AU-RAG: Agent-based Universal Retrieval Augmented Generation

Jisoo Jang
simonjisu@snu.ac.kr
Graduate School of Data Science,
Seoul National University
Seoul, Republic of Korea

Wen-Syan Li
wensyanli@snu.ac.kr
Graduate School of Data Science,
Seoul National University
Seoul, Republic of Korea

Abstract

Retrieval Augmented Generation (RAG) has been effectively used to improve the accuracy of question-answering (Q&A) systems powered by Large Language Models (LLMs) by integrating local knowledge and more up-to-date content. However, traditional RAG methods, including those with re-ranking mechanisms, face challenges when dealing with large, frequently updated data sources or when accessing sources exclusively via APIs, as they require pre-encoding all content into embedding vectors. To address these limitations, we introduce Agent-based Universal RAG (AU-RAG), a novel approach that augments data sources with descriptive metadata, allowing an agent to dynamically search through diverse data pools. This agent-driven system can learn from examples to retrieve and consolidate data from various sources on the fly, functioning as a more flexible and adaptive RAG. We demonstrate AU-RAG's functionality with a financial analysis example and evaluate its performance using a multi-source QA dataset. The results show that AU-RAG performs comparably to RAG with re-ranking in data retrieval tasks while also demonstrating an enhanced ability to intelligently learn and access new data sources from examples, making it a robust solution for dynamic and complex information environments.

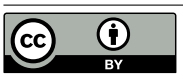
CCS Concepts

• **Information systems** → **Question answering**; **Wrappers (data mining)**; **Data exchange**.

Keywords

Retrieval Augmented Generation, Large Language Models, Agent, Mediation

ACM Reference Format:
Jisoo Jang and Wen-Syan Li. 2024. AU-RAG: Agent-based Universal Retrieval Augmented Generation. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP '24), December 9–12, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3673791.3698416



SIGIR-AP '24, December 9–12, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-0724-7/24/12
https://doi.org/10.1145/3673791.3698416

1 Introduction

Enterprise applications utilize various data sources in the business environment to improve their decision-making processes. The capacity to effectively collect, analyze, and interpret data is essential for extracting actionable insights and driving strategic initiatives.

Since the release of ChatGPT in November 2022 [1], integrating the Large Language Models (LLMs) into business operations has gained substantial popularity, particularly for question-and-answer interactions. People feel more comfortable asking questions in natural language, as it aligns more closely with everyday communication and reduces the barriers to accessing complex information. LLM Q&A interactions typically start with a user posing a question in natural language. The LLM then processes and generates a coherent and contextually relevant response based on its own knowledge of the LLM, which is the data used to train the model.

Context is crucial for LLM to answer users' questions accurately. Although LLM is trained on huge amounts of data, it could still lack domain-specific and up-to-date knowledge to answer business user's questions. One way to resolve this is to train a domain-specific LLM from scratch, such as BloombergGPT [2]. However, training a domain-specific model from scratch can be resource-intensive and time-consuming. Obviously, fine-tuning a pre-trained model with up-to-date knowledge is not realistic.

An alternative and more efficient approach is to enhance the quality of answers by incorporating relevant context into the input prompts. Retrieval-Augmented Generation (RAG) [3] is a method that combines the generative capabilities of LLM with information retrieval techniques to provide more accurate and contextually appropriate responses by augmenting the input prompt with pertinent data retrieved from external sources.

The current RAG methodology faces several significant challenges in adapting to an enterprise application environment, particularly in handling large-scale and heterogeneous data sources. The impact of noise and contradictory information present in the retrieval phase can substantially degrade the quality of the generated responses [4]. One of the reasons for this problem is made worse by the need to convert all contextual data into vectors for retrieval purposes. When dealing with extensive databases or multiple data sources, it becomes impractical to vectorize every piece of content due to the large volume of data. For example, copying every content item into vector form is unfeasible in a huge database environment. Another example is that some data sources provide access to their content through only APIs, so it is not feasible to vectorize the contents in advance. Furthermore, some data sources do not allow duplication of the whole data sources for vectorization due to copyright protection constraints. The other reason is the dependency on similarity search between vectors. Similarity search is a crucial step where vectors representing different pieces of information are

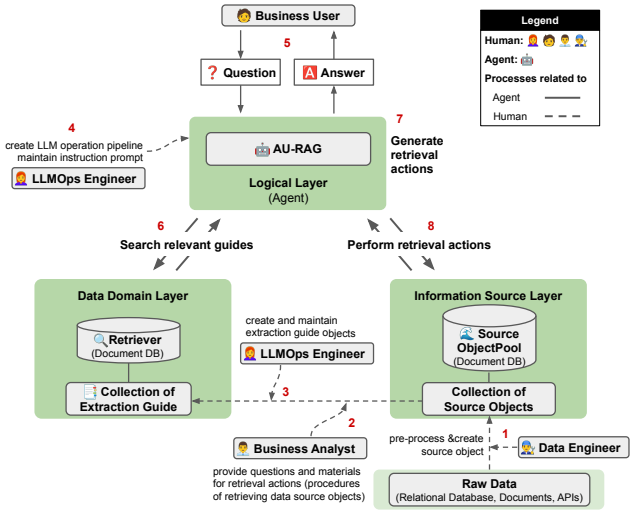


Figure 1: Overview of AU-RAG

compared to find the most relevant matches. It is hard to ensure retrieval accuracy by only relying on similarity measurements when the types of data sources can vary significantly, as we addressed above.

We aim to address two issues and enhance the current RAG system. First, we need to support information retrieval on a large-scale collection of heterogeneous data sources, including data sources supporting access via only restricted APIs beyond typical RAG systems that utilize a single and fixed knowledge base. Second, we need to automate the process to select the correct data source and extract the needed information from that source.

To address these challenges in an enterprise application environment, we propose a novel system: **A**gent-based **U**niversal **R**etrieval **A**ugmented **G**eneration (AU-RAG). Fig. 1 illustrates the interactions and processes within the AU-RAG system. It involves multiple user roles, including business users, LLMOps engineers, data engineers, and business analysts. Each role contributes to augmenting the metadata of the collection of the data sources for more effective information retrieval and generation of answers. Human activities are represented as dotted lines in Fig. 1. The core of AU-RAG is an agent. It can learn from previous experience or provided examples to adapt itself for the environment with a large and diverse collection of available data sources in an enterprise application environment.

The contributions of this paper are as follows:

- **Universal RAG:** Our system design enables efficient information retrieval from large and diverse data sources, eliminating the need to pre-encode all contextual data into vectors for retrieval. To achieve this, we propose a unified object-oriented data management framework, the Source Object Pool, which standardizes data sources in a unified format. Experimental results demonstrate that AU-RAG maintains higher precision in retrieval tasks compared to traditional RAG with a reranker, especially as the number of source objects increases, confirming the system's effectiveness in managing large-scale data environments.
- **Agent-based RAG:** AU-RAG employs an agent to generate precise retrieval actions by mimicking human logical operations, ensuring higher accuracy and relevance in responses. This agent-based system dynamically learns from examples to retrieve data from various source types. Our experiments show that AU-RAG consistently achieves similar F1 scores to RAG with reranking while demonstrating superior precision, even as recall decreases slightly. The integration of retrieval actions based on business analysts' selections facilitates tailored data extraction, meeting specific business needs with increased accuracy.
- **Improvement by dynamic demonstrations:** Our experiments reveal that increasing the number of extraction guide objects (EGs) improves the retrieval performance. AU-RAG benefits from the additional references provided by EGs, showing continuous improvements in precision, recall, and F1 scores as the number of EGs increases. Specifically, performance improves by 1.02% when increasing EGs from 30 to 90, and by a further 3.86% when increasing from 90 to 270, confirming that EGs play a crucial role in enhancing retrieval-augmented generation by providing relevant demonstrations.
- **Stakeholder-Centric System Architecture** AU-RAG is designed to meet the diverse needs of multiple stakeholders in a business environment, including LLMOps engineers, data engineers, business analysts, and business users.

The rest of the paper is organized as follows: In Section 2, we provide the background of user roles in supporting and using LLM-based enterprise applications and the overview of the AU-RAG system. In Section 3, we present the related work in the areas of federated databases, RAG, and agents. In Section 4, we describe the AU-RAG system step by step in detail. In Section 5, we use a financial analysis scenario as an example to show how AU-RAG works in solving a real-world problem. In Section 6, we present an experimental evaluation of AU-RAG using two publicly available data sets. Finally, we offer concluding remarks in Section 7.

2 Overview of Proposed System: AU-RAG

In this section, we start with the background and assumption of a business environment where LLM-based enterprise applications are developed and supported followed by the overview of AU-RAG.

In a business environment, we see there is a large collection of data sources to serve various business functions, ranging from manufacturing, corporate and field operations, sales and distribution, accounting and financial reporting, HR, R&D, etc. There are the following four roles that are relevant to the interaction and maintenance of LLM-based enterprise applications:

Business Users: Business users are experts in business logic and responsible for outcomes, though they may not understand backend systems or data semantics. They need access to all necessary business content for decision-making and can interact with enterprise applications using natural language.

Business Analysts: They are essential in connecting business requirements with technical solutions. They usually ask themselves questions for business analysis and utilize source objects to provide insights to answer questions. Their procedures for retrieving

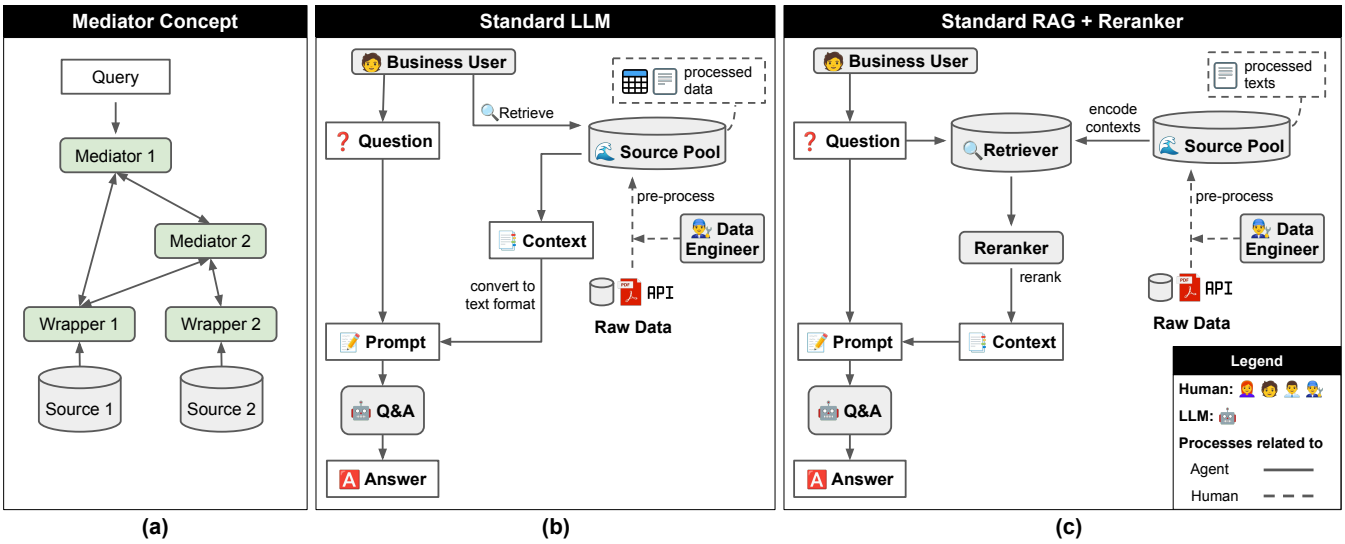


Figure 2: (a) A Network of Mediators and Data Sources from [5], (b) LLM with Standard RAG, and (c) LLM with RAG + Reranker

data, for example, selecting a source and extracting the needed information from that source, become the materials for retrieval actions.

Data Engineers: They are responsible for designing and maintaining the data infrastructure, including databases and knowledge bases. They handle the ETL (extraction, transformation, and load) process to turn raw data into a usable format.

LLMOps Engineers: Large Language Model Ops (LLMOps) [6] encompasses the practices and tools for managing large language models in production. LLMOps engineers oversee all tasks related to LLMs, including model training, performance monitoring, and tuning, as well as designing infrastructure for domain-specific knowledge bases and RAG systems. Prompt engineering, which guides generative AI outputs, can be handled by prompt engineers or LLMOps engineers. In our system, LLMOps engineers manage the LLM pipeline and combine questions with retrieval actions to create Extraction Guides based on business analysts' activities.

The process related to AU-RAG is represented as a solid line in Fig. 1. The architecture is divided into three layers as follows:

The Information Source Layer: This layer aggregates and manages various data sources in a unified object-oriented schema. It contains the Source Object Pool, a repository storing source objects. Data engineers create these objects by pre-processing raw data, including relational databases, documents, and APIs (Fig. 1 step 1). The source objects consist of the data contents with metadata like the ID, name, data types, and tags that point to the raw data. We classify source objects as static (rarely changing, such as text or tables) or dynamic (requiring updates for current information, like APIs or database data). Static objects include text chunks or tables, while dynamic objects have refresh methods to retrieve the latest content.

The Data Domain Layer: The purpose of this layer is to manage the extraction guide objects and support the retrieval process. Retriever is a document database that supports vector search and stores the objects. An extraction guide object contains a natural language question and multiple sources to answer the question. Each source includes source objects, retrieval actions, and retrieved contexts. We define "retrieval actions" as an action space for data extraction, equivalent to data retrieval tools applicable differently by data type. Retrieval actions are essential actions that mirror the logical thought process of humans for retrieving the data to answer the question. The collection of extraction guide objects are references for LLM to generate retrieval actions. Fig. 1 steps 2 and 3 show how they are created from the Information Source Layer by Business Analysts and LLMOps Engineers. Business analysts usually ask themselves questions for business analysis. The first step to finding the answer to the question is retrieving source objects. Their procedures for retrieving data, such as selecting a source object and extracting the needed information from it, become the materials for retrieval actions. LLMOps engineers will combine the question and retrieval actions to create an extraction guide object based on business analysts' activities.

AU-RAG: The core part of our system is an agent with several functions, such as searching extraction guide objects and performing retrieval actions at the Logical Layer. The process begins with a question from a business user, as shown in Fig. 1 step 5. AU-RAG searches for relevant extraction guide objects in the Data Domain Layer by semantic similarity between the user's question and questions in the extraction guide objects (Fig. 1 step 6). Next, AU-RAG generates retrieval actions based on the relevant extraction guide objects to find the correct sources to answer the question (Fig. 1 step 7). Then, AU-RAG performs the retrieval actions to find source objects from the Source Object Pool and extract relevant contexts (Fig. 1 step 8). While retrieving, AU-RAG evaluates whether the agent can answer the user's question with retrieved contexts by utilizing the examples from the extracted guides. If it passes the

evaluation, AU-RAG will generate the final answer. If not, it will repeat steps 7 and 8, which means trying to find other source objects to answer the question.

3 Related Work

In this section, we present the prior work in the areas of federated databases, RAG, and agents.

3.1 Information Retrieval Across Multiple Data Sources

Information retrieval across multiple sources refers to the process of searching, retrieving, and integrating information from various heterogeneous data sources. It is essential in today's data-rich environment, where valuable information is distributed across multiple systems, formats, and locations.

Federated databases [7, 8] are one of the approaches that provide a unified interface to multiple autonomous and heterogeneous databases. IBM Garlic project [9] aimed to develop a multimedia information system that integrates diverse data sources (both databases and non-databases) while maintaining their independence and avoiding data duplication, using an object-oriented schema and advanced database optimization techniques to manage and query heterogeneous collections of data servers efficiently. The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS, [5]) uses the concept of mediators to resolve the integration of heterogeneous information. The key components of integration are mediators and wrappers. Raw data sources are interfaced by wrappers, and mediators are used to answer queries by exchanging the object-exchange models.

The mediator concept, as shown in Fig. 2 (a), can be applied to integrating LLM into enterprise applications. As shown in Fig. 1, the mediator who does the logical operation is AU-RAG in the Logical Layer. The wrapper, the interface for the source objects, is the extraction guide object in the Data Domain Layer. Unlike traditional federated database approaches, AU-RAG utilizes natural language rather than specially designed query languages. Additionally, we use retrieval processes as wrappers instead of translators to communicate with source objects.

3.2 RAG Systems

The evolution of retrieval-augmented generation systems has seen significant advancements. Here, we compare the different methods for Q&A with LLM: Standard LLM, Standard RAG with Reranker, and AU-RAG shown in Fig. 2.

Standard LLM: It is the most straightforward method when the business user interacts with LLM. The user needs to retrieve relevant information from the Source Object Pool to gain a better answer. The data in the Source Object Pool is processed by the data engineer, usually as a text or table form. The retrieved information, generally called context, will be added to the prompt for the QA Agent (LLM) to answer the user's question. One of the weaknesses of this method is the limitation of input prompt length since LLM is still constrained by the maximum length of the input sequence. Even if LLM supports almost infinite lengths of inputs, the user should not put all the relevant information without any filtering into the context since it might be possible to get undesired answers. Liu et al. [10] show that model performance decreases when relevant information is in the middle of its input context. Humans need to retrieve and provide proper contexts to get the desired answer, which is also a pain point.

Standard RAG with Reranker: To improve the previous problems in the Standard LLM method, the Standard RAG with Reranker method tries to enhance the accuracy by searching relevant contexts with the semantic similarity and incorporating a reranking mechanism, which rearranges the relevant contexts based on their relevance to the user's question. In this scenario, the data in the Source Object Pool is also processed by the data engineer but is usually stored in text format. As mentioned in Section 1, in the enterprise application, it is not feasible to encode large-scale and various types of multiple data sources in advance. Also, the Reranker is usually an Interaction Model [11], and continually maintaining it is also a cost.

Compared to both methods, AU-RAG offers advantages by leveraging the mediator concept to access and integrate comprehensive and heterogeneous data sources. Our approach extends beyond simple similarity-based searching by incorporating sophisticated agent-searching capabilities. As a result, AU-RAG can provide more accurate and relevant information by effectively managing and querying diverse data formats and sources. However, this advanced functionality comes with inevitable trade-offs. The increased complexity of the system can lead to performance overhead, potentially slowing down the retrieval process. Additionally, the maintenance and scalability of AU-RAG can be challenging and costly, requiring ongoing adjustments and updates to handle the integration of new data sources and changes in existing ones. Despite these limitations, AU-RAG's ability to utilize a broader range of data sources and provide more nuanced and precise responses marks a significant improvement over traditional methods.

3.3 LLM as an Autonomous Agent

Autonomous agents powered by LLM are expected to handle various tasks by utilizing the human-like abilities inherent in LLM. The construction of LLM-based autonomous agents involves several key modules. The profiling module defines agent roles and characteristics. The memory module stores information for future use, mimicking human short-term and long-term memory, with operations like reading, writing, and reflecting. The planning module helps agents break down complex tasks into smaller steps, using either single-path reasoning or iterative refinement with feedback. The action module translates the agent's action into specific outcomes. The actions are pre-defined in the action spaces. These modules work together to enable effective agent functions [12].

AU-RAG focuses more on the action module, specifically the tool usage of LLM. LLM with tools demonstrated high capabilities to call external tools to solve more complex tasks, such as HuggingGPT [13] and ToolFormer [14]. We define "retrieval actions" as the action state space for data extraction. Each action state is a function call, which means a data retrieval tool that is applicable differently by data type and its arguments. For example, retrieval actions for tabular data are function calls of selecting specific rows or columns of a table, and retrieval actions for text data are function calls like choosing one or multiple sentences by index. Retrieval actions are

Raw Source	Contents Type of Source Object	Source Object Type	Applicable Retrieval Actions
Documents	Text chunk	Static	SelectSentenceIdxs
Documents	Table	Static	SelectCoordinates
Database	Table	Dynamic	SelectCoordinates
APIs	Text chunk	Dynamic	SelectSentenceIdxs

Table 1: Types of Source Objects and Contents

the sequence of mimicking the logical thinking of data analysts to gather data needed to answer questions.

4 Agent-based Universal RAG

In this section, we first introduce the Source Object Pool (Section 4.1) and extraction guide objects(Section 4.2), which provide information for the AU-RAG agent to determine what data sources are relevant to the question asked and how to extract information from the sources. The information retrieval is decomposed into three steps. Here we describe the details of these three steps and the Agent-based Universal Retrieval Augmented Generation (AU-RAG) process in Fig. 3.

4.1 Source Object Pool

Source objects are objects with descriptions of the sources for the agent to use as references when taking actions. The **Source Object Pool** is a repository where source objects are stored. Data engineers process various raw data, such as text and tables in documents, tables in databases, and API call outputs, into content for creating source objects.

We categorize source objects into two kinds (Table. 1). The first is the static source object, which rarely changes, like texts or tables in documents like PDF files. The second is the dynamic source object, which needs regular updates for the latest information, such as databases or APIs. The text-type source objects are document data (called text chunks) processed from the raw documents. They share common semantic meanings, such as the same topic, keywords, etc.

Tables from the documents are also static and processed as an object like Pandas Data Frame [15] that can operate column and row selection operations. Tables from relational databases are also stored as objects similar to static table-type source objects. API call outputs are stored as text-type source objects. Since most enterprise applications use text and tabular data, we assume text and table types are the default content types. Data engineers can also add other types, such as images.

The source object is the base unit for the information source. Each source object contains metadata such as ID, content type, tag, and description. In order to support the dynamic source object, it needs a `refresh` method to get the most up-to-date content. So, if the source object is a table from a database, it includes the SQL to load the table as metadata, and if the source object is an API output, it includes the API function as metadata. When calling the `refresh` method, the dynamic source object will call the SQL or API function within the metadata. The description of a source object is encoded into embeddings in a vector store, which is a part of the Source object Pool. The description embeddings are used in filtering steps for source selection (Fig, 3 step 3).

4.2 Extraction Guide Object

The extraction guide object is designed to facilitate the retrieval process. It consists of questions that business analysts ask themselves to drive decision-making and required sources to answer. Each source contains three attributes: source objects, retrieval actions defined in Section 3.3, and retrieved contexts, which are the output of retrieval actions. Each extraction guide object can transform into an Example object by extracting its attribute values for in-context learning examples. In-context Learning [16] is a method where language models can understand tasks by being shown only a few examples as demonstrations. Khattab, Omar, et al. [17] showed that retrieval-augmented in-context learning boosted retrieval performance by showing the demonstrations, which are training examples that have been designed to present specific expected behaviors. In this context, a single extraction guide object can work for two steps in the AU-RAG process to provide proper examples; one is source selection (Fig. 3 step 2), and the other is the extraction (Fig. 3 step 6).

Think Tank (Fig. 3 step 1) stores the extraction guide objects. It also encodes the question of each extraction guide object into embedding vectors. Given a new question, the retriever encodes the question into a query embedding vector and searches for semantically similar questions, similar to the retrieval process in RAG, to find out which extraction guides we can use to answer the question.

4.3 Problem Decomposition

We can decompose the information retrieval into three parts. The first three are steps for the algorithm, and the last one is a self-evaluation procedure.

(1) **Extraction Guide Retrieval:** What is the best extraction guide object to guide the agent?
(2) **Source Selection:** Where should the agent retrieve to answer the question? How does the agent evaluate the relevant source selection?
(3) **Extraction:** What (or which part) should the agent retrieve from the retrieved sources?

Let us define a natural language question q, the **Source Object Pool** $D_{pool} = \{d_1, \ldots, d_n\}$, and the **Think Tank** retriever R for searching extraction guide objects $G = \{g_1, \ldots, g_m\}$. Let us also use $o[\text{attr}]$ to represent an attribute of an object o. We represent multiple attributes by using $o[\text{attr}_1, \ldots, \text{attr}_k]$. For example, to represent the content and description embedding vector($e(\text{desc})$, e is embedding function) attributes of the source object is $d[\text{content}, e(\text{desc})]$. The extraction guide object g has attributes of question $g[q]$ and relevant multiple sources $S = g[s_1, s_2, \ldots]$. Each source s is also an object that has attributes of multiple source objects ($D = \{d_1, d_2, \ldots\}$) and retrieval actions ($A = \{a_1, a_2, \ldots\}$), it can be represented as $s[D, A]$. Retrieved contexts by retrieval actions are $C = \{c_1, c_2, \ldots\}$. Finally, we use x as an example object transformed from an extraction guide object g for the input prompt of LLM. We will describe each step in the following sub-sections.

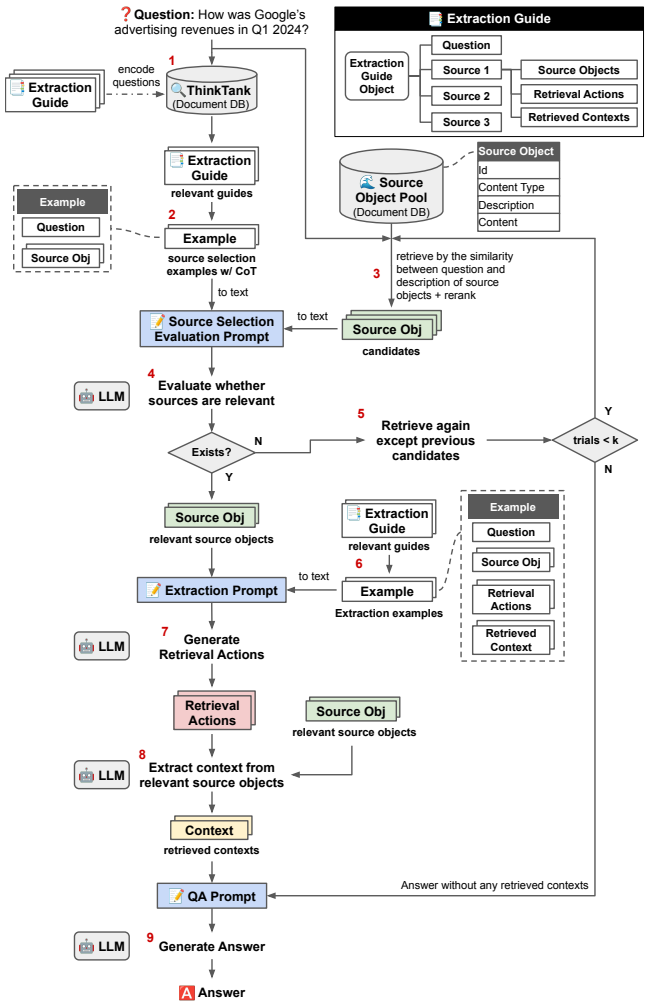


Figure 3: Process Flow of AU-RAG

4.4 Extraction Guide Retrieval

To find the best extraction guide object to guide the agent (Fig. 3 step 1), the retriever searches the relevant extraction guide object by embedding similarity between $e(q)$ and $e(g_i[q])$ for each extraction guide object g_i. This step is similar to the standard RAG process, but the outputs are relevant extraction guide objects (short for *RG*) rather than contexts. These relevant extraction guide objects are essential since they provide a "how-to" knowledge for LLM in two parts: evaluating the source selection and extracting the relevant contexts from the source objects.

4.5 Source Selection

To know where to retrieve to answer the question, an additional iterative evaluation procedure is adapted after the retrieval augmentation generation with a reranker. Steps 2 to 4 in Fig. 3 shows the source selection process. In step 2, the relevant extraction guide objects *RG* are converted to the example set for the source selection prompt: $X_{ss} = \{x_j[q, \{s_k[D] \mid \forall k \in \|S\|\}] \mid \forall j \in \|RG\|\}$. Step 3 is similar to the standard RAG process with a reranker. Let us use a filter function f for the RAG process to get source object candidates, which can be represented as $f(D_{pool})$. The filter function first calculates the embedding similarity between question $e(q)$ and the description of source object $e(d[\text{desc}])$ to get the source object candidates. Then, the reranker [11] rearranges the source object candidates. The system keeps tracking the candidates that are used for retrieval. In Step 4, the outputs of steps 2 and 3 are added to the source selection evaluation prompt as input for LLM to evaluate whether the source object is relevant to answer the question. Since the extraction guide objects only have the answer(target source objects) to the question, incorrect retrieval examples are manually added to the prompt for unbiased in-context learning examples. It is a binary classification problem for each source: $P(z|q, D)$ where D is source object candidates and $z \in (0, 1)$. Within k trials, the system keeps retrieving source objects from the source object pool D_{pool} if all the candidates are irrelevant. The output of the source selection process is relevant source objects *RD*.

4.6 Extraction

The extraction stage is a reasoning problem that determines what should be extracted from the relevant source objects *RD*. Steps 6 to 8 in Fig. 3 shows the extraction process. In step 6, the relevant extraction guide objects *RG* are converted to the example set for the extraction prompt: $X_{ex} = \{x_j[q, S[D, A]] \mid \forall j \in \|RG\|\}$. The relevant source objects *RD* and examples X_{ex} from step 6 are added to the extraction prompt as input. LLM needs to generate retrieval actions A' by learning from in-context demonstration examples X_{ex} about what to extract from the source objects. It can be represented as maximizing $P(A'|q, RD, X_{ex})$. In step 8, the system will perform retrieval actions to extract the relevant contexts (short for *RC*) from relevant source objects *RD*.

4.7 Answer Generation

The system generates the final answer using the retrieved context: $\sum_i^T P(y_i|q, RC)$ where y is the token and T is the length of the final answer.

5 Case Study

In this section, we use a real-world scenario in a financial research firm that utilizes data sources, including in-house databases, quarterly earning reports and earning release press conference transcripts released by all S&P 500 companies, and access to Bloomberg financial news service via API.

Fig. 4 illustrates how raw data is augmented (from left to right) and becomes an extraction guide object by data engineers, business analysts, and LLMOps engineers, respectively. On the left of Fig. 4, the data engineer processes various data types and creates source objects. In this example, an Earnings Report (10-Q) and Earnings Call Transcript, usually in a PDF document, are loaded and processed by a data engineer. The data engineer extracts the textual data and tables to create a static source object. The data engineer then splits the documents into smaller chunks for text-type data and clusters them with other chunks of similar semantic meanings. One way to do this is semantic chunking. Greg Kamradt et al. [18] used the 95 percentile of cosine distance (one minus cosine similarity) between preceding and following sentence embeddings as

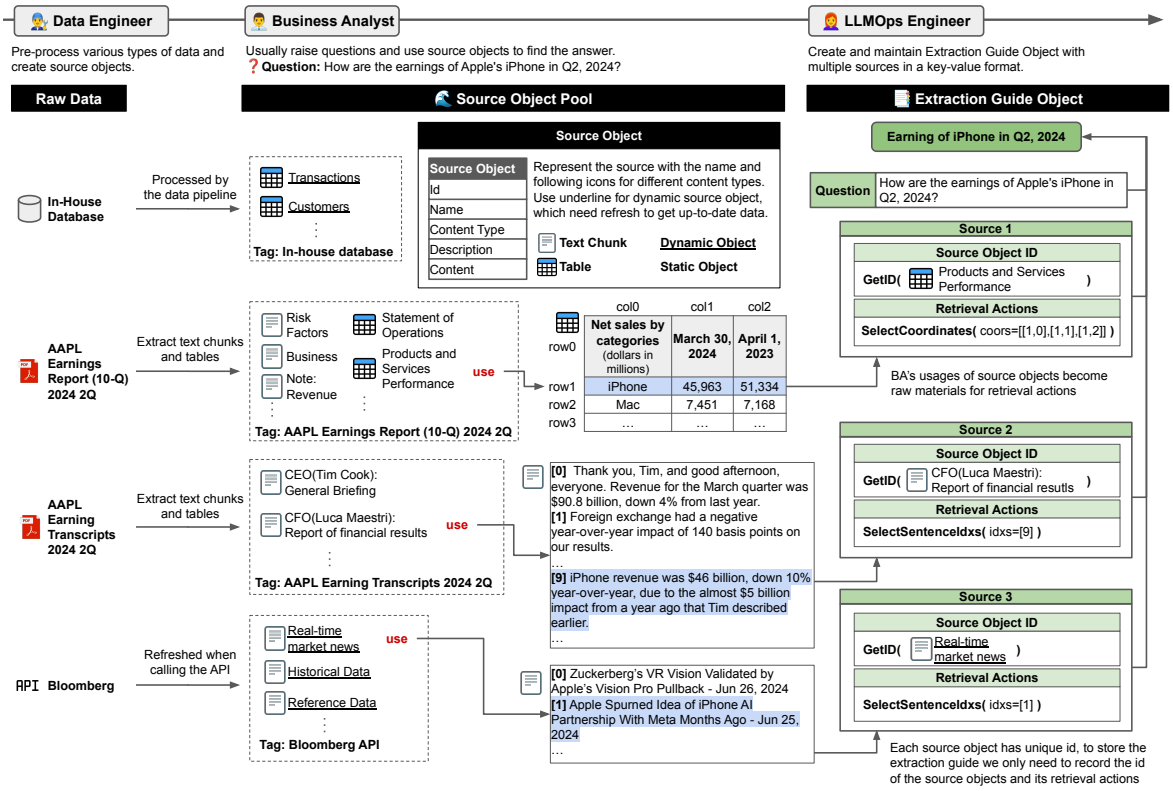


Figure 4: Case Study of How Raw Data Is Augmented and Become Extraction Guides

the break-point of a chunk. Tables from the in-house databases and outputs from Bloomberg API calls are processed and created as dynamic source objects.

In the middle of Fig. 4, A business analyst usually raises questions and uses source objects to find the answers. He asked "How are the earnings of Apple's iPhone in Q2, 2024?". Using three source objects: a table-type static source object (Products and Services Performance), a text chunk static source object (CFO Luca Maestri's Report), and a text chunk dynamic source object (Real-time market news). The analyst extracts relevant information, such as selecting rows and columns from the table, using data retrieval tools. These actions help LLMOps engineers create extraction guide objects. For example, the analyst selects the first row and three columns from the Products and Services Performance object for further revenue analysis of Apple.

On the right of Fig. 4, an LLMOps engineer creates and maintains the extraction guide object for the previously submitted and executed questions by the business analyst and corresponding sources in a key-value format. Each source in the extraction guide object also has source objects, retrieval actions, and retrieved contexts. For example, the question and three sources are associated and an extraction guide object named "Earning of iPhone in Q2, 2024" is created. The extraction guide objects can be viewed as logs and used as examples for later processing questions of similar semantic meaning and structure. For example, suppose a business user or business analyst wants to analyze the performance of major big-tech companies in the online advertising business to write a report for a customer's portfolio. In that case, the extraction guide objects previously created to answer the question about "Apple's iPhone business line earnings in Q2, 2024" can be used as examples (i.e., an extraction guide object) as the context and structure of the question are similar. Therefore, to find information to answer the question about Google's earnings in the advertising business, the agent can infer that it needs (1) Google (Alphabet) Earnings Report (10-Q) in 2024 Q1, (2) Google (Alphabet) Earnings Call Transcripts in 2024 Q1, (3) Bloomberg API for real-time market data and news. Instead of searching many resources to find the answer online, the agent asks the AU-RAG system and generates a plan for answering "How were Google's advertising revenues in Q1 2024?" by referencing a similar extraction guide object, and then the plan will be executed as described previously in Fig. 3.

6 Experimental Evaluation

6.1 Dataset and Experiment Settings

For our experiments, we utilized the TAT-QA (Tabular And Textual dataset for Question Answering) dataset [19]. This dataset comprises both tabular and textual data, along with contextual information such as table coordinates and relevant paragraph indices linked to each question. Since the dataset lacks database or API objects, we focused exclusively on static object types in our

	Method	RAG +Reranker	AU-RAG	RAG +Reranker	AU-RAG	RAG +Reranker	AU-RAG
	Num. SOs	30		90		270	
	Num. Qs	90		276		816	
	Avg. ALR	1.617	1.388	1.461	1.452	1.331	1.283
Retrieval	Precision	43.89%	53.41%	31.09%	41.54%	24.24%	30.28%
	Recall	78.89%	65.34%	71.45%	52.39%	59.52%	43.41%
	F1	53.15%	55.27%	42.29%	43.44%	33.72%	33.06%
Answer	Precision	49.05%	42.40%	50.03%	36.68%	45.58%	34.78%
	Recall	59.64%	47.58%	58.14%	41.05%	51.63%	37.27%
	F1	51.44%	42.75%	51.77%	36.74%	46.47%	34.32%
Retrieval Actions	Precision	-	29.16%	-	25.60%	-	18.76%
	Recall	-	42.88%	-	38.91%	-	33.45%
	F1	-	30.98%	-	28.43%	-	21.49%

Table 2: A comparative analysis of different methods across different numbers of Source Objects (SOs) and methods. Q: Question, ALR: Answer Length Ratio (prediction/target)

	Method	RAG +Reranker	AU-RAG	RAG +Reranker	AU-RAG	RAG +Reranker	AU-RAG
	Num. EGs	30		90		270	
	Num. Qs	816		816		816	
	Avg. ALR	1.876	1.813	1.570	1.529	1.331	1.283
Retrieval	Precision	24.29%	28.55%	24.23%	29.03%	24.24%	30.28%
	Recall	59.56%	41.82%	59.47%	42.22%	59.52%	43.41%
	F1	33.77%	31.51%	33.70%	31.83%	33.72%	33.06%
Answer	Precision	36.80%	28.45%	41.26%	30.18%	45.58%	34.78%
	Recall	49.76%	35.95%	50.41%	35.12%	51.63%	37.27%
	F1	39.44%	29.53%	42.90%	30.57%	46.47%	34.32%
Retrieval Actions	Precision	-	16.48%	-	17.52%	-	18.76%
	Recall	-	31.45%	-	33.15%	-	33.45%
	F1	-	19.29%	-	20.63%	-	21.49%

Table 3: A comparative analysis of different methods across different numbers of Extraction Guides (EGs) and methods. Q: Question, ALR: Answer Length Ratio (prediction/target)

experiments. The training set was employed to generate extraction guide objects, while the test set was used for evaluation. From a total of 554 source objects, we randomly selected subsets of 30, 90, and 270 source objects for our experiments. In all experiments, we employed the ChatGPT 4o-mini model with a temperature setting of 0. From the FAISS [20] vector store, we initially retrieved the top 20 results using the standard RAG model, then reranking to select the top 3 most relevant results.

Baseline Method: In our experiment, RAG with the Reranker method serves as the baseline. After retrieving the relevant source objects in Step 3 of Fig. 3, the retrieved contexts are passed to the QA prompt to generate the final answer.

Dataset Augmentation: The original dataset involves a question-answering task based on a given table and paragraph. To adapt this dataset to our scenario, several additional processing steps were required. First, since the dataset lacks natural language descriptions for each source object, we used a ChatGPT 4o-mini model to generate concise summaries of the content. These summaries were crafted to include important keywords and to describe both the nature of the content (whether a table or paragraph) and its key insights. Second, because the original questions were contextually tied to the provided table and paragraph, they often contained ambiguous pronouns, such as "What does the table show?" with the answer, "It summarizes our available lines of credit and committed and uncommitted lines of credit, . . . " These types of questions were unsuitable for our scenario, which focuses on answering questions through universal source objects, as they were too vague. Therefore, we used the ChatGPT 4o-mini model to rewrite the questions, ensuring that the reformulated questions avoided directly copying the keywords from the original answers. Finally, to evaluate source selection in the extraction guides, we employed the chain-of-thoughts method [21]. The chain-of-thought reasoning steps were also augmented using a large language model (LLM).

Evaluation Metrics: In our experiments, we place greater emphasis on information retrieval rather than question answering. The evaluation metrics for source object retrieval and retrieval actions are precision, recall, and F1 score. The former is measured by the set of retrieved source objects, and the latter is measured

by the set of retrieved source objects and its retrieval actions. If the retrieved object is incorrect, the score should be zero. For the generated answers, we use the ROUGE-L score [22] to assess text generation quality. ROUGE-L measures the longest common subsequence (LCS) between the generated and reference texts, capturing both precision and recall based on the overlap of word sequences. A higher ROUGE-L score indicates that the generated answer more closely matches the reference text, reflecting greater relevance and accuracy in the context of text generation. Additionally, we introduced the average Answer Length Ratio (ALR)—the ratio of the predicted answer's length to the target answer's length—to analyze how the answer length affects the ROUGE score. If the ALR is over 1.0, the length of the prediction text is larger than the target text.

6.2 The number of Source Objects

Table. 2 represents a comparative analysis of different methods across three different settings based on the number of Source Objects (SOs) used.

Scalability and SOs Impact: The decline in retrieval performance as the number of Source Objects (SOs) increases suggests that scalability is a significant factor affecting both precision and recall. This observation implies that methods designed to retrieve from a large number of SOs must adapt their approach to mitigate this drop-off, potentially through more efficient indexing or improved filtering techniques. While RAG + Reranker has better recall as the number of SOs grows, AU-RAG maintains relatively higher precision. This suggests that AU-RAG's retrieval mechanism becomes more selective with increasing SOs, potentially sacrificing some recall for better precision.

Retrieval Performance Analysis: For retrieval, the F1 score of AU-RAG is often comparable to or higher than that of RAG + Reranker despite the lower recall. The trade-off between precision and recall is evident between the two methods. It could be beneficial to emphasize the implications of these differences depending on the context of the application. For instance, higher precision in AU-RAG may be advantageous in scenarios where the quality of the retrieved objects is critical, even if not all relevant objects are retrieved. Conversely, the higher recall of RAG + Reranker may be more suitable for exploratory settings, where comprehensive retrieval is prioritized.

The evaluation of retrieval actions, which takes into account the correctness of both the retrieved objects and the actions taken on them, consistently results in lower scores compared to standard retrieval. This suggests that retrieval actions introduce an additional layer of complexity, which could be attributed to challenges in selecting the correct actions once the right objects are retrieved.

Answer Length Ratio (ALR) Analysis: The Average Answer Length Ratio (ALR) values indicate a notable difference between the methods, with RAG + Reranker tending to produce longer answers. Also, RAG + Reranker achieves a higher ROUGE-L score along with a higher ALR. The reason for this difference could be that the RAG + Reranker method serves all the retrieved contexts for the final QA prompt. However, our method only parses the partial information from the retrieved contexts.

Retrieval: As the number of Source Objects (SOs) increases, retrieval performance declines. RAG + Reranker consistently achieves higher recall compared to AU-RAG across different SO sizes. However, AU-RAG outperforms RAG + Reranker in terms of precision. As a result, the F1 score of our method is either comparable to or exceeds that of the baseline.

6.3 The number of Extraction Guide Objects

Table. 3 shows the different methods across three different settings based on the number of Extraction Guides (EGs) used; the number of the source objects is fixed at 270.

Retrieval Performance Analysis: As observed in the retrieval metrics, the combination of RAG and the Reranker consistently achieves higher recall than AU-RAG, while AU-RAG demonstrates superior precision. This suggests that AU-RAG retrieves fewer false positives and provides more accurate results. The trade-off between the two methods is consistent with previously reported findings.

Impact of EG Number on Performance: Increasing the number of EGs positively impacts the performance of all methods, both in terms of retrieval and answer metrics. Our method shows a continuous improvement in retrieval scores: a 1.02% increase is observed when the number of EGs rises from 30 to 90, and a further 3.86% increase is seen when EGs increase from 90 to 270. This indicates that the addition of references enhances retrieval-augmented generation by providing more relevant demonstrations.

7 Conclusion

Retrieval Augmented Generation (RAG) has brought significant advancements in improving the accuracy of question-answering systems by integrating large language models (LLMs) with local or more up-to-date knowledge. However, traditional RAG methods, including those with re-ranking techniques, struggle to efficiently handle large, diverse, and frequently updated data sources, especially when such sources are accessible only through APIs, as these methods rely on pre-encoded embedding vectors. To address these limitations, we introduced Agent-based Universal RAG (AU-RAG), a novel approach designed to enhance retrieval from various data sources by incorporating descriptive metadata. This enables agents to dynamically search, retrieve, and consolidate relevant content from a pool of diverse sources, adapting to new data sources by learning from previous examples. Our experiments, conducted using a multi-source QA dataset, demonstrate that AU-RAG performs robustly in data retrieval tasks. It maintains higher precision compared to traditional RAG models while effectively learning to access new data sources through examples, making it particularly suited for enterprise applications. The distinct contributions of this work, represented by the concepts of *Universal RAG* and *Agent-based RAG*, highlight AU-RAG's potential to overcome the challenges posed by complex and frequently changing data environments, offering a more flexible and adaptive approach to information retrieval.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(No. RS-2023-00222663, RS-2023-00262885), and the BK21 FOUR (Fostering Outstanding Universities for Research) funded by the Ministry of Education(MOE, Korea).

References

[1] OpenAI Blog. Introducing chatgpt, 2022.

[2] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[4] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[5] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey Ullman, Vasilis Vassalos, and Jennifer Widom. The tsimmis approach to mediation: Data models and languages. *Journal of intelligent information systems*, 8:117–132, 1997.

[6] DataBricks. What is llmops?, 2024.

[7] Dennis Heimbigner and Dennis McLeod. A federated architecture for information management. *ACM Transactions on Information Systems (TOIS)*, 3(3):253–278, 1985.

[8] Amit P Sheth and James A Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 22(3):183–236, 1990.

[9] Mary Tork Roth, Manish Arya, Laura M Haas, Michael J Carey, William Cody, Ronald Fagin, Peter M Schwarz, John Thomas, and Edward L Wimmers. The garlic project. *Sigmod Record*, 25(557-557):10–1145, 1996.

[10] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[11] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. Re2g: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*, 2022.

[12] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26, 2024.

[13] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] The pandas development team. pandas-dev/pandas: Pandas, feb 2020.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[17] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*, 2022.

[18] gkamradt. Retrievaltutorials: 5 levels of text splitting, 2024.

[19] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*, 2021.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

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

[22] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.