# SuperRAG: Beyond RAG with Layout-Aware Graph Modeling

**Jeff Yang**[1], **Duy-Khanh Vu**[1], **Minh-Tien Nguyen**[2]*, **Xuan-Quang Nguyen**[1], **Linh Nguyen**[1], **Hung Le**[3]

[1]Cinnamon AI, 10th floor, Geleximco building, 36 Hoang Cau, Dong Da, Hanoi, Vietnam.
{jeff.yang, klein, albert, linh}@cinnamon.is
[2]Hung Yen University of Technology and Education, Hung Yen, Vietnam.
tiennm@utehy.edu.vn
[3]Deakin University, Australia.
thai.le@deakin.edu.au

## Abstract

This paper introduces layout-aware graph modeling for multimodal RAG. Different from traditional RAG methods that mostly deal with flat text chunks, the proposed method takes into account the relationship of multimodalities by using a graph structure. To do that, a graph modeling structure is defined based on document layout parsing. The structure of an input document is retained with the connection of text chunks, tables, and figures. This representation allows the method to handle complex questions that require information from multimodalities. To confirm the efficiency of the graph modeling, a flexible RAG pipeline is developed using robust components. Experimental results on four benchmark test sets confirm the contribution of the layout-aware modeling for performance improvement of the RAG pipeline.

## 1 Introduction

Retrieval Augmented Generation (RAG) (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022; Izacard et al., 2023) is a new paradigm that helps to reduce the hallucination of large language models (LLMs) (Cao et al., 2020; Raunak et al., 2021; Ji et al., 2023) by providing additional contexts for prompting LLMs (Su et al., 2021; Chen et al., 2024). Recently, the approach has gained considerable attention due to its effectiveness in enhancing the capabilities of LLMs (Guu et al., 2020; Lewis et al., 2020; Su et al., 2021; Xiao et al., 2021; Borgeaud et al., 2022; Izacard et al., 2023). Within this domain, graph-based RAG has emerged, introducing a novel perspective that leverages structured knowledge to improve further performance and interpretability (Panda et al., 2024; Besta et al., 2024; Li et al., 2024; Edge et al., 2024; Sun et al., 2024).

Unlike non-graph-based RAG methods that directly use raw data as individual chunks of text for downstream reasoning or question-answering tasks, the graph-based RAG approach can represent input data as a graph that considers the relationship among text chunks (Panda et al., 2024; Li et al., 2024; Edge et al., 2024). We argue that while most RAG-based pipelines perform effectively within the text modality, handling multimodal inputs—common in real-world business applications—poses substantial challenges to these systems, potentially limiting their broader applicability and impact. The challenge comes from two main reasons. First, input documents contain diverse layouts, structures, and multimodalities that need to be captured in a RAG pipeline. The information on the layout plays an important role, helping LLMs understand the document. Also, the document contains text, tables, and figures which should be encoded into prompts for LLMs' reasoning (Zhao et al., 2023). Second, an input question may require information in different modalities. Let's consider the question: "Please list the standard steps for creating Internet Navigware teaching materials". It requires information in the flow chart on page 27, and text on pages 28, and 29.[1]

This paper introduces a novel graph-based RAG scheme that addresses the two challenges above for actual multimodal QA cases. The pipeline includes four steps: document parsing, data modeling, advanced information retrieval, and reasoning. The document parsing can handle multiple input types using in-house and third-party readers. For data modeling, we introduce a new knowledge graph (KG) that retains the layout and structure of input documents. This is because the layout and structure are important to comprehend the meaning of input documents which enhances the performance of the information retrieval (IR) step. Data modeling in the form of a KG is combined with full-text and vector search to create an advanced IR module

*Corresponding Author.

[1]https://software.fujitsu.com/jp/manual/manualfiles/m150016/b1ww9681/07z000/tutorial.pdf



*Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Track)*, pages 544–557
April 30, 2025 

that uses re-ranking to retrieve the most relevant contexts. The combination of multiple retrievers allows the proposed pipeline to retrieve more relevant information from the contexts. The reasoning step combines an input query and the relevant contexts to form a prompt feed to an LLM for achieving the final answer. In summary, this paper makes three main contributions as follows.

- It introduces a new Layout-Aware Graph Modeling (LAGM) structure to represent input documents for RAG. The structure is created to retain the layout of input documents which is combined with full-text and vector search to improve the quality of the IR step.
- It utilizes state-of-the-art and robust techniques for building a unified RAG pipeline. Experimental results on public benchmark datasets show that the proposed SuperRAG achieves promising results compared to strong other RAG baselines.
- It offers a system where users can experience the proposed RAG pipeline (Appendix 7).

## 2 Related Work

**RAG** RAG is a new method that supports LLMs to fill the gap of out-of-date knowledge (He et al., 2022) and hallucination (Cao et al., 2020; Raunak et al., 2021; Ji et al., 2023). By using relevant information retrieved from external knowledge, RAG can help LLMs to generate more accurate and reliable responses (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022; Izacard et al., 2023; Ren et al., 2023; Shi et al., 2024). With the aid of RAG, LLMs have achieved promising results in many tasks such as code generation (Zhou et al., 2022), domain-specific QA (Cui et al., 2023; Dahl et al., 2024; Pu et al., 2024), or open-domain QA (Izacard and Grave, 2021; Trivedi et al., 2023; Kim et al., 2024; Wang et al., 2024; Yu et al., 2024).

**Graph-based RAG** The graph structure has been adapted to capture relationships among concepts such as Connected Papers tool,[2] a tree of summary nodes for long context (Chen et al., 2023), or multimodal KGs for storing text, diagrams, and source code (Kannan et al., 2020). The graph has also been used to improve the quality of RAG in different ways such as hyper-relational KG (Panda et al., 2024), graph-based agents for long contexts (Li et al., 2024), KG for summarization (Edge et al., 2024), or graph neural networks (Mavromatis and Karypis, 2024). However, we observed that most of these efforts have focused on the text modelity.

We follow the direction of building multimodal KGs for RAG (Sun et al., 2024; Wang et al., 2024). While prior works have explored hierarchical document parsing for RAG, SuperRAG differentiates by emphasizing structured granularity and document layout analysis. We introduce a modern, generalized data model, incorporating Table of Contents (ToC) and master sections to improve retrieval for large documents. These enhancements preserve document structure, enhancing retrieval accuracy and effectiveness. We also share the idea of using the structure of documents for RAG (Saad-Falcon et al., 2023); however, our method empowered by an in-house reader that can handle diverse document types with table and chart understanding rather than only processing the text structure of PDF files as Saad-Falcon et al. (2023).

## 3 Layout-Aware Graph Modeling

Layout-aware Graph Modeling (LAGM) is designed to effectively represent input documents while preserving their original layout and structure. This approach is motivated by the need to enhance the comprehensibility and manageability of property graphs, particularly for applications involving multimodal and complex data. For example, if the query asks for information in a table or chart, the RAG pipeline needs to know which section or subsection it belongs to.

### 3.1 Document Layout Parsing

The first step in constructing LAGM is parsing input documents using specialized readers for different modalities, including text, tables, diagrams, and images. This step outputs a structured format that forms the foundation for graph creation. We leverage an in-house document parser with the enhancement from Azure DI to ensure robust processing across diverse layouts.

**The In-House Document Parser** Our in-house parser is designed as a modular pipeline to process each page independently (Figure 1). It begins with a loader layer for format conversion and pre-processing, followed by AI models for extracting layouts, table structures, OCR, and figure content. The processed data undergoes post-

[2] https://www.connectedpapers.com

processing, such as reading order sorting and relation extraction, and is output in JSON/Markdown.

Key components of the in-house parser include Document Layout Analysis (DLA), reading order detection, table structure recognition, and figure and table classification. The DLA module is pre-trained on DocLayNet (Pfitzmann et al.) and further fine-tuned with 5773 in-house annotated PDF pages, enabling the model to recognize 9 distinct layout labels like titles, tables, and figures.

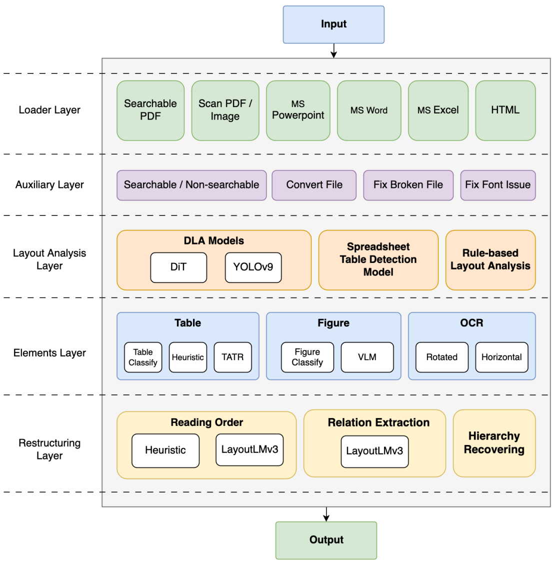


Figure 1: The pipeline of the in-house parser.

For reading order detection, the parser employs the method proposed by Wang et al. (2021), leveraging 5010 annotated document images to extract natural reading sequences. Table structure recognition is implemented using an in-house library designed to identify diverse table formats accurately. Lastly, figure and table classification rely on a curated dataset to categorize tables into subtypes (e.g., full-lined, borderless) and figures into specific types (e.g., charts, diagrams), ensuring precise extraction of visual elements. Table 1 reports

Table 1: Document reading performance.

| Methods | NID | TEDS | TEDS-S |
|---|---|---|---|
| Amazon Textract | 96.71 | 88.05 | 90.79 |
| LlamaParse | **92.82** | 74.57 | 76.34 |
| Unstructured | 91.18 | 65.56 | 70.00 |
| Google Layout Parser | 90.86 | 66.13 | 71.58 |
| Azure DI | 87.69 | 87.19 | 89.75 |
| Our reader DI | 92.43 | **89.76** | **91.14** |

the comparison of the in-house reader with orther strong reading methods. **NID** stands for Normalized Indel Distance for layout and order reading. **TEDS** is Tree Edit Distance-based Similarity for text and table structure recognition. TEDS-S is Tree Edit Distance-based Similarity-Struct for table structure recognition only. We can observe that the in-house reader achieves competitive results which are good to implement actual RAG pipelines.

**Azure DI for PDF Parsing Enhancement** Azure DI enhances the parser by excelling in section-header and paragraph detection. It supports searchable and non-searchable PDFs and aids in creating ToC. To generate the ToC, we use Azure DI outputs for tables, sections, and diagrams, performing the following: (1) Match physical and printed page numbers. (2) Detect ToC based on keywords. (3) Replace printed page numbers with physical page numbers. This integration ensures superior layout-aware graph modeling and improves ToC generation for structured navigation.

### 3.2 Data modeling

After parsing, each document page can be decomposed into title, header, sections, text chunks, tables and diagrams, etc. The data modeling step aims to create a granular-level design for the property graph. Figure 2 shows the definition of LAGM.

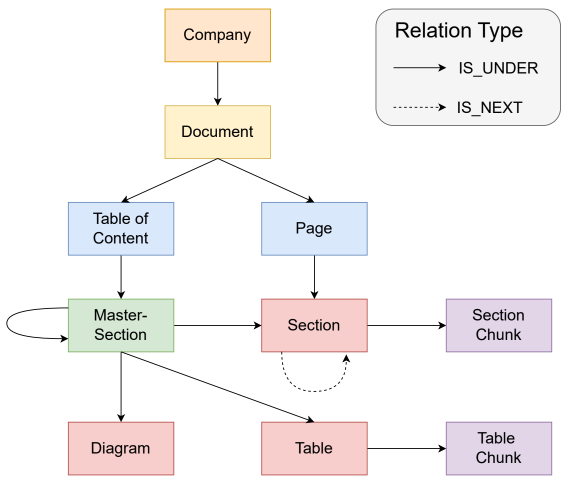


Figure 2: The knowledge graph used for data modeling.

The **Company** node serves as the root, representing the overarching entity or corpus, such as a company, and capturing metadata like the company's name. Each **Document** node, linked to the Company, represents an individual document with attributes such as document name, type, and path.

Documents connect to **Page** nodes, which represent individual pages and include attributes like page index, headers, footers, and textual content.

The **TableOfContents** node, also linked to Document, provides a structural overview of the document and connects to **MasterSection** nodes. MasterSections organize the content hierarchically and link to **Section**, **Table**, and **Diagram** nodes.

**Section** nodes represent logical divisions within a document and include attributes like section headers and content. Sections are connected sequentially via "`has_next`" relationships, ensuring the flow of content. They can also link to finer-grained **SectionChunk** nodes, capturing texts under the section. **Table** nodes, representing tabular data, and **Diagram** nodes, representing visual elements, provide additional structure. Tables may be further connected **TableChunk** nodes for storing textual contents inside the table. These explicit "`is_under`" and "`has_next`" relationships reflect the natural hierarchy and flow of documents. This design supports layout-aware graph modeling and efficient information retrieval, enhancing applications like RAG pipelines by enabling precise navigation and knowledge extraction.

### 3.3 The SuperRAG Framework

Building on layout-aware graph modeling (LAGM), we introduce an advanced retrieval expansion framework combining LLM-based and heuristic-driven approaches for flexible and efficient information retrieval. This framework enhances RAG-based pipelines by improving adaptability and scalability across applications.

**LLM-Based Graph Traversal.** This approach leverages a Large Language Model (LLM) to perform context-aware graph traversal. Using the graph schema (visualized in Fig. 2) as input, the LLM dynamically generates Cypher queries, enabling intelligent and relationship-driven retrieval. It is particularly effective for complex, multimodal data and intricate document structures encoded in the graph. Detailed information of the prompt for the LLM is mentioned at the end of the appendix.

**Heuristic-Based Retrieval.** Complementing the LLM-based approach, the framework processes ToC, tables, and diagrams as heuristics for IR enhancement. For ToC, the framework uses structured output from the LLM with prompt engineering (Fig. 4) and heuristics to extract the ToC during indexing. This is because ToC contains important structured information for retrieval. During retrieval, it computes semantic similarity scores between section titles and the query for targeted content retrieval. Additionally, few-shot prompting is used to ask the LLM to directly extract the relevant page based on a given query. For table processing, the DETR model (Carion et al., 2020) for table detection and recognition is used, followed by an OCR engine to reconstruct the table structure before indexing. This ensures that tables are accurately captured and searchable within the SuperRAG pipeline. For diagram processing, OCR models are used to extract text from diagrams and feed both images and text information into a multimodal LLM (e.g., GPT-4o) for better interpretation. This allows context-aware understanding of visual elements, ensuring better integration of diagrams in retrieval and reasoning. These methods are computationally efficient, effective, and robust for dealing with structured content.

**Comparative Insights.** The dual framework balances flexibility and efficiency, with LLM-based traversal excelling in unstructured, exploratory tasks, and heuristics providing predictable performance for high-throughput systems. Together, they enable scalable and adaptive RAG pipelines, leveraging graph structures for optimal retrieval.

### 3.4 Graph Augmentation

To enrich the LAGM, we employ the $K$-Nearest Neighbors (KNN) (Cover and Hart, 1967) as a graph augmentation technique to create new `is_similar` relationships between nodes within the graph. The KNN algorithm calculates similarity between nodes based on their properties, using metrics such as cosine similarity, Jaccard similarity, or Euclidean distance, depending on the data type. Also, `has_stem` relationships are generated using synonyms or words sharing the same stem, linking nodes representing conceptually related terms.

## 4 Applications

Figure 3 shows the pipeline of LAGM that integrates multiple retrievers and re-rankers, combining heuristic graph traversal, similarity search, and language model-based techniques for efficient retrieval and ranking. The pipeline is flexible in several ways. First, it merges cross-page context using the graph representation. Second, a TOC retriever is included for documents with structured information, improving context quality for specific queries. Additionally, the pipeline uses diagram/table expansion for queries needing information from tables and diagrams, with a self-reflection layer to

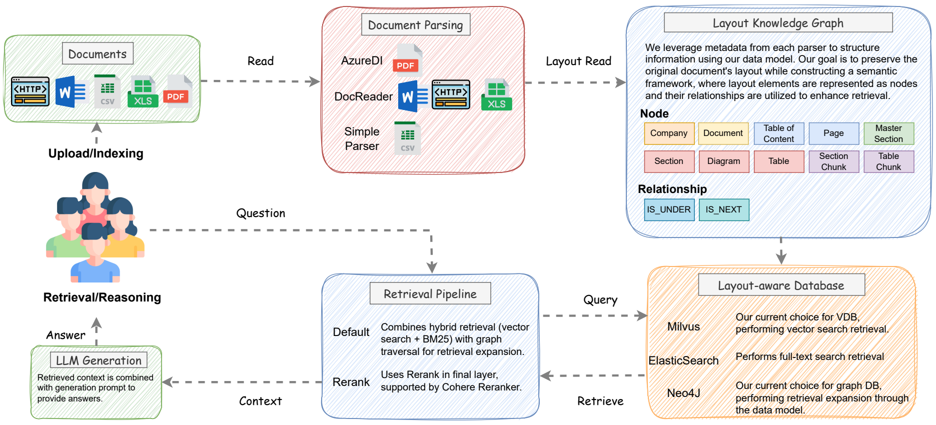


Figure 3: The proposed SuperRAG framework.

evaluate whether table or diagram information is necessary based on the query intent. It selectively integrates these elements only when they contribute to a more accurate answer, reducing irrelevant content retrieval. Notably, LAGM is pipeline-agnostic and can integrate into any RAG pipeline.

## 5 Experimental Settings

### 5.1 Datasets

We examine the following datasets for evaluation.

**DOCBENCH** is a benchmark designed to evaluate LLM-based document reading systems (Zou et al., 2024). It features 1,102 questions and 229 PDF documents from five domains: academia, finance, government, laws, and news, with an average of 66 pages and 46,377 tokens per document.

**SPIQA** includes 27K research papers in three tasks: direct QA with figures and tables, direct QA with full papers, and CoT QA. The evaluation contains test-A (666 filtered questions), test-B (228 human-written questions from QASA), and test-C (493 from QASPER), all emphasizing reasoning with figures and tables.

### 5.2 Detailed Implementation

Milvus was used as a vector database. ElasticSearch was used for full-text search. Neo4J was implemented as a graph database. The embedding model uses embedding-v3-large from Open AI. LLM for completion uses GPT-4o with version 2024-05-01. The hyper-parameters include selecting the top 3 tables and diagrams, the top 20 for relevant contexts, and the top 10 for re-ranking.

### 5.3 Evaluation Metrics

All models were assessed using a GPT-4-based evaluator, which has demonstrated a 98% agreement with human annotators, ensuring robust and reliable accuracy measurement (Zou et al., 2024).

## 6 Results and Discussion

This section first reports the performance comparison of SuperRAG with other strong RAG-based methods, and then shows the ablation study, output observation. It finally describes the demo system.

### 6.1 Performance on RAG Tasks

**Layout-aware vs. non-layout-aware** The first comparison includes two settings: layout-aware and non-layout-aware. The layout-aware approach leverages document structure—such as headers, tables, figures, and sections—to provide contextual cues that are often critical for accurately understanding and retrieving information across varied domains. In contrast, the non-layout-aware model only uses Hybrid Search for IR with a flat structure.

The first part of Table 2, and Table 3, demonstrate that layout-aware modeling significantly enhances performance across domains and tasks. On DOCBENCH, the layout-aware model achieves an average accuracy of 75.8%, outperforming the non-layout model's 68.5% by 7.3 points. Notably, in academia and finance, gains are 11.9 and 9.8 points, respectively, showing the value of structural cues in complex documents. On SPIQA in Table 3, the layout-aware model improves Test-A accuracy by 4.5 points (59.% vs. 55.4%) and Test-B by 1.3 points (63.1% vs. 61.8%). In the challenging Test-C, it achieves an average accuracy gain of 9 points (57.2% vs. 48.2%), with notable im-

Table 2: The comparison on DOCBENCH.

| System | Aca | Fin | Gov | Laws | News | Text | Multi | Meta | Una | Avg. Acc |
|---|---|---|---|---|---|---|---|---|---|---|
| Layout-aware vs. non-layout-aware data modeling | | | | | | | | | | |
| Non-layout | 64.0 | 70.1 | 64.2 | 62.8 | 83.7 | 77.7 | 74.4 | 46.1 | 70.2 | 68.5 |
| Layout-aware | **75.9** | **79.9** | **71.6** | **65.4** | 83.7 | **84.7** | **85.1** | **50.4** | **75.8** | **75.8** |
| Layout-aware vs. SOTA RAG methods | | | | | | | | | | |
| GPT4 (API) | 65.7 | 65.3 | 75.7 | 69.6 | 79.6 | **87.9** | 74.7 | 50.8 | 37.1 | 69.8 |
| GPT-4o (API) | 56.4 | 56.3 | 73.0 | 65.5 | 75.0 | 85.0 | 62.7 | 50.4 | 17.7 | 63.1 |
| KimiChat (Web) | 62.4 | 61.8 | **77.0** | 78.5 | 87.2 | 87.6 | 65.3 | 50.4 | 71.8 | 70.9 |
| Claude 3 Opus (Web) | 73.9 | 40.6 | 70.3 | **79.1** | **86.6** | 80.8 | 64.6 | **54.3** | 58.9 | 67.6 |
| SuperRAG (Ours) | **75.9** | **79.9** | 71.6 | 65.4 | 83.7 | 84.7 | **85.1** | 50.4 | **75.8** | **75.8** |

provement in table handling. These results confirm layout awareness as a key factor in improving contextual understanding and retrieval accuracy.

Table 3: Layout-aware vs. non-layout-aware on SPIQA Test-B and Test-C. ColPali is used for Qwen 2B, 7B, Claude, and GPT-o4.

| System | Figure | Table | Avg. Acc |
|---|---|---|---|
| Test-A | | | |
| Non-layout | 53.9 | 57.2 | 55.4 |
| Layout-aware | **57.4** | **63.7** | **59.9** |
| Test-B | | | |
| Non-layout | 62.4 | 61.0 | 61.8 |
| Layout-aware | **66.1** | **58.9** | **63.1** |
| Test-C | | | |
| Non-layout | 57.5 | 44.6 | 48.2 |
| Layout-aware | **58.2** | **56.7** | **57.2** |

**Comparison with SOTA methods** The proposed data modeling was compared to state-of-the-art RAG methods. On DOCBENCH, we compare our method against state-of-the-art LLM-based document reading systems, including proprietary pipelines like GPT-4, KimiChat, and Claude-3. For SPIQA, since the benchmarked results only measure baseline QA performance using full gold context without including the IR component of the RAG system, a direct comparison would be unfair. To address this, we reran several strong baselines using a full IR pipeline instead of relying on reported numbers from original papers. Additionally, we evaluated ColPali (Faysse et al., 2024), an open-source retrieval model that generates contextualized embeddings from document page images, contrasting with our layout-focused method.

As shown in the second part of Table 2 and Table 4, our approach SuperRAG consistently outperforms other systems across both DOCBENCH and SPIQA benchmarks. On DOCBENCH, SuperRAG achieves the highest overall accuracy (75.8%), particularly excelling in the Financial and multi-type questions. In comparison, proprietary systems like GPT-4 and KimiChat perform strongly in specific categories, but their overall accuracies fall short by at least 6% compared to our method. Notably, SuperRAG's ability to handle a wide range of question types, especially complex multi-type and unatype questions, highlights its superior document comprehension capabilities.

Table 4: The performance on SPIQA Test-B and Test-C. ColPali is used for Qwen 2B, 7B, Claude-3.5 Sonnet.

| System | Figure | Table | Avg. Acc |
|---|---|---|---|
| Test-A | | | |
| GPT-4o (API) | 51.6 | 54.2 | 52.7 |
| Qwen 2-7B | 48.3 | 40.5 | 45.9 |
| Claude-3.5 Sonnet | **58.1** | 56.8 | 57.6 |
| SuperRAG (Ours) | 57.4 | **63.5** | **59.9** |
| Test-B | | | |
| GPT-4o (API) | 63.1 | 53.6 | 59.2 |
| Qwen 2-7B | 41.3 | 45.2 | 42.9 |
| Claude-3.5 Sonnet | 53.3 | 44.2 | 49.5 |
| SuperRAG (Ours) | **66.2** | **58.9** | **63.2** |
| Test-C | | | |
| GPT-4o (API) | 43.1 | 40.9 | 41.5 |
| Qwen 2-7B | 40.2 | 28.5 | 31.8 |
| Claude-3.5 Sonnet | 46.0 | 42.3 | 43.4 |
| SuperRAG (Ours) | **58.2** | **56.7** | **57.2** |

For SPIQA, SuperRAG demonstrates superior performance across all three test sets, excelling in both figure and table-based QA tasks. In Test-A, it achieves the highest average accuracy (59.9%), with a notable 63.5% on table-based questions, outperforming the best baseline by 7%. For Test-B, SuperRAG again leads with an average accuracy of

63.2%, surpassing the strongest baseline Claude3.5 Sonet (49.5%). It achieves 66.2% on figure-related tasks and 58.9% on table-based tasks, showcasing balanced strengths across modalities. In Test-C, SuperRAG achieves 57.2% overall, with standout performances in both figures (58.2%) and tables (56.7%). In comparison, the runner-up Claude-3.5 Sonnet trails at 46.0%, marking a substantial gap of 12.2%. These results underscore SuperRAG's ability to handle multimodal inputs effectively, even when competing with enterprise systems.

### 6.2 Ablation Study

We investigate the flexibility of the pipeline by testing with three settings. The first setting is the non-layout method which uses the hybrid search + cross-page context merger (**1**). The second setting is the layout-aware method which uses the hybrid search + cross-page context merger + TOC integration + table-diagram expansion (**2**). The TOC integration is to extract the Table-of-Content in documents. The table-diagram expansion expands the context with tables and diagrams relevant to the input query. The final setting is also our proposed layout-aware method which is similar to the second setting but using self-reflection (**3**). Self-reflection means that the pipeline decides whether to use information from tables and diagram expansion based on the input query.

Table 5 presents the accuracy results across various settings. Our method, equipped with all functionalities, consistently achieves the highest accuracy, highlighting the effectiveness of each component in enhancing overall system performance.

Table 5: Component contribution. DOC: DOCBENCH.

| Setting | DOC | Test-A | Test-B | Test-C |
|---|---|---|---|---|
| 1 | 68.5 | 55.4 | 61.8 | 48.2 |
| 2 | 71.7 | 53.0 | 60.9 | 53.1 |
| 3 | **75.8** | **59.9** | **63.1** | **57.2** |

### 6.3 Output Observation

The performance of RAG pipelines was observed to show their behavior on raw samples. To do that, the observation was done with three methods: non-layout, layout-aware, and ColPali (using Sonnet). Tables 6 and 7 show the outputs of the three pipelines. For the first sample in Table 6, the non-layout-aware pipeline could not output correct answer. This is because it could not retrieve correct relevant context for RAG. The ColPali method gives an uncertain answer because the rank of the paper retriever page image from Colpali (topk=1 or topk=3) does not contain enough information and the reasoning capability on the image of VLM still have some disadvantage. The layout-aware gives the correct answer (retrieval information from both images (in page 2 - Reference 2 in Page 3) and text content from page 3 and another page). It shows the efficiency of the proposed layout-aware method for retrieving relevant context. For the second sample in Table 7, both layout and non-layout model are all based on the benchmark tables for accuracy data and cannot retrieve information about test errors in figure `d`. The ColPali method can not retrieve extract page contain figure `d` with top 1 or top 3. As the result, it could not output a correct answer. In this case, all the RAG pipelines could not retrieve the figure `d`. I suggests that the retrieval of visual components in documents should be improved.

### 6.4 The Demo System

Figure 5 provides an interface where users can experience the system. The right panel includes settings for uploading files, IR types, and other settings. The central panel consists of a text box for inputting queries. After putting a query, the system retrieves relevant context based on the layout-aware graph modeling and responses the final answer. The right panel provides evidence of the answer, that contains confidence scores and relevant chunks. Related information is highlighted in the relevant chunks. The open source version can be found at `https://github.com/Cinnamon/kotaemon`.

## 7 Conclusion

The paper introduces layout-aware graph modeling for multimodal data construction used by RAG. The modeling takes into account the structure of input documents for building a graph that contains the relationship among text chunks, tables, and figures. A RAG pipeline has also been developed to confirm the efficiency of the modeling. Experimental results on four public test sets show two important points. First, layout-aware modeling is beneficial for improving the performance of RAG compared to non-layout-aware and strong other RAG pipelines. Second, the designed RAG pipeline is flexible, and adding more sophisticated RAG-related components improves the performance of the system. The modeling and RAG pipeline are practical for business scenarios.

## Limitations

First, our approach relies heavily on accurate document layout parsing and high-quality data modeling. If these components are misaligned or if document structure extraction tools are limited, the pipeline's effectiveness may be reduced. In particular, noisy layouts or variations in document structures across domains could impact the quality of information retrieval (IR) and subsequently the reasoning performance of the pipeline. Moreover, integrating tables, figures, and non-text elements into a coherent graph structure may introduce computational overhead, making the pipeline resource-intensive. This can affect scalability, especially in real-world applications requiring high throughput or settings with limited computational resources.

## Ethics Statement

Our framework presents no major ethical concerns, as it has been designed with a genuine focus on improving the accuracy of information retrieval in LLM-based systems. Our method does not generate or alter content independently but instead organizes multimodal information from existing documents, ensuring that outputs remain faithful to the source material. Privacy risks are minimized by following data protection regulations and implementing strict anonymization protocols where needed, particularly for sensitive data.

## References


Maciej Besta, Ales Kubicek, Roman Niggli, Robert Gerstenberger, Lucas Weitzendorf, Mingyuan Chi, Patrick Iff, Joanna Gajda, Piotr Nyczyk, Jürgen Müller, et al. 2024. Multi-head rag: Solving multi-aspect problems with llms. *arXiv preprint arXiv:2406.05085*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. 2020. Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*.

Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E Ho. 2024. Large legal fictions: Profiling legal hallucinations in large language models. *Journal of Legal Analysis*, 16(1):64–93.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Manuel Faysse, Hugues Sibille, Tony Wu, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*.

Gautier Izacard and Édouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Amar Viswanathan Kannan, Dmitriy Fradkin, Ioannis Akrotirianakis, Tugba Kulahcioglu, Arquimedes Canedo, Aditi Roy, Shih-Yuan Yu, Malawade Arnav, and Mohammad Abdullah Al Faruque. 2020. Multimodal knowledge graph for deep learning papers and code. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3417–3420.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. *arXiv preprint arXiv:2404.13081*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. 2024. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. *arXiv preprint arXiv:2406.14550*.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, et al. 2024. Holmes: Hyper-relational knowledge graphs for multi-hop question answering using llms. *arXiv preprint arXiv:2406.06027*.

B Pfitzmann, C Auer, M Dolfi, AS Nassar, and PWJ Staar. Doclaynet: A large humanannotated dataset for document-layout analysis (2022). *URL: https://arxiv. org/abs/2206*, 1062.

Hongxu Pu, Xincong Yang, Jing Li, and Runhao Guo. 2024. Autorepo: A general framework for multimodal llm-based automated construction reporting. *Expert Systems with Applications*, page 124601.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019*.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, David Seunghyun Yoon, Ryan A Rossi, and Franck Dernoncourt. 2023. Pdftriage: Question answering over long, structured documents. *arXiv preprint arXiv:2309.08872*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8364–8377.

Yixuan Su, Yan Wang, Deng Cai, Simon Baker, Anna Korhonen, and Nigel Collier. 2021. Prototype-to-style: Dialogue generation with style-aware editing on retrieval memory. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2152–2161.

Qiang Sun, Yuanyi Luo, Wenxiao Zhang, Sirui Li, Jichunyang Li, Kai Niu, Xiangrui Kong, and Wei Liu. 2024. Docs2kg: Unified knowledge graph construction from heterogeneous documents assisted by large language models. *arXiv preprint arXiv:2406.02962*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. 2021. Layoutreader: Pre-training of text and layout for reading order detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4735–4744.

Fei Xiao, Liang Pang, Yanyan Lan, Yan Wang, Huawei Shen, and Xueqi Cheng. 2021. Transductive learning for unsupervised text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2510–2521.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, et al. 2024. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. *arXiv preprint arXiv:2410.10594*.

Ruochen Zhao, Hailin Chen, Weishi Wang, Fangkai Jiao, Xuan Long Do, Chengwei Qin, Bosheng Ding, Xiaobao Guo, Minzhi Li, Xingxuan Li, et al. 2023. Retrieving multimodal information for augmented generation: A survey. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4736–4756.

Shuyan Zhou, Uri Alon, Frank F Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig. 2022. Docprompting: Generating code by retrieving the docs. *arXiv preprint arXiv:2207.05987*.

Anni Zou, Wenhao Yu, Hongming Zhang, Kaixin Ma, Deng Cai, Zhuosheng Zhang, Hai Zhao, and Dong Yu. 2024. Docbench: A benchmark for evaluating llm-based document reading systems.

## Appendix

**Prompt for LLM-based Graph Traversal** The ToC prompt example is shown in Fig. 4.

```
DEFAULT_TABLEOFCONTENTS_TEMPLATE = ( "Assume
that you are reading a book. You have a query
and need to find the relevant lines in the
Table of Contents. \n" "Return the
corresponding lines without any
explanation.\n" "Think step by step and
return the answer in the final step only.\n"
"### Table of Contents\n"
"{table_of_content_exp}" "\n### Query\n"
"{query_exp}" "\n###Answer" "{answer_exp}"
"\n\n\n### Table of Contents\n"
"{table_of_contents}" "\n### Query\n"
"{query}" "\n###Answer" )
```

Figure 4: The proposed SuperRAG framework.

An example prompt for LLMs to generate Cypher graph queries is included at the end of the appendix (7).

**The output observation** The examples of output observation are shown in Tables 6 and 7.

**The demo system** The user interface of the system is shown in Fig. 5.

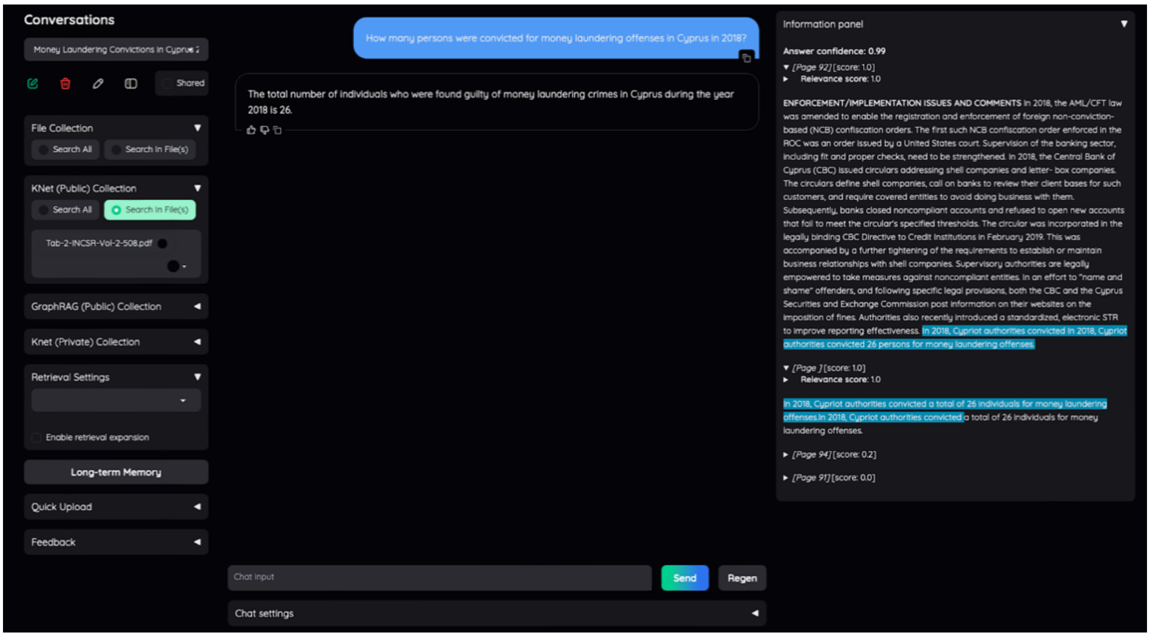


Figure 5: The demo system with the sample from the DOCBENCH dataset. The input question is *"How many persons were convicted for money laundering offenses in Cyprus in 2018?"* and the answer is *"26 persons"*.

Table 6: Output observation on two samples of SPIQA.

**Question:** Which of the two approaches, density estimation or mixture of classifiers, is more robust to outliers? (https://arxiv.org/pdf/1805.06431v4)
**Correct answer:** Mixture of classifiers
**Evidence:** The figure shows that the mixture of classifiers approach is able to correctly classify the data points even when there are outliers present. This is because the mixture of classifiers approach is able to learn the different modes of the data distribution, while the density estimation approach is not.

---

**Non-layout:** Based on the provided context, particularly from References 4, 6, and 7, the proposed method (ChoiceNet) incorporates a mixture of densities to robustly learn the underlying target distribution in the presence of noisy (corrupt) training data, whereas the mixture of classifiers is also discussed but seems to face challenges in distinguishing clean and corrupt patterns independently.
In the regression tasks and synthetic examples described in References 4 and 7, it is shown that the proposed method, which is based on a mixture density network, outperforms or shows comparable results with low outlier rates and demonstrates superior performance with high outlier rates (greater than 50%). This is evident from the comparison with other methods, including a standard mixture density network (MDN) and other robust loss methods. Specifically, Figures 6(a)-(c) in Reference 8 illustrate that the proposed method maintains lower average fitting errors across different outlier rates, and Figures 6(d)-(f) in Reference 8 show better fitting results even with 60% outliers.
On the other hand, the mixture of classifiers approach, while theoretically sound, faces practical challenges. As stated in Reference 3, the mixture of classifiers might not work well in practice because both classifiers can independently learn corrupt patterns, leading to a mixture weight adhering to 1/2 and making it hard to distinguish clean from corrupt data without additional dependencies or regularization.
Therefore, based on the empirical evidence and detailed explanations given in the provided context, the density estimation approach (specifically the proposed mixture density network method) is more robust to outliers compared to the mixture of classifiers approach.

---

**Claude 3 Opus:** Based on the information provided in the image, it's difficult to definitively state which approach - density estimation or mixture of classifiers - is more robust to outliers. The figure shows both approaches being used as part of a process for binary classification on corrupt data, but doesn't explicitly compare their robustness to outliers.
However, we can make some inferences:
1. The mixture of classifiers approach (part b of the figure) seems to separate the data points more clearly, with distinct decision boundaries (shown as red lines labeled W). This could potentially make it more robust in handling outliers or corrupt data points.
2. The density estimation approach (part a) shows overlapping distributions of data points. While this may capture the overall distribution well, it might be more susceptible to being influenced by outliers.
3. The text mentions that the mixture of classifiers approach (using $\hat{W}$) is "expected to reveal the reverse patterns by minimizing the following mixture classification loss". This suggests that this approach might be designed to better handle corrupt data or outliers.
4. The motivation section discusses how the mixture approach aims to "discriminate the corrupt data by using another linear classifier $\hat{W}$", which could imply better handling of outliers.
Given these points, there's a slight indication that the mixture of classifiers approach might be more robust to outliers. However, to make a definitive statement, we would need more specific information about how each method handles outliers, which isn't explicitly provided in the image or text.

---

**Layout-aware:** The approach described in the document for handling outliers involves two primary methods: density estimation and mixture of classifiers. Based on the provided references, the robustness to outliers can be inferred from the empirical results and the theoretical framework described.
1. **Density Estimation Approach**: - The density estimation approach leverages a mixture density network (MDN) to model the target conditional density using a parameterized distribution, typically a Gaussian distribution (Reference 3). - The robustness of this approach is evaluated in regression tasks with synthetic data and real-world datasets like the Boston Housing dataset (Reference 4). The results show that while the proposed method, built on top of an MDN, outperforms or shows comparable results with low outlier rates, it exhibits superior performance with high outlier rates (> 50%) (Reference 4). - Additionally, the MDN's performance degrades severely with extreme noise levels (60%), indicating that without dependencies among different mixture components, the MDN struggles to allocate high mixture probability correctly (Reference 4).
2. **Mixture of Classifiers Approach**: - The mixture of classifiers approach involves modeling the choice probability of different classifiers and estimating the corruption probability using the mixture weight $T_w(x)$ (Reference 2). - This method employs a linear functional with negative correlation between classifiers to help distinguish clean and corrupt patterns (Reference 3). - An illustrative example in a binary classification task using the MNIST dataset shows that ChoiceNet (which uses a mixture of classifiers) successfully infers the clean data distribution and effectively handles corrupted labels (Reference 7). - The final test accuracies reported (81.7% train accuracy and 98.1% test accuracy) suggest that this approach can robustly handle noisy labels (Reference 7).
**Comparison and Conclusion**: - The density estimation approach shows that it can handle high outlier rates, but its performance degrades significantly with extreme noise levels. - The mixture of classifiers approach, particularly as implemented in ChoiceNet, demonstrates strong performance in distinguishing clean from corrupted data and maintains high accuracy even with a significant proportion of noisy labels.
Based on the empirical results and the robustness demonstrated, **the mixture of classifiers approach appears to be more robust to outliers** compared to the density estimation approach. The mixture of classifiers method effectively identifies and manages corrupted data through its correlation-guided mechanism, leading to better handling of noisy datasets.

Table 7: Output observation of the second sample from SPIQA.

| |
|---|
| **Question:** Which model performs the best in terms of test error? (https://arxiv.org/pdf/1805.06431v4)<br>**Correct answer:** CDAN (M)<br>**Evidence:** Figure (d) shows the test error for different models. CDAN (M) has the lowest test error. |
| **Non-layout:** Based on the provided context, the model that performs the best in terms of test error is the "CDAN+E (w/o random sampling)" model. This model achieves the highest average accuracy across all transfer tasks, as shown in the table in Reference 8. Specifically, it achieves an average accuracy of 87.7%, which is higher than the other variants of CDAN+E (with Gaussian sampling and with uniform sampling).<br>The performance across specific tasks such as A -> W, D -> W, W -> D, A -> D, D -> A, and W -> A also supports this conclusion, where the CDAN+E (w/o random sampling) consistently shows superior or comparable performance in comparison to the other models. Therefore, it can be concluded that "CDAN+E (w/o random sampling)" performs the best in terms of test error among the models listed.<br>**Claude 3 Opus:** I apologize, but I don't see any information in this image about model performance or test errors. The image contains technical details about generalization error analysis, domain adaptation theory, and descriptions of various datasets used in experiments. It does not show any results or comparisons of model performance. To answer which model performs best in terms of test error, we would need to see actual experimental results or performance metrics, which are not provided in this particular image. |
| **Layout-aware:** Based on the provided context, particularly from Reference 1 (page 7), the model that performs the best in terms of test error across various datasets and tasks is the **CDAN+E** model. This can be observed from the consistently high classification accuracies reported for CDAN+E across multiple benchmark datasets when compared to other models.<br>Here is a detailed breakdown from the tables:<br>1. **Office-31 Dataset (AlexNet and ResNet-50 backbones):** - CDAN+E achieves the highest average accuracies of 77.7% (AlexNet) and 87.7% (ResNet-50).<br>2. **ImageCLEF-DA Dataset:** - CDAN+E records an average accuracy of 80.3% (AlexNet) and 87.7% (ResNet-50), again being the top performer.<br>3. **Office-Home Dataset:** - CDAN+E shows the highest average accuracy of 65.8% (ResNet-50).<br>4. **Digits Dataset (MNIST, USPS, SVHN):** - CDAN+E achieves an average accuracy of 94.3%, outperforming other methods.<br>5. **VisDA-2017 Synthetic to Real Dataset:** - CDAN+E achieves an accuracy of 70.0%, which is higher than other comparison methods.<br>Therefore, across these datasets, CDAN+E consistently outperforms other domain adaptation methods, indicating it has the lowest test error and is the best-performing model overall. |

```
CYPHER_QUERY_TEMPLATES = """You are required to construct a Cypher query to
retrieve the requested information from the graph database.  The graph schema
is provided below for reference.
{graph_schema}
Instructions for Cypher Query Generation:
1.  Schema Adherence:
- Use only the provided relationship types and properties.
2.  Response Guidelines:
- Generate a Cypher query as plain text without any additional formatting.
- Include only the Cypher statement; exclude any explanations, apologies, or
unrelated content.
3.  Conditions for Query Construction:
- Use pageIdx and parentPageIdx to identify the page.  Do not use pageNumber.
- Use the docType attribute to identify the document type.
- If docName is provided, use it to filter nodes.
4.  Handling Uncertainty:
- If unsure about the user's request or if no Cypher query is applicable,
return nothing.
5.  Things to Avoid:
- Do not generate generic queries.  If the request lacks specifics, return
nothing.
- Do not use or infer any additional relationship types or properties.
- Don't generate overly complex queries.  Keep the queries simple and focused
on the user's request.
- Don't generate keyword queries unless explicitly requested.
- Don't write queries that could return all SECTION, TABLE, or DIAGRAM nodes
from the document.
Good Examples:
---
MATCH (s)-[:S_IS_UNDER_P]->(p:PAGE)
WHERE toString(p.pageIdx) IN $pages AND s.parentDocName IN $doc_id
RETURN s;
---
Bad Examples:
---
MATCH (s:SECTION)
WHERE s.parentDocName IN ['<dir>', '<doc_name>']
RETURN s;
---
MATCH (s:SECTION)-[:S_IS_UNDER_P]->(p:PAGE)
WHERE s.parentDocName IN ['<dir>', '<doc_name>']
RETURN s;
---
User Request:  {user_request}
docName:  {doc_name}
Cypher Query (Generate a Cypher query as plain text without any additional
formatting):"""
```