Can we Retrieve Everything All at Once? ARM: An Alignment-Oriented LLM-based Retrieval Method

Peter Baile Chen[1] **Yi Zhang**[2] **Michael Cafarella**[1] **Dan Roth**[3]
[1]MIT [2]AWS AI Labs [3]University of Pennsylvania & Oracle AI
peterbc@mit.edu, imyi@amazon.com, michjc@csail.mit.edu, danroth@seas.upenn.edu

Abstract

Real-world open-domain questions can be complex, especially when answering them requires integrating information from multiple sources. Effectively identifying the necessary information involves *aligning* it with the available data and its organization. However, existing RAG solutions address the alignment problem in a limited manner. Using off-the-shelf LLMs for question decomposition lacks awareness of the available data and its structure, often resulting in suboptimal retrieval performance. Alternatively, iteratively generating follow-up queries and interacting with the data collection, as explored in agentic RAG approaches, shows potential but is often *inefficient* since each successive query depends on previous results rather than being guided by the overall organization of the available data. To address the *alignment* problem, we introduce an LLM-based retrieval method — ARM, designed to better align questions with the organization of the data collection. Instead of solely matching query utterance, ARM explores *relationships among data objects*, enabling a retrieve-all-at-once solution for complex queries. Experimental results demonstrate that ARM significantly outperforms existing RAG methods on various complex open-domain QA tasks across multiple modalities, achieving superior retrieval performance and downstream accuracy while significantly lowering monetary costs.[1]

1 Introduction

Answering real-world questions can be complicated, especially when required information is distributed across *heterogeneous* information sources, such as text corpus, databases, and even image collections. Consider an example question *"What is the highest eligible free rate for K-12 students in the schools in the most populous county in California?"* and the data collection shown in Figure 1. To answer this question, one passage (A) from the text corpus and three joinable tables (B, C, D) from the database are needed. Identifying these requires exploring the available text and tables in the data collection, reasoning about their relationships (e.g., A connects to B through entities like California counties, while C connects to D via joinable columns such as school name), and determining the best organization of these objects that can fully answer the question. In other words, an optimal retrieval strategy would leverage the *alignment* between the user query and available data and their internal organization. This approach would ensure the retrieval of objects A, B, C, and D, even if B and C are not explicitly mentioned in the user question.

One approach to tackling the alignment problem is to use off-the-shelf LLMs to decompose complex questions into simpler subqueries (Khot et al., 2022; Zhou et al., 2022; Jhamtani et al., 2023), increasing the chances of aligning them with objects in the data collection. As shown in Figure 1, this method retrieves A and D. However, such decomposition lacks awareness of the available data and its organization. Therefore, alignment may fail because important data objects relevant to the question can be missed, especially if they are not directly mentioned in the question (this method can miss bridging tables B and C).

Another approach to address the alignment problem iteratively is by retrieving relevant information at each step, which has been explored in the context of agentic RAG (Yao et al., 2022; Trivedi et al., 2022; Press et al., 2022; Asai et al., 2023; Zhang et al., 2024). Typically, an LLM-based agent, which has demonstrated impressive performance in reasoning and acting (Kojima et al., 2022; Wei et al., 2022; Shinn et al., 2024; Yao et al., 2024), *iteratively* reasons about actions (i.e., search queries). Search queries are then issued, and the retrieved passages are fed back to the agent, which continues reasoning about the next action until it de-

[1]Data and code are available at https://peterbaile.github.io/arm/.

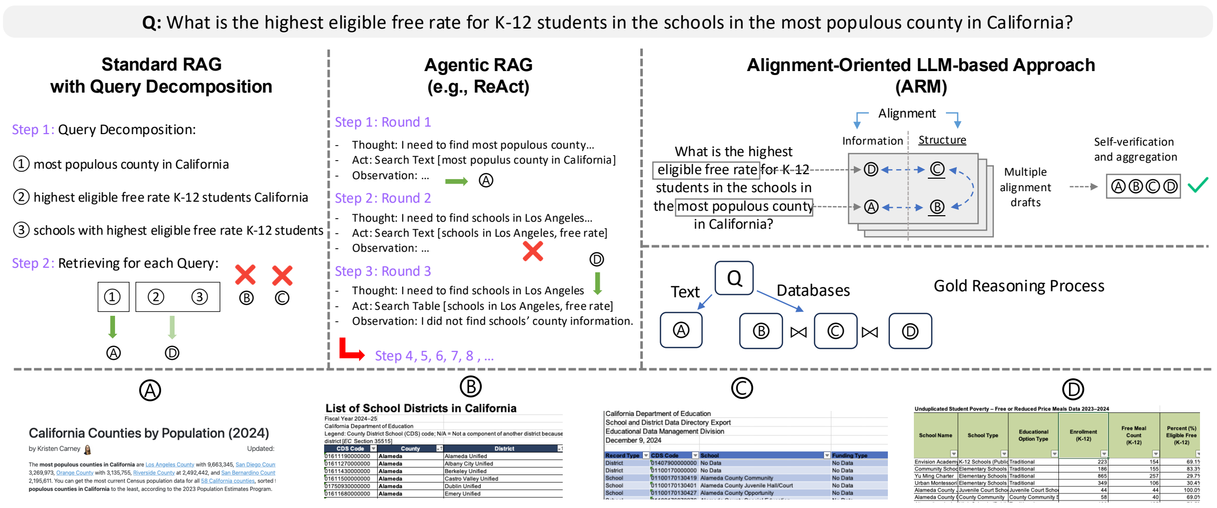

Q: What is the highest eligible free rate for K-12 students in the schools in the most populous county in California?
Standard RAG with Query Decomposition
Step 1: Query Decomposition:
① most populous county in California
② highest eligible free rate K-12 students California
③ schools with highest eligible free rate K-12 students
Step 2: Retrieving for each Query:
Agentic RAG (e.g., ReAct)
Step 1: Round 1
Thought: I need to find most populous county...
Act: Search Text [most populus county in California]
Observation: ...
Step 2: Round 2
Thought: I need to find schools in Los Angeles...
Act: Search Text [schools in Los Angeles, free rate]
Observation: ...
Step 3: Round 3
Thought: I need to find schools in Los Angeles
Act: Search Table [schools in Los Angeles, free rate]
Observation: I did not find schools' county information.
Step 4, 5, 6, 7, 8 , ...
Alignment-Oriented LLM-based Approach (ARM)
Alignment
Information
Structure
What is the highest eligible free rate for K-12 students in the schools in the most populous county in California?
Multiple alignment drafts
Self-verification and aggregation
Text
Q
Databases
Gold Reasoning Process
California Counties by Population (2024)
List of School Districts in California
California Department of Education
Unduplicated Student Poverty – Free or Reduced Price Meals Data 2023–2024

Figure 1: A summary of our approach ARM, and a comparison with retrieval in standard RAG, which leverages LLMs for query decomposition, and agentic RAG, which employs LLM-based agents to iteratively generate queries.

termines that the question has been fully answered. Although adopting such agent-based iterative solutions has shown promising results (such as retrieving A and D, as shown in Figure 1), these approaches do not *explicitly* consider available data objects and their organization. Each action is driven by the agent's decision about what information is still needed to fully answer the question, based on previously retrieved objects. This can lead to *inefficient* exploration with unnecessary iterations, increasing inference time and cost. Moreover, an iterative approach based on past experience cannot enable a *joint optimization*, where early and later steps are planned together. As shown in (Yao et al., 2022), agentic solutions face a critical issue known as *reasoning derailment*, where agents fail to recover from an initial erroneous action, resulting in consecutive rounds of incorrect reasoning.

In this work, we argue that enabling an efficient and comprehensive retrieval for complex questions requires *aligning* the question and its decomposition with existing data objects and their organization. This means that, beyond performing semantic matching between the question and data objects (allowing the retrieval of A and D), the retrieval process must also simultaneously reason about how candidate data objects can be connected and whether they involve bridging entities between documents or connecting columns between tables (ensuring the retrieval of B and C as well). Therefore, we propose an LLM-based **a**lignment-oriented **r**etrieval **m**ethod, ARM, for complex open-domain questions. Through reasoning about a better alignment to the data collection and its organization, we want to have a more *effective* and *efficient* retrieval solution, where we can achieve *retrieve-everything-all-at-once*. Inspired by (Chen et al., 2024c), which can be considered as a retrieve-all-table-at-once solution that leverages a solver to reason about table relationships, we integrate its reasoning module as a way to identify the relationships among data objects, and therefore we can *search* for potential alignments from the query to the data collection for an LLM. We then use LLM to verify and aggregate the candidate alignment outputs and produce the final results. Unlike agentic RAG solutions tailored for multi-hop question answering (Press et al., 2022; Trivedi et al., 2022), which rely mainly on bridging entities in passages to generate partial answers for guiding subsequent retrieval, our reasoning module is more general. It can integrate business-specific logic, such as detecting joinable columns between tables, to reason about relationships and handle a broader range of complex QA tasks.

2 Overview

This section provides an overview of our alignment-oriented LLM-based retrieval solution. Each component will be elaborated in Section 3. Intuitively, our solution adopts the idea of retrieving while generating (Jain et al., 2024), so that we can take the most advantage of LLM's reasoning capability to infer the alignment between the question and the data organization. Rather than simply interleaving "retrieval" by constrained decoding based on evidence from the text corpus, and "thoughts" by unconstrained decoding (Jain et al., 2024), we propose to retrieve everything jointly, guided by information from data objects available in the data collection, a solver's reasoning, and LLM's self-verification.

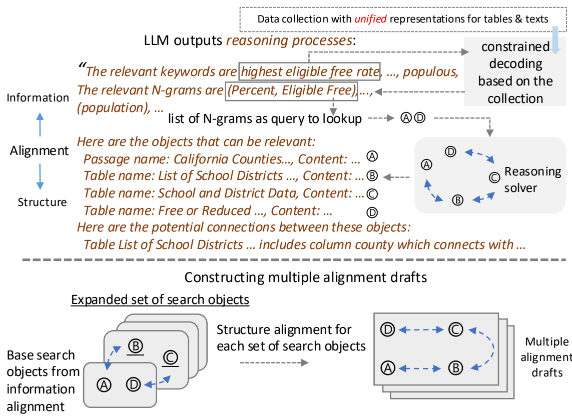
Specifically, we consider the retrieval problem as an *alignment* problem, aiming to identify all necessary data objects from the data collection and map them to the information needed to answer the question. An off-the-shelf LLM faces two key challenges in solving this problem. First, while an LLM can use its reasoning capabilities to identify potentially useful information, it cannot determine how to map that information to existing data objects without direct access to the data collection. Second, LLMs may lack the domain-specific knowledge to reason about how different data objects that are semantically related to the question are connected. They might also struggle to identify whether additional data objects are required to connect these objects, particularly when the question involves private databases (Chen et al., 2024b). Therefore, LLMs need guidance from the data collection and its organization to reason about the alignment.

To solve the challenges mentioned above, we mainly consider two alignment steps and one self-verification step. The first is *information alignment*, where we draft the key information needed to answer the question directly. This is achieved by constrained decoding using N-grams extracted from existing data objects. The second is *structure alignment*, where we reason about how different pieces of key information from existing data objects can be connected, potentially with additional objects, to answer the question through a reasoning solver. The alignment results are then fed to the "reasoning process" as drafts, and the LLM self-verifies the relatedness of the data objects to the question as well as their connections and selects the final data objects that can fully answer the question. The overall idea is illustrated in Figure 1.

3 Methodology

3.1 Indexing

In this paper, we unify tables and passages and consider them as *textual data objects*. We chunk each serialized data object, compute the embeddings of each chunk, and further represent and index it as an N-gram collection. N-grams are used to summarize the key information from a chunk of data objects, enabling a quick lookup of its contents. Embeddings are used to support semantic similarity search. Both embeddings and N-grams are used to guide LLMs' reasoning of what data objects should be used to answer the question (Section 3.2.1). Details can be found in Appendix A.



Data collection with unified representations for tables & texts
LLM outputs reasoning processes:
constrained decoding based on the collection
"The relevant keywords are highest eligible free rate, ..., populous,
The relevant N-grams are (Percent, Eligible Free), ..., (population), ...
list of N-grams as query to lookup
Ⓐ Ⓓ
Information
Alignment
Structure
Here are the objects that can be relevant:
Passage name: California Counties..., Content: ... Ⓐ
Table name: List of School Districts ..., Content: ... Ⓑ
Table name: School and District Data, Content: ... Ⓒ
Table name: Free or Reduced ..., Content: ... Ⓓ
Here are the potential connections between these objects:
Table List of School Districts ... includes column county which connects with ...
Reasoning solver
Constructing multiple alignment drafts
Expanded set of search objects
Base search objects from information alignment
Structure alignment for each set of search objects
Multiple alignment drafts

Figure 2: Information and structure alignment in ARM. Gray dotted lines refer to guidance received from external sources/ tools. Blue dotted lines refer to the inferred data organization required to answer the question.

3.2 Alignment

3.2.1 Information Alignment

As described in Section 2, we instruct the LLM to generate a reasoning process with multiple intermediate steps. The first step is to determine the key information required to answer the question. As an off-the-shelf LLM lacks access to the data collection, its analysis of useful information may not align with information from the available data objects. To address this, we propose to instruct the LLM to first decompose the question by extracting keywords independently of the data collection, and then guide it to rephrase each keyword using N-grams available from the data objects in the collection through constrained decoding.

Constrained Beam Decoding. Our constrained-decoding-based information alignment is based on the model's extracted keywords from the user question. Since these keywords might not appear as directly in the corpus, we instruct the model to rephrase them to align with N-grams indexed from our data collection as mentioned in Section 3.1.

This alignment is performed by constraining the model's output space during the decoding process. The constrained beam decoding starts once a '(' is decoded which indicates that model is performing alignment and continues until ')' is decoded which indicates that the model has finished alignment for one keyword. As an N-gram can be composed of multiple tokens, we use a suffix tree to keep track of valid continuations of generated tokens. With beam search, we maintain top lists of N-grams with the highest scores and decode the list with the highest score. The score of an N-gram is calculated as the average logits across all of its tokens.

Each list of N-grams decoded is used as a query to search for chunks, considering both the exact match and semantic similarity to objects in the data collection. Objects with the highest overall score form a *base* set of search objects that serve as the foundation for constructing a draft for the continuation of LLM's "reasoning process".

3.2.2 Structure Alignment

Information alignment guides us towards a set of search objects from the data collection that is very likely to help answer the question. However, there can be *redundant* and *missing* information. For instance, several passages identified may address the same aspect of the question. The information about the necessary bridging entities (Yang et al., 2018) or bridging tables (Chen et al., 2024c), and the information that has to be derived from those bridging entities or tables can still be missing. To address this, we further design a structure alignment module that reasons about a complete list of search objects with their organization, so that it can match the information required and fully answer the question.

The key challenge of structure alignment is LLM does not have a global view of all available data objects and may lack specific knowledge to identify the missing objects needed to correctly connect the candidate objects that have been identified, especially when the data collection belongs to a specialized domain that the LLM may not have encountered extensively during its training.

Therefore, we propose to use an external solver to solve this structure alignment problem, where we can formulate the objective and include any business or domain-specific reasoning logic. Specifically, for a given list of search objects, the solver is tasked with returning a subset of the input search objects that are *connected* and can be combined to best answer the question. We then use the content of these selected objects to construct a partial "draft" for the LLM to continue its "reasoning process"(Section 3.3).

Inference using Mixed-Integer Programming. We formulate structure alignment for retrieval as an optimization problem. Following (Chen et al., 2024c), we use a mixed-integer linear program (MIP) to solve it, leveraging its flexibility to inject any business or domain-specific logic into the objective. Specifically, the goal of the MIP program is to select a list of k objects from a given list of M search objects $\{O_i\}_{i=1}^{M}$ that can simultaneously maximize the relevance between the question and selected objects and compatibility (strength of connection) among the selected objects. The output of the MIP consists of both the k selected objects and the *connections* between these objects (connecting entities or joinable columns). Details can be found in Appendix B.

Relevance. Relevance between a user question Q and an object O_i, denoted as R_i, is defined as the cosine similarity between the embedding of Q and the embedding of the serialized object O_i.

Compatibility. Object-object compatibility between two objects O_i and O_j, denoted as C_{ij}, measures the strength of connection between objects. It is computed based on both semantic similarity and exact-value similarity between contents from both objects.

Objective. The objective function is to maximize the relevance of the selected objects and compatibility among the selected objects: $\arg\max \sum_i R_i b_i + \sum_{i,j} C_{ij} c_{ij}$ where binary decision variable b_i denotes whether object O_i is selected and c_{ij} denotes whether object O_i connect with object O_j.

Constructing Multiple Alignment Drafts. Although our reasoning solver aims to ensure alignment by considering data organization to fully answer the question, it depends on the base set of data objects retrieved during the information alignment stage. However, the base set of objects may not include sufficient information to answer the user question. For instance, when questions require multiple connected objects (and bridging objects), the base objects might need to be expanded to include sufficient information. To address this, we propose expanding the base search objects in different ways, creating *multiple* sets of search objects. Specifically, this expansion follows a multi-hop approach, where for each base object, we add its k most compatible objects (determined using the compatibility score defined above). This process can be repeated for l steps, progressively introducing more relevant objects. We then apply the aforementioned MIP

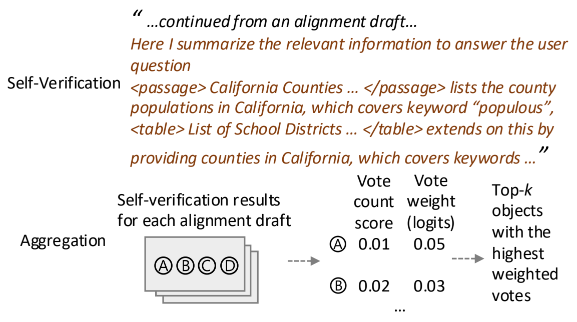

“ …continued from an alignment draft…
Here I summarize the relevant information to answer the user question
Self-Verification
<passage> California Counties … </passage> lists the county populations in California, which covers keyword “populous”,
<table> List of School Districts … </table> extends on this by providing counties in California, which covers keywords … ”
Self-verification results for each alignment draft
Aggregation
Ⓐ Ⓑ Ⓒ Ⓓ
Vote count score
Vote weight (logits)
Ⓐ 0.01 0.05
Ⓑ 0.02 0.03
…
Top-k objects with the highest weighted votes

Figure 3: Self-verification and aggregation in ARM.

solver to each set, generating *multiple* alignment drafts that integrate both information and structure alignment. In our experiments, we considered three expanded sets of search objects.

3.3 Self-Verfication and Aggregation

Finally, each draft generated by the MIP solver, which includes both objects and their connections, is verbalized (details can be found in Appendix C) and provided to models. Models will select a set of objects that can potentially be used to answer the user question, acting as a verifier that collectively evaluates all information decoded so far to perform object selection. An aggregation mechanism, based on model confidence, is used to combine the selection results from the different drafts. This is shown in Figure 3.

Model as Self-Verifier. At this stage, model has generated a decomposition of the user question, and has alignment knowledge on both information and structure. Using these knowledge, model performs the verification process by checking if selected objects includes content that can cover different aspects of the decomposition and that selected objects are connected. We use constraint decoding to guide this selection process to ensure factuality. In particular, we ensure that objects chosen by models must be present in the draft.

Aggregation of alignment drafts. After verification is performed for each draft, the model's *reasoning process* is completed. We aggregate the selected data objects from the multiple alignment drafts by factoring in the model's confidence for each selected object. Specifically, the aggregation process is treated as a weighted voting process, where each draft acts as a voter who votes for a selected object. In this context, an object's confidence can be measured by the weight of votes and the number of votes it receives. The weight of a vote is measured using logits. In particular, tokens generated in the *reasoning process* that correspond to the object's name are identified, and the logits of these tokens are averaged to be the weight of a vote. The number of votes an object receives is the count of its occurrences across all drafts, with softmax applied to normalize this count. Then, the confidence in an object is computed as the weighted sum of the average vote weight and the normalized number of votes. Finally, the data objects with the highest confidence are selected as the final set of retrieved objects.

4 Experiments

4.1 Experimental setup

Datasets. We evaluated our approach and baselines on open-domain question-answering tasks involving multiple information sources and modalities (text and tables). As such, we selected representative complex QA datasets for each modality: 2WikiMultiHop (Ho et al., 2020) for text, Bird (Li et al., 2024) for tables, and OTT-QA (Chen et al., 2020) for a combination of both text and tables. Specifically, 2WikiMultiHop consists of multi-hop questions over multiple passages with short-text answers, Bird focuses on questions involving multiple tables with answers in the form of SQL statements, and OTT-QA includes multi-hop questions spanning both passages and tables, with short-text answers. For each dataset, we used the dev split for the user questions and constructing the data collection. Details can be found in Appendix D.1.

Baselines. We evaluated our approach against two baseline methods: the standard RAG and agentic RAG approaches. For the standard RAG baseline, we considered two variations: dense retrieval and dense retrieval followed by a cross-encoder reranker. Additionally, we augmented the standard RAG approach by incorporating an LLM-based query decomposition. To have a fair comparison with our approach, we used the same model (Llama 3.1-8B-Instruct) as the LLM to perform query decomposition. Table 4 contains prompts used for generating sub-questions.

In our experiments, the embedding models used for dense retrieval were UAE-Large-V1 (Li and Li, 2023) and Snowflake-arctic-embed-m-v2.0 (Yu et al., 2024) , and we used bge-reranker-v2-minicpm-layerwise (Li et al., 2023; Chen et al., 2024a) as the reranking model. Additionally, ReAct and IRCoT were chosen as representatives of the agentic RAG approach. As mentioned in Section 1, IRCoT is specifically designed for multi-hop question answering, relying on generating partial answers to guide the next round of retrieval. However, this approach is difficult to adapt to scenarios where decomposing the final answer into partial answers is not straightforward, such as SQL-based answers in the Bird dataset. As a result, we did not evaluate IRCoT on the Bird dataset. Details for running the baselines can be found in Appendix D.2.

Environment. We used the Python-MIP[2] package with Gurobi as the external Mixed-integer program solver and a cluster of V100 GPUs for performing inference. Both Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct were used as models for running ARM to perform the retrieval process. To perform downstream tasks, we used Llama-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, and GPT-4o-mini. Agentic RAG baselines were executed on the same three models. The same three ICL examples were provided to models for running ARM and downstream tasks. Table 7 contains the 3-shot prompts used for ARM.

4.2 Metrics

We evaluated both the retrieval performance of the retrieved objects and the end-to-end performance on downstream tasks. We further quantified the efficiency of different methods by tracking the number of input and output tokens, computing the total cost of the end-to-end process, and measuring retrieval iterations.

For retrieval performance, we adopted the standard metrics of precision, recall, and F1 of the retrieved objects compared to the gold objects. However, the recall metric could be misleading because, in an extreme scenario, a retriever can achieve high recall by retrieving a significant portion of gold objects for every question, but with none of the questions having *all* gold objects retrieved. This is problematic as a question can usually only be answered when all information is provided. Therefore, we further augmented existing metrics with the percentage of questions with all gold objects retrieved, denoted as perfect recall (PR). For agentic RAG baselines, we assessed retrieval performance by comparing the objects provided to models across all iterations with the gold objects.

To evaluate end-to-end performance on downstream tasks, we measured exact match (EM) and F1 score by comparing the predicted and gold short answers for OTT-QA and 2WikiMultiHop. For Bird, following (Li et al., 2024), execution accuracy was defined as 1 if the predicted and gold SQL statements produced the same execution results and 0 otherwise.

To evaluate efficiency, we tracked the number of input and output tokens along with the retrieval iterations used by LLMs for agentic RAG baselines and ARM. For agentic RAG baselines, this includes the entire iterative process. For ARM, this includes the retrieval step and the final LLM answer generation. The overall cost is determined based on the number of input and output tokens, following the pricing of proprietary models. We adopted OpenAI's pricing model[3] as OpenAI models are among the most widely used. In particular, since our experiments utilize GPT-4o-mini, we based our cost analysis on its pricing strategy. Additionally, unlike standard RAG baselines or ARM, which perform a single round of retrieval, agentic RAG can perform multiple retrieval iterations. Therefore, we also measured the number of retrieval iterations for these methods.

4.3 Retrieval performance

Table 1 shows the retrieval performance of all methods. On Bird, ARM retrieves on average 5.00 objects, achieving a recall of 96.5 and perfect recall of 92.2. In comparison, the best-performing standard RAG baseline, dense retrieval, retrieves 5 objects with a recall of 88.3 and perfect recall of 77.7, which is 8.1 and 14.6 points lower compared to ARM, respectively. Additionally, compared to ReAct running on Llama3.1-8B-Instruct, ARM reduces retrieval iterations by 4.34 and retrieves 11.6 fewer objects while maintaining comparable recall and perfect recall but achieving a 32.3 higher F1, potentially reducing noise.

On OTT-QA, ARM retrieves an average of 5.0 objects, achieving a recall of 79.6 and a perfect recall of 61.4. In comparison, the best-performing standard RAG baseline, dense retrieval with reranker, retrieves 5 objects with a recall of 74.5 and a perfect recall of 52.4, which is 5.1 and 9.0 points lower than ARM, respectively. Additionally, compared to ReAct running on Llama3.1-8B-Instruct, ARM reduces retrieval iterations by

[2] https://www.python-mip.com/

[3] https://openai.com/api/pricing/

Table 1: Retrieval performance of all methods. PR is the percentage of questions with all gold objects retrieved. DR refers to dense retrieval. DRR refers to dense retrieval with reranker. X-D refers to method X with query decomposition. X_{L}, X_{Q}, and X_{G} refers to method X executed on Llama3.1-8B-Instruct, Qwen2.5-7B-Instruct, and GPT-4o-mini, respectively. For agentic baselines, X(num_1 ; num_2) refers to X involves num_1 retrieval iterations when using UAE-Large-V1 as the embedding model and num_2 iterations when using Snowflake-arctic-embed-m-v2.0. **Bolded** and underlined numbers indicate the best and second-best performance on local models, respectively. **Gray bolded** numbers indicate the performance of proprietary models when they outperform local models.

	UAE-Large-V1					Snowflake-arctic-embed-m-v2.0				
	Avg #obj. ↓	P	R	F1	PR	Avg #obj.	P	R	F1	PR
Bird										
DR	5.00	33.7	88.9	47.7	78.4	5.00	33.2	87.7	47.0	76.9
DRR	5.00	33.0	86.2	46.5	74.7	5.00	32.5	85.4	45.9	73.9
DR-D	3.72	38.9	74.9	49.1	56.5	3.57	42.5	76.6	52.5	59.8
DRR-D	5.00	33.0	86.4	46.6	74.7	5.00	33.1	86.8	46.8	75.4
ReAct$_{\text{L}}$(5.24 ; 5.44)	16.1	15.5	96.4	25.3	93.0	17.1	13.6	95.2	22.8	91.2
ReAct$_{\text{Q}}$(2.22 ; 2.17)	6.92	28.5	92.9	42.2	86.0	7.09	27.6	92.0	41.0	84.7
ReAct$_{\text{G}}$(3.16 ; 2.95)	10.4	25.6	96.6	38.4	92.5	10.0	24.8	**96.4**	37.7	**92.2**
ARM$_{\text{L}}$	5.00	42.9	**96.9**	56.2	**93.1**	4.95	43.1	**96.0**	56.4	**91.3**
ARM$_{\text{Q}}$	3.79	**56.6**	95.1	**67.1**	89.6	3.79	**55.0**	94.0	**65.6**	87.5
OTT-QA										
DR	5.00	34.6	69.0	44.0	43.1	5.00	32.6	65.0	41.4	35.8
DRR	5.00	37.4	75.2	47.8	53.8	5.00	36.8	73.7	46.9	51.0
DR-D	4.79	32.7	62.3	40.9	32.7	4.72	32.6	61.5	40.7	30.6
DRR-D	5.00	37.1	74.5	47.4	52.5	5.00	36.9	73.9	47.0	51.3
ReAct$_{\text{L}}$(4.52 ; 4.60)	18.1	15.5	75.0	23.4	54.0	18.9	14.1	71.2	21.4	48.1
ReAct$_{\text{Q}}$(2.38 ; 2.43)	9.88	24.0	74.9	33.5	53.8	10.2	22.5	72.7	31.6	49.7
ReAct$_{\text{G}}$(3.70 ; 4.00)	13.4	21.9	**80.1**	31.2	61.7	14.9	19.7	78.6	28.5	59.4
IRCoT$_{\text{L}}$(4.13 ; 4.27)	16.2	13.6	63.6	20.7	41.5	16.9	12.9	62.3	19.6	39.1
IRCoT$_{\text{Q}}$(1.00 ; 1.00)	5.02	4.3	8.7	5.5	5.9	5.01	4.0	8.1	5.1	4.6
IRCoT$_{\text{G}}$(5.31 ; 5.78)	22.9	13.1	**81.8**	20.7	**65.3**	25.0	11.6	**80.4**	18.6	**63.3**
ARM$_{\text{L}}$	4.99	47.4	**80.0**	**55.1**	**62.6**	5.00	46.7	**79.1**	**54.6**	**60.2**
ARM$_{\text{Q}}$	4.98	**48.6**	77.3	54.3	57.8	5.00	**48.9**	77.0	54.3	56.9
2WikiMultihop										
DR	5.00	33.2	71.7	44.6	42.7	5.00	33.3	71.7	44.6	42.2
DRR	5.00	34.1	73.5	45.8	45.7	5.00	34.4	74.0	46.1	46.1
DR-D	5.00	33.0	71.0	44.2	41.2	5.00	33.2	71.3	44.4	41.1
DRR-D	5.00	34.3	73.9	46.0	46.2	5.00	34.4	73.8	46.0	45.7
ReAct$_{\text{L}}$(2.58 ; 2.50)	10.3	28.0	97.2	42.1	94.0	10.3	28.0	97.1	42.0	93.9
ReAct$_{\text{Q}}$(1.86 ; 1.85)	8.31	28.1	88.7	41.5	76.4	8.34	27.9	88.4	41.1	75.2
ReAct$_{\text{G}}$(2.39 ; 2.33)	9.09	29.2	**98.0**	44.0	**95.8**	9.11	29.3	98.0	44.0	**95.6**
IRCoT$_{\text{L}}$(3.15 ; 3.18)	11.6	15.4	70.6	24.4	51.1	11.7	15.7	71.3	24.8	51.3
IRCoT$_{\text{Q}}$(1.00 ; 1.00)	5.02	4.3	9.6	5.8	6.0	5.05	4.4	9.9	5.9	6.1
IRCoT$_{\text{G}}$(6.04 ; 6.30)	25.4	12.2	89.6	20.0	79.8	26.4	11.8	89.7	19.3	79.9
ARM$_{\text{L}}$	4.96	62.8	**97.6**	71.7	**94.7**	4.97	61.7	**98.0**	71.2	**95.4**
ARM$_{\text{Q}}$	4.67	**66.3**	96.6	**74.5**	92.4	4.59	**67.2**	97.3	**75.3**	93.9

3.56 and retrieves 13.5 fewer objects while achieving 6.5 points higher recall and 10.4 points higher perfect recall.

On 2WikiMultiHop, ARM retrieves an average of 5.0 objects, achieving a recall of 97.8 and a perfect recall of 95.1. In comparison, the best-performing standard RAG baseline, dense retrieval with query decomposition and reranker, retrieves 5 objects with a recall of 73.9 and 46.0, which is 24.0 and 49.1 points lower than ARM, respectively. Additionally, compared to ReAct running on Llama3.1-8B-Instruct, ARM reduces retrieval iterations by 1.54 and retrieves 5.3 fewer objects while achieving 0.6 points higher recall and 1.1 points higher perfect recall.

These results demonstrate that ARM outperforms standard RAG baselines in recall and perfect recall, indicating a higher likelihood of retrieving all necessary information to answer user questions. Additionally, compared to agentic RAG baselines executed on the same model, ARM achieves comparable or superior retrieval performance while using much fewer retrieval iterations and retrieving fewer objects.

Table 2: End-to-end performance of all methods. Refer to Table 1 for notations.

	UAE-Large-V1					Snowflake-arctic-embed-m-v2.0				
	Bird	OTT-QA		2WikiMultiHop		Bird	OTT-QA		2WikiMultiHop	
	Acc.	Exact	F1	Exact	F1	Acc.	Exact	F1	Exact	F1
Model for answer generation: Llama3.1-8B-Instruct										
DR@5	17.7	34.1	40.6	30.7	37.9	17.4	29.4	35.6	28.3	35.5
DRR@5	17.0	39.8	46.8	31.5	38.7	15.6	38.1	45.3	30.7	38.9
DR-D@5	14.1	27.2	33.3	28.6	35.9	16.4	26.1	31.9	29.4	36.4
DRR-D@5	16.4	38.7	46.2	30.5	38.1	18.8	38.7	45.7	29.8	37.9
ReAct$_{L}$	5.0	27.2	34.4	33.7	43.8	4.3	26.2	33.2	34.1	44.6
IRCoT$_{L}$	–	18.0	21.9	14.7	18.9	–	17.4	21.0	14.7	18.7
ARM$_{L}$	**20.6**	**44.1**	**52.0**	**45.0**	**54.4**	19.5	**43.8**	**50.8**	43.5	53.8
ARM$_{Q}$	20.1	42.4	49.9	44.5	54.3	**21.8**	42.1	49.2	**44.3**	**54.4**
Model for answer generation: Qwen-2.5-7B-Instruct										
DR@5	19.5	30.0	35.5	35.6	40.6	19.8	27.0	32.2	35.0	40.3
DRR@5	19.9	35.9	40.9	36.6	41.7	22.0	35.2	41.3	35.3	40.2
DR-D@5	16.8	24.0	29.1	34.7	39.6	19.6	23.6	28.4	34.4	39.3
DRR-D@5	20.0	36.5	42.5	34.9	40.5	22.9	36.5	42.2	33.7	38.8
ReAct$_{Q}$	11.9	34.7	44.1	31.9	45.4	12.3	35.0	44.5	33.7	46.0
IRCoT$_{Q}$	–	4.4	5.3	6.1	8.7	–	3.7	4.5	6.7	9.4
ARM$_{L}$	21.3	**40.4**	**46.8**	**48.5**	**56.2**	22.7	**40.0**	**46.6**	46.7	**54.5**
ARM$_{Q}$	**23.2**	37.3	43.7	45.5	53.1	**26.1**	38.0	44.0	**47.1**	54.2
Model for answer generation: GPT-4o-mini										
DR@5	29.9	39.9	47.8	40.7	47.5	32.0	33.9	41.7	40.5	47.5
DRR@5	30.1	45.9	55.2	41.7	48.6	31.5	44.3	53.9	41.1	48.4
DR-D@5	24.1	32.6	39.7	40.4	46.9	26.2	30.2	37.8	41.7	47.9
DRR-D@5	27.1	45.6	55.4	41.6	48.2	26.8	44.5	53.7	41.1	48.3
ReAct$_{G}$	27.1	41.2	50.2	52.7	**68.8**	27.8	35.1	45.4	**54.1**	**70.1**
IRCoT$_{G}$	–	18.6	25.4	26.3	28.2	–	15.4	20.8	23.3	25.1
ARM$_{L}$	31.8	**49.2**	**59.2**	**53.6**	65.3	33.0	**48.4**	**58.4**	53.2	64.7
ARM$_{Q}$	**33.4**	47.2	57.0	52.8	64.2	**34.4**	47.4	56.9	53.3	64.0

Table 3: Cost of agentic RAG baselines relative to ARM executed using Llama-3.1-8B-Instruct. See Table 8 for details on the number of tokens and absolute cost.

	UAE-Large-V1						Snowflake-arctic-embed-m-v2.0					
	Llama-3.1-8B-Instruct		Qwen-2.5-7B-Instruct		GPT-4o-mini		Llama-3.1-8B-Instruct		Qwen-2.5-7B-Instruct		GPT-4o-mini	
	ReAct	IRCoT	ReAct	IRCoT	ReAct	IRCoT	ReAct	IRCoT	ReAct	IRCoT	ReAct	IRCoT
Bird	5.34x	–	2.27x	–	3.05x	–	5.18x	–	2.06x	–	2.61x	–
OTT-QA	4.06x	1.49x	2.39x	0.27x	3.38x	2.01x	4.09x	1.49x	2.42x	0.28x	3.63x	2.22x
Wikihop	1.44x	0.88x	1.06x	0.23x	1.32x	2.20x	1.36x	0.87x	1.05x	0.24x	1.27x	2.27x

4.4 End-to-end performance

Table 2 shows the end-to-end results on all datasets across all approaches while Table 3 shows the overall cost of agentic RAG baselines relative to ARM.

Averaging across all models, ARM demonstrates superior performance on all datasets. On Bird, it outperforms the best-performing standard RAG baseline, dense retrieval, by 2.94 points in execution accuracy and agentic RAG by 10.9 points. On OTT-QA, ARM achieves 3.27 points higher exact match and 3.59 points higher F1 match compared to dense retrieval with query decomposition and reranker, while outperforming the agentic RAG by 10.1 points in exact match and 9.2 points in F1 match. On 2WikiMultiHop, ARM achieves 12.0 points higher exact match and 15.0 points higher F1 match compared to dense retrieval with reranker, while outperforming the agentic RAG by 8.1 points in exact match and 4.6 points in F1 match. Moreover, compared to ARM, the highest-performing agentic RAG baseline, ReAct, incurs significantly higher costs—405.9% more on Bird, 204.5% more on OTT-QA, and 279.8% more on 2WikiMultiHop.

The results show that ARM outperforms standard RAG baselines by retrieving higher-quality objects, enhancing downstream performance while retrieving a similar number of objects. Moreover, compared to agentic RAG baselines, ARM

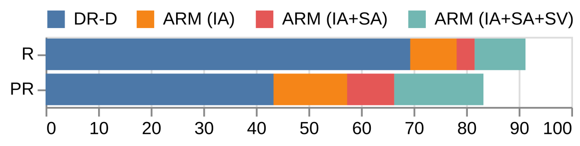


Figure 4: The average recall and perfect recall of dense retrieval with decomposition (DR-D) and our method with successive modules: information alignment (IA), structure alignment (SA), and self-verification and aggregation (SV) for the top-5 retrieved objects across all datasets.

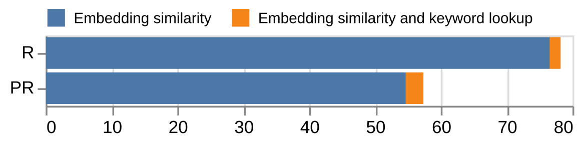


Figure 5: The average recall and perfect recall for the information alignment module evaluated using embedding similarity alone and when combined with keyword lookup for the top-5 retrieved objects across all datasets.

achieves superior downstream performance at a significantly lower cost. This highlights ARM as a more *effective* and *efficient* solution.

4.5 ReAct analysis

We manually analyzed 50 randomly sampled questions per dataset from ReAct, the best-performing agentic RAG approach. Our analysis identified two main errors in models' iterative reasoning process: forgetting previously generated information and repeatedly searching for similar keywords despite retrieving relevant objects. These issues lead to inefficiency, increasing retrieval iterations and cost. Detailed examples can be found in Appendix E.

4.6 Ablation studies

Significance of different modules. ARM includes three modules: information alignment, structure alignment, and self-verification and aggregation. Figure 4 illustrates the performance of dense retrieval with query decomposition and our method with successive modules for the top-5 retrieved objects. Information alignment involves decomposing the original question into keywords and retrieving relevant objects, similar to the baseline of dense retrieval with query decomposition. To highlight the benefits of information alignment, we compare the two methods. On average, information alignment outperforms dense retrieval with query decomposition by 8.83 points in recall and 14.0 points in perfect recall across all datasets. The inclusion of structure alignment boosts recall by 3.42 points and perfect recall by 8.96 points, building on the gains from information alignment. Finally, the complete method with all three modules enhances recall by 9.68 points and perfect recall by 17.0 points. The improvement with each successive module demonstrates the contribution of every module to the overall retrieval performance.

Information alignment. Information alignment retrieves relevant objects through two components: keyword lookup using the decomposed and aligned keywords via BM25, and embedding similarity. Figure 5 illustrates the performance with embedding similarity alone and when combined with keyword lookup. Adding keyword lookup to embedding similarity improves recall by 1.67 points and perfect recall by 2.70 points on average across all datasets, clearly demonstrating the contribution of each component.

5 Conclusion

Understanding the available data objects in a collection and their organization is essential for answering complex open-domain questions that involve heterogeneous information sources. Off-the-shelf LLMs decompose queries without considering available data in the collection, leading to a suboptimal retrieval. While agentic RAG can interact with the data collection, it generates queries based on prior retrieval results rather than the available data objects and their organization. Therefore, it is often inefficient, requiring more iterations to retrieve all necessary data objects. In this work, we propose ARM, an alignment-oriented retrieval method that identifies key data objects needed to answer a question and navigates their data organization to retrieve all relevant objects, even if not explicitly mentioned in the query. Experiments demonstrate that ARM outperforms baselines in retrieval and downstream performance while being more efficient in retrieval iterations and cost.

6 Limitations

First, we acknowledge the limitations of our chosen evaluation datasets. While our retrieval framework, along with the data representation method in Section 3.1, is adaptable to various modalities, our experiments primarily focus on text and tables. Developing a benchmark dataset that requires joint reasoning across text, tables, and images, as well as evaluating our method on such data, is left for future work. Secondly, our method is only compatible with open-source models and cannot be applied to models that are exclusively accessible via APIs.

Acknowledgments

We gratefully acknowledge the support of DARPA ASKEM Award HR00112220042, the ARPA-H Biomedical Data Fabric project, a grant from Liberty Mutual, the Office of Naval Research (ONR N00014-23-1-2364), and the Croucher Scholarship.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Peter Baile Chen, Fabian Wenz, Yi Zhang, Moe Kayali, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2024b. Beaver: An enterprise benchmark for text-to-sql. *arXiv preprint arXiv:2409.02038*.

Peter Baile Chen, Yi Zhang, and Dan Roth. 2024c. Is table retrieval a solved problem? exploring join-aware multi-table retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2687–2699.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. 2020. Open question answering over tables and text. *arXiv preprint arXiv:2010.10439*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Palak Jain, Livio Baldini Soares, and Tom Kwiatkowski. 2024. From rag to riches: Retrieval interlaced with sequence generation. *arXiv preprint arXiv:2407.00361*.

Harsh Jhamtani, Hao Fang, Patrick Xia, Eran Levy, Jacob Andreas, and Ben Van Durme. 2023. Natural language decomposition and interpretation of complex utterances. *arXiv preprint arXiv:2305.08677*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. 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.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506.*

Weinan Zhang, Junwei Liao, Ning Li, and Kounianhua Du. 2024. Agentic information retrieval. *arXiv preprint arXiv:2410.09713.*

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625.*

A Serialization and chunking

Serialization. In our data collection, each passage chunk is serialized as a concatenation of its passage title and the content of the chunk. A table chunk is serialized as a concatenation of its table name, title, description (if any), and rows.

For instance, a serialized document is as follows:

```
Document id: /wiki/Ajim
Document name: /wiki/Ajim
Document content: Ajim ( Arabic : Ağīm
) is a commune and port located on the
Island of Djerba off the coast of Tunisia.
It is Djerba 's main fishing port and the
closest city to the African continent.
It had a population of 24,294 at the 2014
census.
```

An example of a serialized table is as follows (table content is represented as a markdown):

```
Table id: IPSC_European_Handgun_
Championship_2
Table name: IPSC European Handgun
Championship
Table description: The IPSC European
Handgun Championship is an IPSC level 4
championship hosted every third year in
Europe.
Champions - Junior category

Table content:
| Year | Division | Gold | Silver |
Bronze | Venue |
|---:|:----|:------|:-------|:---|:----|
| 2013 | Open | Simo Partanen | Daniil
Karchev | Francois Belloni | Barcelos ,
Portugal |
| 2013 | Standard | Kenneth Handberg |
Sotirios-Thomas Zafeiridis | Mats Selven
| Barcelos , Portugal |
| 2013 | Production | Pavel Torgashov |
Konstantin Kryuchin | Nikita Kryuchin |
Barcelos , Portugal |
| 2016 | Standard | Danila Pakhomov | Ilya
Sologub | Kirill Fedorov | Felsőtárkány
, Hungary |
```

N-Grams and Embeddings. We constructed N-grams for each table and passage chunk, varying N between one to three. Moreover, we computed the embedding for each table and passage chunk using the serialized objects.

B Mixed-integer program

The objective function is subject to the following constraints:
Constraint 1 ensures decision variables are binary

$$b_i, c_{ij} \in \{0, 1\} \tag{1}$$

Constraint 2 sets the maximum number of objects and encourages selected objects to be connected.

$$\sum_i b_i = k, \sum_{i,j} c_{ij} \leq 2(k-1) \tag{2}$$

Constraint 3 ensures that only connections between selected objects are considered.

$$2c_{ij} \leq o_i + o_j, \forall i, j \tag{3}$$

Object-object compatibility between two objects O_i and O_j, denoted as C_{ij}, measures the strength of connections between objects. It is computed based on both semantic similarity (cosine similarity of embeddings) and exact-value similarity between contents from both objects.

Table-table compatibility is determined by pairwise column comparisons between the two tables. Since only one pair of columns is necessary to connect two tables, table compatibility is determined by the highest compatibility score among all possible column pairs. Each column-column compatibility is calculated as the weighted sum of the semantic similarity between the column headers and the exact-value similarity (Jaccard similarity) of the column rows.

Table-passage compatibility is determined by comparing all cells within a table and all sentences in a passage and taking the pair with the highest compatibility. Each cell-sentence compatibility is calculated as the weighted sum of the semantic similarity and exact-value similarity (overlap coefficient) between the cell content and the sentence.

Passage-passage compatibility is computed by comparing all sentences in a passage with all sentences in the other passage and taking the pair with the the highest compatibility. Each sentence-sentence compatibility is calculated as the weighted sum of semantic similarity and exact-value similarity (overlap coefficient).

C Draft verbalization

Each draft that includes both objects and their connections and is verbalized as follows:

- Objects: each object is serialized in the same way as described in Appendix A, but only the k ($k = 5$ in our setting) most similar (based on embedding similarity) table rows or passage sentences are selected, so that the model can focus on the most relevant content.
- Connections: connections that represent joinable columns between tables are serialized as "`column {column name} in {table name} connects with column {column name} in {table name}`". Connections that represent connecting entities between tables/ documents and documents are serialized as "`{table name}/{document name} includes {cell}/"{sentence}" which connects with sentence "{sentence}" in {passage name}`".

D Experimental setup

D.1 Dataset details

For Bird, we constructed the data collection by merging tables from all databases used by the dev questions. For 2WikiMultiHop, we randomly sampled 1800 questions from the dev set and merged all supporting and non-supporting passages associated with each question. Similarly, for OTT-QA, we constructed the data collection by merging the tables and passages used to answer the questions in the dev set. In our experiments, we removed the questions with incomplete annotations (missing either required tables or passages through our manual inspection) from OTT-QA. In total, there are 1800 questions and 10640 documents for 2WikiMultiHop, 1834 questions and 3862 objects (3073 passages and 789 tables) for OTT-QA, and 1534 questions and 75 tables in the data collection for Bird. After chunking (described in Section 3.1), there are 10640 chunks, 4407 chunks, and 249515 chunks for 2WikiMultiHop, OTT-QA, and Bird, respectively.

D.2 Baselines details

All objects are chunked and serialized in the way described in Section 3.1. Dense retrievers compute the embedding for each chunk and outputs the top-k objects with the highest cosine similarity with the user question. If an object is divided into multiple chunks, its similarity score is the highest similarity score across all its chunks. For the reranking model, it was provided with the top-50 objects retrieved using dense retrievers. The reranker model assigns a score for each pair of user question and object, and it outputs top-k objects with the highest scores. When query decomposition is applied, 30 objects were retrieved for each sub-question using dense retrievers and further reranked to output the top-k objects.

ReAct was implemented following the original design of interleaving thought, action, and observation. A *thought* step allows models to reason about the current situation. An *action* can be of two types: (1) the model can generate some keywords to search for relevant objects from the corpus (2) or finish generation with an answer. An *observation* step involves calling a dense retriever, which retrieves the 5 serialized objects with the highest similarity to the model-generated keywords in *action*. Because most questions in both datasets can be answered using 4 objects, we set the maximum number of iterations to 8. The process continues until either the answer is found or the maximum limit of 8 rounds is reached. Table 5 contains the 3-shot prompts used for ReAct.

IRCoT was implemented following the original design of interleaving chain-of-thought (CoT) reasoning with retrieval, where the last CoT sentence is used for object retrieval. Similar to the implementation for ReAct, each retrieval step uses a dense retriever to retrieve the 5 serialized objects with the highest similarity to the last CoT sentence. The iterative process continues until an answer is found or the 8-round limit is reached. Table 6 contains the 3-shot prompts used for IRCoT.

E Examples where models fail using ReAct

Below are examples where models failed to generate correct answers using ReAct.

As seen in Table 9, the model can forget information generated in previous iterations. It was trying to search the population of Barcelos, but concluded with the population of Ajim.

As seen in Table 10, the model fell into a loop of searching using similar keywords, even when gold tables have already been retrieved. The gold tables are `financial.card`, `financial.disp`, `financial.client`

Table 4: 3-shot prompt used for query decomposition.

You are given a user question, your task is to decompose the user question into simpler sub-questions. The sub-questions should be separated by newline characters.
Here are some examples.

User question: what is the full name of the jesus college alumni who graduated in 1960 ?
Sub-questions:
Who are the alumni of Jesus College?
Which alumni of Jesus College graduated in 1960?
What are the full names of those who graduated from Jesus College in 1960?

User question: the home stadium of the bray wanderers of 2006 league of ireland is situated behind what station ?
Sub-questions:
What is the home stadium of the Bray Wanderers in the 2006 League of Ireland?
Where is the Bray Wanderers' home stadium located geographically?
Which station is situated in front of this stadium?
What is the name of the station that is located behind the home stadium of the Bray Wanderers?

User question: what is the enrollment of the school with a delta sigma theta chapter chartered on may 8 , 1971 ?
Sub-questions:
What is Delta Sigma Theta?
Which schools have a Delta Sigma Theta chapter?
Which schools have a Delta Sigma Theta chapter chartered on May 8, 1971?
What is the enrollment of the school that has a Delta Sigma Theta chapter chartered on May 8, 1971?

User question: {user question}

Table 5: 3-shot prompt used for ReAct.

```
Solve a question answering task with interleaving Thought, Action, Observation steps. A Thought step must be followed by a
Action step, a Action step must be followed by a Observation step, and a Observation step must be followed by a Thought step.
Thought can reason about the current situation, and Action can be of two types:
(1) Search[keyword], which returns five new documents or tables relevant to the keywords.
(2) Finish[answer], which returns the answer and finishes the task.
You answer should be a short answer to the user question.
Here are some examples.

User question: what is the full name of the jesus college alumni who graduated in 1960 ?
Thought: I need to search jesus college alumni, find the people that graduated in 1960, then find their full names
Action: Search[jesus college aumni]
Observation:
(...5 tables omitted...)

Thought: The person who graduated in 1960 is not mentioned in any table. So I need to see more content relevant to jesus
college alumni to decide.
Action: Search[jesus college alumni]
Observation:
(...5 tables omitted...)

Thought: The person who graduated in 1960 is not mentioned in any table. So I need to see more content relevant to jesus
college alumni to decide.
Action: Search[jesus college alumni]
Observation:
(...2 tables omitted...)
Table id: List_of_politicians,_lawyers,_and_civil_servants_educated_at_Jesus_College,_Oxford_1
Table name: List of politicians, lawyers, and civil servants educated at Jesus College, Oxford
Table description: Jesus College is one of the constituent colleges of the University of Oxford in England. ...
Table content:
| Name | M | G | Degree | Notes |
|:——————–|——:|——:|:————————————————————|:——————————————————————
—————————————————————————————————|
| Lalith Athulathmudali | 1955 | 1960 | BA Jurisprudence ( 2nd , 1958 ) , BCL ( 2nd , 1960 ) | President of the Oxford Union (
1958 ) ; a Sri Lankan politician ; killed by the Tamil Tigers in 1993 |
| Neal Blewett ( HF ) | 1957 | 1959 | BA PPE ( 2nd ) | Member of the Australian House of Representatives ( 1977-1994 ) ,
Government Minister ( 1983-1994 ) , High Commissioner to the UK ( 1994-1998 ) |
(...2 tables omitted...)

Thought: Table List_of_politicians,_lawyers,_and_civil_servants_educated_at_Jesus_College,_Oxford_1 lists Lalith Athulath-
mudali as receiving his BCL degree in 1960 from jesus college. So the person who graduated in 1960 is Lalith Athulathmudali. I
need to search Lalith Athulathmudali to find their full name.
Action: Search[Lalith Athulathmudali]
Observation:
Document id: /wiki/Lalith_Athulathmudali
Document name: /wiki/Lalith_Athulathmudali
Document content: Lalith William Samarasekera Athulathmudali , PC ( Sinhala : ; 26 November 1936 - 23 April 1993 ) , known
as Lalith Athulathmudali , was Sri Lankan statesman . He was a prominent member of the United National Party , who served as
Minister of Trade and Shipping ; Minister National Security and Deputy Minister of Defence ; Minister of Agriculture , Food
and Cooperatives and finally Minister of Education . Following a failed impeachment of President Premadasa , he was removed
from the UNP and formed his own party .
(...4 documents omitted...)

Thought: Document Lalith_Athulathmudali provides the full name: Lalith William Samarasekera Athulathmudali. So the answer
is Lalith William Samarasekera Athulathmudali.
Action: Finish[Lalith William Samarasekera Athulathmudali]

(...2 examples omitted...)

User question: {user question}
```

Table 6: 3-shot prompt used for IRCoT.

```
(...3 documents omitted)...
Table id: List_of_politicians,_lawyers,_and_civil_servants_educated_at_Jesus_College,_Oxford_1
Table name: List of politicians, lawyers, and civil servants educated at Jesus College, Oxford
Table description: Jesus College is one of the constituent colleges of the University of Oxford in England. ...
Table content:
| Name | M | G | Degree | Notes |
|:————————-|——:|——:|:————————————————————————|:——————————————————————————————————————————————————————————————————|
| Lalith Athulathmudali | 1955 | 1960 | BA Jurisprudence ( 2nd , 1958 ) , BCL ( 2nd , 1960 ) | President of the Oxford Union ( 1958 ) ; a Sri Lankan politician ; killed by the Tamil Tigers in 1993 |
| Neal Blewett ( HF ) | 1957 | 1959 | BA PPE ( 2nd ) | Member of the Australian House of Representatives ( 1977-1994 ) , Government Minister ( 1983-1994 ) , High Commissioner to the UK ( 1994-1998 ) |

Document id: /wiki/Lalith_Athulathmudali
Document name: /wiki/Lalith_Athulathmudali
Document content: Lalith William Samarasekera Athulathmudali , PC ( Sinhala : ; 26 November 1936 - 23 April 1993 ) , known as Lalith Athulathmudali , was Sri Lankan statesman . He was a prominent member of the United National Party , who served as Minister of Trade and Shipping ; Minister National Security and Deputy Minister of Defence ; Minister of Agriculture , Food and Cooperatives and finally Minister of Education . Following a failed impeachment of President Premadasa , he was removed from the UNP and formed his own party .

User question: what is the full name of the jesus college alumni who graduated in 1960 ?
Answer: The jesus college alumni who graduated in 1960 is Lalith Athulathmudali. The full name of Lalith Athulathmudali is Lalith William Samarasekera Athulathmudali. Finish[Lalith William Samarasekera Athulathmudali]

(...2 examples omitted...)

User question: {user question}
```

Table 7: 3-shot prompt used for ARM.

```
You are given a user question, your task is to decompose the user question into contiguous, non-overlapping substrings that can cover different information mentioned in the user question. For each substring, generate n-grams that are the most relevant to the substring. Based on the generated relevant n-grams, generate a list of relevant objects, including their names, content, and connections between these objects. From these candidate objects, you should identify the minimum number of objects that can be used to answer the user question based on the relevance between the object name, object content and user question as well as the relevance of the object connections. You should end your response with <>.

User question: what is the full name of the jesus college alumni who graduated in 1960 ?
The relevant keywords are full name | jesus college | alumni | graduated | 1960
The relevant n-grams are full name ( name) | jesus college ( jesus college) | alumni ( alumni, former, university) | graduated ( degree, educated, postgraduate) | 1960 ( 1960)

Here are the objects that can be relevant:
(...4 tables omitted...)
Table id: List_of_politicians,_lawyers,_and_civil_servants_educated_at_Jesus_College,_Oxford_1
Table name: List of politicians, lawyers, and civil servants educated at Jesus College, Oxford
Table description: Jesus College is one of the constituent colleges of the University of Oxford in England. ...
Table content:
| Name | M | G | Degree | Notes |
|:———————-|——:|——:|:——————————————————————|:——————————————————————
————————————————————————————————————————————|
| Lalith Athulathmudali | 1955 | 1960 | BA Jurisprudence ( 2nd , 1958 ) , BCL ( 2nd , 1960 ) | President of the Oxford Union ( 1958 ) ; a Sri Lankan politician ; killed by the Tamil Tigers in 1993 |
| Neal Blewett ( HF ) | 1957 | 1959 | BA PPE ( 2nd ) | Member of the Australian House of Representatives ( 1977-1994 ) , Government Minister ( 1983-1994 ) , High Commissioner to the UK ( 1994-1998 ) |
(...2 documents omitted...)
Document id: /wiki/Lalith_Athulathmudali
Document name: /wiki/Lalith_Athulathmudali
Document content: Lalith William Samarasekera Athulathmudali , PC ( Sinhala : ; 26 November 1936 - 23 April 1993 ) , known as Lalith Athulathmudali , was Sri Lankan statesman . He was a prominent member of the United National Party , who served as Minister of Trade and Shipping ; Minister National Security and Deputy Minister of Defence ; Minister of Agriculture , Food and Cooperatives and finally Minister of Education . Following a failed impeachment of President Premadasa , he was removed from the UNP and formed his own party .
(...2 documents omitted...)

Here are the potential connections between these objects:
List_of_politicians,_lawyers,_and_civil_servants_educated_at_Jesus_College,_Oxford_1 includes Lalith Athulathmudali which connects with sentence “Lalith William Samarasekera Athulathmudali , PC ( Sinhala : ; 26 November 1936 - 23 April 1993 ) , known as Lalith Athulathmudali , was Sri Lankan statesman .” in /wiki/Lalith_Athulathmudali
(...4 connections omitted...)

Here I summarize the relevant information to answer the user question
<table> List_of_politicians,_lawyers,_and_civil_servants_educated_at_Jesus_College,_Oxford_1 </table> lists Lalith Athulathmudali as receiving his BCL degree in 1960 from jesus college, which covers the keywords "jesus college alumni graduated 1960" | <document> /wiki/Lalith_Athulathmudali </document> extends on this by providing his full name: Lalith William Samarasekera Athulathmudali, which covers the keywords "full name" <>

(...2 examples omitted...)

User question: {user question}
```

Table 8: Number of input tokens, output tokens, and absolute cost for each method. Cost is calculated based on OpenAI's pricing for GPT-4o-mini: \$0.150/1M input tokens, \$0.600/1M output tokens. Costs are multiplied by 1000 for better display.

	Bird			OTT-QA			2WikiMultiHop		
	#input tok. ↓	#output tok. ↓	cost ↓	#input tok.	#output tok.	cost	#input tok.	#output tok.	cost
Embedding model: UAE-Large-V1, Answer generation: Llama3.1-8B-Instruct									
ReAct	160877.7	644.5	24.52	117382.9	388.4	17.84	19861.7	239.0	3.12
IRCoT	–	–	–	42947.7	189.8	6.56	11885.0	193.3	**1.90**
ARM$_{\mathrm{L}}$	28474.2	523.6	**4.59**	26076.0	795.2	**4.39**	11833.5	651.6	2.17
Embedding model: UAE-Large-V1, Answer generation: Qwen-2.5-7B-Instruct									
ReAct	67974.5	291.2	10.37	69130.0	208.4	10.49	14566.1	160.2	2.28
IRCoT	–	–	–	7622.5	89.7	**1.20**	3040.1	59.9	**0.49**
ARM$_{\mathrm{L}}$	28474.2	497.7	**4.57**	26076.0	790.5	4.39	11833.5	647.5	2.16
Embedding model: UAE-Large-V1, Answer generation: GPT-4o-mini									
ReAct	92301.1	276.5	14.01	97549.2	256.9	14.79	18237.5	188.8	2.85
IRCoT	–	–	–	57557.8	249.5	8.78	30663.4	260.0	4.76
ARM$_{\mathrm{L}}$	28617.5	495.7	**4.59**	25973.9	791.3	**4.37**	11824.8	646.7	**2.16**
Embedding model: Snowflake-arctic-embed-m-v2.0, Answer generation: Llama-3.1-8B-Instruct									
ReAct	165887.4	691.7	25.30	119261.0	401.1	18.13	18727.2	236.4	2.95
IRCoT	–	–	–	43400.1	184.7	6.62	11765.6	192.6	**1.88**
ARM$_{\mathrm{L}}$	30304.1	553.5	**4.88**	26286.4	805.8	**4.43**	11808.9	657.0	2.17
Embedding model: Snowflake-arctic-embed-m-v2.0, Answer generation: Qwen-2.5-7B-Instruct									
ReAct	65812.0	288.8	10.05	70552.6	211.9	10.71	14454.4	160.9	2.26
IRCoT	–	–	–	7748.8	90.1	**1.22**	3155.9	61.8	**0.51**
ARM$_{\mathrm{L}}$	30304.1	537.9	**4.87**	26286.4	801.0	4.42	11808.9	653.6	2.16
Embedding model: Snowflake-arctic-embed-m-v2.0, Answer generation: GPT-4o-mini									
ReAct	83968.4	267.6	12.76	105764.0	265.3	16.02	17552.6	185.4	2.74
IRCoT	–	–	–	64113.1	262.5	9.77	31678.8	266.8	4.91
ARM$_{\mathrm{L}}$	30475.0	533.9	**4.89**	26185.0	802.1	**4.41**	11801.4	653.0	**2.16**

Table 9: Example where models forget information generated in previous iterations using ReAct.

User question: how many people live in the city where the first ipsc european handgun championship , junior category , event took place ?

Thought: I need to search the first IPSIC European Handgun Championship, Junior Category, to find the event location, then search the city to find its population.
Action: Search[IPSC European Handgun Championship, Junior Category]
Observation:
Table id: IPSC_European_Handgun_Championship_2
Table name: IPSC European Handgun Championship
Table description: The IPSC European Handgun Championship is an IPSC level 4 championship hosted every third year in Europe.
Champions – Junior category

Table content:

Year	Division	Gold	Silver	Bronze	Venue
2013	Open	Simo Partanen	Daniil Karchev	Francois Belloni	Barcelos , Portugal
2013	Standard	Kenneth Handberg	Sotirios-Thomas Zafeiridis	Mats Selven	Barcelos , Portugal
2013	Production	Pavel Torgashov	Konstantin Kryuchin	Nikita Kryuchin	Barcelos , Portugal
2016	Standard	Danila Pakhomov	Ilya Sologub	Kirill Fedorov	Felsőtárkány , Hungary

(...4 objects omitted...)

Thought: Table IPSC_European_Handgun_Championship_2 lists the venue for 2013 event in Barcelos, Portugal. So the event took place in Barcelos, Portugal. I need to search Barcelos to find its population.
Action: Search[**Barcelos**]
(...retrieved objects omitted...)

Thought: The content of the documents includes no information about population. I need to search Barcelos to find its population.
Action: Search[**Barcelos**]
(...retrieved objects omitted...)

Thought: The content of the documents includes no information about population. I need to search Barcelos to find its population.
Action: Search[**Barcelos**]
(...retrieved objects omitted...)

Thought: The content of the documents includes no information about population. I need to search Barcelos to find its population.
Action: Search[**Barcelos**]
Observation:
(...four objects omitted...)
Document id: /wiki/Ajim
Document name: /wiki/Ajim
Document content: Ajim (Arabic : Ağīm) is a commune and port located on the Island of Djerba off the coast of Tunisia . It is Djerba 's main fishing port and the closest city to the African continent . It had a population of 24,294 at the 2014 census .

Thought: Document **/wiki/Ajim** lists a population of 24,294 at the 2014 census. So the population is 24,294.
Action: Finish[24,294]

Table 10: Example where models loop on similar keywords using ReAct. Gold tables are bolded.

User question: provide the ids and age of the client with high level credit card, which is eligible for loans.

Thought: I need to find tables that can cover client ids, age, credit card level, and loan eligibility. So I should search for client information, credit card details, and loan eligibility status.
Action: Search[client information, credit card details, loan eligibility]
Observation:
Table name: **financial.client**
client_id	gender	birth_date	district_id
————:	:———	:————-	—————:
1039	F	1980-10-11	70

Table name: financial.loan
loan_id	account_id	date	amount	duration	payments	status
———-:	————-:	:————	———:	————:	————:	:———
7264	11141	1997-10-07	44940	12	3745	A

Table name: **financial.disp**
disp_id	client_id	account_id	type
———-:	————:	————-:	:————
525	525	436	DISPONENT

Table name: debit_card_specializing.customers
CustomerID	Segment	Currency
————-:	:————-	:————
4643	SME	EUR

Table name: debit_card_specializing.products
ProductID	Description
————:	:———————————-
544	Guarantee Credit Note

Thought: The tables I have found do not include information about credit card levels or loan eligibility. I need to search for more specific tables that might contain credit card details and loan eligibility status.
Action: Search[credit card level, loan eligibility status]
Observation:
Table name: financial.loan
loan_id	account_id	date	amount	duration	payments	status
———-:	————-:	:————	———:	————:	————:	:———
7308	11362	1996-12-27	129408	24	5392	A

Table name: **financial.client**
client_id	gender	birth_date	district_id
————:	:———	:————-	—————:
1039	F	1980-10-11	70

Table name: debit_card_specializing.products
ProductID	Description
————:	:—————————
544	Guarantee Credit Note

Table name: **financial.card**
card_id	disp_id	type	issued
———-:	———-:	:——	:————
162	967	junior	1996-12-23

Table name: debit_card_specializing.customers
CustomerID	Segment	Currency
————-:	:————-	:————
123	SME	EUR

Thought: The tables I have found still do not include information about credit card levels or loan eligibility. I need to search for more specific tables that might contain credit card details and loan eligibility status.
Action: Search[credit card level, loan eligibility status]
(...retrieved tables omitted...)

Thought: I still haven't found the necessary information regarding credit card levels and loan eligibility. I need to search for more specific tables that might contain this information.
Action: Search[credit card level, loan eligibility status]
...