NeuSym-RAG: Hybrid Neural Symbolic Retrieval with Multiview Structuring for PDF Question Answering

Ruisheng Cao[12]*, **Hanchong Zhang**[12*], **Tiancheng Huang**[12*], **Zhangyi Kang**[12], **Yuxin Zhang**[12], **Liangtai Sun**[12], **Hanqi Li**[124], **Yuxun Miao**[12], **Shuai Fan**[23], **Lu Chen**[124†] **and Kai Yu**[124†]
[1]MoE Key Lab of Artificial Intelligence, Shanghai, China
X-LANCE Lab, School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
[2]Jiangsu Key Lab of Language Computing, Suzhou, China
[3]AISpeech Co., Ltd., Suzhou, China [4]Suzhou Laboratory, Suzhou, China
{211314,htc981,narcisss,chenlusz,kai.yu}@sjtu.edu.cn

Abstract

The increasing number of academic papers poses significant challenges for researchers to efficiently acquire key details. While retrieval augmented generation (RAG) shows great promise in large language model (LLM) based automated question answering, previous works often isolate neural and symbolic retrieval despite their complementary strengths. Moreover, conventional single-view chunking neglects the rich structure and layout of PDFs, e.g., sections and tables. In this work, we propose NeuSym-RAG, a hybrid neural symbolic retrieval framework which combines both paradigms in an interactive process. By leveraging multi-view chunking and schema-based parsing, NeuSym-RAG organizes semi-structured PDF content into both the relational database and vectorstore, enabling LLM agents to iteratively gather context until sufficient to generate answers. Experiments on three full PDF-based QA datasets, including a self-annotated one AIRQA-REAL, show that NeuSym-RAG stably defeats both the vector-based RAG and various structured baselines, highlighting its capacity to unify both retrieval schemes and utilize multiple views.

1 Introduction

With the exponential growth in academic papers, large language model (LLM) based question answering (QA) systems show great potential to help researchers extract key details from emerging studies. However, individual PDFs often exceed prompt limits, and user queries may span multiple documents. To tackle these challenges, retrieval-augmented generation (RAG, Huang and Huang, 2024) is effective for knowledge-intensive QA.

Despite its wide application, the classic *neural retrieval* (Guu et al., 2020) often fails when handling precise queries involving mathematical operations, comparisons, or aggregations. For example, in the top-left of Figure 1, the total number of tables cannot be determined through retrieved chunks as they are scattered across the document. On the other hand, *symbolic retrieval* such as TAG (Biswal et al., 2024) relies on semantic parsing (Berant et al., 2013) techniques, e.g., text-to-SQL (Li et al., 2024b), to directly extract the target information from the structured database. Unfortunately, such precise queries often break down in semantic fuzzy matching or morphological variations, e.g., "*graph-based RAG*" versus "*GraphRAG*". Previous literature investigates these two paradigms in isolation.

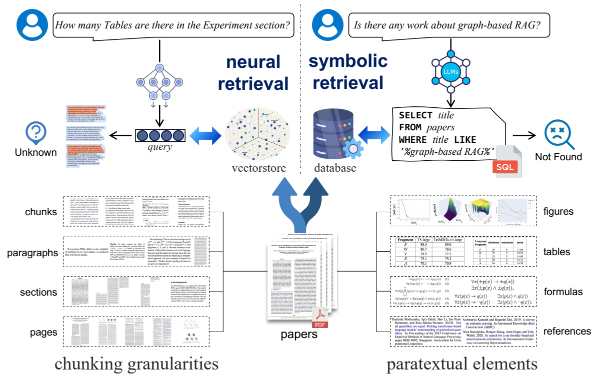

Figure 1: Motivation of the proposed NeuSym-RAG. (Upper) Two paradigms of retrieval strategies. (Bottom) PDF documents can be split based on different granularities and they contain many paratextual elements.

Furthermore, the most widely utilized scheme to segment documents into chunks is based on a fixed length of consecutive tokens, possibly considering sentence boundaries or more delicate granularities (Yang, 2023). However, for semi-structured PDF documents of research papers, this common practice neglects the intrinsic structure of sections and the salient features of paratextual tables and figures (illustrated at the bottom of Figure 1). The distinct layout of PDF files offers a more structured view towards segmenting and arranging content.

To this end, we propose a hybrid **Neu**ral **Sym**bolic retrieval framework (NeuSym-RAG) for

*Equal contribution.
†The corresponding authors are Lu Chen and Kai Yu.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6211–6239
July 27 - August 1, 2025

PDF-based question answering, which combines both retrieval paradigms into an interactive procedure. During pre-processing, the PDF file of each paper is fed into a pipeline of parsing functions (§ 2.2) to complete the metadata, segment raw texts based on different views, and extract various embedded paratextual elements (e.g., tables and figures). These identified elements are populated into a relational database, specifically designed for semi-structured PDF. Then, we select encodable column values from the populated database for storage into the vectorstore (§ 2.3). These cell values present diverse views in interpreting PDF content. Moreover, the database schema graph is leveraged to organize the vectors into a well-formed structure. To answer the user question, LLM agents can adaptively predict executable actions (§ 2.4.1) to retrieve desired information from either backend environment (database or vectorstore) in a multi-round fashion. This agentic retrieval terminates when the collected information suffices to answer the input question, in which case LLM agent will predict a terminal "GENERATEANSWER" action.

To validate NeuSym-RAG, we convert and annotate three full PDF-based academic QA datasets, including a self-curated one AIRQA-REAL with 553 samples and 18 instance-specific evaluation metrics (§ 3.1). Experiments on both closed- and open-source LLMs demonstrate that it remarkably outperforms the classic neural RAG (17.3% points on AIRQA-REAL) and various structured methods like HybridRAG (Sarmah et al., 2024) and GraphRAG (Edge et al., 2024). The ablation study highlights: 1) the positive impact of model size increase on agentic retrieval, 2) the superiority of integrating multiple chunking perspectives, and 3) the low sensitivity to LLM choice for pre-processing and high sensitivity for subjective evaluation.

To sum up, our contributions are three-fold:

- We are the first to integrate both vector-based neural retrieval and SQL-based symbolic retrieval into a unified and interactive NeuSym-RAG framework through executable actions.
- We incorporate multiple views for parsing and vectorizing PDF documents, and adopt a structured database schema to systematically organize both text tokens and encoded vectors.
- Experiments on three realistic full PDF-based QA datasets *w.r.t.* academic research validate the superiority of NeuSym-RAG over various neural and symbolic baselines.

2 Framework of NeuSym-RAG

This section presents the complete framework of NeuSym-RAG, comprising three stages (Figure 2).

2.1 The Overall Procedure

To answer user questions about a semi-structured PDF file, the entire workflow proceeds as follows:

1) **Parsing:** Firstly, we pass the raw PDF file into a pipeline of functions to segment it in multi-view, extract non-textual elements, and store them in a schema-constrained database (DB).

2) **Encoding:** Next, we identify those encodable columns in the DB, and utilize embedding models for different modalities to obtain and insert vectors of cell values into the vectorstore (VS).

3) **Interaction:** Finally, we build an iterative Q&A agent which can predict executable actions to retrieve context from the backend environment (either DB or VS) and answer the input question.

2.2 Multiview Document Parsing

At this stage, for each incoming PDF file, we aim to parse it with different perspectives into a relational database DuckDB (Mühleisen and Raasveldt, 2024). The pipeline of parsing functions includes (in the top middle of Figure 2): 1) Querying scholar APIs (e.g., `arxiv`) to obtain the metadata such as the authors and published conference, such that we can support metadata-based filtering (Poliakov and Shvai, 2024) during retrieval. 2) Splitting the text based on different granularities with tool PyMuPDF (Artifex Software, 2023), e.g., pages, sections, and fixed-length continuous tokens. 3) Leveraging OCR models to extract non-textual elements (we choose the tool `MinerU`, Wang et al., 2024a). 4) Asking large language models (LLMs) or vision language models (VLMs) to generate concise summaries of the parsed texts, tables, and images (Xu et al., 2023). The retrieved metadata, parsed elements and predicted summaries will all be populated into the symbolic DB. We handcraft the DB schema in advance, which is carefully designed and universal for PDF documents (see the middle part of Figure 2 or App. E for visualization).

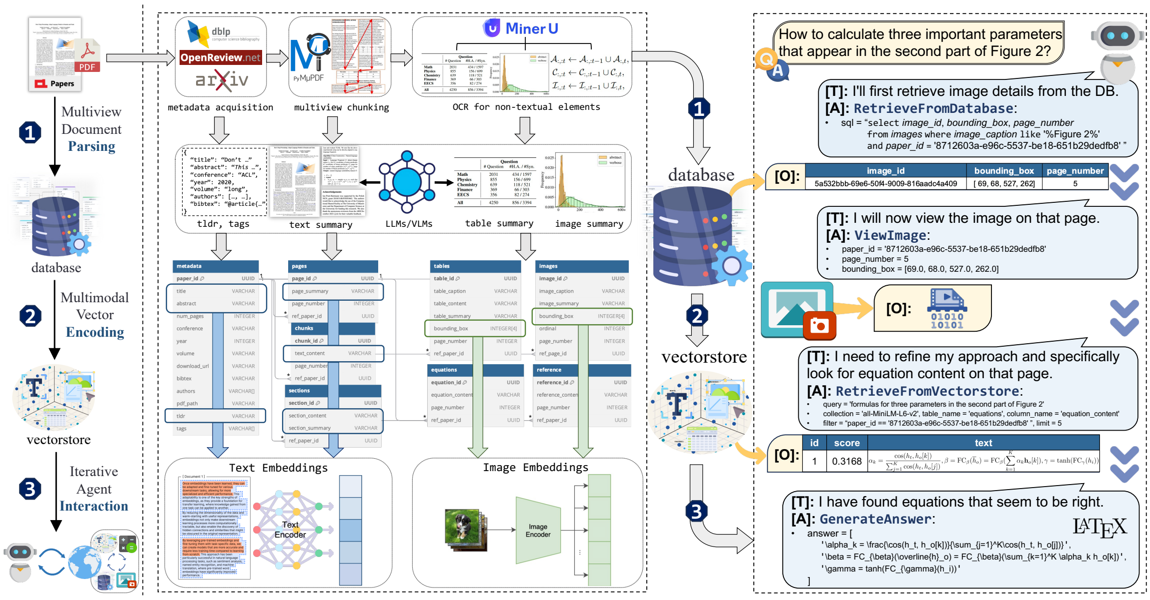

Figure 2: Overview of the NeuSym-RAG framework. The demonstration example comes from a real and simplified case in our labeled dataset (§ 3.1). "[T]", "[A]", and "[O]" represent thought, action and observation, respectively.

2.3 Multimodal Vector Encoding

After the completion of the symbolic part, we switch to the neural encoding of parsed elements. Firstly, we label each column in the DB schema as "encodable" or not. For text modality, we mark those columns of data type "`varchar`" as encodable whose cell values are relatively long (e.g., column "`pages.page_summary`" in Figure 3); while for images, the "`bounding_box`" columns which record 4 coordinates of figures or tables are recognized to extract the square region in the PDF. Next, we resort to encoding models of both modalities to vectorize text snippets or cropped images.

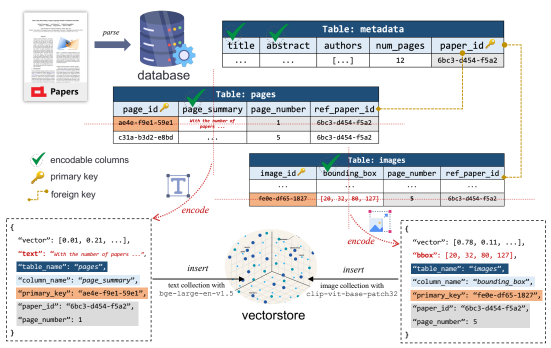

Figure 3: Illustration of how to convert an encodable cell value in the database to one data entry in the vectorstore.

To build a one-to-one mapping between each cell value in the DB and neural vector in the VS, we supplement each data point in the VS with its corresponding table name, column name, and primary key value for that row in the DB. This triplet can uniquely identify each value in the DB. Besides, we add 2 extra fields, namely "`paper_id`" and "`page_number`", into the JSON dict to enable metadata filtering. These data entries will be inserted into the VS, categorized into different collections based on the encoding model and modality [1].

Through the first two stages (§ 2.2 and § 2.3), various chunking perspectives in the VS are intrinsically connected via a structured DB schema (visualized in Figure 11). Moreover, long-form texts or visual bounding boxes in the DB are vectorized into the VS to support fuzzy semantic matching.

2.4 Iterative Agent Interaction

Now that the database and vectorstore have been populated, we can build an RAG agent which proactively retrieves context from both the DB and VS.

2.4.1 Action Space Design

Firstly, we introduce the 5 parameterized actions with arguments that agents can take during interaction, namely RETRIEVEFROMVECTORSTORE, RETRIEVEFROMDATABASE, VIEWIMAGE, CALCULATEEXPR, and GENERATEANSWER.

RETRIEVEFROMVECTORSTORE This action converts the classic static retrieval into real-time

[1] We create 4 collections in the Milvus (Wang et al., 2021) vectorstore: for text embeddings, we use BM25 (Robertson and Zaragoza, 2009), `all-MiniLM-L6-v2` (Wang et al., 2020), and `bge-large-en-v1.5` (Xiao et al., 2023), while `clip-vit-base-patch32` (Radford et al., 2021) for images.

dynamic retrieval. It has multiple arguments which are adjustable (see Lst. 1), e.g., "query" encourages LLMs to rewrite the user intention more clearly (Ma et al., 2023), while "table_name" and "column_name" request the agent to select an appropriate perspective for retrieval. During execution, the environment will retrieve context concerning "query" with specified constraints "filter" (e.g., page_number=1) from the VS and return the observation in a tabular format (see App. F.2.3).

```
RetrieveFromVectorstore(
    # user input can be rephrased
    query: str,
    # select encoding model/modality
    collection_name: str,
    # (table_name, column_name) together
        defines which view to search
    table_name: str,
    column_name: str,
    # allow fine-grained meta filtering
    filter: str = '',
    limit: int = 5
)
```

Listing 1: RETRIEVEFROMVECTORSTORE action with its parameters in function calling format.

RETRIEVEFROMDATABASE This action accepts a single parameter "sql". It operates by executing the provided SQL query against the prepared DB and fetching the resulting table to the agent. Similar to ToolSQL (Wang et al., 2024b), through iterative symbolic retrieval, agents can predict different SQLs to explore the DB and exploit multiple pre-parsed views on interpreting PDF content.

VIEWIMAGE To address potential PDF parsing errors during the first stage (§ 2.2) and leverage features of more advanced VLMs, we devise this action to extract a cropped region (defined by "bounding_box") in one PDF page and send the base64 encoded image back to agents. Notably, the concrete coordinates can be obtained through retrieval from either the DB or VS during interaction. This way, the agent will acquire the desired image as observation and then reason based on it.

```
ViewImage(
    paper_id: str,
    page_number: int,
    # 4-tuple of float numbers, if [],
       return the image of entire page
    bounding_box: List[float] = []
)
```

Listing 2: VIEWIMAGE action with its parameters in function calling format.

CALCULATEEXPR Inspired by Xu et al. (2024), we also include this action which accepts a Python "expr" (e.g., 2 + 3 * 4) and returns the calculation result. This simple integration stably reduces hallucination *w.r.t.* math problems in our pilot study.

GENERATEANSWER This is the *terminal* action that returns the final answer when agents determine that the retrieved context suffices to resolve the input question. The only parameter "answer" can be of any type depending on the user requirement. Refer to App. F.2.1 for formal action definitions.

2.4.2 Hybrid Neural Symbolic Retrieval

In each turn, agents predict one action (see App. F.2.2 for different action formats) to interact with the environment and obtain the real-time observation. We adopt the popular ReAct (Yao et al., 2023) framework, which requires agents to output their thought process first (see the right part of Figure 2). This iterative retrieval (Shao et al., 2023) enables agents to leverage complementary strengths of both retrieval paradigms in continuous steps.

For example, agents can firstly filter relevant rows by executing a SQL program in the database. RETRIEVEFROMDATABASE is particularly adept at handling structured queries and metadata-based constraints. After extracting primary key values of rows for the intermediate output, we can further narrow down the final result set by inserting those key values into the "filter" parameter of action RETRIEVEFROMVECTORSTORE to conduct neural semantic matching, i.e.,

`filter='primary_key in [pk values of sql exec]'.`

Conversely, agents can firstly query the VS via neural search to select the most relevant entries, and then treat this transitional set as a temporary table or condition in the subsequent SQL retrieval.

3 Experiment

This section introduces our human-labeled dataset AIRQA-REAL, the main experiment and ablation study. Code and data are publicly available [2].

3.1 Q&A Dataset on AI Research Papers

AIRQA-REAL Previous Q&A datasets on academic papers typically focus on simple questions based on a single page or merely the abstract of the PDF, which is far from reflecting real-world scenarios. To address this, we annotate a more complex Q&A dataset regarding full and even multiple

[2] https://github.com/X-LANCE/NeuSym-RAG

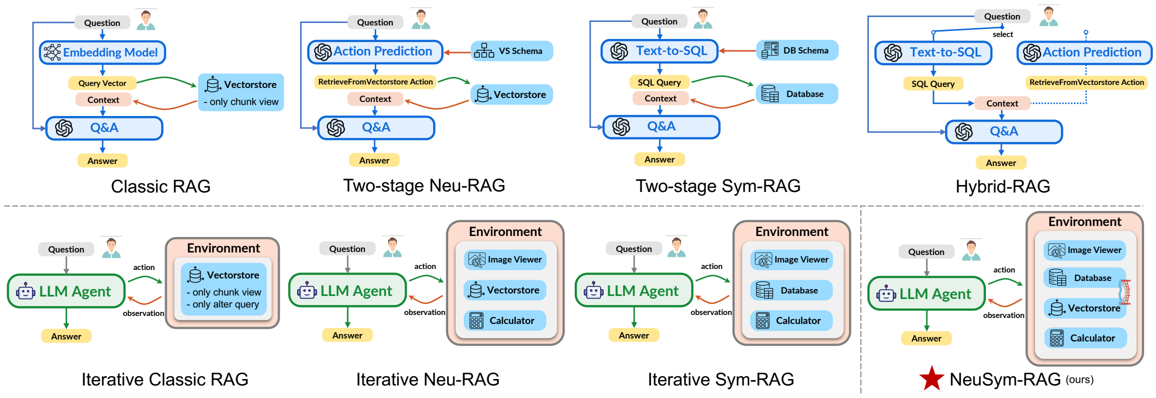

Figure 4: Comparison of different structured baselines with our NeuSym-RAG. Refer to App. C for text description.

Category	Question
text	What are the main components of ERRA model?
table	On the dataset proposed by this work, how much does the GPT-3.5-turbo model improve its GPT4score after using Graph-CoT?
image	Considering the performance of ChatDev agent on DSEval-LeetCode benchmark, what is the most common cause of the errors?
formula	How does Multi-DYLE combine the three different losses as the objective of training?
metadata	Which conference was the paper 'Fact-Checking Complex Claims with Program-Guided Reasoning' published in? Is it a long, short or findings?

Table 1: Examples of single-doc type from AIRQA-REAL dataset. See App.A.2 for the other two types.

PDFs, called AIRQA-REAL. Based on published AI papers in 2023 and 2024, 16 researchers manually annotate 553 questions which span across: a) 3 task types (i.e., single-doc details, multi-doc analysis, and paper retrieval), b) 5 categories (i.e., text, table, image, formula, and metadata), and c) 2 evaluation genres (i.e., hard-coding objective metrics and LLM-based subjective assessment). We showcase one example for each category in Table 1.

Other Benchmarks To validate the universality, we also convert two other full-PDF-based Q&A datasets, namely M3SciQA (Li et al., 2024a) and SciDQA (Singh et al., 2024). These two benchmarks have 452 and 2937 test samples, respectively. And we adjust them into the data format of AIRQA-REAL. See App. A for more details about datasets.

Evaluation Metrics While most long-form Q&A datasets utilize LLMs or human experts for accuracy (including the aforementioned M3SciQA and SciDQA), we propose more flexible and precise evaluation metrics. Specifically, we 1) adopt answer formatting similar to DABench (Hu et al., 2024), which imposes Python-style restrictions on output (e.g., "*Your answer should be a Python list of two strings.*"), and 2) implement instance-specific, execution-based evaluation by designing 18 functions with optional parameters. These functions are categorized as either subjective or objective, depending on whether they involve the judgement from LLMs (see App. D).

3.2 Experiment Settings

Baselines We implement 11 methods (partly illustrated in Figure 4), ranging from trivial input questions, to more advanced structured models. The main differences lie in: a) Single-view or Multi-view: the Classic RAG only has access to text chunks, while the others can select from multiple pre-parsed perspectives; b) Two-stage or Iterative: the major distinction between the two rows in Figure 4 is whether agentic retrieval is permitted; c) Composition of the environment: the complete backend contains relational database, vectorstore, image viewer and calculator. Different baselines may only include part of them, which further restricts the permissible actions (see App. F.2.1).

LLMs and Hyper-parameters We evaluate NeuSym-RAG with various LLMs. For closed-source ones, we use GPT-4o-mini-2024-07-18 and GPT-4-1106-vision-preview. Regarding state-of-the-art open-source LLMs, we include InternLM2.5 (Cai et al., 2024), Llama-3.3-70B-Instruct (Hugging Face Team, 2023), Qwen2.5-VL-72B-Instruct (Team, 2025), and DeepSeek-R1 (Guo et al., 2025). As for hyper-parameters, the temperature is set to 0.7 and top_p is fixed to 0.95. The maximum retrieved tokens in each turn and the cutoff for full-text input are both limited to $5k$. And the threshold of interaction turns is 20.

Model	AIRQA-REAL						M3SciQA			SciDQA			
	text	table	image	formula	metadata	AVG	table	image	AVG	table	image	formula	AVG
Classic-RAG													
GPT-4o-mini	12.3	11.9	12.5	16.7	13.6	13.4	17.9	10.6	15.6	59.4	60.4	59.3	59.8
GPT-4V	13.2	13.9	10.0	13.9	13.6	14.7	12.1	8.8	11.1	56.6	56.8	58.1	57.4
Llama-3.3-70B-Instruct	8.7	7.9	9.5	16.7	0.0	10.0	12.7	8.1	11.3	56.8	58.8	58.9	58.0
Qwen2.5-VL-72B-Instruct	9.6	5.9	11.9	11.1	13.6	10.5	11.6	11.6	11.6	54.8	56.9	56.3	56.2
DeepSeek-R1	11.7	13.9	9.5	30.6	9.1	13.9	11.9	9.5	11.2	63.9	61.3	61.7	62.4
NeuSym-RAG													
GPT-4o-mini	33.0	12.9	11.9	19.4	18.2	30.7	18.7	16.6	18.0	63.0	63.6	62.5	63.0
GPT-4V	38.9	**18.8**	**23.8**	**38.9**	**27.3**	37.3	13.7	13.4	13.6	62.6	63.5	63.2	63.1
Llama-3.3-70B-Instruct	30.6	11.9	16.7	16.7	**27.3**	29.3	**26.3**	17.6	**23.6**	55.5	57.3	56.6	56.4
Qwen2.5-VL-72B-Instruct	**43.4**	15.8	11.9	25.0	**27.3**	**39.6**	20.2	**22.7**	21.1	60.2	60.6	61.8	60.5
DeepSeek-R1	33.2	16.8	11.9	27.8	18.2	32.4	19.0	13.7	17.4	**64.3**	**64.6**	**63.9**	**64.5**

Table 2: Main results of different LLMs using Classic-RAG or NeuSym-RAG on three datasets.

Method	Neural	Symbolic	Multi-view	# Interaction(s)	sgl.	multi.	retr.	subj.	obj.	AVG
Question only	✗	✗	✗	1	5.7	8.0	0.4	9.4	2.7	4.0
Title + Abstract				1	5.7	14.0	0.0	13.1	3.6	5.4
Full-text w/. cutoff				1	28.3	10.7	0.4	26.2	7.6	11.2
Classic RAG	✓	✗	✗	1	18.2	4.0	9.4	8.4	11.0	10.5
Iterative Classic RAG				≥ 2	8.2	10.0	15.2	5.6	13.2	11.8
Two-stage Neu-RAG	✓	✗	✓	2	19.5	10.0	5.3	15.9	9.4	10.7
Iterative Neu-RAG				≥ 2	**37.7**	18.7	48.4	**32.7**	38.3	37.3
Two-stage Sym-RAG	✗	✓	✓	2	12.2	5.4	9.4	10.6	8.7	9.1
Iterative Sym-RAG				≥ 2	32.1	14.7	33.6	27.1	28.3	28.0
Graph-RAG	✓	✗	✓	2	22.2	11.1	0.0	21.1	11.5	15.6
Hybrid-RAG	✓	✓	✓	2	23.3	9.3	5.7	16.8	10.5	11.8
NeuSym-RAG (ours)				≥ 2	28.3	**32.3**	**58.2**	27.1	**42.6**	**39.6**

Table 3: Performances of different RAG methods on AIRQA-REAL dataset, where "# Interaction(s)" denotes the number of LLM calls for a single question. See App. C for detailed introduction of each RAG baseline.

Detailed resource consumption, including running time and LLM API cost, is presented in App. B.

3.3 Main Experiment

Based on our pilot study (Table 9 in App. B), we fix the action format to `markdown` and the observation format to `json` for all experiments below.

In Table 2, we show performances of our proposed method on different LLMs and datasets. Accordingly, 1) NeuSym-RAG remarkably outperforms the Classic RAG baseline across all datasets, with a minimal 17.3% improvement on AIRQA-REAL for all LLMs. With more customizable actions and more trials, NeuSym-RAG can interact with the backend environment with higher tolerance and learn through observations to retrieve proper information for question answering. 2) VLMs perform better in tasks that require vision capability, e.g., in M3SciQA where LLMs have to view an anchor image in the first place. 3) Open-source LLMs are capable of handling this complicated interactive procedure in a zero-shot paradigm, and even better than closed-source LLMs. NeuSym-RAG with Qwen2.5-VL-72B-Instruct elevates the overall accuracy by 29.1%, compared to the Classic RAG baseline. Surprisingly, it surpasses GPT-4o-mini by an impressive 8.9%, which highlights the universality of NeuSym-RAG.

Next, to figure out the contribution of each component in NeuSym-RAG, we compare the performances of different structured RAG agent methods described in § 3.2. From Table 3, we can observe that: 1) Two-stage Neu-RAG outperforms Classic RAG, while Hybrid RAG achieves even further improvements. This consistent increase can be attributed to the fact that agents can adaptively determine the parameters of actions, e.g., which chunking view to select for neural retrieval. 2) Iterative retrievals are superior to their two-stage variants. Through multi-turn interaction, the agent can explore the backend environment and select the most relevant information to answer the question. 3) As the number of interactions increases, objective scores rise faster than subjective scores, indicating that with more retrievals, LLMs generate more rationally. On the whole, NeuSym-RAG defeats all adversaries. It verifies that providing multiple views and combining two retrieval strategies both contribute to the eventual performance.

3.4 Ablation Study

For brevity, in this section, `GPT` denotes GPT-4o-mini and `Qwen` represents Qwen2.5-72B-Instruct. And unless otherwise specified, we utilize `Qwen`.

Different Model Scales In Figure 5, we show the performance changes by varying the model scales. Accordingly, we observe that: 1) The performances consistently improve as the scales of LLMs increase. 2) Unfortunately, the R1-Distill-Qwen series models lag far behind their counterparts (Qwen2.5-Instruct). Although they generate valuable thoughts during the interaction, they struggle to exactly follow the action format specified in our prompt. We hypothesize that they severely overfit the instructions during R1 post-training.

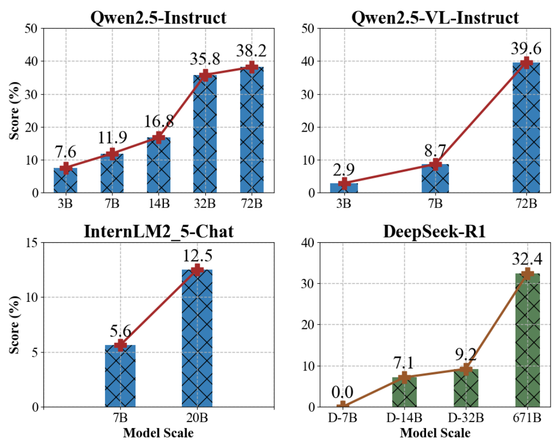

Figure 5: Performances of different model sizes on AIRQA-REAL dataset. "D-" denotes "Distill-Qwen-".

Chunking with Multiple Views In the Classic RAG baseline, we segment raw documents following continuous 512 tokens. We wonder whether different chunking views lead to varied outcomes. Results in Table 4 are insightful: the classic chunking strategy is still the best in general. However, chunking views tailored to specific aspects may achieve the best results in their particular categories.

Choices of Text Embedding Models Table 5 demonstrates the performance of NeuSym-RAG with different vector collections (*i.e.*, encoding models). The experimental results uncover that a single BM25 text collection suffices to attain the best performance. We speculate that there might be two possible reasons for this observation: 1) existing LLMs are still not capable of exploring retrieved context from multiple encoding models and then selecting the optimal one. This behavioral pattern requires profound reasoning abilities. 2) Another potential insight is that, for different encoding models, the best practice is to directly choose the highest-performing one instead of leaving the burden to the agent. And in our preliminary experiments, we also find similar phenomena that instead of creating different DB tables to store the parsed PDF content with different auxiliary tools, directly choosing the top-performing tools, that is PyMuPDF (Artifex Software, 2023) and MinerU (Wang et al., 2024a) in § 2.2, and using a single DB table achieves the best performance.

Retrieved Column	text	table	image	formula	Overall (%)
`chunk_content`	13.95	**15.00**	**22.22**	28.57	**18**
`section_content`	**18.60**	10.00	5.56	14.29	12
`table_content`	13.95	**15.00**	0.00	9.52	9
`image_summary`	4.65	0.00	16.67	4.76	6
`equation_content`	6.98	0.00	5.56	**33.33**	8

Table 4: Performances of different retrieval options of the Classic RAG baseline on a subset of AIRQA-REAL.

BM25	`MiniLM`	bge	text	table	image	formula	meta	Overall (%)
✓	✗	✗	48.8	35.0	38.9	61.9	45.5	46
✗	✓	✗	41.9	25.0	27.8	33.3	27.3	33
✗	✗	✓	39.5	25.0	16.7	52.4	54.6	37
✓	✓	✗	44.2	30.0	27.8	52.4	27.3	40
✓	✗	✓	41.9	20.0	33.3	42.9	27.4	35
✗	✓	✓	34.9	10.0	16.7	47.6	45.5	30
✓	✓	✓	39.5	20.0	16.7	47.6	45.5	35

Table 5: Performance of NeuSym-RAG on AIRQA-REAL dataset with different text encoding model (s).

Agent Model	Summary Model	objective	subjective	Overall (%)
`GPT`	`GPT`	24.62	34.29	28
	`Qwen`	23.08	34.29	27
`Qwen`	`GPT`	30.77	57.14	40
	`Qwen`	30.77	54.28	39

Table 6: Performances of different models with different LLM summary models on a subset of AIRQA-REAL.

Details of Pre-processing and Evaluation Aiming to reduce the high cost of API calls, we explore the impact of using open-source v.s. closed-source LLMs during 1) PDF pre-parsing (the summary generation), and 2) subjective evaluation. As shown in Table 6, cheaper open-source LLMs appear to be a viable alternative to closed-source ones, since the final results are insensitive to element

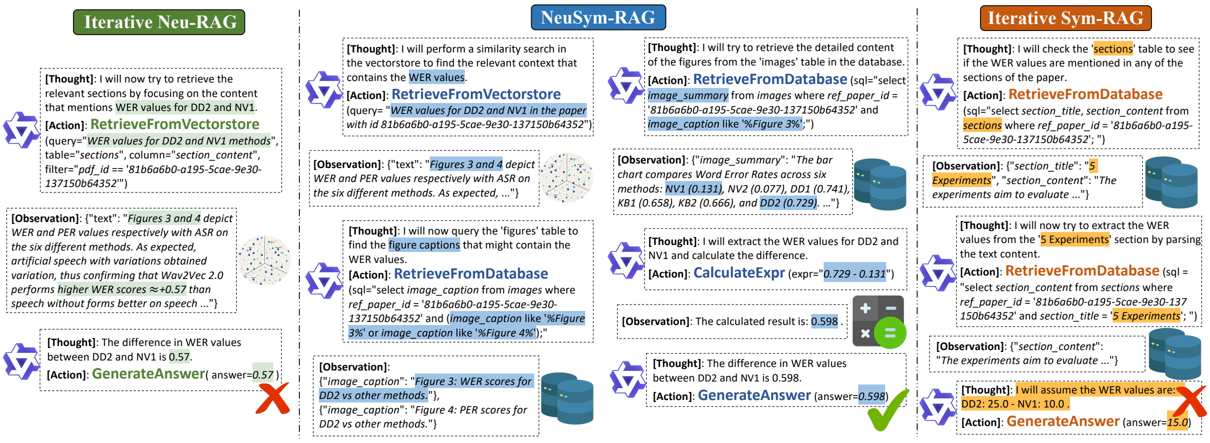

Figure 6: Case study on AIRQA-REAL dataset. The input question is "*In terms of WER values with ASR across the six different methods tested in the paper, how much higher is DD2 compared to NV1?*".

summaries produced by different LLMs. As for evaluation, we randomly sample 100 long-form answers requiring subjective assessment and recruit AI researchers to independently measure the semantic equivalence compared to the reference answer. Figure 7 indicates that, in most cases, LLM evaluations align closely with human judgments. Notably, closed-source GPT-4o aligns exactly with human judgments, whereas open-source LLMs are occasionally either too strict or too lenient. Therefore, we choose closed-source LLMs for subjective evaluation throughout the experiments.

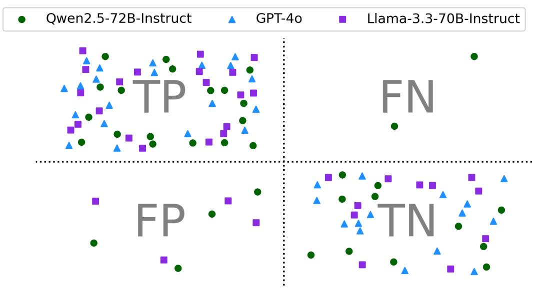

Figure 7: The consistency between human and LLM-based evaluation. We classify examples that are judged as correct by human as positive samples.

3.5 Case Study

In Figure 6, we serialize the interaction trajectory of 3 iterative methods for the same question. NeuSym-RAG firstly predicts a RETRIEVE-FROMVECTORSTORE action to search chunks regarding "*WER values*". After detecting relevant figures, it predicts a RETRIEVEFROMDATABASE action to access their captions and summaries. In contrast, Iterative Neu-RAG fails to exploit the figures due to the absence of the DB, and Iterative Sym-RAG struggles to locate relevant context. In this case, we can clearly observe one benefit or mode of combining both retrieval actions: the VS helps agents identify relevant context, while the DB for precise queries. This synergy achieves the accurate answer unattainable by either retrieval type alone, emphasizing the significance of their collaboration.

4 Related Work

Structured and Iterative Retrieval Previous literature has proposed various RAG methods for structured knowledge sources. GraphRAG (Edge et al., 2024) builds a knowledge graph (KG) to summarize partial and final answers. HybridRAG (Sarmah et al., 2024) integrates KG-based and vector-based methods for financial documents. FastRAG (Abane et al., 2024) employs schema and script learning to extract structured data, combining text search with KG. Another promising line of work proves that agents can benefit from iterative retrieval. IRCoT (Trivedi et al., 2023) handles multi-step QA by interleaving retrieval with chain-of-thought. Iter-RetGen (Shao et al., 2023) integrates retrieval and generation iteratively for collective knowledge. CoA (Zhang et al., 2024) uses multiple agents for long-context tasks, with a manager agent combining their results. Our NeuSym-RAG integrates neural and symbolic retrieval in an agentic interaction to fetch multi-view context.

PDF-based Q&A Benchmarks Existing question answering datasets over PDF documents often overlook the underlying structure and layout. QASPER (Dasigi et al., 2021) generate questions from merely titles and abstracts, while FinQA (Chen et al., 2021) targets single-page Q&A. QASA (Lee et al., 2023) retains section structures but misses other key elements like figures and tables. More recent works, like Visconde (Pereira et al., 2023), M3SciQA (Li et al., 2024a), and SciDQA (Singh et al., 2024), focus on long-form subjective evaluation. This work addresses these limitations by tackling questions from full PDFs of AI research papers, including objective metrics.

5 Conclusion

In this work, we propose a hybrid neural symbolic retrieval framework (NeuSym-RAG) for semi-structured PDF-based question-answering (Q&A). Experiments on three human-labeled realistic datasets regarding AI research, especially on the self-annotated AIRQA-REAL, show that NeuSym-RAG remarkably outperforms the Classic RAG baseline by more than 17.3% points. Future works include: 1) training a specialized LLM agent of medium size to further improve performances and efficiency, and 2) extending Q&A tasks to other vertical domains such as healthcare and law which heavily rely on external well-structured PDF files.

Limitations

Despite the superiority of the proposed NeuSym-RAG framework, it still has the following limitations: 1) The pipeline of parsing and encoding the PDF content is relatively slow, especially when higher parsing accuracy is demanded. That is, we need OCR models typically with larger capacity and longer processing time. This method is suitable if all documents are pre-stored in a repository, since the pre-processing can be conducted once and for all. However, it may be unsatisfactory in scenarios where users expect real-time uploads of new PDFs. 2) The iterative chain-of-retrieval procedure incurs extra time delay compared to classic RAG. This is anticipated and Snell et al. (2024) points out that scaling inference computation is more effective than scaling model parameters. 3) Different from GraphRAG (Edge et al., 2024) which can pre-parse any free-form texts, NeuSym-RAG requires that the external documents exhibit implicit structures or layouts. And PDF files in AI research naturally conform to this constraint. Thus, integrating the two retrieval paradigms can maximize the utilization of the inherent advantages of semi-structured documents. 4) Currently, AIRQA-REAL are limited to the domain of AI research papers. We are actively working on expanding our efforts to other vertical domains, including finance and law.

Ethics Statement

All papers utilized in the construction of our curated Q&A dataset, AIRQA-REAL, are sourced from the websites `ACL Anthology` [3], `arXiv.org` [4], and `OpenReview.net` [5], which are all licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). The use of these materials complies fully with their licensing terms, strictly for academic and research purposes. No private, sensitive, or personally identifiable information is used in this work. The study adheres to the ethical guidelines outlined by ACL, ensuring integrity, transparency, and reproducibility.

Acknowledgements

We would like to thank Hankun Wang, Danyang Zhang, Situo Zhang, Yijie Luo, Ye Wang, Zichen Zhu, Dingye Liu, and Zhihan Li for their careful annotation on the AIRQA-REAL dataset. We also thank all reviewers for their constructive and thoughtful advice. This work is funded by the China NSFC Projects (62120106006, 92370206, and U23B2057) and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

References

Amar Abane, Anis Bekri, and Abdella Battou. 2024. Fastrag: Retrieval augmented generation for semi-structured data. *arXiv e-prints*, pages arXiv–2411.

Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. Litsearch: A retrieval benchmark for scientific literature search.

Inc. Artifex Software. 2023. Pymupdf - a python binding for mupdf. `https://pymupdf.readthedocs.io/en/latest/`. Version 1.24.9, accessed on January 25, 2025.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

[3]ACL Anthology: `https://aclanthology.org/`

[4]arXiv.org API: `https://info.arxiv.org/help/api/index.html`

[5]OpenReview.net APIv2: `https://docs.openreview.net/reference/api-v2`

Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E Gonzalez, Carlos Guestrin, and Matei Zaharia. 2024. Text2sql is not enough: Unifying ai and databases with tag. *arXiv preprint arXiv:2408.14717*.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: line graph enhanced text-to-sql model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2541–2555. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R. Routledge, and William Yang Wang. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3697–3711. Association for Computational Linguistics.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics.

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. *CoRR*, abs/2404.16130.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. Infiagent-dabench: Evaluating agents on data analysis tasks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yizheng Huang and Jimmy Huang. 2024. A survey on retrieval-augmented text generation for large language models. *CoRR*, abs/2404.10981.

Hugging Face Team. 2023. Llama 3.3-70b-instruct model. `https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct`. Accessed: 2023-02-11.

Yoonjoo Lee, Kyungjae Lee, Sunghyun Park, Dasol Hwang, Jaehyeon Kim, Hong-in Lee, and Moontae Lee. 2023. Qasa: advanced question answering on scientific articles. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Chuhan Li, Ziyao Shangguan, Yilun Zhao, Deyuan Li, Yixin Liu, and Arman Cohan. 2024a. M3SciQA: A multi-modal multi-document scientific QA benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15419–15446, Miami, Florida, USA. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Hannes Mühleisen and Mark Raasveldt. 2024. *duckdb: DBI Package for the DuckDB Database Management System*. R package version 1.1.3.9017, https://github.com/duckdb/duckdb-r.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing text-to-sql capabilities of large language models: A study on prompt design strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14935–14956.

Jayr Pereira, Robson Fidalgo, Roberto Lotufo, and Rodrigo Nogueira. 2023. Visconde: Multi-document qa with gpt-3 and neural reranking. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part II*, page 534–543, Berlin, Heidelberg. Springer-Verlag.

Mykhailo Poliakov and Nadiya Shvai. 2024. Multi-meta-rag: Improving rag for multi-hop queries using database filtering with llm-extracted metadata. *arXiv preprint arXiv:2406.13213*.

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. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. 2024. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF 2024, Brooklyn, NY, USA, November 14-17, 2024*, pages 608–616. ACM.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Shruti Singh, Nandan Sarkar, and Arman Cohan. 2024. SciDQA: A deep reading comprehension dataset over scientific papers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Qwen Team. 2025. Qwen2.5-vl.

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.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. 2024a. Mineru: An open-source solution for precise document content extraction.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627.

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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zhongyuan Wang, Richong Zhang, Zhijie Nie, and Jaein Kim. 2024b. Tool-assisted agent on SQL inspection and refinement in real-world scenarios. *CoRR*, abs/2408.16991.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Hongshen Xu, Zichen Zhu, Situo Zhang, Da Ma, Shuai Fan, Lu Chen, and Kai Yu. 2024. Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback. In *First Conference on Language Modeling*.

S. Yang. 2023. Advanced rag 01: Small-to-big retrieval. Accessed on 2024-12-11.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

A AirQA-Real Dataset

A.1 Data Format

In this section, we briefly introduce the data format of our AirQA-Real dataset. Each data instance is represented as a JSON dictionary which contains the following fields:

- `uuid`: Globally unique uuid of the current task example.
- `question`: The user question about the given papers.
- `answer_format`: The requirements for the output format of LLMs, e.g., a Python list of text strings or a single float number, such that we can evaluate the answer conveniently.
- `tags`: A list of tags denoting different types, categories or genres. Feasible tags include [“single”, “multiple”, “retrieval”, “text”, “image”, “table”, “formula”, “metadata”, “subjective”, “objective”].
- `anchor_pdf`: A list of PDF uuids that are directly related to or explicitly mentioned in the question.
- `reference_pdf`: A list of PDF uuids that may or may not help answer the question.
- `conference`: A list of conference names plus year. This field is only useful in the “*paper retrieval*” setting to limit the scope of search space, as shown in the task example of Listing 4.
- `evaluator`: A dictionary containing 2 fields, `eval_func` and `eval_kwargs`, which defines how to evaluate the model outputs. Concretely,
  - the “`eval_func`” field defines the name of our customized Python function (or metric) which is used to compare the predicted result and the expected ground truth;
  - the “`eval_kwargs`” field defines the arguments for the corresponding evaluation function, which usually contain the gold or reference answer and other optional parameters.

  For example, in Listing 3, we utilize the function “`eval_structured_object_exact_match`” to evaluate the LLMs output according to the given gold answer [“SCG-NLI”, “false”].

```
{
    "uuid": "927ff9af-42f7-5216-a6f9-f106e8ff6759",
    "question": "On the HEML sentence level with AUC metric, which baseline outperforms MIND on specific conditons? Is it the best variant according to the paper that proposed that baseline?",
    "answer_format": "Your answer should be a Python list of two strings. The first string is the name of the baseline (with variant) that outperforms MIND, as proposed in the anchor PDF. The second string is either `true` or `false`.",
    "tags": [
        "multiple",
        "text",
        "table",
        "objective"
    ],
    "anchor_pdf": [
        "621d42a1-dbab-5003-b7c5-625335653001"
    ],
    "reference_pdf": [
        "ab661558-432d-5e5e-b49c-a3660a40986e",
        "1a21b653-3db0-55e8-9d34-8b6cd3dcbefa",
        "85111b8b-4df0-5a9a-8d11-a7ae12eebcf6",
        "0597ce2b-cd8c-5b5b-b692-e8042d8548de",
        "6df0f3f3-e2e1-5d7a-9d70-3114ceac5939",
        "02f7fff5-cec7-5ac8-a037-f5eb117b9547"
    ],
    "conference": [],
    "evaluator": {
        "eval_func": "eval_structured_object_exact_match",
```

```
        "eval_kwargs": {
            "gold": [
                "SCG-NLI",
                "false"
            ],
            "ignore_order": false,
            "lowercase": true
        }
    }
}
```

Listing 3: A multi-doc analysis task example of JSON format from the dataset AirQA-Real.

```
{
    "uuid": "4b4877cd-4cdc-5d52-ac20-edfaa6dd7e32",
    "question": "Is there any paper leverages knowledge distillation of language models for textual out-of-distribution detection or anomaly detection?",
    "answer_format": "Your answer should be the title of the paper WITHOUT ANY EXPLANATION.",
    "tags": [
        "retrieval",
        "text",
        "objective"
    ],
    "anchor_pdf": [],
    "reference_pdf": [],
    "conference": [
        "acl2023"
    ],
    "evaluator": {
        "eval_func": "eval_paper_relevance_with_reference_answer",
        "eval_kwargs": {
            "reference_answer": "Multi-Level Knowledge Distillation for Out-of-Distribution Detection in Text"
        }
    }
}
```

Listing 4: A paper retrieval task example of JSON format from the dataset AirQA-Real.

A.2 Data Examples

Table 7 shows 3 examples in AirQA-Real with different task types, namely single-doc details, multi-doc analysis, and paper retrieval. Different numbers of PDF documents are involved in different task types. For instance, single-doc examples only specify one paper in its "anchor_pdf" field, while multi-doc examples must mention multiple papers in the "anchor_pdf" or "reference_pdf" fields. And for paper retrieval, we relax the search scope to the entire conference venue, such as "acl2023" in Listing 4.

Category	Question	Answer Format
`single`	On the ALFWorld dataset experiments, how much did the success rate improve when the authors used their method compared to the original baseline model?	Your answer should be a floating-point number with one decimal place.
`multiple`	I would like to reproduce the experiments of KnowGPT, could you please provide me with the websites of the datasets applied in the experiment?	Your answer should be a Python list of 3 strings, the websites. Note that you should provide the original URL as given in the papers that proposed the datasets.
`retrieval`	Find the NLP paper that focuses on dialogue generation and introduces advancements in the augmentation of one-to-many or one-to-one dialogue data by conducting augmentation within the semantic space.	Your answer should be the title of the paper WITHOUT ANY EXPLANATION.

Table 7: Demonstration examples of different task types from the dataset AirQA-Real, where `single`, `multiple` and `retrieval` represent single-doc details, multi-doc analysis and paper retrieval, respectively.

A.3 Dataset Statistics

In total, the entire AIRQA-REAL contains 6797 PDF documents and 553 questions with respect to AI research papers mostly in year 2023 and 2024. Among them, 6384 PDFs are published papers from ACL 2023 and ICLR 2024, while the PDFs left are also from publicly available websites `arXiv.org` and `OpenReview.net`. Table 8, Figure 8 and Figure 9 show the statistics of the dataset AIRQA-REAL. Note that, one task example can be labeled with more than one category tag (i.e., text, table, image, formula, and metadata). As for the 553 examples, 240 are converted from the LitSearch (Ajith et al., 2024) benchmark (a paper retrieval task set). And only examples in LitSearch whose ground truth belongs to ACL 2023 or ICLR 2024 venues are selected and revised to be compatible with our AIRQA-REAL data format. Correspondingly, we set the value of field "`conference`" (defined in App. A.1) to be "`acl2023`" or "`iclr2024`" in order to restrict the paper search space.

Data Splits	Number	Ratio(%)
single	159	29
multiple	150	27
retrieval	244	44
text	470	85
table	101	18
image	42	8
formula	36	7
metadata	22	4
objective	446	81
subjective	107	19
Overall	553	100

Table 8: The task number and ratio of different data splits on the dataset AIRQA-REAL.

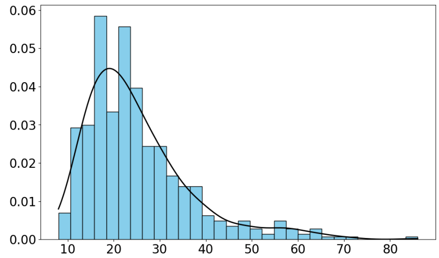

Figure 8: Frequency distribution of question lengths.

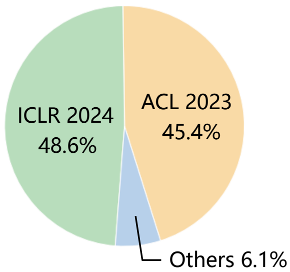

Figure 9: Conference distribution of the papers used.

A.4 Other Full-PDF-based Academic Research Q&A Benchmarks

Apart from AIRQA-REAL, we also convert two other existing Q&A benchmarks M3SciQA (Li et al., 2024a) and SciDQA (Singh et al., 2024) in academic research to validate the universality of our NeuSym-RAG framework, which gives us 492 and 2937 more test samples, respectively. Specifically,

M3SciQA The answers of the test set on M3SciQA are not publicly available yet till 2025-05-30. Thus, we only utilize the complete 452 labeled samples in the validation set as the test suite which are licensed under Creative Commons Attribution 4.0. Notice that, each question in M3SciQA involves multiple papers. To obtain the final answer, agents need to firstly find the correct paper which belongs to the references of the anchor paper. Since our constructed database contains the entire paper set on M3SciQA, we only provide the title of the anchor PDF without referenced paper titles in the prompt to make the testing scenario more realistic. And agents need to find the target paper title by itself in the reference section. PDFs of all relevant papers can be downloaded from the arXiv.org website. As for evaluation, we follow the official evaluation metric [6] and leverage large language model based assessment with exactly the same prompt, model (gpt-4-0125-preview) and temperature (0.0) to determine whether the final long-form answer is correct.

SciDQA We convert the original 2937 test samples which are licensed under Open Data Commons Attribution License (ODC-By) v1.0. The relevant paper PDFs can all be downloaded from OpenReview.net and we categorize each input question as "table", "image" and "formula" based on heuristic

[6]Evaluation for M3SciQA: `https://github.com/yale-nlp/M3SciQA/blob/main/src/evaluate_detail.py`

rules (e.g., whether a specific keyword such as "equation" is mentioned in the question or ground truth). As for evaluation, we adopt the official LLM-based judgement and answer extraction scripts [7] except that the large language model is replaced with gpt-4o-mini from Llama-3.1-70B-Instruct due to its performance being more closely aligned with humans according to Figure 7.

Two separate evaluation functions are encapsulated for these two benchmarks (see Table 11). The authors declare that the usage of these two existing benchmarks strictly obeys the aforementioned licenses.

B Supplementary Experiments and Settings

For NeuSym-RAG, we also investigate different formats of the action and observation spaces (exemplified in App. F.2.2 and App. F.2.3). According to Table 9: 1) The action format has a more substantial impact on the results compared to the observation format. 2) Specifically, both LLMs perform the best when the action format is "*markdown*", which resembles the function calling fashion in Python. Another possible explanation is that the other 3 action formats generally impose stricter formatting requirements. 3) Different LLMs may prefer different observation formats on account of their training corpus. But the gaps, especially for the top-performing 2 choices, are relatively small.

Type	Format	Overall(%)	Type	Format	Overall(%)
GPT-4o-mini-2024-07-18			Qwen2.5-72B-Instruct		
action	**markdown**	**40**	action	**markdown**	**28**
	json	35		json	27
	xml	28		xml	19
	yaml	32		yaml	26
observation	**json**	**40**	observation	json	28
	markdown	31		markdown	27
	html	31		html	26
	string	39		**string**	**30**

Table 9: Ablation study on different action and observation formats on a subset of AIRQA-REAL dataset.

In Table 10, we present some statistics of iterative methods on different models in our experiments. In general, GPT-series LLMs complete the task with fewer interaction rounds, indicating that closed-source LLMs exhibit stronger capabilities and confidence. Using the Qwen2.5-72B-Instruct LLM, the NeuSym-RAG approach completes the task with more interaction rounds than the other two iterative methods, since the NeuSym-RAG approach can integrate the context from both retrieval paradigms.

Model	RAG Method	# Interaction(s)	# Prompt Token(s)	# Completion Token(s)	Time (s)	Cost ($)
Llama-3.3-70B-Instruct	NeuSym-RAG	4.45	36521	832	40	-
Qwen2.5-72B-Instruct	Iterative Sym-RAG	5.16	43548	2730	105	-
	Iterative Neu-RAG	4.19	48600	2395	94	-
	NeuSym-RAG	5.26	58262	1987	83	-
GPT-4o-mini	NeuSym-RAG	3.59	29306	518	20	0.0059
GPT-4V	NeuSym-RAG	3.61	54657	1098	17	0.1464

Table 10: Statistics of the number of interaction(s), accumulated prompt / completion token(s), time consumption, and LLM cost per sample with different models and RAG methods on 100 samples from AIRQA-REAL.

C Agent Baselines

In this section, we elaborate the implementation of each RAG method utilized in Table 3.

- **Classic RAG** fetches question-related chunks from the vectorstore and directly provides them as the context for LLMs to answer the question. The chunk size is fixed to 512 tokens using the

[7] Evaluation for SciDQA: https://github.com/yale-nlp/SciDQA/tree/main/src/evals

RecursiveCharacterTextSplitter from langchain [8] and the retrieved top-K size is set to 4. In other words, we pre-fetch the content of column "chunks.text_content" based on the raw input question and insert them as the context in the prompt. The default text embedding model is fixed to the widely used sentence transformer all-MiniLM-L6-v2 (Wang et al., 2020).

- **Iterative Classic RAG** enables LLMs to repeatedly alter their query texts and iteratively retrieve chunks until the answer can be obtained. But the view is always fixed to column "chunks.text_content". It can be regarded as one simple implementation of the popular IRCoT (Trivedi et al., 2023).
- **Two-stage Neu-RAG** splits the task into two stages. At stage one, LLMs predict a RETRIEVEFROMVECTORSTORE action with only one chance. But they can predict the parameters in that action, e.g., the query text, the (table, column) perspective to choose, the embedding collection, and the returned top-K size. While at stage two, agents must output the final answer with retrieved context.
- **Iterative Neu-RAG** is developed from the two-stage one, incorporating additional chances in multi-turn interactions. LLMs can predict multiple parameterized actions to retrieve from the vectorstore until the interaction trajectory suffices to answer the question. And we incorporate another two useful actions VIEWIMAGE and CALCULATEEXPR (formally defined in § 2.4.1 and App. F.2.1) into the multi-turn interaction.
- **Two-stage Sym-RAG** requires LLMs to generate a SQL query first. After SQL execution upon the backend database, they must predict the answer using the retrieved context, with only one chance. Indeed, this symbolic retrieval belongs to another field of text-to-SQL (Cao et al., 2021).
- **Iterative Sym-RAG** is the multi-round version, where LLMs can iteratively interact with the symbolic database and try different SQLs to better conclude the final answer. The additional two action types VIEWIMAGE and CALCULATEEXPR are also included in the action space like Iterative Neu-RAG.
- **Graph-RAG** (Edge et al., 2024) is implemented by following the official guideline of library graphrag [9]. And we adopt the local search mode since it performs better than global in our pilot study. To reduce the graph construction time, we build a separate graph for each PDF document, and restrict the retrieval scope for each question to its tied papers.
- **Hybrid-RAG** (Sarmah et al., 2024) can be considered as a two-stage structured baseline, where the LLM agent firstly determines which action (RETRIEVEFROMVECTORSTORE or RETRIEVEFROMDATABASE) to use. After predicting the parameters of the action, and fetching the context from either the vectorstore or database, it needs to generate the final answer with only one chance.

D Evaluation Metrics

The detailed information of all 18 evaluation functions we design is presented in Table 11.

[8] Langchain text splitter: https://python.langchain.com/docs/how_to/recursive_text_splitter/.
[9] GraphRAG website: https://microsoft.github.io/graphrag/get_started/.

Eval Type	Sub-Type	Function	Description
objective	match	eval_bool_exact_match	Evaluate the output against the answer using exact boolean match.
		eval_float_exact_match	Evaluate the output against the answer using exact float match with variable precision or tolerance.
		eval_int_exact_match	Evaluate the output against the answer using exact integer match.
		eval_string_exact_match	Evaluate the output against the answer using exact string match.
		eval_structured_object _exact_match	Evaluate the output against the answer recursively by parsing them both as Python-style lists or dictionaries.
	set	eval_element_included	Evaluate whether the output is included in the answer list.
		eval_element_list_included	Evaluate whether each element in the output list is included in the answer list.
		eval_element_list_overlap	Evaluate whether the output list overlaps with the answer list.
	retrieval	eval_paper_relevance_with _reference_answer	Evaluate whether the retrieved paper is the same as the reference answer.
subjective	semantic	eval_reference_answer _with_llm	Evaluate the output against the reference answer using LLMs.
		eval_scoring_points_with _llm	Evaluate whether the scoring points are all mentioned in the output using LLMs.
		eval_partial_scoring_points _with_llm	Evaluate whether the scoring points are partially mentioned in the output using LLMs.
	formula	eval_complex_math_form ula_with_llm	Evaluate the mathematical equivalence between the output and the answer formatted in Latex using LLMs.
logical		eval_conjunction	Evaluate the conjunction of multiple evaluation functions. The output passes the evaluation if and only if all the elements in the output pass the corresponding sub-evaluations.
		eval_disjunction	Evaluate the disjunction of multiple evaluation functions. The output passes the evaluation if and only if at least one of the element in the output passes the corresponding sub-evaluation.
		eval_negation	Evaluate the negation of an evaluation function. The output passes the evaluation if and only if it doesn't pass the original evaluation function.
others		eval_m3sciqa	Evaluate examples in dataset M3SciQA with the encapsulated original LLM-based function.
		eval_scidqa	Evaluate examples in dataset SciDQA with the encapsulated original LLM-based function.

Table 11: The checklist of the 18 used evaluation functions, including their categories, names, and descriptions.

E Database Schema and Encodable Columns in Vectorstore

The complete database schema to store parsed PDF content is visualized in Figure 10, with free online tool drawSQL [10]. All encodable columns are listed in Table 12. These columns provide various perspectives towards interpreting the PDF content. And these different views are inherently connected by the sub-graph of the original database schema (as illustrated in Figure 11).

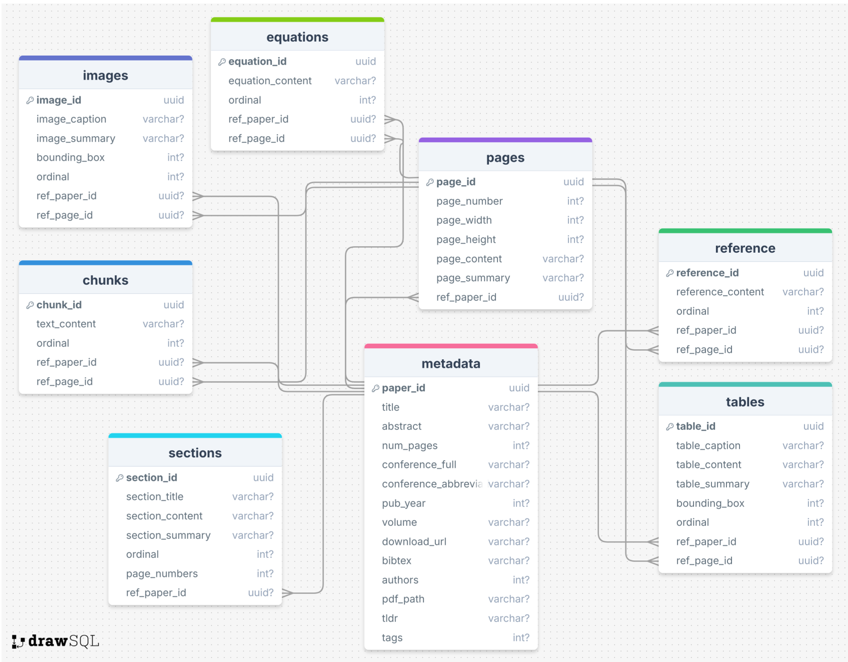

Figure 10: The complete and universal database schema to store the parsed elements of each PDF file. Note that, since the visualization tool `drawSQL` cannot display "`ARRAY`" data types, the actual data types of the columns "*bounding_box*", "*authors*", "*tags*", and "*page_numbers*" are `INT[4]`, `VARCHAR[]`, `VARCHAR[]`, and `INT[]`, respectively.

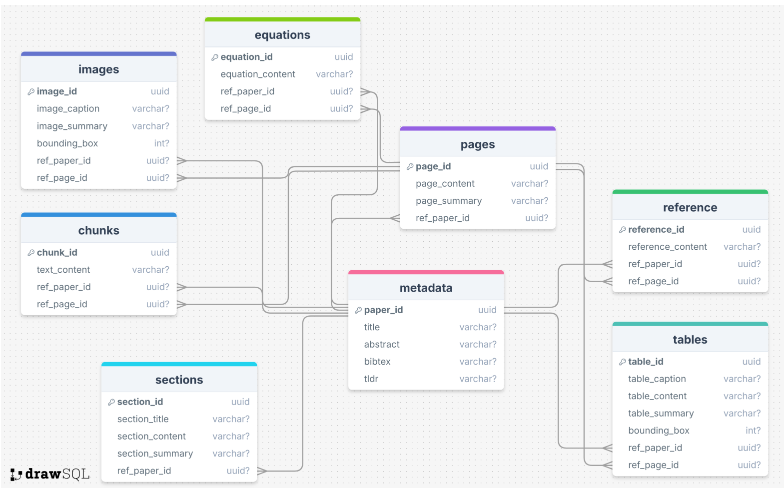

Figure 11: All encodable columns which are inherently connected by the schema sub-graph. Note that, since the visualization tool `drawSQL` cannot display "`ARRAY`" data types, the actual data types of columns "*images.bounding_box*" and "*tables.bounding_box*" are both `INT[4]`.

[10] https://drawsql.app/

Table	Column	Description
metadata	title	The title of this paper.
	abstract	The abstract of this paper.
	bibtex	The bibtex of this paper.
	tldr	A brief summary of the paper's main idea or findings generated by LLM based on title and abstract.
pages	page_content	The content of the page.
	page_summary	A brief summary of the page content, generated by LLM, focusing on key information and describing the page content.
images	image_caption	The caption of this image, empty string if not found.
	image_summary	A brief summary of the image, generated by LLM, focusing on key information and describing the image.
	bounding_box	The bounding box of the figure in the format $[x_0, y_0, w, h]$, where (x_0, y_0) represents the coordinates of the top-left corner and (w, h) represents the width and height which are used to determine the shape of the rectangle. The cropped image is encoded.
tables	table_caption	Caption of the table, showing key information of the table.
	table_content	The content of the table in html format.
	table_summary	A brief summary of the table content generated by LLM, focusing on key information and describing the table content.
	bounding_box	The bounding box of the table in the format $[x_0, y_0, w, h]$, where (x_0, y_0) represents the coordinates of the top-left corner and (w, h) represents the width and height. The cropped image is encoded.
sections	section_title	The title of the current section.
	section_content	The text content of the current section.
	section_summary	A brief summary of the section content generated by LLM, focusing on key information and describing the section content.
chunks	text_content	The text content of the current chunk.
equations	equation_content	Content of the equation in latex format.
reference	reference_content	Text content of each reference.

Table 12: The checklist of all encodable columns with their descriptions.

F Prompt Template

This section presents the detailed structure of the prompts used in various agent baselines (Figure 4), outlining the components that shape the agent's interactions and reasoning process. The overall prompt template includes the following five key components.

Overall Prompt Composition

[System Prompt]: Defines the agent's role and describes the task to tackle, clarifying the context of the task.

- - - -

[Action and Observation Space Prompt]: Defines the list of all feasible actions that the agent can take, including the action description, observation space, syntax and parameters, and use cases for each type. The detailed specification varies depending on the action format.

- - - -

[Interaction Framework Prompt]: Outlines the main interaction procedure and template.

- - - -

[Hint Prompt]: Provides hints or suggestions to help the agent refine its planning and reasoning process.

- - - -

[Task Prompt]: Defines the task input, usually including the input question, required answer format, retrieved context, database schema, vectorstore schema and operators.

F.1 System Prompt

The system prompt is used to define the ultimate goal that agents should achieve, the environment (if exists) that agents can interact with and the expected agent behavior to conduct. Notice that, the prompts for different agent baselines differ from each other.

System Prompts for Different Agent Baselines

For NeuSym-RAG:
You are an intelligent agent with expertise in **retrieving useful context from both the DuckDB database and the Milvus vectorstore through SQL execution and similarity search** and **answering user questions**. You will be given a natural language question concerning PDF files, along with the schema of both the database and the vectorstore. Your ultimate goal is to answer the input question with pre-defined answer format. The DuckDB database contains all parsed content of raw PDF files, while the Milvus vectorstore encodes specific column cells from the database as vectors. You can predict executable actions, interact with the hybrid environment (including database and vectorstore) across multiple turns, and retrieve necessary context until you are confident in resolving the question.

- - - -

Task Description
Each input task consists of the following parts:
[Question]: A natural language question from the user regarding PDF files, e.g., Is there any ...?
[Answer Format]: Specifies the required format of the final answer, e.g., the answer is Ÿesör N̈owithout punctuation.
[Database Schema]: A detailed serialized schema of the DuckDB database for reference when generating SQL queries. It includes 1) tables, 2) columns and their data types, 3) descriptions for these schema items, and 4) primary key and foreign key constraints.
[Vectorstore Schema]: A detailed serialized schema of the Milvus vectorstore for reference when generating executable retrieval actions with specific parameters. It includes 1) collections, 2) fields, 3) encodable (table, column) pairs in the relational database where the vectorized content originates, and 4) grammar for valid filter rules.

For Classic RAG:
You are intelligent agent who is expert in **answering user questions** based on the retrieved context. You will be given a natural language question concerning a PDF file, and your task is to answer the input question with predefined output format using the relevant information.

For Two-stage Neu-RAG:
[Stage 1] You are intelligent agent who is expert in **predicting a well-formed retrieval action** to search useful information to answer the user question. You will be given a natural language question concerning a PDF file and a vectorstore schema which defines all usable collections and fields in them. The vectorized contents in the vectorstore all come from cell values in another relational database which stores the parsed content of the PDF files. And your task is to predict a parametrized retrieval action to find useful information based on vector similarity search. Please refer to the concrete vectorstore schema to produce a valid retrieval action.
[Stage 2] You are intelligent agent who is expert in **answering user question** given the retrieved context. You will be given a natural language question concerning a PDF file and the retrieved context. Your task is to predict the final answer based on given question and context. Please refer to the answer format to produce the valid answer.

System Prompt

For Iterative Classic RAG and Iterative Neu-RAG:
You are intelligent agent who is expert in **retrieving useful context from the vectorstore based on similarity search** and **answering user questions**. You will be given a natural language question concerning a PDF file and a vectorstore schema of Milvus, and your ultimate task is to answer the input question with pre-defined output format. The Milvus vectorstore encodes various context from the parsed PDF in multi-views. You can predict executable actions, interact with the vectorstore in multiple turns, and retrieve desired context to help you better resolve the question.
- - - -
Task Description
Each input task consists of the following parts:
[Question]: A natural language question from the user regarding PDF files, e.g., Is there any ...?
[Answer Format]: Specifies the required format of the final answer, e.g., the answer is "Yes" or "No" without punctuation.
[Vectorstore Schema]: A detailed serialized schema of the Milvus vectorstore for reference when generating executable retrieval actions with specific parameters. It includes 1) collections, 2) fields, 3) encodable (table, column) pairs in the relational database where the vectorized content originates, and 4) grammar for valid filter rules.

For Two-stage Sym-RAG:
[Stage 1] You are intelligent agent who is expert in **writing SQL programs** to retrieve useful information. You will be given a natural language question concerning a PDF file and a database schema which stores the parsed PDF content, and your task is to predict SQL to retrieve content from the database. Please refer to the concrete database schema to produce the valid SQL.
[Stage 2] the same as method Two-stage Neu-RAG

For Iterative Sym-RAG:
You are intelligent agent who is expert in **leveraging SQL programs to retrieve useful information** and **answer user questions**. You will be given a natural language question concerning a PDF file and a database schema of DuckDB which stores the parsed PDF content, and your ultimate task is to answer the input question with predefined output format. You can predict intermediate SQLs, interact with the database in multiple turns, and retrieve desired information to help you better resolve the question.
- - - -
Task Description
Each input task consists of the following parts:
[Question]: A natural language question from the user regarding PDF files, e.g., Is there any ...?
[Answer Format]: Specifies the required format of the final answer, e.g., the answer is Ÿesör N̈owithout punctuation.
[Database Schema]: A detailed serialized schema of the DuckDB database for reference when generating SQL queries. It includes 1) tables, 2) columns and their data types, 3) descriptions for these schema items, and 4) primary key and foreign key constraints.

For Hybrid-RAG:
[Stage 1] You are intelligent agent who is expert in predicting a well-formed retrieval action to search useful information to answer the user question. You will be given a natural language question concerning a PDF file, a database schema which stores the parsed PDF content, and a vectorstore schema which defines all usable collections and fields in them. The vectorized contents in the vectorstore all come from cell values in the database. And your task is to predict a parametrized retrieval action to find useful information. Please refer to the concrete schema to produce a valid retrieval action.
[Stage 2] the same as methods Two-stage Neu-RAG and Two-stage Sym-RAG

F.2 Action and Observation Space Prompt

In total, there are 5 actions (see § 2.4.1) for the proposed NeuSym-RAG framework. We choose the RETRIEVEFROMVECTORSTORE action as an example, and serialize it in JSON format below:

Action and Observation Space Prompt for NeuSym-RAG (JSON format)

Action and Observation Space
All allowable action types include **["RetrieveFromVectorstore", "RetrieveFromDatabase", "CalculateExpr", "ViewImage", "GenerateAnswer"]**. Here is the detailed specification in **JSON** format for them:

Action Type
RetrieveFromVectorstore

Description
Given a query text, retrieve relevant context from the Milvus vectorstore. Please refer to the schema of different collections and fields for each stored data entry.

Observation
The observation space is the retrieved top-ranked entries from the Milvus vectorstore based on input parameters.

Action and Observation Space Prompt for NeuSym-RAG (JSON format) – continued

Syntax and Parameters (JSON Format)

```
{
    "action_type": "RetrieveFromVectorstore",
    "parameters": {
        "query": {
            "type": "str",
            "required": true,
            "description": "This query will be encoded and used to search for relevant context.
                You can rephrase the user question to obtain a more clear and structured requirement."
        },
        "collection_name": {
            "type": "str",
            "required": true,
            "description": "The name of the collection in the Milvus vectorstore to search for
                relevant context. Please ensure the collection does exist in the vectorstore."
        },

        "table_name": {
            "type": "str",
            "required": true,
            "description": "The table name is used to narrow down the search space. And it will
                be added to the filter condition. Please ensure this table has encodable columns."
        },
        "column_name": {
            "type": "str",
            "required": true,
            "description": "The column name is used to narrow down the search space. And it will
                be added to the filter condition. Please ensure it is encodable in `table_name`."
        },
        "filter": {
            "type": "str",
            "required": false,
            "default": "",
            "description": "The filter condition to narrow down the search space. Please refer to
                the syntax of filter rules. By default, it is empty. It is suggested to restrict
                `primary_key`, `pdf_id`, or `page_number` to refine search results."
        },
        "limit": {
            "type": "int",
            "required": false,
            "default": 5,
           "description": "The number of top-ranked context to retrieve. Please ensure that it is a
                positive integer. And extremely large limit values may be truncated."
        }
    }
}
```

Use Cases (JSON Format)

Case 1

Search the Mivlus collection `text_bm25_en`, which use BM25 sparse embeddings, with the filter condition “`table_name == 'chunks' and column_name == 'text_content' and pdf_id == '12345678' and page_number == 1`” to restrict the content source and return the top 10 relevant entries.

[Action]:

```
{"action_type": "RetrieveFromVectorstore", "parameters": {"query": "Does this paper discuss LLM-based
    agent on its first page?", "collection_name": "text_bm25_en", "table_name": "chunks",
   "column_name": "text_content", "filter": "pdf_id == '12345678' and page_number == 1", "limit": 10}}
```

Case 2

... more cases in JSON format ...

... specification for other types of actions, the prompt can be easily inferred ...

F.2.1 Syntax and Parameters for Other Action Types (JSON Format)

This subsection formally describes the syntax and parameters for the other 4 action types, including RETRIEVEFROMDATABASE, CALCULATEEXPR, VIEWIMAGE, GENERATEANSWER.

```
{
    "action_type": "RetrieveFromDatabase",
    "parameters": {
        "sql": {
            "type": "str",
            "required": true,
            "description": "The concrete DuckDB SQL query to execute and retrieve results."
        }
    }
},
{
    "action_type": "CalculateExpr",
    "parameters": {
        "expr": {
            "type": "str",
            "required": true,
            "description": "The expression to calculate, e.g., '13 * 42'."
        }
    }
},
{
    "action_type": "ViewImage",
    "description": "You can retrieve the visual information of the paper by taking this action. Please provide the paper id, the page number, and the optional bounding box.",
    "observation": "The observation space is the image that you want to view. We will show you the image according to your parameters. The error message will be shown if there is any problem with the image retrieval.",
    "parameters": {
        "paper_id": {
            "type": "str",
            "required": true,
            "description": "The paper id to retrieve the image."
        },
        "page_number": {
            "type": "int",
            "required": true,
            "description": "The page number (starting from 1) of the paper to retrieve the image."
        },
        "bounding_box": {
            "type": "List[float]",
            "required": false,
            "default": [],
            "description": "The bounding box of the image to retrieve. The format is [x_min, y_min, delta_x, delta_y]. The complete PDF page will be retrieved if not provided."
        }
    }
},
{
    "action_type": "GenerateAnswer",
    "parameters": {
        "answer": {
            "type": "Any",
            "required": true,
            "description": "The final answer to the user question. Please adhere to the answer format for the current question."
        }
    }
}
```

Listing 5: Syntax and Parameters of JSON Format for Other Action Types in NeuSym-RAG

Note that, except for the Classic RAG baseline, which has no action space, the feasible action types differ among the other baselines described in Figure 4:

- **Iterative Classic RAG:** it only accepts the RETRIEVEFROMVECTORSTORE and GENERATEANSWER actions. For the former one, we further fix the perspective to be "`chunks.text_content`" and the collection to text embedding model `all-MiniLM-L6-v2`;
- **Two-stage Neu-RAG:** it only accepts action RETRIEVEFROMVECTORSTORE at the first stage;
- **Iterative Neu-RAG:** all available action types during the iterative neural retrieval includes [RETRIEVEFROMVECTORSTORE, CALCULATEEXPR, VIEWIMAGE, GENERATEANSWER];
- **Two-stage Sym-RAG:** it only accepts action RETRIEVEFROMDATABASE at the first stage;
- **Iterative Sym-RAG:** all available action types during the iterative symbolic retrieval includes [RETRIEVEFROMDATABASE, CALCULATEEXPR, VIEWIMAGE, GENERATEANSWER];
- **Hybrid-RAG:** it accepts both actions RETRIEVEFROMVECTORSTORE and RETRIEVEFROMDATABASE at the first stage, and no action at the second stage.

F.2.2 Other Action Formats Apart From JSON

This subsection introduces other serialized action formats apart from JSON, namely MARKDOWN, XML and YAML. Take action RETRIEVEFROMVECTORSTORE as an example (others can be easily inferred):

MARKDOWN Format

Syntax and Parameters (MARKDOWN Format)

```
RetrieveFromVectorstore(query: str, collection_name: str, table_name: str, column_name: str, filter:
str = '', limit: int = 5)
    - query: str, required. The query text will be encoded and used to search for relevant context.
        You can rephrase the user question to obtain a more clear and structured requirement.
    - collection_name: str, required. The name of the collection in the Milvus vectorstore to
        search for relevant context. Please ensure the collection does exist in the vectorstore.
    - table_name: str, required. The table name is used to narrow down the search space. And it
        will be added to the filter condition. Please ensure this table has encodable columns.
    - column_name: str, required. The column name is used to narrow down the search space. And it
        will be added to the filter condition. Please ensure it is encodable in `table_name`.
    - filter: str, optional, default to ''. The filter condition to narrow down the search space.
        Please refer to the syntax of filter rules. By default, it is empty. It is suggested to
        restrict `primary_key`, `pdf_id`, or `page_number` to refine search results.
    - limit: int, optional, default to 5. The number of top-ranked context to retrieve. Please
        ensure that it is a positive integer. And extremely large limit values may be truncated.
```

Use Cases (MARKDOWN Format)

Case 1

Search the Mivlus collection `text_bm25_en`, which use BM25 sparse embeddings, with the filter condition "`table_name == 'chunks' and column_name == 'text_content' and pdf_id == '12345678' and page_number == 1`" to restrict the content source and return the top 10 relevant entries.

[Action]:

```
RetrieveFromVectorstore(query="Does this paper discuss LLM-based agent on its first page?",
    collection_name='text_bm25_en', table_name='chunks', column_name='text_content',
    filter="pdf_id == '12345678' and page_number == 1", limit=10)
```

Case 2

Perform a vector-based similarity search on all cell values from the `abstract` column in the 'metadata' table in the database, using the `MiniLM-L6-v2` setence transformer embeddings. By default, the top 5 most relevant entries will be returned.

[Action]:

```
RetrieveFromVectorstore(query="Is there any work about the topic structured RAG?", collection_name=
    'text_sentence_transformers_all_minilm_l6_v2', table_name='metadata', column_name='abstract')
```

Case 3

.. more cases in MARKDOWN format ...

XML Format

Syntax and Parameters (XML Format)

```
<action>
    <action_type>RetrieveFromVectorstore</action_type>
    <parameters>
        <query>
            <type>str</type>
            <required>true</required>
            <description>The query text will be encoded and used to search for relevant context.
            You can rephrase the original user question to obtain a more clear and structured query
            requirement.</description>
        </query>
        <collection_name>
            <type>str</type>
            <required>true</required>
            <description>The collection name in the Milvus vectorstore to search for relevant
            context. Please ensure the collection does exist in the vectorstore.</description>
        </collection_name>
        <table_name>
            <type>str</type>
            <required>true</required>
            <description>The table name is used to narrow down the search space. It will be added
            to the filter condition. Please ensure this table has encodable columns.</description>
        </table_name>
        <column_name>
            <type>str</type>
            <required>true</required>
            <description>The column name is used to narrow down the search space. It will be added
            to the filter condition. Please ensure it is encodable in `table_name`.</description>
        </column_name>
        <filter>
            <type>str</type>
            <required>false</required>
            <default></default>
            <description>The filter condition to narrow down the search space. Please refer to the
            syntax of filter rules. By default, it is empty. It is suggested to restrict `pdf_id`,
            `page_number`, or `primary_key` to refine search results.</description>
        </filter>
        <limit>
            <type>int</type>
            <required>false</required>
            <default>5</default>
            <description>The number of top-ranked context to retrieve. Please ensure that it is a
            positive integer. And extremely large limit values may be truncated.</description>
        </limit>
    </parameters>
</action>
```

Use Cases (XML Format)

Case 1
Search the Mivlus collection `text_bm25_en`, which use BM25 sparse embeddings, with the filter condition "`table_name == 'chunks' and column_name == 'text_content' and pdf_id == '12345678' and page_number == 1`" to restrict the content source and return the top 10 relevant entries.

[Action]:

```
<action><action_type>RetrieveFromVectorstore</action_type><parameters><query>Does this paper discuss
LLM-based agent on its first page?</query><collection_name>text_bm25_en</collection_name>
<table_name>chunks</table_name><column_name>text_content</column_name><filter>pdf_id == '12345678'
and page_number == 1</filter><limit>10</limit></parameters></action>
```

Case 2
.. more cases in XML format ...

YAML Format

Syntax and Parameters (YAML Format)

```
action_type: RetrieveFromVectorstore
parameters:
    query:
        type: str
        required: true
        description: The query text will be encoded and used to search for relevant context. You
            can rephrase the user question to obtain a more clear and structured requirement.
    collection_name:
        type: str
        required: true
        description: The name of the collection in the Milvus vectorstore to search for relevant
            context. Please ensure the collection does exist in the vectorstore.
    table_name:
        type: str
        required: true
        description: The table name is used to narrow down the search space. It will be added to
            the filter condition. Please ensure this table has encodable columns.
    column_name:
        type: str
        required: true
        description: The column name is used to narrow down the search space. It will be added to
            the filter condition. Please ensure it is encodable in `table_name`.
    filter:
        type: str
        required: false
        default: ''
        description: The filter condition to narrow down the search space. Please refer to the
            syntax of filter rules. By default, it is empty. It is suggested to restrict `pdf_id`,
            `page_number` or `primary_key` to refine search results.
    limit:
        type: int
        required: false
        default: 5
        description: The number of top-ranked context to retrieve. Please set it to
            a positive integer to limit the number of returned results. Extremely
            large limit values may be truncated.
```

Use Cases (YAML Format)

Case 1
Search the Mivlus collection `text_bm25_en`, which use BM25 sparse embeddings, with the filter condition "`table_name == 'chunks' and column_name == 'text_content' and pdf_id == '12345678' and page_number == 1`" to restrict the content source and return the top 10 relevant entries.
[Action]:

```
action_type: RetrieveFromVectorstore
parameters:
    query: Does this paper discuss LLM-based agent on its first page?
    collection_name: text_bm25_en
    table_name: chunks
    column_name: text_content
    filter: pdf_id == '12345678' and page_number == 1
    limit: 10
```

Case 2
.. more cases in YAML format ...

F.2.3 Observation Format

In this subsection, we discuss different observation formats to organize the retrieved entries. The execution result of action RETRIEVEFROMDATABASE naturally forms a *table*. While for action RETRIEVE-FROMVECTORSTORE, we extract the `text` field for text type and the `bounding_box` field for image type respectively, and also concatenate the top-K entries as a *table*. We support the following 4 observation formats to serialize the table-stye observation: MARKDOWN, JSON, STRING, and HTML.

Take the RETRIEVEFROMDATABASE action as an example, the SQL query to execute is:

select *title*, *pub_year* from *metadata* where *conference_abbreviation* = 'ACL' limit 3;

Then, the returned observation texts in different formats are:

Observation in MARKDOWN Format

```
+----------------------------------------------------------+----------+
|                          title                           | pub_year |
+----------------------------------------------------------+----------+
| ContraCLM: Contrastive Learning For Causal Language Model |   2023   |
|      Mitigating Label Biases for In-context Learning      |   2023   |
|     mCLIP: Multilingual CLIP via Cross-lingual Transfer   |   2023   |
+----------------------------------------------------------+----------+
```

In total, 3 rows are displayed in MARKDOWN format.

Observation in JSON Format

```
{"title":"ContraCLM: Contrastive Learning For Causal Language Model","pub_year":2023}
{"title":"Mitigating Label Biases for In-context Learning","pub_year":2023}
{"title":"mCLIP: Multilingual CLIP via Cross-lingual Transfer","pub_year":2023}
```

In total, 3 rows are displayed in JSON format.

Observation in STRING Format

```
                                                    title  pub_year
ContraCLM: Contrastive Learning For Causal Language Model      2023
          Mitigating Label Biases for In-context Learning      2023
      mCLIP: Multilingual CLIP via Cross-lingual Transfer      2023
```

In total, 3 rows are displayed in STRING format.

Observation in HTML Format

```
<table border="1" class="dataframe">
  <thead>
    <tr style="text-align: right;">
      <th>title</th>
      <th>pub_year</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>ContraCLM: Contrastive Learning For Causal Language Model</td>
      <td>2023</td>
    </tr>
    <tr>
      <td>Mitigating Label Biases for In-context Learning</td>
      <td>2023</td>
    </tr>
    <tr>
      <td>mCLIP: Multilingual CLIP via Cross-lingual Transfer</td>
      <td>2023</td>
    </tr>
  </tbody>
</table>
```

In total, 3 rows are displayed in HTML format.

F.3 Interaction Framework Prompt

This section describes the interaction framework for iterative retrieval methods. For Classic and Two-stage RAG baselines, this part is omitted. We follow the popular ReAct framework (Yao et al., 2023) to encourage stepwise thought process before predicting actions.

Interaction Framework Prompt

Interaction Framework
The main interaction procedure proceeds like this:

- - - -

[Thought]: reasoning process, why to take this action
[Action]: which action to take, please strictly conform to the action specification
[Observation]: execution results or error message after taking the action

... more interleaved triplets of ([Thought], [Action], [Observation]) ...

[Thought]: reasoning process to produce the final answer
[Action]: the terminal action `GenerateAnswer`, there is no further observation

- - - -

In general, the main interaction loop consists of an interleaved of triplets ([Thought], [Action], [Observation]), except the last `GenerateAnswer` action which does not have "[Observation]:". You need to predict the "[Thought]: ..." followed by the "[Action]: ..." for each turn, and we will execute your action in the environment and provide the "[Observation]: ..." for the previous action.

F.4 Hint Prompt

This type of prompt provides a list of hints to guide the interaction. It highlights best practices for information retrieval, iterative refinement, and sequential decision-making throughout the task-solving process. It is highly customizable and can be easily extended to accommodate LLM prediction errors. Therefore, we only provide one demonstration example for the complete NeuSym-RAG framework.

Hint Prompt for NeuSym-RAG

Suggestions or Hints for Agent Interaction

1. Explore multiple retrieval strategies. For example:
- Experiment with different (table, column) pairs to extract diverse types of information.
- Query various embedding models (collections) to find the most relevant context.

2. Combine both structured and unstructured data. Concretely:
- Use SQL queries to retrieve precise facts and structured data. Pay special attention to morphological variations in cell values.
- Perform similarity searches in the vectorstore to capture semantic relationships and hidden insights.

3. Iterate and refine:
- If SQL execution result is not satisfactory, try alternative SQL queries to explore the database content carefully.
- If the vector-based neural retrieval is insufficient, try alternative approaches or parameter settings.
- Use your findings to validate or enrich the final response.

4. Ensure confidence. That is, only make a final decision when you are confident that the retrieved information fully addresses the user's query.

F.5 Task Prompt

The task prompt defines the concrete input for the current user question, which should at least include: 1) the user input question, 2) the answer format, and 3) the database or vectorstore schema (if needed). Following Nan et al. (2023), we use the code representation format for database schema and incorporate schema descriptions to enhance the schema linking. As for the vectorstore schema, we introduce 1) all available collections, 2) fields for each stored data entry, 3) encodable (table, column) pairs from the corresponding DuckDB, and 4) valid operators to use in the `filter` condition during vector search. We only present one input case for NeuSym-RAG, while the task prompt for other methods can be easily inferred depending on whether the database or vectorstore will be integrated as the backend environment.

Task Prompt for NeuSym-RAG

Remember that, for each question, you only have 20 interaction turns at most. Now, let's start!
[Question]: What are the main questions that this paper tries to resolve or answer?"
[Answer Format]: Your answer should be a Python list of text strings, with each element being one critical problem that this paper analyzes, e.g., ["question 1", "question 2", ...].
[Database Schema]: The database schema for "ai_research" is as follows:

```
/* database ai_research: This database contains information about AI research papers. Each PDF file is
   represented or parsed via different views, e.g., pages, sections, figures, tables, and references.
    We also extract the concrete content inside each concrete element via OCR. */
/* table metadata: This table stores metadata about each paper. */
CREATE TABLE IF NOT EXISTS metadata (
    paper_id UUID, -- A unique identifier for this paper.
    title VARCHAR, -- The title of this paper.
    abstract VARCHAR, -- The abstract of this paper.
    pub_year INTEGER, -- The year when this paper was published.
    ... [more columns with their data types and descriptions, omitted] ...
    PRIMARY KEY (paper_id)
);
... [the remaining tables and columns using the CREATE statement] ...
```

[Vectorstore Schema]: The vectorstore schema for ai_research is as follows. You can try collections with different encoding models or modalities:

```
[
    {
        "collection_name": "text_bm25_en",
        "description": "This collection is used to store the sparse embeddings generated by the
            BM25 model for all encodable text content in another relational database. The semantic
            search is based on field `vector` with metric inner-product (IP).",
        "fields": [
            {"name": "vector", "dtype": "SPARSE_FLOAT_VECTOR", "desc": "attained by BM25 model"},
            {"name": "text", "dtype": "VARCHAR", "desc": "cell value from the database"},
            {"name": "pdf_id", "dtype": "VARCHAR", "desc": "unique id of the PDF file"},
            {"name": "page_number", "dtype": "INT16", "desc": "source page of the `text` field"},
            {"name": "table_name", "dtype": "VARCHAR", "desc": "source table of `text` field"},
            {"name": "column_name", "dtype": "VARCHAR", "desc": "source column of `text` field"},
            {"name": "primary_key", "dtype": "VARCHAR", "desc": "primary key value for the row
                that contains the `text` field in the relational database"}
        ]
    },
    {
        "collection_name": "text_sentence_transformers_all_minilm_l6_v2",
        "description": "This collection is used to store the embeddings generated by the model
            MiniLM-L6-v2 model for all encodable text content in another relational database. The
            semantic search is based on field `vector` with metric COSINE.",
        "fields": "The fields of this collection are the same as those in `text_bm25_en`."
    },
    ... [other collections with their fields] ...
]
```

Here are all encodable (table_name, column_name) tuples from the corresponding DuckDB database, where the encoded vector entries are sourced. Different columns together provide multiple perspectives for vector search.

```
[("metadata", "title"), ("metadata", "abstract"), ... [more encodable (table, column) pairs] ...]
```

Here are the operators that you can use in the `filter` parameter for RetrieveFromVectorstore action:

```
[
    {
        "symbol": "and",
        "example": "expr1 and expr2",
        "description": "True if both expr1 and expr2 are true."
    },
    {
        "symbol": "+",
        "example": "a + b",
        "description": "Add the two operands."
    },
    ... [more operators with detailed description] ...
]
```