading Operating Profit 12.5 bn
...

--- Page 6 ---
Delivering the Nestle Model in 2011

Former Nestlé Model FY 2011

Organic growth 5-6% +7.5%
...

in constant FX

Improvement in capital efficiency

ep . Nestlé

--- Page 7 ---
10 years of the Nestle Model
Driving Organic Growth & EBIT Margin
Improvement os
14.0% 14:3% 14.6% Pe

14.4%

10-year annual averages
Organic Growth 6.4%

Real Internal Growth 3.6%

m Organic Growth
@ EBIT margin *
@ TOP margin

2001 2002 2003 2004 2005 2006 2007 2008
2009 2010

* Before net trading items Data corresponds to
published figures — nol restated for accounting
changes

February 21st, 2012

--- Page 9 ---
A winning strategy driving superior shareholder
returns

Nestlé SA Total Shareholder Returns
2002-2011
...
```

Figure 4.2: This figure shows the first retrieved chunk for a SlideVQA dataset query using two different chunking methods: page-level chunking and document-level chunking. Note that this figure displays only a portion of the document-level chunking result, as the complete chunk content is substantially longer.

26% more chunks for both datasets. This increase occurs for the same fundamental reason as SlideVQA, but the effect is much less significant because these datasets contain denser text per page than the slides. For example, if a page in PaperTab contains 2,800 characters, it still creates only one chunk, whereas document-level chunking might combine it with text from adjacent pages to create a perfectly-sized 3,000-character chunk.

Additionally, SciGraphQA and FetaTab demonstrate intermediate patterns, with about 37% increases for both datasets. These datasets contain a mix of text-dense and text-sparse pages, such as pages with large figures, tables and mathematical content that reduces the available text for chunking.

4.3.1 Summary

The data reveals a clear correlation between document text density and number of chunks generated. For instance, datasets with consistent, dense textual content show minimal differences in number of chunks between chunking strategies, while those with sparse content per page experience significant chunk number multiplication with page-level approaches.

4.4 Dataset Performance

For analyzing the performance of each dataset, we begin by comparing the F1 scores across the baseline, V1, and V2 configurations with Gemini-1.5-Flash to assess whether V1 and V2 preserve answer quality while reducing runtime compared to the baseline. More precisely, we consider that V1 and V2 preserve answer quality if their F1 scores are not more than 3 percentage point lower than the baseline performance. Afterwards, we compare the F1 scores of V1 and V2 across the three LLM backbones (Gemini-1.5-Flash, Llama3-70b, and Pixtral-Large) to evaluate whether more advanced LLM backbones can improve the response quality.

SPIQA dataset

The results were as follows:

Macro-Average Word-overlap F1 scores (%)	**Baseline**
Gemini-1.5-Flash	15.40

Table 4.6: Macro-averaged word-overlap F1 (%) for each backbone under Baseline running on SPIQA. Higher is better.

Macro-Average Word-overlap F1 scores (%)	**V1**	**V2**
Gemini-1.5-Flash	14.51	13.59
Llama3-70b	31.15	30.94
Pixtral-Large	27.16	22.56

Table 4.7: Macro-averaged word-overlap F1 (%) for each backbone under the three pipelines (V1, V2) running on SPIQA. Higher is better.

The baseline performance with Gemini-1.5-Flash (Table 4.6) achieves only 15.40% F1 score, establishing a low performance level. Importantly, both V1 and V2 optimizations show decreased answer quality compared to the baseline, with V1 at 14.51% and V2 at 13.59% (Table 4.7), representing decreases of 0.89 and 1.81 percentage points respectively. Since these decreases are both less than our 3 percentage point threshold, we consider that the optimization strategies maintain acceptable answer quality while achieving substantial runtime improvements for SPIQA. Furthermore, V1 slightly outperforms V2 by 0.92 percentage point in terms of F1 score for this dataset.

Next, we evaluate V1 and V2 configurations with Gemini-1.5-Flash, Llama3-70b and Pixtral-Large on SPIQA (Table 4.7). The results show Llama3-70b significantly outperforming others at 31.15% for V1 and 30.94% for V2. Meanwhile, Pixtral-Large shows moderate performance at 27.16% for V1 and 22.56% for V2. Notably, both LLMs achieve substantially better F1 scores in the optimized configurations than Gemini-1.5-Flash achieved in V1 and V2 (V1: 14.51%, V2: 13.59%).

Additionally, the F1 scores of V1 always outperform V2 whether with Llama3-70b (by 0.21 percentage point), Pixtral-Large (by 4.6 percentage point) or Gemini-1.5-Flash (by 0.92 percentage point). Interestingly, recall from Table 4.5 that we observed the page-level chunking of V2 generates more chunks than the document-level chunking of V1 for SPIQA, but the end-to-end F1 scores clearly show that SPIQA doesn't benefit from this increase. This might be due to the fact that while page-level chunking generates more chunks, these chunks lose cross-boundary content that spans multiple pages. Since the number of chunks we retrieved for each query is fixed, the retrieved chunks from page-level chunking may contain less comprehensive information than those from document-level chunking, which can include content that spans across page boundaries.

Finally, based on all the observations above, the best performing combination for SPIQA is Llama3-70b with V1, achieving 31.15% F1 score.

SlideVQA Dataset

The results were as follows:

Macro-Average Word-overlap F1 scores (%)	**Baseline**
Gemini-1.5-Flash	4.70

Table 4.8: Macro-averaged word-overlap F1 (%) for each backbone under Baseline running on SlideVQA. Higher is better.

Macro-Average Word-overlap F1 scores (%)	**V1**	**V2**
Gemini-1.5-Flash	3.6	4.66
Llama3-70b	5.64	5.66
Pixtral-Large	3.53	7.58

Table 4.9: Macro-averaged word-overlap F1 (%) for each backbone under the three pipelines (V1, V2) running on SlideVQA. Higher is better.

The overall results for SlideVQA are worse than that of SPIQA. Specifically, the baseline performance with Gemini-1.5-Flash achieves only 4.70% (Table 4.8), which is even lower than SPIQA's already poor baseline performance. Regarding the V1 and V2 with Gemini-1.5-Flash (Table 4.9), V1 decreases the performance to 3.6% (a 1.1 percentage point drop), while V2 also shows a slight decrease to 4.66% (a 0.04 percentage point drop). Since these decreases in F1 scores are both less than our 3 percentage points threshold, we conclude that while V1 and V2 achieve substantial runtime improvements over the baseline, they still maintain the answer quality well for SlideVQA with Gemini-1.5-Flash.

With Llama3-70b, V1 and V2 both perform similarly well, achieving 5.64% for V1 and 5.66% for V2, where V2 outperforms V1 slightly by 0.02 percentage point. These scores represent improvements over V1 and V2's combinations with Gemini-1.5-Flash (2.04 and 1 percentage point improvement respectively). Moreover, Pixtral-Large shows the most dramatic variation, with V1 achieving 3.53% (below V1's combination with Gemini-1.5-Flash by 0.07 percentage point) while V2 reaches 7.58%, representing the highest performance across all the configurations for this dataset.

These results show that V2 outperforms V1 across all the LLM backbones, demonstrating that V2's optimization (page-level chunking) is particularly beneficial for answer generation accuracy on slide-based documents, despite V2 not achieving superior runtime performance compared to V1 for this dataset (as stated in Section 4.2.2). This might because slides naturally align with page boundaries, with minimal cross-page content spanning multiple slides. Consequently, applying page-level chunking to slide-based documents preserves content integrity without losing meaningful cross-boundary information. In contrast, document-level chunking combines textual content from multiple slides into a single chunk, but the content within each slide is often thematically unrelated. As a result, with a fixed number of retrieved chunks for each query, page-level chunks are more likely to contain focused, query-relevant information from individual slides. Meanwhile, document-level chunks may include substantial unrelated content from multiple slides, potentially confusing the LLMs during answer generation. Therefore, V2's page-level approach achieves better end-to-end performance for SlideVQA by providing more targeted, contextually coherent chunks that align with the natural structure of slide presentations. In addition, the example from Section 4.3 (Figure 4.2) illustrates this principle: for the query 'How much is the Trading Operating Profit in the year Nestlé achieved the third largest Organic Growth in 10 years?', the top chunk from page-level chunking contains only content from page 7, while the document-level chunk spans pages 1 through 9. Since the ground truth relevant pages for this query are only pages 5 and 7, the document-level chunk contains substantial irrelevant information from unrelated slides that could confuse the LLM.

Finally, based on all the observations above, the best performing combination for SlideVQA is Pixtral-Large with V2, achieving 7.58% F1 score.

SciGraphQA Dataset

The results were as follows:

Macro-Average Word-overlap F1 scores (%)	**Baseline**
Gemini-1.5-Flash	15.06

Table 4.10: Macro-averaged word-overlap F1 (%) for each backbone under Baseline running on SciGraphQA. Higher is better.

Macro-Average Word-overlap F1 scores (%)	**V1**	**V2**
Gemini-1.5-Flash	14.99	15.5
Llama3-70b	26.94	27.13
Pixtral-Large	20.98	24.03

Table 4.11: Macro-averaged word-overlap F1 (%) for each backbone under the three pipelines (V1, V2) running on SciGraphQA. Higher is better.

By observing Table 4.11 and 4.10, we see that for Gemini-1.5-Flash, the baseline performance achieves 15.06% F1 score while V1 and V2 achieve 14.99% and 15.5% respectively. These results show that while V2 reduces the runtime significantly over the baseline, it also improves the F1 scores of the responses generated by Gemini-1.5-Flash (0.44 percentage point increase). In contrast, V1 decreases the F1 scores by 0.07 percentage point compared to the baseline. However, this 0.07 percentage point drop is within the 3 percentage point threshold, which is acceptable. Therefore, while V2 runs faster than V1 on SciGraphVQA and improves answer quality, V1 also maintains acceptable answer quality.

Regarding the F1 scores of V1 and V2 using Llama3-70b and Pixtral Large, Llama3-70b shows V2 achieving 27.13% compared to V1's 26.94%, a modest 0.19 percentage point advantage. More significantly, Pixtral-Large demonstrates the largest performance gap, with V2 reaching 24.03% versus V1's 20.98%, representing a substantial 3.05 percentage point improvement. Additionally, Llama3-70b and Pixtral-Large both significantly outperform gemini-1.50-flash for V1 and V2 configurations, while Llama3-70b also achieves higher F1 scores than Pixtral-Large overall.

Furthermore, the consistent V2 superiority across all LLM backbones indicates that page-level chunking is particularly well-suited for SciGraphQA. This might because scientific papers have content that naturally aligns with page boundaries, where figures, captions, and related textual descriptions are typically co-located on the same page. Consequently, page-level chunking preserves these figure-text relationships that might be fragmented by document-level chunking.

Finally, based on all the observations above, the best performing combination for SciGraphVQA is Llama3-70b with V2, achieving 27.13% F1 score.

PaperTab Dataset

The results were as follows:

Macro-Average Word-overlap F1 scores (%)	**Baseline**
Gemini-1.5-Flash	14.04

Table 4.12: Macro-averaged word-overlap F1 (%) for each backbone under Baseline running on PaperTab. Higher is better.

Macro-Average Word-overlap F1 scores (%)	**V1**	**V2**
Gemini-1.5-Flash	13.78	14.76
Llama3-70b	16.71	16.65
Pixtral-Large	15.10	10.46

Table 4.13: Macro-averaged word-overlap F1 (%) for each backbone under the three pipelines (V1, V2) running on PaperTab. Higher is better.

From Table 4.12 and Table 4.13, we see that for Gemini-1.5-Flash, the baseline achieves 14.04% F1 score, while the optimization results show V2 outperforming both the baseline and V1, achieving 14.76% compared to V1's 13.78%. This represents a 0.72 percentage point improvement over baseline for V2, while V1 shows a 0.26 percentage point decrease. However, V1's decrease is within the threshold, so V1 still preserves the answer quality.

Regarding the results of V1 and V2 with alternative LLM backbones, Llama3-70b demonstrates relatively balanced performance between the two optimization strategies, with V1 achieving 16.71% and V2 reaching 16.65%, representing only a marginal 0.06 percentage point difference in favor of V1. Additionally, Llama3-70b configurations significantly outperform the Gemini-1.5-Flash results for V1, V2.

However, Pixtral-Large shows a notable divergence, with V1 achieving 15.10% while V2 drops to 10.46%, representing a substantial 4.64 percentage point decrease. This suggests that for PaperTab documents, Pixtral-Large performs significantly better with document-level chunking than with page-level chunking, though the underlying reasons require further investigation.

Finally, based on all the observations above, the best performing combination for PaperTab is Llama3-70b with V1, achieving 16.71% F1 score.

FetaTab Dataset

The results were as follows:

Based on the results from Table 4.14 and the Table 4.15, we see that the FetaTab dataset demonstrates notably higher baseline performance compared to the previous datasets, with Gemini-1.5-Flash achieving 29.13% F1 score. Moreover, V1 and V2 shows decreased performance running

Macro-Average Word-overlap F1 scores (%)	**Baseline**
Gemini-1.5-Flash	29.13

Table 4.14: Macro-averaged word-overlap F1 (%) for each backbone under Baseline running on FataTab. Higher is better.

Macro-Average Word-overlap F1 scores (%)	**V1**	**V2**
Gemini-1.5-Flash	26.89	28.65
Llama3-70b	52.91	53.73
Pixtral-Large	26.06	27.93

Table 4.15: Macro-averaged word-overlap F1 (%) for each backbone under the three pipelines (V1, V2) running on FetaTab. Higher is better.

with the same Gemini model compared to the baseline, with V1 achieving 26.89% (a 2.24 percentage point drop) and V2 reaching 28.65% (a 0.48 percentage point drop). Since both decreases are within our 3 percentage point threshold, our optimizations preserve the answer quality for FetaTab while achieving substantial runtime improvements.

Regarding the results of V1 and V2 with Llama3-70b and Pixtral-Large, we see that Llama3-70b demonstrates the strongest overall performance across all configurations, achieving 52.91% for V1 and 53.73% for V2. This represents a substantial improvement over V1 and V2 with Gemini-1.5-Flash and shows V2 with a modest 0.82 percentage point advantage over V1. In comparison, Pixtral-Large shows relatively modest performance with 26.06% for V1 and 27.93% for V2, with V2 outperforming V1 by 1.87 percentage points. However, unlike Llama3-70b, the F1 scores of Pixtral-Large for both V1 and V2 are worse than that of Gemini-1.5-Flash.

Moreover, it's noticeable that V2 consistently outperforms V1 across all LLM backbones, suggesting that V2 optimization (page-level chunking) is beneficial for FetaTab. And the reason for this is similar to the explanations provided in Section **??** for SciGraphQA.

Finally, based on all the observations above, the best performing combination for FetaTab is Llama3-70b with V2, achieving 53.73% F1 score.

Summary of Performance

The comprehensive F1 score evaluation across all five VisDoMBench datasets reveals complex relationships between optimization strategies, document types, and LLM backbones.

Specifically, baseline performance with Gemini-1.5-Flash varies dramatically from 4.70% F1 score (SlideVQA) to 29.13% F1 score (FetaTab), indicating that some datasets are inherently more challenging for multimodal question answering using Gemini-1.5-Flash than others. Additionally, the optimization strategies (V1 and V2) show inconsistent effects on answer quality preservation over different datasets.

Furthermore, for the two extended LLM backbones (Llama3-70b and Pixtral-Large), Llama3-70b always performs the best with V1 and V2 across all the datasets compared to all other LLM backbones, with F1 scores ranging from 5.64% to 53.73%. In contrast, Pixtral-Large shows variable results. For instance, it outperforms Gemini-1.5-Flash on datasets like SPIQA, SciGraphVQA while it underperforms on others (e.g., PaperTab V2, SlideVQA V1).

In addition, we notice that V2 consistently outperforms V1 on visually-oriented datasets like

SlideVQA, SciGraphQA and FetaTab across all LLM backbones. This occurs because the page-level chunking used by V2 is more beneficial for datasets that have sparse textual content or large figures, as well as content that aligns with page boundaries, compared with the document-level chunking used by V1. Conversely, text-dense datasets like SPIQA and PaperTab benefit from document-level chunking that preserves cross-page context.

4.5 Error Analysis

4.5.1 Purpose and Overview

To supplement aggregate Word Overlap F1 scores, we conducted detailed error analysis on SlideVQA with V2 across all the three LLM backbones (Gemini-1.5-Flash [49], Llama3-70b [29, 30], and Pixtral-Large [31] [32]) to understand the types of errors made by different model architectures. As shown in the F1 scores results above, SlideVQA represents a challenging case where all models achieve relatively low F1 scores (4.66%, 5.66%, and 7.58% respectively). Therefore, through this error analysis, we categorize each incorrect response into specific categories to systematically characterize failure modes across different model architectures. Furthermore, the detailed definitions for each error type will be introduced in Section4.5.2.

The pipeline has two main steps:

- **Normalisation & Scoring**: For each question item, we recorded the following attributes: question text, model-predicted answer, gold standard answer, source file name, token lengths, and the Word-Overlap F1 scores. Items achieving $F_1 >= 50\%$ were classified as correct responses naively. All remaining items became candidates for systematic error analysis.
- **LLM Labelling**: Non-correct items are systematically labelled using GPT-4o-mini [33, 34]. To ensure reproducibility and auditability, the model's outputs are strictly constrained to a JSON schema format, eliminating subjective free-form reasoning and enabling transparent error categories. However, manual spot-checking revealed some inaccuracies in the automated labeling process. Therefore, the error analysis results should be interpreted as preliminary insights into error patterns rather than definitive error classifications..

4.5.2 Error Type Definitions and Examples

- **Correct but Verbose/Paraphrase** This category includes responses that contain or paraphrase the reference answer, and sometimes append irrelevant clauses, inflating response length and reducing precision scores despite accurate core content. These predictions are semantically correct or contain the exact gold answer, but receive low Word-Overlap F1 scores for stylistic reasons rather than substantive errors. Furthermore, the responses may use different wording such as synonyms or quantified restatements of qualitative claims, which reduce token overlap even tough the semantic meaning remains equivalent. By capturing these stylistically penalized but semantically correct cases, this category effectively distinguishes metric artifacts from genuine reasoning or grounding failures.

Example (1).
Q: Referring to the figure that illustrates the impact of increasing $\beta 2$ on precision and recall in the

salient object detection model optimized by the FLoss function, how does increasing $\beta 2$ influence the trade-off between precision and recall?
Gold: Increasing the value of $\beta 2$ decreases the precision and increases the recall of the model.
Pred: Increasing $\beta 2$ in the salient object detection model optimized by the FLoss function results in a trade-off where recall improves at the expense of precision, as the model prioritizes reducing false negatives over false positives.
Word Overlap F1 (%): 29.27

Example (2).
Q: Based on the camera-ablation study figure in the HUMBI dataset paper, how does the accuracy of the garment reconstruction change with varying numbers of cameras?
Gold: The accuracy of the garment reconstruction increases as the number of cameras used increases.
Pred: The accuracy of garment reconstruction in the HUMBI dataset improves with an increasing number of cameras, with optimal accuracy achieved at 60 cameras and optimal density at 90 cameras, indicating that fewer cameras can achieve similar accuracy to the full 107-camera setup.
Word Overlap F1 (%): 28.57

Example (3).
Q: Based on Table 3, how does the model's use of tone as a covariate lead to distinct word associations for anti-immigration and pro-immigration perspectives within the same topics, such as contrasting d̈etaineesv̈ersus c̈riminal?̈
Gold: The model captures different perspectives on immigration by highlighting contrasting words associated with the same topic, depending on whether the tone is anti-immigration or pro-immigration.
Pred: The model's incorporation of tone as a covariate enables it to differentiate between anti-immigration and pro-immigration perspectives within the same topics by associating distinct words that reflect the tonal nuances of each stance, such as d̈etaineesv̈ersus c̈riminalf̈or negative tones.
Word Overlap F1 (%): 23.73

- **Empty Prediction**: The system produced no answer (or only whitespace).

Example (1).
Q: How does Figure 9 in the paper *Disentangling Language and Knowledge in Task-Oriented Dialogs* illustrate the transformation of point-of-interest properties between the original and pre-processed SMD Navigate datasets?
Gold: The pre-processed SMD Navigate data combines all the properties (such as distance, address) of a point of interest (POI) into a single subject with the object being p̈oi.̈ The original data had separate entries for each property.
Pred: ""
Word Overlap F1 (%): null

Example (2).
Q: What insights does Figure 7 provide about the network's shift in focus from generating masks to producing sharper and more realistic images as the training epochs progress in the unsupervised holistic image generation framework?
Gold: he network initially focuses on predicting a good mask. As the epoch increases, the input parts become sharper. Finally, the network concentrates on generating realistic images.
Pred: ""
Word Overlap F1 (%): null

Example (3).
Q: Based on the benchmark figure in the FALCON paper, how does increasing the number of output

classes affect the setup and online times for the Softmax function in the context of Fourier Transform-based computations with homomorphic encryption?
Gold: The setup and online time for the Softmax increases as the number of classes increases.
Pred: ""
Word Overlap F1 (%): null

- **Insufficient Evidence**: This category includes the predictions where the model explicitly states that it cannot answer from the provided evidence and does not present a concrete candidate answer.

Example (1).
Q: What year is printed on the t-shirt the man is wearing in the photograph in which he is holding a serving plate full of what gives a utility of $+10$ from consuming 1 assuming standard consumption economics?
Gold: 2007
Pred: Unfortunately, it's not possible to provide an answer to this question without more information, specifically the photograph mentioned in the question.
Word Overlap F1 (%): 0.0

Example (2).
Q: What four things is the animal that the five people are staring at being compared to in the cartoon?
Gold: RUG, ROPE, SNAKE, TREE TRUNK
Pred: Unfortunately, without the image or a text description of the cartoon, it is not possible to provide a specific answer to the question. The answer would depend on the content of the cartoon, which is not provided.
Word Overlap F1 (%): 0.0

Example (3).
Q: In which region is the average order size per individual who says, "I closed the deal, didn't I" the smallest?
Gold: Southwest
Pred: The provided text does not contain the information needed to answer this question.
Word Overlap F1 (%): 0.0

- **Direction flip**: This error occurs when the predicted answer asserts the opposite trend to the reference answer, such as claiming a variable "increase" when it actually "decreases". This polarity confusion frequently occurs in figure interpretation tasks where understanding visual trends is crucial.

Example (1).
Q: Referring to the figure that illustrates the impact of increasing $\beta 2$ on precision and recall in the salient object detection model optimized by the FLoss function, how does increasing $\beta 2$ influence the trade-off between precision and recall?
Gold: Increasing the value of $\beta 2$ decreases the precision and increases the recall of the model.
Pred: Increasing $\beta 2$ shifts the trade-off toward higher precision at the expense of recall.
Word Overlap F1 (%): 45.45

Example (2).
Q: What trend is observed in the test accuracy of different models on the CUB-200-2011 dataset as λ increases, based on the logarithmic scale in the left plot of the figure from the Pairwise Confusion

paper?
Gold: The test accuracy of all models decreases as λ increases.
Pred: As λ increases on a logarithmic scale, the test accuracy of different models on the CUB-200-2011 dataset exhibits a general trend of increasing or stabilizing, as observed in the left plot of the figure from the Pairwise Confusion paper.
Word Overlap F1 (%): 34.15

Example (3).
Q: As depicted in the right panel of the figure in the "Distillation for Membership Privacy" (DMP) paper, how does the generalization gap change with increasing average X_{ref} entropy?
Gold: The generalization gap increases as the average X_{ref} entropy increases.
Pred: As depicted in the right panel of the figure in the "Distillation for Membership Privacy" (DMP) paper, the generalization gap decreases with increasing average X_{ref} entropy, indicating improved model generalization to unseen data as the entropy of the reference data increases.
Word Overlap F1 (%): 32.56

- **Numeric off-by-many**: This error captures cases where the predicted number differs from the reference by two substantial numerical error rather than a near-miss. Moreover, this error point to fundamental failures in numerical comprehension, such as systematic miscounting, significant digit misrecognition, or deeper misunderstanding of quantitative relationships in the visual or textual content.

Example (1).
Q: How does the F1-score of the model change when trained and tested on two-view versus three-view data, as demonstrated in the scalability analysis figure showing precision, recall, and F1-score metrics?
Gold: The performance of the model generally improves as the number of views increases. For example, when the model is trained and tested on two-view data, the F1-score is 29.67. However, when the model is trained and tested on three-view data, the F1-score increases to 30.2. This suggests that the model is able to learn more effectively from data with more views.
Pred: The F1-score increases from 0.91 to 0.95.
Word Overlap F1 (%): 13.56
Explanation: F1-score values are significantly different.

Example (2).
Q: In the figure comparing the performance of various models on the NIPS Abstracts dataset, as presented in the paper S̈entence Ordering and Coherence Modeling using Recurrent Neural Networks,ḧow does the proposed model's accuracy compare to other models, and what specific accuracy does it achieve?
Gold: The proposed model has the highest accuracy on the NIPS Abstracts dataset, with an accuracy of 51.55.
Pred: The proposed model (RNN + Attn + LM) achieves the highest accuracy of 64.1%, outperforming all other computational models.
Word Overlap F1 (%): 35.71
Explanation: Accuracy values are significantly different.

Example (3).
Q: Based on Table 4, how does the classification accuracy of the discriminator in the REAT method compare to the conventional discriminator in AL, and what specific numerical evidence supports this comparison?
Gold: The discriminator in the author's approach achieves higher accuracy (95.72%) compared to

the conventional discriminator in AL (94.01%).
Pred: The classification accuracy of the discriminator in the REAT method is 92.7%, which is 7.4% higher than the 85.3% accuracy of the conventional discriminator in AL, demonstrating a significant improvement in performance.
Word Overlap F1 (%): 39.02
Explanation: Predicted accuracy is significantly lower than gold.

- **Generic or Off-Topic**: This error captures cases where the model produces verbose, templated, or high-level responses that contain little specific content relevant to the question, resulting very low lexical overlap with the reference answer. This category helps isolate instances where models produce hallucinated summaries or padding text instead of providing the requested atomic answer.

  Example (1).
  Q: Based on Table 1, how do Seq2Sick's innovations in search strategy (group lasso and gradient regularization) and targeted keyword attack compare to the other RNN-based attack methods listed?
  Gold: Seq2Sick differs from existing attack methods in two key aspects:
  1. Search Strategy: While previous methods primarily rely on greedy search, which becomes increasingly inefficient for longer sequences, Seq2Sick employs group lasso regularization and projected gradient descent with gradient regularization. This allows for simultaneous searching of all replacement positions, leading to improved efficiency.
  2. Targeted Attack Type: Existing methods focus on targeting specific classes or binary classifications, while Seq2Sick introduces a novel k̈eywordẗarget type, allowing attacks to be directed towards specific keywords within the generated sequence.
  Pred: Seq2Sick achieves superior or equivalent success rates (99.8-99.9%) compared to other RNN-based methods while maintaining a more efficient perturbation rate than the similarly successful TextFooler.
  Word Overlap F1 (%): 9.25
  Explanation: Prediction discusses success rates instead of innovations.

  Example (2).
  Q: Based on the figure illustrating the training setup for 3D mesh prediction in the HUMBI dataset, how does the decoder refine or process the intermediate outputs from the regression network to generate the final 3D mesh model of human body expressions?
  Gold: The decoder is responsible for generating the final 3D mesh from the intermediate representations produced by the regression network.
  Pred: The decoder uses a graph convolutional network (GCN) with residual blocks to progressively refine vertex positions through spectral graph convolutions and nonlinear activations.
  Word Overlap F1 (%): 11.11
  Explanation: Prediction does not directly answer the question.

  Example (3).
  Q: In the Projection Block of the Grassmann Network architecture, as illustrated in the figure, what is the role of the ReOrth Layer in re-orthogonalizing the output from the FRMap Layer?
  Gold: The ReOrth Layer re-orthogonalizes the output of the FRMap Layer.
  Pred: It performs a QR decomposition to project the full-rank matrix onto the Stiefel manifold, yielding an orthonormal basis for Grassmann manifold computations.
  Word Overlap F1 (%): 0
  Explanation: Prediction does not match the gold answer's focus.

- **Entity or attribute mismatch**: This error occurs when the model refers to an incorrect entity, attribute, or data column despite producing fluent and grammatically correct text, such as identifying the wrong object, colour, axis, datas series, or category. This category specifically targets grounding, retrieval, and selection failure where the model demonstrates adequate language generation capabilities but incorrectly identifies or extracts the relevant information from the source material.

  Example (1).
  Q: Based on the data presented in Table 1 of the B̈DD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learningp̈aper, which object category has the highest combined total of bounding box and instance track annotations in the BDD100K MOT dataset?
  Gold: Cars have the largest total number of annotations.
  Pred: Person.
  Word Overlap F1 (%): 0
  Explanation: Prediction identifies wrong object category.

  Example (2).
  Q: Based on the ablation study results in Table 1 of the Devon paper, which specific architectural modification led to the largest increase in end-point error, rising to 15.64, on the KITTI 2015 dataset?
  Gold: Removing the normalization in the relation modules had the most significant negative impact on performance for the KITTI 2015 dataset.
  Pred: The removal of cross-attention in the decoder.
  Word Overlap F1 (%): 9.52
  Explanation: Prediction discusses a different architectural change.

  Example (3).
  Q: How does Figure 9 in the paper *Disentangling Language and Knowledge in Task-Oriented Dialogs* illustrate the transformation of point-of-interest properties between the original and pre-processed SMD Navigate datasets?
  Gold: The pre-processed SMD Navigate data combines all the properties (such as distance, address) of a point of interest (POI) into a single subject with the object being p̈oi.̈ The original data had separate entries for each property.
  Pred: Figure 9 illustrates the transformation by showing a reduction in the number of unique values for each POI property after normalization.
  Word Overlap F1 (%): 20.41
  Explanation: Prediction discusses unique values, not property transformation.

- **Other(uncategorised)**: This category serves as a residual bucket for cases where no specific error type can be assigned with sufficient confidence, functioning as the default fallback when all other classification checks fail. By maintaining this default category, the classification remains high-precision and mutually exclusive, ensuring that every error receives a label while avoiding focused classifications that could compromised the integrity of more specific categories.

  Example (1).
  Q: In Figure 1(f) of the paper, which illustrates thematic divergence, how does the argument structure of the AMR parsing tree for Ï like eatingd̈iffer from that of Ï like grapes?̈
  Gold: The parsing tree for Ï like eatingḧas only one argument, while the parsing tree for Ï like grapesḧas two arguments.
  Pred: For Ï like grapes,̈ the AMR tree has a flat structure with ARG1 as

SlideVQA-V2	correct($F1 > 50\%$)	verbose/ paraphrase correct	empty pred.	insufficient evidence	direction flip	numeric off-by-many	generic or off-topic	entity/attribute mismatch	other
Gemini-1.5-Flash	18	41	**0**	**464**	7	4	0	17	0
Llama3-70b	2	**122**	0	67	63	**157**	3	**137**	0
Pixtral-Large	3	56	**442**	0	3	13	2	32	0

Table 4.16: Error Analysis Results for SlideVQA with V2 Configuration Across LLM Backbones

Word Overlap F1 (%): 37.50
Explanation: Prediction is incomplete.

.

4.5.3 Results

In Table 4.16, we present the distribution of responses from the three LLMs for SlideVQA across different error categories. This analysis reveals distinct behavioral patterns across the three LLM backbones when processing SlideVQA with V2 configuration. Specifically, first, Gemini-1.5-Flash demonstrates highly conservative behavior, with the vast majority of responses (464/551, 84%) classified as insufficient evidence. This suggests the Gemini model frequently determines that neither the visual nor textual pipelines provided adequate information for combination. The cause of this problem could be retrieval limitations, conflicting evidence between pipelines, or conservative combination strategies. Additionally, the 41 verbose/paraphrased correct responses show that when accounting for semantically correct but differently worded or verbose answers, the true performance accuracy of Gemini-1.5-Flash on SlideVQA is $(18 + 41)/551 \times 100\% = 10.71\%$, which is more than double its F1 score (4.66%) reported in the last section.

Moreover, regarding the results for Llama3-70b, its true performance accuracy is approximately 22.5% ($(2 + 122)/551 \times 100\%$) compared to its reported F1 score of only 5.66%. This demonstrates how word overlap metrics can severely underestimate the performance of systems that produce detailed, analytical responses. However, they are also many responses that fall in to the numeric off-by-many (157 responses) and entity/attribute mismatch (137 responses) categories. This suggests that Llama3-70b may be more willing than Gemini-1.5-Flash to attempt answers even when evidence is limited, potentially leading to these types of factual errors.

Finally, Pixtral-Large exhibits the most problematic combination behavior with 453/551 (82%) empty predictions. This suggests the model frequently fails to execute the combination process entirely, possibly due to difficulties parsing the complex prompt structure or inability to handle conflicting evidence from visual and textual pipelines. However, despite this high failure, when accounting for the 3 correct responses and the 56 verbose/paraphrased correct responses, the true performance accuracy of Pixtral-Large on SlideVQA should be $(3 + 56)/551 \times 100\% = 10.71\%$.

4.5.4 Summary of Error Analysis

Through the analysis above, the key discovery was that Word Overlap F1 scores severely underestimate actual performance. Specifically, when accounting for verbose or differently worded but semantically correct responses, the true accuracy rates were 22.5% for Llama3-70b, 10.71%

for both Gemini-1.5-Flash and Pixtral-Large, compared to their F1 scores of 5.66%, 4.66%, and 7.58% respectively. This demonstrates that evaluation metrics based on exact word matching fail to capture the performance of systems that produce analytical responses through multi-modal evidence combination.

4.6 Summary

This comprehensive evaluation assessed the effectiveness of optimization strategies for the VisDoMRAG multimodal retrieval-augmented generation system through runtime analysis, chunk generation analysis, answer quality measurement, and error pattern examination on SlideVQA across three LLM backbones.

Overall, the evaluation demonstrates that VisDoMRAG optimization requires dataset-specific approaches rather than universal solutions.

Specifically, runtime improvements were consistently achieved across all configurations, with V1 and V2 strategies transforming processing times from hours to minutes through parallelization and architectural changes. Next, the chunk generation analysis revealed how document-level chunking (V1) and page-level chunking (V2) perform differently across various document types. In particular, page-level chunking proved superior for datasets with sparse textual content or large visual figures where content aligns with page boundaries, like SlideVQA, while document-level chunking benefited text-dense materials requiring cross-page context preservation, like SPIQA. Consequently, this explains why V1 and V2 optimizations show varying effectiveness across different datasets.

Then, through the answer quality analysis, we see that V1 and V2 both preserve the answer quality when running with Gemini-1.5-Flash compared to the baseline with the same LLM backbone. And we also see that V2 works better on datasets SlideVQA, SciGraphQA, and FetaTab with all the LLM backbones than V1, while V1 workes better on SPIQA and PaperTab. Therefore, optimization strategies must be chosen based on dataset characteristics.

Moreover, for the selection of LLM backbones, we notice that Llama3-70b consistently performs best across most datasets. However, for SlideVQA, Pixtral-Large achieves the best results. Thus, the selection of LLM backbones also depends on the type and characteristics of the dataset.

Finally, the error analysis on SlideVQA with V2 optimization across the three LLM backbones revealed that Word Overlap F1 scores significantly underestimate system performance by penalizing correct but either verbose or paraphrased answers. This shows that our system's performance on SlideVQA is not as bad as the reported F1 scores suggest. However, this analysis also highlights future work opportunities for deeper investigation of SlideVQA challenges, such as replacing the BGE-1.5 textual retriever with BM25, as the original VisDoM paper (Table 2.3) indicated that BM25 performs better on SlideVQA.

Chapter 5

Related Work

Recent advances in multimodal document understanding have increasingly explored various architectural and methodological approaches to effectively process and reason over documents containing both textual and visual elements. Specifically, these approaches can be broadly categorized into three main directions: first, multimodal RAG architectures that focus on how to retrieve and process cross-modal information; second, advanced reasoning frameworks that emphasize systematic question decomposition and iterative retrieval; and third, information organization strategies that structure multimodal evidence at the indexing level. Consequently, the following sections examine how these different approaches relate to and complement VisDoM's [48] dual-pipeline architecture with consistency-constrained fusion.

5.1 Multimodal RAG Architectures

- **M3DocRAG** [8]. M3DocRAG develops a versatile multimodal RAG system that effectively handles various scenarios including closed and open domains, single and multi-hop reasoning, and different evidence formats such as text, charts, and images. Specifically, the approach centers on a multimodal retriever combined with an LLM that maintains visual information throughout the entire process, thereby allowing retrieval across multiple pages without converting everything to text. In contrast to VisDoM [48], which uses separate visual and textual pipelines that are subsequently fused with a consistency constraint, M3DocRAG instead employs a single unified multimodal retriever designed for broader applications. Furthermore, as future work, M3DocRAG serves as a strong baseline for several key research directions: (i) learning a single retriever that returns cross-modal evidence; (ii) conducting ablation studies on whether parallel (VisDoM-style) versus unified (M3DocRAG-style) retrieval yields better precision on visually dense slides and tables; and (iii) testing how retrieval scale (single versus many documents) affects answer faithfulness.

- **VLMT** [25]. VLMT introduces a unified Vision-Language Multimodal Transformer that directly incorporates visual tokens into a sequence-to-sequence language model's embedding space without requiring additional projection layers, while using three-stage pretraining to effectively align cross-modal representations. Subsequently, it implements a two-stage multimodal QA pipeline with a multimodal reranker for retrieval and a QA component for grounded generation. In contrast to VisDoM [48], which takes a system-focused approach with dual RAG pipelines and inference-time fusion, VLMT instead emphasizes model-centric pre-trained fusion within the backbone architecture itself. Moreover, as an ablation study, there is potential to compare VLMT-style token-level fusion (single backbone) versus VisDoM-style late fusion (dual pipelines) on VisDoMBench scenarios with charts and tables to quantify the benefit of pretraining versus modularity in future research.

- **SimpleDoc** [18]. SimpleDoc offers a streamlined multimodal document visual question answering framework that enhances page collection through dual-cue page retrieval: first, it performs initial vector similarity-based retrieval, then it filters and re-ranks the results using page summaries. Subsequently, a single vision-language model iteratively calls the retriever and updates working memory until confident answers are achieved. In contrast, while VisDoM uses explicit consistency constraints to fuse parallel pipelines, SimpleDoc instead prioritizes retrieval efficiency and summary-aware re-ranking as its core approach. Furthermore, this approach inspires an engineering baseline for VisDoMRAG [48], suggesting the addition of a summary-guided filter in front of VisDoM's visual retriever to reduce chart and figure noise while maintaining recall.

5.2 Advanced Reasoning Frameworks

- **Chain-of-Action (CoA)** [35]. Chain-of-Action (CoA) presents a planner-executor framework for multimodal, retrieval-enhanced question answering, specifically designed to minimize hallucinations and enhance compositional reasoning. In particular, the method breaks down complex queries into reliable "actions" that systematically alternate between tool usage (retrieval) and grounded text generation, while incorporating explicit mechanisms to ensure answer consistency with retrieved information. In contrast to VisDoM [48], which focuses on fusing visual and textual evidence, CoA instead addresses how the LLM conducts step-by-step reasoning and utilizes tools effectively. Moreover, integrating a CoA-style controller on top of VisDoM's dual pipelines could potentially (a) improve verifiability, (b) reduce cross-modal drift, and (c) yield auditable chains that cite page-level evidence.

- **Doc-React** [55]. Doc-React addresses multi-page documents containing mixed content (images and text), specifically arguing that standard single-turn RAG fails to retrieve sufficient fine-grained context. Consequently, the model continuously proposes and refines sub-queries, re-ranks evidence, and employs an LLM for both evaluation and generation to effectively balance information acquisition with uncertainty reduction. In contrast to VisDoM's [48] concurrent visual-text retrieval followed by fusion, Doc-React's innovation instead lies in its iterative retrieval process that is optimized through mutual information approximation. Furthermore, in the context of VisDoMRAG, Doc-React suggests adding an entropy-aware retrieval controller that keeps pulling new pages and tables until uncertainty drops below a threshold, which is particularly useful in long reports or slide decks where critical facts are scattered.

- **MDocAgent** [14]. MDocAgent approaches document understanding through multi-agent collaboration, where specialized agents (general, critical, text, image, and summarizing) work together to effectively retrieve multimodal context and create comprehensive answers. In contrast to VisDoM's [48] two-pipeline fusion approach, MDocAgent instead distributes responsibilities across agents to systematically specialize retrieval and reasoning by modality and task type. Moreover, this suggests a variant of our work that keeps VisDoM's dual retrievers but introduces an "evidence critic" agent to adjudicate conflicts (e.g., chart vs. paragraph) before final fusion, potentially reducing inconsistent cross-modal attributions.

5.3 Information Organization and Indexing

- **MMRAG-DocQA** [13]. MMRAG-DocQA creates a hierarchical indexing system with flattened within-page chunks and topological cross-page connections to effectively capture correlations across modalities and pages, specifically targeting hallucinations and cross-page fragmentation in lengthy documents. In contrast, rather than addressing consistency at inference time like VisDoM, MMRAG-DocQA instead tackles the issue during indexing by systematically structuring multi-level evidence for coherent cross-modal and cross-page retrieval. Furthermore, a promising hybrid for our work is to: (i) adopt MMRAG-style hierarchical/multi-granularity indexing to pre-wire inter-page relations; (ii) keep VisDoM's [48] dual pipelines; and (iii) study whether structured indices reduce the need for heavy inference-time fusion heuristics.

5.4 Summary

In conclusion, the aforementioned works reveal several promising research directions for future investigation. First, there is significant potential to integrate VisDoM-style [48] dual pipelines with CoA [35] and Doc-React [55] controllers to achieve more faithful iterative retrieval processes. Second, a valuable comparison could be conducted between model-centric VLMT [25] fusion approaches versus modular late fusion strategies to better understand their respective trade-offs. Finally, it would be worthwhile to evaluate whether hierarchical indexing methods (as proposed in MMRAG-DocQA [13]) and agentic critic systems (as demonstrated in MDocAgent [14]) can effectively reduce cross-modal conflicts when tested on VisDoMBench [48] and its extended datasets.

Chapter 6

Conclusion

This chapter summarizes our research on optimizing the VisDoMRAG [48] system performance across the VisDoMBench [48] dataset. First, we present the key findings derived from our evaluation of different configurations. Then, we provide optimal configuration recommendations based on our experimental results and analysis. Finally, we discuss the limitations of our research methodology and outline directions for future work.

6.1 Key Findings

- Firstly, both V1 and V2 optimization strategies achieved substantial runtime improvements, reducing processing times by up to 98%, while preserving answer quality within acceptable performance thresholds across all the datasets.
- Secondly, through analysis of document-level and page-level chunking approaches and the number of chunks they generated, we discovered that while page-level chunking achieves greater runtime efficiency by eliminating the metadata recovery process, its effectiveness varies significantly across document types. Specifically, page-level chunking (V2) proves more effective for datasets like SlideVQA, SciGraphVQA, and FetaTab, while document-level chunking (V1) performs better for text-dense datasets like SPIQA and PaperTab.
- Thirdly, we expanded the system by incorporating two additional LLM backbones (Llama3-70b [29, 30] and Pixtral-Large [31] [32]) beyond the original VisDoMRAG implementation. The results demonstrate that Llama3-70b with V1 and V2 consistently achieves superior performance across most datasets. However, SlideVQA presents a notable exception, where Pixtral-Large with V2 configuration outperforms other models. Additionally, Pixtral-Large shows variable performance across different dataset-optimization combinations, with some configurations performing worse than Gemini-1.5-Flash.
- Finally, through error analysis of SlideVQA with the V2 configuration, we discovered that Word-Overlap F1 [41] scores significantly underestimate system performance by penalizing verbose or differently worded responses that are semantically correct, highlighting fundamental limitations of the Word-Overlap F1 [41] metric. Additionally, the analysis revealed that

many responses indicate insufficient evidence in the retrieved content, suggesting potential challenges in the retrieval stage of the pipeline.

6.2 Optimal Configuration Recommendations

Based on our comprehensive evaluation across five datasets and multiple optimization strategies, we present the best-performing configurations for each dataset:

- **SPIQA**: Llama3-70b with V1 configuration (31.15% Word-Overlap F1 [41])
- **SlideVQA**: Pixtral-Large with V2 configuration (7.58% Word-Overlap F1 [41])
- **SciGraphVQA**: Llama3-70b with V2 configuration (27.13% Word-Overlap F1 [41])
- **PaperTab**: Llama3-70b with V1 configuration (16.71% Word-Overlap F1 [41])
- **FetaTab**: Llama3-70b with V2 configuration (53.73% Word-Overlap F1 [41])

These recommendations provide practical guidance for deploying optimized multimodal RAG systems based on document characteristics.

6.3 Limitations

- Our error analysis relied on automated labeling, which carries acknowledged accuracy limitations that may affect the reliability of our specific error categorization findings. However, despite these methodological constraints, the core discovery regarding Word-Overlap F1 [41] metric limitations remains valid and significant.
- The evaluation was necessarily restricted to three LLM backbones. Given the rapid pace of development in language model architectures, newer models may exhibit different optimization preferences that could potentially alter our conclusions about effective model selection strategies.

6.4 Future Work

- **SlideVQA-Specific Improvements:** Our error analysis revealed that LLMs consistently fail to obtain adequate supporting information for SlideVQA queries, leading to widespread answer refusal and consequently poor model performance on this dataset. To address this issue, we propose conducting deeper error analysis on both textual and visual responses generated for SlideVQA. Furthermore, since the original VisDoM paper [48] demonstrates that BM25 serves as a more effective textual retriever for SlideVQA, we should evaluate our pipeline using this alternative retrieval method to determine whether performance improvements can be achieved.
- **Model Expansion and Architecture Refinement:** Future iterations should incorporate more advanced LLMs to evaluate whether newer architectures can better leverage the optimized pipeline configurations we identified. Additionally, building upon the related work

discussed earlier, we should explore alternative enhancement strategies for the VisDoMRAG pipeline, potentially including improved multimodal fusion techniques, enhanced retrieval mechanisms, and more sophisticated answer generation approaches.

Chapter 7

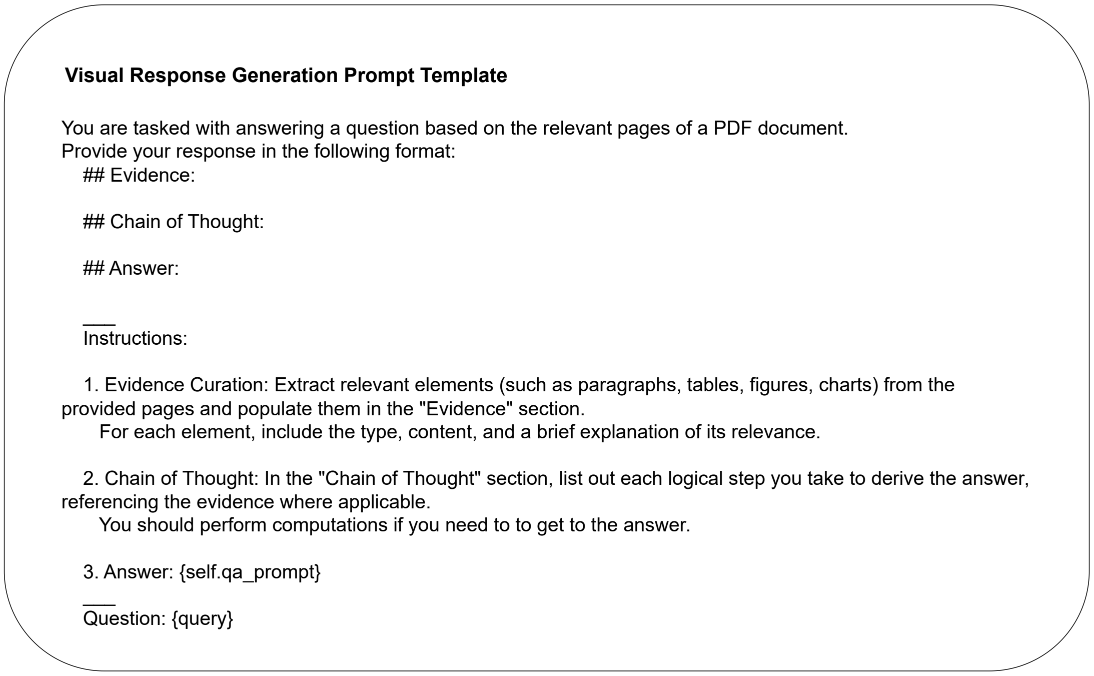
Appendices

7.1 Prompt Template

The following figures (7.1, 7.2, 7.3) show the prompt templates for visual, textual and combined responses generation.

7.2 GenAI Assistance Claim:

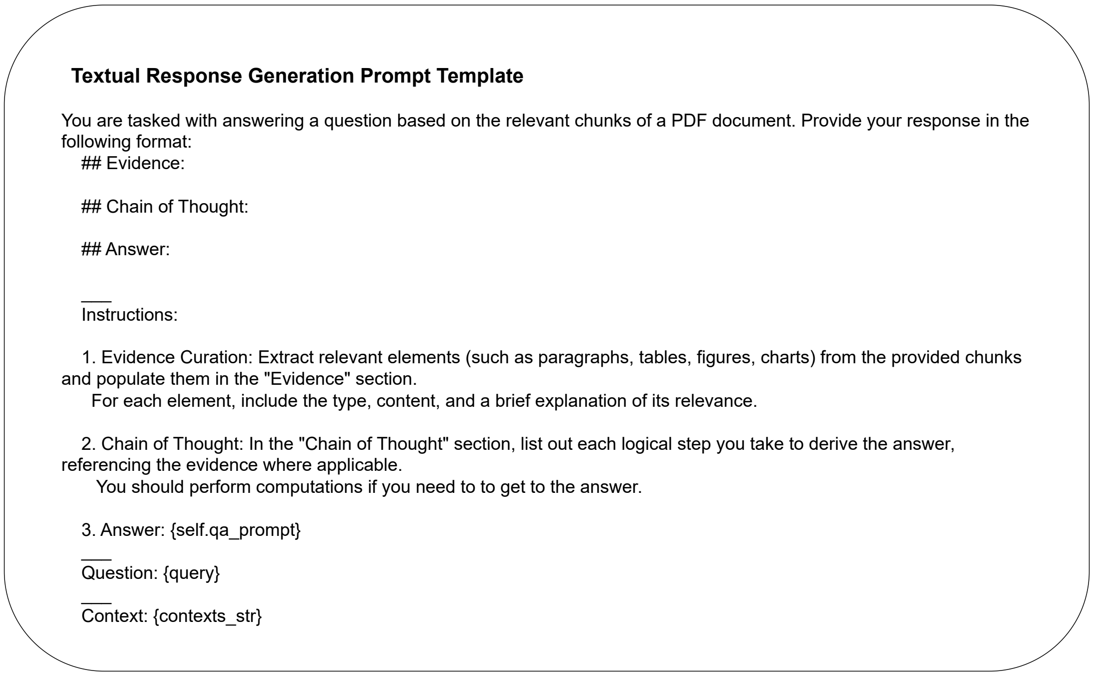
This research involved the use of generative artificial intelligence tools, specifically Claude(Sonnet 4) and ChatGPT-5, for the following purposes: (1) assistance with academic writing structure and clarity, and (2) code debugging support for system implementation. All research concepts, experimental methodology, data collection, analysis, and interpretation of results are entirely the original work of the author. The core contributions regarding ViSDoMRAG optimization strategies, performance analysis, and research findings were developed independently without AI assistance.



Visual Response Generation Prompt Template

```
You are tasked with answering a question based on the relevant pages of a PDF document.
Provide your response in the following format:
    ## Evidence:

    ## Chain of Thought:

    ## Answer:

    ___
    Instructions:

    1. Evidence Curation: Extract relevant elements (such as paragraphs, tables, figures, charts) from the
provided pages and populate them in the "Evidence" section.
        For each element, include the type, content, and a brief explanation of its relevance.

    2. Chain of Thought: In the "Chain of Thought" section, list out each logical step you take to derive the answer,
referencing the evidence where applicable.
        You should perform computations if you need to to get to the answer.

    3. Answer: {self.qa_prompt}
    ___
    Question: {query}
```

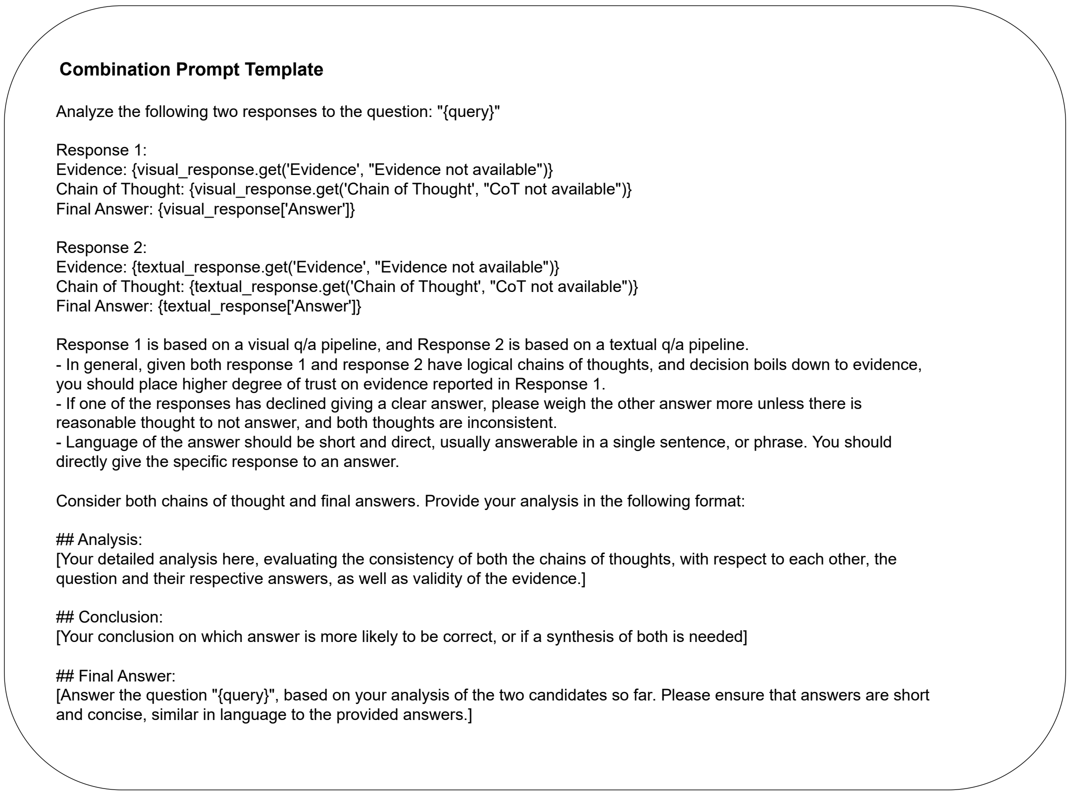
Figure 7.1: Prompt Template for Visual Response Generation



Textual Response Generation Prompt Template

```
You are tasked with answering a question based on the relevant chunks of a PDF document. Provide your response in the
following format:
    ## Evidence:

    ## Chain of Thought:

    ## Answer:

    ___
    Instructions:

    1. Evidence Curation: Extract relevant elements (such as paragraphs, tables, figures, charts) from the provided chunks
and populate them in the "Evidence" section.
      For each element, include the type, content, and a brief explanation of its relevance.

    2. Chain of Thought: In the "Chain of Thought" section, list out each logical step you take to derive the answer,
referencing the evidence where applicable.
        You should perform computations if you need to to get to the answer.

    3. Answer: {self.qa_prompt}
    ___
    Question: {query}
    ___
    Context: {contexts_str}
```

Figure 7.2: Prompt Template for Textual Response Generation



Combination Prompt Template

```
Analyze the following two responses to the question: "{query}"

Response 1:
Evidence: {visual_response.get('Evidence', "Evidence not available")}
Chain of Thought: {visual_response.get('Chain of Thought', "CoT not available")}
Final Answer: {visual_response['Answer']}

Response 2:
Evidence: {textual_response.get('Evidence', "Evidence not available")}
Chain of Thought: {textual_response.get('Chain of Thought', "CoT not available")}
Final Answer: {textual_response['Answer']}

Response 1 is based on a visual q/a pipeline, and Response 2 is based on a textual q/a pipeline.
- In general, given both response 1 and response 2 have logical chains of thoughts, and decision boils down to evidence,
you should place higher degree of trust on evidence reported in Response 1.
- If one of the responses has declined giving a clear answer, please weigh the other answer more unless there is
reasonable thought to not answer, and both thoughts are inconsistent.
- Language of the answer should be short and direct, usually answerable in a single sentence, or phrase. You should
directly give the specific response to an answer.

Consider both chains of thought and final answers. Provide your analysis in the following format:

## Analysis:
[Your detailed analysis here, evaluating the consistency of both the chains of thoughts, with respect to each other, the
question and their respective answers, as well as validity of the evidence.]

## Conclusion:
[Your conclusion on which answer is more likely to be correct, or if a synthesis of both is needed]

## Final Answer:
[Answer the question "{query}", based on your analysis of the two candidates so far. Please ensure that answers are short
and concise, similar in language to the provided answers.]
```

Figure 7.3: Prompt Template for Combined Response Generation

Bibliography

[1] Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-Eval: Instituting Standardized Evaluation for Long Context Language Models. `https://doi.org/10.48550/arXiv.2307.11088` arXiv:2307.11088 [cs.CL]

[2] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. arXiv:2308.12966 `https://arxiv.org/abs/2308.12966`

[3] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. arXiv:2308.14508 [cs.CL] `https://arxiv.org/abs/2308.14508`

[4] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. 2024. PaliGemma: A versatile 3B VLM for transfer. arXiv:2407.07726 [cs.CV] `https://arxiv.org/abs/2407.07726`

[5] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, et al. 2021. Improving language models by retrieving from trillions of tokens. arXiv:2112.04426 [cs.CL] `https://arxiv.org/abs/2112.04426`

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] `https://arxiv.org/abs/2005.14165`

[7] Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W. Cohen. 2022. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. arXiv:2210.02928 [cs.CL] `https://arxiv.org/abs/2210.02928`

[8] J. Cho et al. 2024. M3DocRAG: Multi-modal Retrieval is What You Need for Multi-page Multi-document Understanding. *arXiv preprint arXiv:2411.04952* (2024).

[9] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. arXiv:1409.1259 [cs.CL] `https://arxiv.org/abs/1409.1259`

[10] Chroma Team. 2025. Chroma. `https://docs.trychroma.com/`

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] `https://arxiv.org/abs/1810.04805`

[12] Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. ColPali: Efficient Document Retrieval with Vision Language Models. arXiv:2407.01449 [cs.IR] `https://arxiv.org/abs/2407.01449`

[13] Z. Gong et al. 2025. MMRAG-DocQA: A Multi-Modal Retrieval-Augmented Generation Method for Document Question-Answering with Hierarchical Index and Multi-Granularity Retrieval. *arXiv preprint arXiv:2508.00579* (2025).

[14] Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. MDocAgent: A Multi-Modal Multi-Agent Framework for Document Understanding. *arXiv preprint arXiv:2503.13964* (2025). `https://github.com/aiming-lab/MDocAgent`

[15] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TAPAS: Weakly Supervised Table Parsing via Pre-training. arXiv:2004.02349 [cs.IR] `https://arxiv.org/abs/2004.02349`

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] Gautier Izacard and Edouard Grave. 2020. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. arXiv:2007.01282 [cs.CL] `https://arxiv.org/abs/2007.01282`

[18] Chelsi Jain, Yiran Wu, Yifan Zeng, Jiale Liu, Shengyu Dai, Zhenwen Shao, Qingyun Wu, and Huazheng Wang. 2025. SimpleDoc: Multi-Modal Document Understanding with Dual-Cue Page Retrieval and Iterative Refinement. *arXiv preprint arXiv:2506.14035* (2025).

[19] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. arXiv:2305.06983 [cs.CL] `https://arxiv.org/abs/2305.06983`

[20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG] `https://arxiv.org/abs/2001.08361`

[21] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906 [cs.CL] `https://arxiv.org/abs/2004.04906`

[22] Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. 2021. Mind the Gap: Assessing Temporal Generalization in Neural Language Models. arXiv:2102.01951 [cs.CL] `https://arxiv.org/abs/2102.01951`

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL] `https://arxiv.org/abs/2005.11401`

[24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. arXiv:2301.12597 [cs.CV] `https://arxiv.org/abs/2301.12597`

[25] Qi Zhi Lim, Chin Poo Lee, Kian Ming Lim, and Kalaiarasi Sonai Muthu Anbananthen. 2025. VLMT: Vision-Language Multimodal Transformer for Multimodal Multi-hop Question Answering. *arXiv preprint arXiv:2504.08269* (2025).

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. arXiv:2304.08485 [cs.CV] `https://arxiv.org/abs/2304.08485`

[27] Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, et al. 2024. MMLONGBENCH-DOC: Benchmarking Long-Context Document Understanding with Visualizations. arXiv:2407.01523 [cs.CL] `https://arxiv.org/abs/2407.01523`

[28] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. arXiv:2212.10511 [cs.CL] `https://arxiv.org/abs/2212.10511`

[29] Meta AI. 2024. *Introducing Meta Llama 3.* `https://ai.meta.com/blog/meta-llama-3/` Overview and release of Meta-Llama-3-70B-Instruct; accessed 2025-08-26.

[30] Meta AI. 2024. Meta-Llama-3-70B-Instruct — Model Card. https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct. Weights, license, and usage notes; accessed 2025-08-26.

[31] Mistral AI. 2024. Pixtral Large. https://mistral.ai/news/pixtral-large. Official announcement; accessed 2025-08-26.

[32] Mistral AI. 2025. *Models Overview.* `https://docs.mistral.ai/getting-started/models/models_overview/` Official documentation for model IDs and the `-latest` alias; accessed 2025-08-26.

[33] OpenAI. 2024. GPT-4o mini: Advancing Cost-Efficient Intelligence. Blog post. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ July 18, 2024.

[34] OpenAI. 2024. GPT-4o System Card. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[35] Z. Pan et al. 2024. Chain-of-Action: Faithful and Multimodal Question Answering through Large Language Models. *arXiv preprint arXiv:2403.17359* (2024).

[36] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases? arXiv:1909.01066 [cs.CL] https://arxiv.org/abs/1909.01066

[37] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2010.08191 [cs.CL] https://arxiv.org/abs/2010.08191

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV] https://arxiv.org/abs/2103.00020

[39] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. OpenAI technical report. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[40] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. OpenAI technical report. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Austin, Texas, 2383–2392. https://doi.org/10.18653/v1/D16-1264

[42] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. 2021. Measuring Attribution in Natural Language Generation Models. arXiv:2112.12870 [cs.CL] https://arxiv.org/abs/2112.12870

[43] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? arXiv:2002.08910 [cs.CL] https://arxiv.org/abs/2002.08910

[44] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3) (NIST Special Publication, Vol. 500-225)*, Donna K. Harman (Ed.). National Institute of Standards and Technology, Gaithersburg, MD, 109–126. https://trec.nist.gov/pubs/trec3/t3_proceedings.html

[45] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. arXiv:2305.15294 [cs.CL] https://arxiv.org/abs/2305.15294

[46] Ray Smith. 2007. An Overview of the Tesseract OCR Engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

[47] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. arXiv:2004.09297 [cs.CL] https://arxiv.org/abs/2004.09297

[48] Manan Suri, Puneet Mathur, Franck Dernoncourt, Kanika Goswami, Ryan A. Rossi, and Dinesh Manocha. 2024. VisDoM: Multi-Document QA with Visually Rich Elements Using Multimodal Retrieval-Augmented Generation. arXiv:2412.10704 [cs.CL] https://arxiv.org/abs/2412.10704

[49] Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530 [cs.CL] https://arxiv.org/abs/2403.05530

[50] Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2022. Hierarchical multimodal transformers for Multi-Page DocVQA. arXiv:2212.05935 [cs.CV] https://arxiv.org/abs/2212.05935

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

[52] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. arXiv:2409.12191 https://arxiv.org/abs/2409.12191

[53] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957 [cs.CL] https://arxiv.org/abs/2002.10957

[54] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL] https://arxiv.org/abs/2201.11903

[55] Junda Wu, Yu Xia, Tong Yu, Xiang Chen, Sai Sree Harsha, Akash V. Maharaj, Ruiyi Zhang, Victor Bursztyn, Sungchul Kim, Ryan A. Rossi, Julian McAuley, Yunyao Li, and Ritwik Sinha. 2025. Doc-React: Multi-page Heterogeneous Document Question-answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vienna, Austria, 67–78. https://doi.org/10.18653/v1/2025.acl-short.6

[56] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2023. C-Pack: Packed Resources For General Chinese Embeddings. arXiv:2309.07597 [cs.CL] https://arxiv.org/abs/2309.07597

[57] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. arXiv:2007.00808 [cs.IR] https://arxiv.org/abs/2007.00808

[58] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2019. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. arXiv:1912.13318 [cs.CL] https://arxiv.org/abs/1912.13318

[59] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2020. LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding. arXiv:2012.14740 [cs.CL] https://arxiv.org/abs/2012.14740

[60] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, et al. 2024. Qwen2 Technical Report. arXiv:2407.10671 [cs.CL] https://arxiv.org/abs/2407.10671

[61] Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, Qian Qi, Ji Zhang, and Fei Huang. 2023. mPLUG-DocOwl: Modularized Multimodal Large Language Model for Document Understanding. arXiv:2307.02499 [cs.CL] https://arxiv.org/abs/2307.02499

[62] Lei Zhang, Yunshui Li, Ziqiang Liu, Jiaxi Yang, Junhao Liu, Longze Chen, Run Luo, and Min Yang. 2023. Marathon: A Race Through the Realm of Long Context with Large Language Models. arXiv:2312.09542 [cs.CL] https://arxiv.org/abs/2312.09542