

# ATOMR: Atomic Operator-Empowered Large Language Models for Heterogeneous Knowledge Reasoning

Amy Xin\*  
xin-x23@mails.tsinghua.edu.cn  
Tsinghua University  
Beijing, China

Jinxin Liu\*  
liujinxi20@mails.tsinghua.edu.cn  
Tsinghua University  
Beijing, China

Zijun Yao  
yaozj20@mails.tsinghua.edu.cn  
Tsinghua University  
Beijing, China

Zhicheng Li  
lizhiche23@mails.tsinghua.edu.cn  
Tsinghua University  
Beijing, China

Shulin Cao  
shulincao97@163.com  
Zhipu AI  
Beijing, China

Lei Hou  
houlei@tsinghua.edu.cn  
Tsinghua University  
Beijing, China

Juanzi Li<sup>†</sup>  
lijuanzi@tsinghua.edu.cn  
Tsinghua University  
Beijing, China

## Abstract

Recent advancements in large language models (LLMs) have led to significant improvements in various natural language processing tasks, but it is still challenging for LLMs to perform knowledge-intensive complex question answering due to LLMs' inefficacy in reasoning planning and the hallucination problem. A typical solution is to employ retrieval-augmented generation (RAG) coupled with chain-of-thought (CoT) reasoning, which decomposes complex questions into chain-like sub-questions and applies iterative RAG at each sub-question. However, prior works exhibit sub-optimal reasoning planning and overlook dynamic knowledge retrieval from heterogeneous sources. In this paper, we propose **ATOMR**, a novel heterogeneous knowledge reasoning framework that conducts multi-source reasoning at the *atomic* level. Drawing inspiration from the graph modeling of knowledge, ATOMR leverages large language models (LLMs) to decompose complex questions into combinations of three *atomic knowledge operators*, significantly enhancing the reasoning process at both the planning and execution stages. We also introduce **BLENDQA**, a novel evaluation benchmark tailored to assess complex heterogeneous knowledge reasoning. Experiments show that ATOMR significantly outperforms state-of-the-art baselines across three single-source and two multi-source reasoning benchmarks, with notable performance gains of

9.4% on 2WikiMultihop and 9.5% on BLENDQA. We release our code and datasets<sup>1</sup>.

## CCS Concepts

• **Information systems** → **Question answering**; • **Computing methodologies** → *Natural language generation*.

## Keywords

Retrieval-Augmented Generation, Large Language Models, Multi-Source Reasoning, Knowledge-Intensive QA

## ACM Reference Format:

Amy Xin, Jinxin Liu, Zijun Yao, Zhicheng Li, Shulin Cao, Lei Hou, and Juanzi Li. 2024. ATOMR: Atomic Operator-Empowered Large Language Models for Heterogeneous Knowledge Reasoning. In *Proceedings of (Preprint)*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 Introduction

Knowledge-intensive complex question answering is a challenging task that requires the ability to reason over vast amounts of knowledge with various reasoning skills, such as multi-hop inference, comparison and calculation [3, 10, 31, 45]. Although the recent advancements of Large Language Models (LLMs) [23] have enabled them to excel in various natural language processing tasks [1, 50], it is arduous for LLMs to perform knowledge-intensive reasoning due to their inefficacy in reasoning planning [12] and the hallucination problem [1, 15], namely the phenomenon that LLMs confidently make up factually incorrect answers.

In order to address the above issues, a typical solution is to leverage retrieval augmentation with chain-of-thought [36] reasoning techniques. Specifically, most recent works [4, 26, 33, 38] propose to perform question decomposition on complex questions into simple questions, then retrieve knowledge facts for sub-question answering, thus alleviating the hallucination. However, there still exists three main challenges: **C1. Sub-optimal reasoning planning** due to the inadequate question decomposition result. Most

\*Equal contribution.

<sup>†</sup>Corresponding Author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*Preprint, Under Review, THU-KEG*

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM  
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

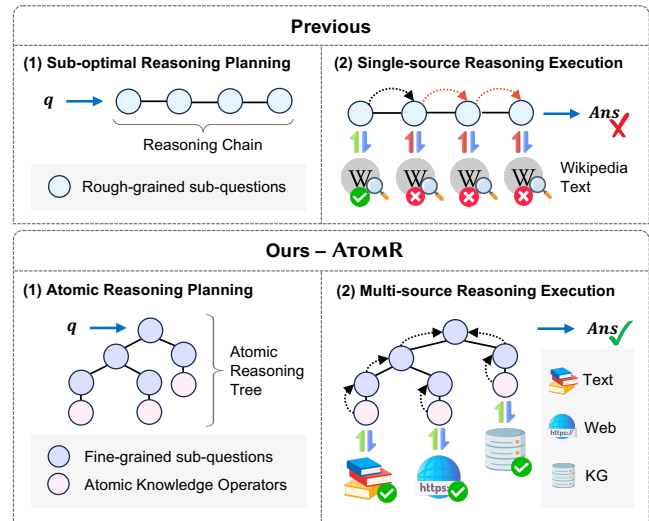
<sup>1</sup><https://github.com/THU-KEG/AtomR.git>

previous works attempt to decompose complex questions into multiple sub-questions with different structures, such as chain-structure [26, 27, 33, 38] or tree structure [4, 42]. However, since they leverage the free-form decomposition, the granularity of the sub-questions is often insufficient, which frequently leads to sub-optimal reasoning paths and mistakes; **C2. The limited support for multiple heterogeneous knowledge sources** in previous retrieval systems. Online web pages, local text corpus, and knowledge bases contain rich knowledge that complement each other. However, most existing work retrieve on a fixed source for all questions [4, 33, 38]. A few [21] support multiple sources but lacks in-depth exploration of knowledge sources such as web pages, only using the top1 snippet from Google; **C3. The absence of high quality test data built on heterogeneous knowledge sources.** Existing benchmarks that are built on multiple knowledge sources either have a narrow knowledge scope, typically only encompassing encyclopedic knowledge from Wikipedia and Wikidata [43, 48, 49], or lack instance-level design, allowing individual questions to be answered using a single knowledge source without requiring cross-knowledge source querying and comparison [6, 40].

Inspired by the modular nature of knowledge [29], which means that knowledge is divisible, diverse, and can be represented by discrete nodes and edges in a graph, in this paper we propose **ATOMR**, an **Atomic** operator-empowered **Reasoning** framework with dynamic knowledge retrieval over heterogeneous knowledge sources. Building on the basic program functions leveraged by previous knowledge base question answering (KBQA) systems [3, 10, 44] to facilitate knowledge manipulation on knowledge graphs (KGs), we refine and summarize a set of atomic operators suitable for various knowledge sources. The atomic knowledge operators possess the fundamental properties of indivisibility and orthogonality, meaning that each one corresponds to a distinct atomic operation without any functional overlap.

The proposed reasoning framework can effectively address the above-mentioned challenges. One one hand, by asking LLMs to decompose the input question to the most fine-grained level, where each leaf node in the sub-question tree corresponds to one atomic knowledge operation, ATOMR derives an adequately decomposed reasoning path, thus minimizing the impact of challenge C1 to the greatest extent. On the other hand, in order to address the challenge C2, ATOMR performs dynamic knowledge retrieval from multiple heterogeneous knowledge sources at necessary nodes, e.g. the leaves. Additionally, thanks to the orthogonality of atomic knowledge operator, ATOMR can avoid redundant retrieval in different branches. As shown in Figure 1, our framework employs a more fine-grained decomposition and a more subtle retrieval strategy.

To be more specific about the pipeline of the proposed reasoning framework, first, ATOMR decomposes the input question into a tree of sub-questions, where each leaf node that requires external knowledge corresponds to an atomic knowledge operator. Then, ATOMR resolves the question by executing the decomposed reasoning tree from the bottom-up in post-order traversal. For the leaf nodes of atomic knowledge operators, ATOMR includes a unique implementation for each operator, enabling knowledge manipulation across three external sources. For other nodes, the output is either produced by (1) child aggregation, or (2) direct retrieval-augmented



**Figure 1: Comparison of ATOMR to previous methods. Previous methods suffer from ineffective decomposition and redundant retrieval. ATOMR employs a more fine-grained decomposition, and only performs dynamic retrieval at necessary nodes, avoiding redundant retrieved information.**

reasoning if child-aggregation fails. The hierarchical execution process helps to locate the optimal reasoning path, and enhances the framework’s robustness, ensuring that if the lower-level nodes fail, the higher-level nodes still could obtain the answer.

Finally, to address the challenge of lacking high quality test data built on heterogeneous knowledge sources (C3), we construct an evaluation benchmark BLENDQA across diverse knowledge sources through a combination of LLM-generated content and manual verification, covering knowledge graphs, online web pages, and local text corpora. We adopt a bottom-up construction approach, first creating sub-questions within different knowledge sources, then merging them into one through a common bridging entity. For example, for web pages, we mainly collect recent news articles and use LLMs to ask questions about relevant entities mentioned in the articles. It includes various types of questions such as multi-hop, comparison, true or false, and long answer question. In this way, the data we constructed covers a wide range of both new and old knowledge, with minimal overlap between different knowledge sources. Experiments also show that our reasoning framework utilizes the three knowledge sources at roughly the same rate.

We conduct extensive experiment to evaluate the performance of ATOMR on three Single-Source datasets and two Multi-Source datasets (including BLENDQA). We employ GPT4o as the base LLM, use full Wikipedia dump as text knowledge source, Google API as web knowledge source, and Wikidata as KG knowledge source. Experiments show that AtomR yields significant improvements over SOTA baselines, achieving 5.4%, 9.4%, and 1.2% F1 score improvements on Single-Source datasets, and 9.5%, 6.6% F1 score improvements on multi-source datasets. Our contribution in this paper is three-fold: (1) Introduce ATOMR, an atomic operator-empowered hybrid reasoning framework; (2) Present evaluation benchmark

BLENDQA built across three heterogeneous knowledge sources; (3) Assess AtomR with extensive experiments and analysis, yielding new SOTA results on 3 and 2 multi-source datasets.

## 2 Related Work

### 2.1 Knowledge-Intensive Reasoning

Although the language models are capable of generating highly coherent text and have strong reasoning abilities, the knowledge required for some tasks exceeds the average knowledge level of humans and cannot be obtained from the input’s local context. External knowledge sources are necessary for such tasks, which we refer to as knowledge-intensive tasks. A wide range of tasks are knowledge-intensive, such as open-domain question answering [5, 22], knowledge base question answering [10, 44], fact checking [30, 35], knowledgeable open dialogue [7, 47] and so on. Knowledge-intensive tasks fully reflect a model’s ability to use external world knowledge to solve real-world problems, and many benchmarks have been created to facilitate the evaluation [25, 46]. Our work falls into the category of question answering, and similar to some recent efforts [8, 19, 21], we focus more on multi-knowledge source reasoning rather than relying on a single knowledge source. Furthermore, we extend previous work by supporting not only commonly used encyclopedic knowledge like Wikipedia and Wikidata [34], but also private offline corpora and online web pages.

### 2.2 Retrieval-Augmented Language Models

It has been validated that retrieval-augmented methods can improve the performance of language models in various natural language tasks [11, 14, 16, 37]. For example, early works [20] leverage dense vector of text to retrieve relevant passage of the input, and recent advances incorporate trainable retrievers [28] or search engines [26] to augment the input context. Previous works usually adopt the one-time retrieval strategy [2, 17], where the retriever is only called once to solve one question, which may not able to perform refined retrieval for complex questions. Therefore, A few recent approaches instead adopt the multi-time retrieval strategy. IRCOT [33] performs a retrieval action in each step of the CoT reasoning process. ITER-RETGEN [27] iteratively call the retriever in every turn of the CoT steps based on the previous turn’s generation result and the original question. Self-Ask [26] utilizes LLMs to decompose questions into sub-questions and performs the retrieval for each sub-question. ProbTree [4] proposes to decompose a question into a tree structure, and accordingly perform retrieval at each node of the tree. Compared to previous methods, ATOMR decomposes a complex question into fine-grained atomic knowledge operators, effectively improving the success rate of relevant knowledge retrieval at each node of the reasoning tree.

## 3 Methodology

In this section, we introduce the design of ATOMR. The overall architecture of ATOMR is illustrated in Figure 2. At the core of our framework are three fundamental **Atomic Knowledge Operators** (Section 3.1), which we design to retrieve and manipulate knowledge at the atomic level. These operators then steer the process of ATOMR through two main stages: **(1) Atomic Reasoning Planning** (Section 3.2), where the system effectively generates a fine-grained

**Table 1: The correspondence of each ATOMR atomic knowledge operator to SPARQL and Cypher clauses and KoPL functions. “[REL]” represents inter-nodal relationship in Cypher, while “WHERE(n)” and “WHERE(e,r)” represents using SPARQL’s WHERE clause with the entity name constraint and the entity-relation constraint, respectively. We omit the SPARQL SELECT clause, as it is a fundamental main clause that is inherently included in all queries.**

Function	SPARQL	Cypher	KoPL	ATOMR
Entity Disambiguation	WHERE (n)	MATCH	Find	<b>Search</b>
One-hop Inference	WHERE (e,r)	[REL]	Relate, QueryAttr, QueryRelation	<b>Relate</b>
Entity Filtering	FILTER	WHERE	FilterConcept, FilterStr, FilterNum, FilterYear, FilterDate	<b>Filter</b>

Atomic Reasoning Tree (ART), and **(2) Atomic Reasoning Execution** (Section 3.3), where the system performs bottom-up reasoning across multiple knowledge sources at each atomic node. We first introduce the design of our three atomic knowledge operators in Section 3.1, then detail the two main stages of our framework in Sections 3.2 and 3.3. The LLM prompts used in each step are included in Appendix B.

### 3.1 Atomic Knowledge Operators

Knowledge-intensive QA demands accurate knowledge retrieval and sophisticated knowledge manipulation. Hence, the *granularity* at which we operate knowledge is crucial. Previous knowledge base question answering systems [3] exemplify fine-grained knowledge manipulation by breaking down knowledge into atomic elements—entities, relations, and attributes. In this light, we aim to refine the reasoning process of LLMs to an atomic level.

We design three general *atomic knowledge operators* for LLMs: Search, Relate, and Filter, which can be applied across various knowledge sources. These operators distill essential operations from existing graph query languages. Table 1 illustrates how operations of three widely-used graph query languages, SPARQL, Cypher, and KoPL [3, 9, 24], can be induced into ATOMR’s three atomic knowledge operators.

**Search.** The Search operator is designed for *entity disambiguation*: to accurately retrieve the desired entities from a massive entity pool, especially when multiple entities share similar names. In graph knowledge models, Search is analogous to locating the initial entity node(s), which sets the foundation for subsequent inter-nodal reasoning. ATOMR’s Search operator is defined as follows:

$$list[entity] = Search(entity\_name, \{optional\}descriptor)$$

It contains two inputs: the entity name and an optional entity descriptor to facilitate disambiguation. For example, Search (“Michael

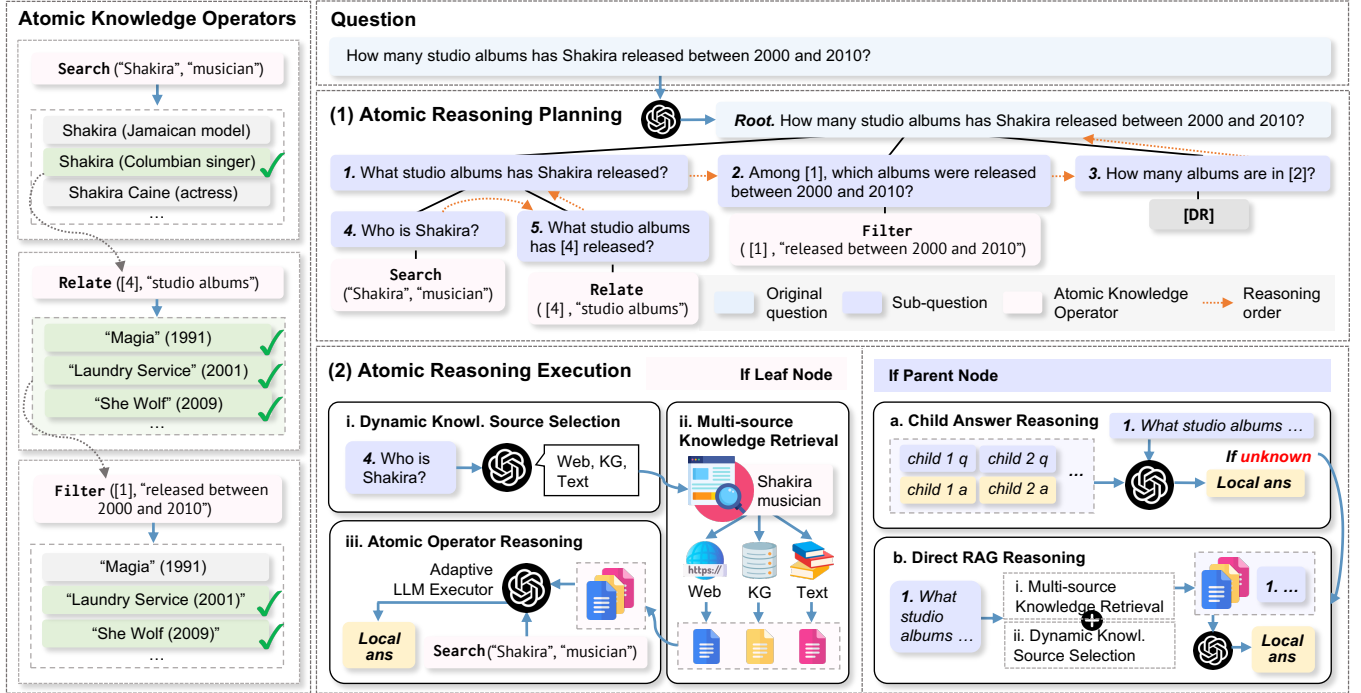


Figure 2: The Overall Framework of AtomR.

*Jordan*, “scientist”) returns the entity [“Michael I. Jordan”], successfully distinguishing the machine learning scientist “Michael Jordan” from other “Michael Jordan” entities.

During execution, the Search function first initiates multi-source knowledge retrieval using the entity name concatenated with the optional descriptor  $\{entity\_name\} \{descriptor\}$  as the query. Then, the retrieved knowledge is inputted to an *adaptive LLM executor* to conduct entity disambiguation through in-context learning. Finally, the LLM executor outputs the disambiguated entity list.

**Relate.** The Relate operator is designed for *one-hop inference*. In graph knowledge models, Relate is analogous to handling knowledge from a head entity node to a tail entity node across a relation edge. There are three possibilities for one-hop inference: (1) retrieve tail entities given a head entity and relation, (2) retrieve an attribute value given an entity and attribute, and (3) retrieve a relation between a head and tail entity. The three possibilities are depicted in Equations 1, 2, 3, where  $entity_h$  and  $entity_t$  denotes head and tail entity, respectively:

$$list[entity_t] = Relate(entity_h, relation) \quad (1)$$

$$value = Relate(entity_h, attribute) \quad (2)$$

$$relation = Relate(entity_h, entity_t) \quad (3)$$

An example of case (1) would be  $Relate(\text{“Barack Obama”, “child”})$ , which returns the tail entity list [“Malia Obama”, “Sasha Obama”]. Similarly, for case (2),  $Relate(\text{“Barack Obama”, “date of birth”})$  returns the attribute [“August 4th, 1961”]. An example for case (3) would be

to reversely retrieve the relation of Malia Obama to Barack Obama, where  $Relate(\text{“Malia Obama”, “Barack Obama”})$  returns [“child”].

During execution, similar as Search, Relate first initiates multi-source knowledge retrieval using the concatenated function parameters as the query, then inputs the retrieved knowledge into the adaptive LLM executor with an in-context learning prompt. Finally, the LLM executor outputs the answer: either an entity list for case (1), an attribute value for case (2), or a relation for case (3).

**Filter.** The Filter operator is designed for *entity filtering*. In graph knowledge models, Filter is analogous to filtering an entity subset that satisfies an attributal condition from an initial entity set. It is defined as follows:

$$list[entity] = Filter(list[entity], condition)$$

For example,  $Filter(\text{“Lionel Messi”, “Steven Jobs”, “Bill Gates”}, \text{“born in 1955”})$ , returns [“Bill Gates”, “Steve Jobs”].

The execution of Filter is slightly more complicated. First, Filter initiates multi-source knowledge retrieval for every entity in its input  $list[entity]$ , where each retrieval query is formulated as  $\{entity\_name\} \{condition\}$ . Subsequently, for each entity  $e_i$ , Filter concatenates all retrieved passages and calculates the overlapping coefficient  $O_i$  between the query  $q_i$   $\{entity\_name\} \{condition\}$  and the concatenated passage  $p_i$ :

$$O_i(q_i, p_i) = \frac{|q_i \cap p_i|}{\min(|q_i|, |p_i|)}$$

The overlapping coefficient of each entity is then compared to a hyperparameter threshold  $t$ . Entities with  $O_i < t$  will be directly discarded to remove excessive noise from the retrieved knowledge.

Then, the remaining entities and passages are inputted to the adaptive LLM executor to perform entity filtering via in-context learning. Finally, the LLM executor outputs the filtered entity list.

### 3.2 Atomic Reasoning Planning

To effectively model the complex reasoning structures of knowledge-intensive QA, ATOMR employs reasoning planning in a tree structure [4, 42]. Given a complex question, ATOMR leverages an LLM planner to decompose the question into an *Atomic Reasoning Tree* (ART), where the root node is the original complex question, while each non-root node is a decomposed sub-question of its parent. The decomposition continues until each leaf question is a fine-grained *atomic question*, which could be directly answered by either (a) calling one of the three predefined atomic knowledge operators—Search, Relate, or Filter, or (b) analyzing answers of previous sibling sub-questions. Then, for each atomic question of case (a), the LLM planner appends the question’s corresponding operator call with parameters as the final set of leaf nodes for the ART. We generate ARTs using in-context learning, and the prompt is included in Figure 5 of the Appendix.

Following Cao et al. [4], we index ART nodes in breadth-first search (BFS) order. To reference intermediate answers, we use reference placeholders denoted as  $[i]$ , which will be substituted with actual answers of sub-question  $[i]$  during the reasoning execution stage. Figure 2(1) illustrates an example ART, where the original question “How many studio albums has Shakira released between 2000 and 2010?” is decomposed into a reasoning tree of three leaf atomic knowledge operators, linked to sub-questions 4, 5, and 2, respectively. The grey *[DR]* (Direct Reasoning) mark attached to sub-question 3 “How many albums are in  $[2]$ ?” indicates that it does not trigger a leaf operator call but instead formulates an answer by direct LLM reasoning based on the output of its sibling sub-questions, in this case sub-question 2.

We argue that an ART is *atomic* because a question is decomposed until each sub-question reaches an atomic granularity that could be handled by a predefined atomic knowledge operator. Although previous efforts have sought to achieve fine-grained question decomposition, they often fall short due to the lack of effective atomic constraints. By explicitly inducing and defining a set of atomic knowledge operators, ATOMR successfully manoeuvres the LLM’s reasoning planning process at a highly fine-grained level.

### 3.3 Atomic Reasoning Execution

In this stage, given the decomposed ART, we conduct bottom-up reasoning over each node via post-order-traversal. Algorithm 1 depicts the detailed reasoning procedure, while Figure 2(2) visualizes the main modules involved.

**3.3.1 Leaf Node Reasoning.** In an ART, each leaf node is an atomic knowledge operator. The execution of an atomic knowledge operator consists of three steps: (1) Dynamic knowledge Source Selection, (2) Multi-source knowledge retrieval, and (3) Atomic Operator Reasoning.

**Dynamic Knowledge Source Selection.** While previous works retrieve knowledge from a static knowledge source, ATOMR employs dynamic knowledge source selection at each sub-question. This enables ATOMR to flexibly identify the most suitable knowledge

---

#### Algorithm 1 Atomic Reasoning Execution

---

```

1: Require: Atomic Reasoning Tree  $ART$ 
2: for  $n_i$  in  $PostOrder(ART)$  do
3:   if  $n_i$  is leaf then
4:     prepare  $n_i$ ’s sub-question  $q_i$ , child  $c_i$ 
5:      $s_i \leftarrow KnowledgeSourceSelection(q_i)$ 
6:      $k_i \leftarrow KnowledgeRetrieval(c_i, s_i)$ 
7:     try
8:        $a_i \leftarrow AtomicOperatorReasoning(q_i, c_i, k_i)$ 
9:     catch Operator Execution Failure
10:     $a_i \leftarrow DirectRAGReasoning(q_i, k_i)$ 
11:   end try
12:   else
13:     prepare  $n_i$ ’s sub-question  $q_i$ , children nodes  $c_i$ 
14:     if  $c_i$  is  $[DR]$  then
15:        $a_i \leftarrow SiblingAnswerReasoning(q_i, s_i)$ 
16:     else
17:        $a_i \leftarrow ChildAnswerReasoning(q_i, c_i)$ 
18:       if  $a_i$  is Unknown then
19:          $s_i \leftarrow KnowledgeSourceSelection(q_i)$ 
20:          $k_i \leftarrow KnowledgeRetrieval(q_i, s_i)$ 
21:          $a_i \leftarrow DirectRAGReasoning(q_i, k_i)$ 
22:       end if
23:     end if
24:   end if
25: end for
26: return  $a_{Root}$ 

```

---

sources to answer each sub-question. Specifically, we leverage an LLM as a dynamic knowledge source selector through in-context learning. The LLM outputs the selected knowledge sources, which are used in the subsequent multi-source knowledge retrieval step. **Multi-source Knowledge Retrieval.** After selecting the appropriate knowledge sources for each sub-question, ATOMR initiates knowledge retrieval from each source. The retrieval method varies by source: Web sources are queried via a search engine API, text sources through a dense retriever, and knowledge graphs (KG) via a structured graph query language. For the Web and Text knowledge sources, ATOMR retrieves the top  $k$  passages and articles, respectively, where  $k$  is a pre-defined hyperparameter. For the KG knowledge source, the full structured answer list is returned. We formulate a customized retrieval query for each atomic operator as detailed in Section 3.1.

**Atomic Operator Reasoning.** Equipped with the sub-question, the atomic knowledge operator is now ready for execution. To ensure flexibility and robustness, we utilize an *Adaptive LLM Executor*, instead of static symbolic code, for operator execution. We refer to this process as *Atomic Operator Reasoning*. The detailed implementation of each atomic knowledge operator is explained in Section 3.1. Finally, the output of the adaptive LLM executor serves as the local answer for the associated sub-question.

**3.3.2 Parent Node Reasoning.** In an ART, all nodes that don’t initiate an atomic knowledge operator call are considered parent nodes. A parent node may go through (1) Child Answer Reasoning, (2) Sibling Answer Reasoning, and (3) Direct RAG Reasoning.

**Child Answer Reasoning.** Child answer reasoning is the process of deducing an answer for a parent node by synthesizing answers of its child nodes, formally:

$$a_i = \text{ChildAnswerReasoning}(q_i, [q_{c_1}, a_{c_1}, q_{c_2}, a_{c_2}, \dots])$$

where  $q_i$  is the sub-question for the current node and  $[q_{c_1}, a_{c_1}, \dots]$  is the list of question-answer pairs for  $q_i$ 's child nodes. For example, in Figure 2(1), sub-question 1 can be answered by synthesizing the answers of its child nodes 4 and 5. Child answer reasoning is achieved through in-context learning, where the LLM is provided with the child question-answer pairs along with an instruction prompt.

**Sibling Answer Reasoning.** Similar to child answer reasoning, sibling answer reasoning targets  $[DR]$  nodes that are answered by analyzing answers of its previous sibling nodes, formally:

$$a_i = \text{SiblingAnswerReasoning}(q_i, [q_{s_1}, a_{s_1}, q_{s_2}, a_{s_2}, \dots])$$

where  $[q_{s_1}, a_{s_1}, \dots]$  is the list of question-answer pairs for  $q_i$ 's sibling nodes that are referenced in the current question  $q_i$ . For example, in Figure 2(1), sub-question 3 can be answered by analyzing answers of its referenced sibling node 2. Sibling answer reasoning is also achieved through LLM in-context learning.

**Direct RAG Reasoning.** Each parent node is primarily resolved through either child answer reasoning or sibling answer reasoning. However, if a parent node fails to obtain an answer, *direct RAG reasoning* is employed. Parallel to leaf node reasoning, direct rag reasoning first performs dynamic knowledge source selection for the current sub-question  $q_i$ , then initiates multi-source knowledge retrieval with  $q_i$  as the query. Finally, given  $q_i$  and the retrieved knowledge, an LLM is employed to conduct standard RAG through in context learning. This design ensures that while ATOMR primarily focuses on knowledge retrieval at the leaf nodes, it can also flexibly access external knowledge at inner nodes when necessary.

## 4 Experiments

### 4.1 Experimental Setup

**4.1.1 Datasets and Evaluation Metrics.** To conduct a comprehensive assessment of ATOMR, we conduct experiments in both single-source and multi-source settings. In the single-source setting, we evaluate ATOMR on three Wikipedia-based multi-hop QA benchmarks: HotpotQA [41], 2WikiMultiHop [13], and Musique [32]. We adopt the test and development sets released by IRCOT [33], which include a 500-entry test set and a 100-entry development set sampled from the original development set. In the multi-source setting, we evaluate ATOMR on two recent datasets: CRAG [39] by Meta and BLENDQA, our original dataset. Both CRAG and BLENDQA are constructed based on three heterogeneous knowledge sources—the Wikidata knowledge base, the Wikipedia text corpus, and the Google Web search engine. We will provide a more thorough introduction of BLENDQA in the next section. For BLENDQA, we evaluate ATOMR on the full dataset; for CRAG, we only consider static questions and sample a 500-entry test set with a 100-entry development set. Following IRCOT and ProbTree, we adopt token-level F1 as the evaluation metric.

**4.1.2 BLENDQA.** The evaluation benchmark BLENDQA is designed to evaluate models' reasoning capabilities across three heterogeneous knowledge sources. In practice, we use the Wikidata and Wikipedia as the knowledge graph and text corpus, respectively. The LLM used for construction is *gpt-4o-2024-08-06*. The general process is to construct two sub-questions  $sub-q_1$  and  $sub-q_2$  in two different sources that shares a common bridging entity, and merge them to form a cohesive query. Based on the knowledge source used, there are three types of questions: the KG-Text, KG-Web, Text-Web. **KG-Text:** In the construction of KG-Text category, the  $sub-q_1$  is sampled from Natural Questions (NQ) [18], which is built on Wikipedia. We identify the topic entity of it as the bridging entity. Then we sample several triples of the entity in the KG, using them generate the second  $sub-q_2$ .

**KG-Web:** In KG-Web, there are two sub-types depending on the anchor of the construction. (1) The first *kg2web* is anchored in KG, where we randomly sample entities from KG as bridging entities, sample their triples in KG to generate  $sub-q_1$ , and search the bridging entity for relevant news to generate  $sub-q_2$ . (2) The second *web2kg* is anchored in web, where we collect a large number of news articles in various domains (politics, business, etc.), asking the LLM to identify a bridging entity in an article and ask a question about it as  $sub-q_1$ . The  $sub-q_2$  is generated in KG with the same procedure described in (1).

**Text-Web:** There are also two sub-types in Text-Web. (1) The *web2text* is anchored in web, where we identify the topic entity as bridging entity in the sampled  $sub-q_1$  from NQ, then search the entity on the web (exclude wikipedia) to collect its descriptions. Using these descriptions, we leverage the LLM to generate a unique tag of the entity as  $sub-q_2$ , e.g. "Neil Armstrong" - "first man to walk on the moon". (2) The second *text2web* is anchored in web, where we sample  $sub-q_1$  from NQ, and use the answer of it as the bridging entity (if valid). Finally we search entity for news to generate  $sub-q_2$ .

### 4.1.3 Baselines.

**Single-source Baselines.** We evaluate both Closebook and Openbook baselines. For the Closebook setting, we evaluate Standard Prompting and CoT using two-shot prompts. For the Openbook setting, we first evaluate Standard RAG with one-time retrieval, then evaluate three state-of-the-art reasoning frameworks: two chain-structured frameworks IRCOT [33] and SearChain [38], and one tree-structured framework ProbTree [4].

**Multi-source Baselines.** We evaluate both Closebook and Openbook approaches using Standard Prompting, CoT, and Standard RAG with two-shot prompts. We also assess three established frameworks: Self-Ask [26], which integrates only a Google search engine; ProbTree [4], which combines a Google engine with a Wikipedia corpus; and Chain-of-Knowledge (CoK) [21], which supports all three knowledge sources—Google, Wikipedia, and Wikidata.

**4.1.4 Implementation Details.** For all experiments, we use *gpt-4o-2024-08-06* as our base LLM. For web knowledge retrieval, we use SERPAPI to obtain the most up-to-date results from Google. For text knowledge retrieval, we use ColBERTv2 as our dense retriever to retrieve passages from a local Wikipedia corpus. We use HotpotQA's official Wikipedia abstract dump from October 2017 to evaluate HotpotQA, and the Dec 2021 full Wikipedia dump (following Atlas) to evaluate the other four datasets. For KB knowledge retrieval, we

**Table 2: Results for Single-Source Reasoning. Token-level F1 is reported, with the best results in bold and the second best results underlined. (†) indicates the overall gain of ATOMR compared to the second-best baseline for each dataset.**

	HotpotQA			2Wikimultihop					Musique			
	Overall	Bridge	Comp.	Overall	Bridge	Infer.	Comp.	B.C.	Overall	2hop	3hop	4hop
<i>Without Retrieval (Closebook)</i>												
Standard Prompting	50.02	45.22	72.50	41.09	14.44	32.24	71.63	63.13	19.31	21.84	16.03	17.81
CoT	58.38	55.48	71.91	58.32	38.24	54.83	77.27	77.17	29.03	36.26	24.07	17.38
<i>With Retrieval (Wikipedia)</i>												
Standard RAG	60.31	61.88	52.97	47.94	34.96	54.32	56.96	57.27	22.07	28.51	15.81	14.77
IRCoT	60.20	58.00	<u>69.40</u>	63.80	46.20	45.50	<u>91.60</u>	79.00	34.20	<u>44.20</u>	26.30	20.10
SearChain	59.04	<b>69.73</b>	51.12	63.10	48.85	50.38	81.41	84.21	31.68	38.89	28.38	17.28
ProbTree	<u>65.91</u>	65.50	67.81	<u>69.32</u>	<u>52.45</u>	<u>64.08</u>	88.17	<u>90.00</u>	<u>34.92</u>	42.52	<b>30.38</b>	<b>21.53</b>
<b>ATOMR (ours)</b>	<b>71.27(† 5.4)</b>	<u>68.96</u>	<b>82.07</b>	<b>78.72(† 9.4)</b>	<b>59.07</b>	<b>80.04</b>	<b>97.48</b>	<b>93.33</b>	<b>36.11(† 1.2)</b>	<b>45.63</b>	<u>29.94</u>	<u>20.15</u>

**Table 3: Results for Multi-Source Reasoning. Token-level F1 is reported. Web, Text, and KG denote the integration of knowledge from Google, the Wikipedia corpus, and the Wikidata knowledge base, respectively.**

	Web	Text	KG	BlendQA			CRAG					
				Overall	KG-Web	KG-Text	Text-Web	Overall	Simple	S.C.	Comp.	M.H.
<i>Without Retrieval (Closebook)</i>												
Standard Prompting	-	-	-	23.26	16.49	24.71	27.64	59.18	56.46	53.87	70.52	55.86
CoT	-	-	-	30.61	18.30	34.37	37.24	63.59	63.22	60.09	69.53	61.40
<i>With Retrieval (Multi-Source)</i>												
Standard RAG	✓	✓	✓	33.78	<u>26.62</u>	32.76	<u>41.21</u>	59.27	<u>66.23</u>	56.70	56.67	56.98
Self-Ask	✓	-	-	26.25	20.69	27.59	29.68	48.40	47.52	43.22	60.27	42.30
ProbTree	✓	✓	-	<u>33.85</u>	24.48	<u>38.81</u>	36.70	58.78	58.95	49.00	67.90	59.28
Chain-of-Knowledge	✓	✓	✓	33.08	22.66	37.55	37.38	<u>64.75</u>	57.82	<u>62.15</u>	<u>74.04</u>	<b>65.40</b>
<b>ATOMR (ours)</b>	✓	✓	✓	<b>43.32(† 9.5)</b>	<b>45.63</b>	<b>39.69</b>	<b>45.22</b>	<b>71.39(† 6.6)</b>	<b>70.10</b>	<b>68.51</b>	<b>81.71</b>	<u>64.95</u>

use KoPL as the query language and KQA Pro’s Wikidata dump as the knowledge base. We only retrieve text knowledge for single-source datasets, and retrieve knowledge from as many sources as supported for each baseline on multi-source datasets. We retrieve knowledge from all three sources for multi-source Standard RAG. For ATOMR, we set  $k = 3$  to retrieve the top 3 web and text results, and set  $t = 0.5$  for the Filter function. The results for IRCoT are from Cao et al. [4], while all other baselines are reproduced by ourselves using the above settings.

## 4.2 Main Results

**4.2.1 Single-source Results.** As shown in Table 2, ATOMR surpasses all baselines across all three datasets. Compared to the previous SOTA ProbTree, ATOMR achieves F1 improvements by 5.4%, 9.4%, and 1.2% on HotpotQA, 2WikiMultiHop, and Musique, respectively. ATOMR also demonstrates outstanding performance on all types of questions, notably achieving F1 scores as high as 97.48% and 93.33% on the Comparison and Bridge Comparison question types. Such improvements demonstrate the effectiveness of ATOMR’s fine-grained reasoning planning and atomic knowledge operator design.

While ATOMR achieves the best overall performance on Musique, it yields slightly lower results on Musique’s 3-hop and 4-hop questions. Upon closer investigation of the dataset and our test outputs, we find that the sub-questions defined in the Musique dataset are relatively coarse-grained and can usually be broken down into two to three operators by ATOMR. As a result, 3-hop and 4-hop reasoning often involve a large number of sub-questions, which easily leads to error propagation.

**4.2.2 Multi-source Results.** ATOMR achieves the best results on both BLENDQA and CRAG, with overall F1 improvements of 9.5% and 6.6%, respectively. On BLENDQA, ATOMR outperforms all baselines consistently over each knowledge-source setting, demonstrating the effectiveness of our dynamic knowledge selection process. We also observe that compared to ATOMR, the two previous SOTA frameworks ProbTree and Chain-of-Knowledge are more prone to LLM hallucination because they rely heavily on the LLM’s memorized factual knowledge during the reasoning process. In contrast, ATOMR prioritizes reasoning with accurately retrieved fine-grained knowledge, leading to more trustworthy results. Furthermore, comparing to ProbTree, ATOMR achieves more superior results with fewer LLM and retriever calls, which is detailed in Section 4.3.3.

**Table 4: Ablating the integration of knowledge sources.**

Knowledge Sources	BLENDQA	CRAG
Text, Web, KG	<b>46.97</b>	<b>70.23</b>
w/o KG	46.52(↓ 0.5)	68.88(↓ 1.4)
w/o Text	43.34(↓ 3.6)	67.03(↓ 3.2)
w/o Web	30.30(↓ 16.7)	58.27(↓ 12.0)

### 4.3 Analysis

**4.3.1 Case study: How does ATOMR outperform previous methods?** We present a real example from CRAG to showcase three advantages of ATOMR comparing to two SOTA frameworks - ProbTree and CoK.

**(1) Effective Reasoning Planning.** In this test case, CoK decomposes the question into two sequential rationales, each recalling only a few of Shakira’s studio albums, which leads to an incomplete answer. ProbTree decomposes the input into two overlapping complex question, resulting in serious redundancy and low accuracy results. In contrast, ATOMR, achieves fine-grained reasoning planning with three atomic knowledge operators, each designed to fulfill a unique, orthogonal function.

**(2) Accurate Knowledge Retrieval.** The sub-optimal reasoning planning of CoK and ProbTree directly leads to inefficiencies in knowledge retrieval. CoK only retrieves information of the albums mentioned in each rationale, capturing just a subset of the required answer set. ProbTree’s compositional sub-questions result in noisy and inaccurate retrievals. In contrast, ATOMR’s atomic reasoning planning ensures fine-grained and precise knowledge retrieval.

**(3) Preventing LLM hallucination.** Both CoK and ProbTree rely heavily on LLM parametric knowledge. CoK employs LLM-generated factual rationales, while ProbTree uses close-book answers of LLMs. Such characteristics make both frameworks susceptible to LLM hallucination. For instance, ProbTree’s close-book answer for q-1 contains four factual errors. In contrast, ATOMR primarily uses LLMs as reasoning agents over retrieved knowledge, only tapping into LLM’s parametric knowledge when external sources are lacking, thereby enhancing answer accuracy and reliability.

**4.3.2 Effect of Incorporating Multi-Source Knowledge.** We conduct an ablation study to assess the impact of multi-source knowledge integration by sampling 80 entries each from BLENDQA and CRAG. The results, detailed in Table 4, indicate that removing any knowledge source reduces ATOMR’s performance, with Web knowledge having the most significant impact and KG the least. Removing the Text source results in a noticeable decline of over 3%, highlighting that Wikipedia corpus retrieval is necessary and cannot be fully substituted by Web searches. We also report the distribution of selected knowledge sources in Figure 4 of the Appendix, which demonstrates that the Web knowledge source is the most generally selected knowledge source, while the other two sources also contribute adequately to ATOMR’s performance.

**4.3.3 Comparative Cost Analysis.** While both ATOMR and ProbTree employ tree reasoning structures, we design ATOMR to be more accurate while also more *cost-efficient*. Table 5 illustrates that ATOMR requires fewer LLM and knowledge retriever calls compared

**Table 5: Comparing ATOMR’s API consumption with the previous SOTA ProbTree.**

Single-source	HotpotQA		2Wiki		Musique	
	LLM	Text	LLM	Text	LLM	Text
ProbTree	4049	1875	5127	3863	4620	3683
<b>ATOMR</b>	3629	1978	4996	2573	4120	2734

Multi-source	BLENDQA			CRAG		
	LLM	Text	Web	LLM	Text	Web
ProbTree	3683	1854	1854	3917	1694	1694
<b>ATOMR</b>	3465	1158	1504	2880	946	1236

to ProbTree. ProbTree necessitates two basic LLM calls per tree node—one for close-book answering and another for open-book answering—whereas ATOMR demands just one LLM call, whether for atomic operator reasoning at a leaf node or for child-or-sibling Answer Reasoning at a parent node. Similarly, while ProbTree initiates a knowledge retriever call at every node, ATOMR only initiates retriever calls when external knowledge is needed—either at an atomic leaf node, or at a parent node that fails to formulate an answer with child answer reasoning. This is made possible by the *atomic* nature of our method: ATOMR decomposes complex questions into a highly fine-grained reasoning tree, which generally requires retrieval only at leaf nodes.

## 5 Conclusion

In this paper, we propose ATOMR, an atomic operator-empowered large language models reasoning framework for heterogeneous knowledge sources. ATOMR breaks down the reasoning process to the atomic level of knowledge, outperforming previous SoTA reasoning systems across three single-source and two multi-source datasets by large margins. Ablation studies further validate the effectiveness of the proposed atomic knowledge operators, which guide the model in effective reasoning planning and precise knowledge retrieval, enhancing accuracy while reducing the overhead.

## References

- [1] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. *CoRR* abs/2302.04023 (2023). <https://doi.org/10.48550/arXiv.2302.04023>
- [2] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 2206–2240. <https://proceedings.mlr.press/v162/borgeaud22a.html>
- [3] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA Pro: A Dataset with Explicit Compositional Programs for Complex Question Answering over Knowledge Base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 6101–6119. <https://doi.org/10.18653/v1/2022.acl-long.422>



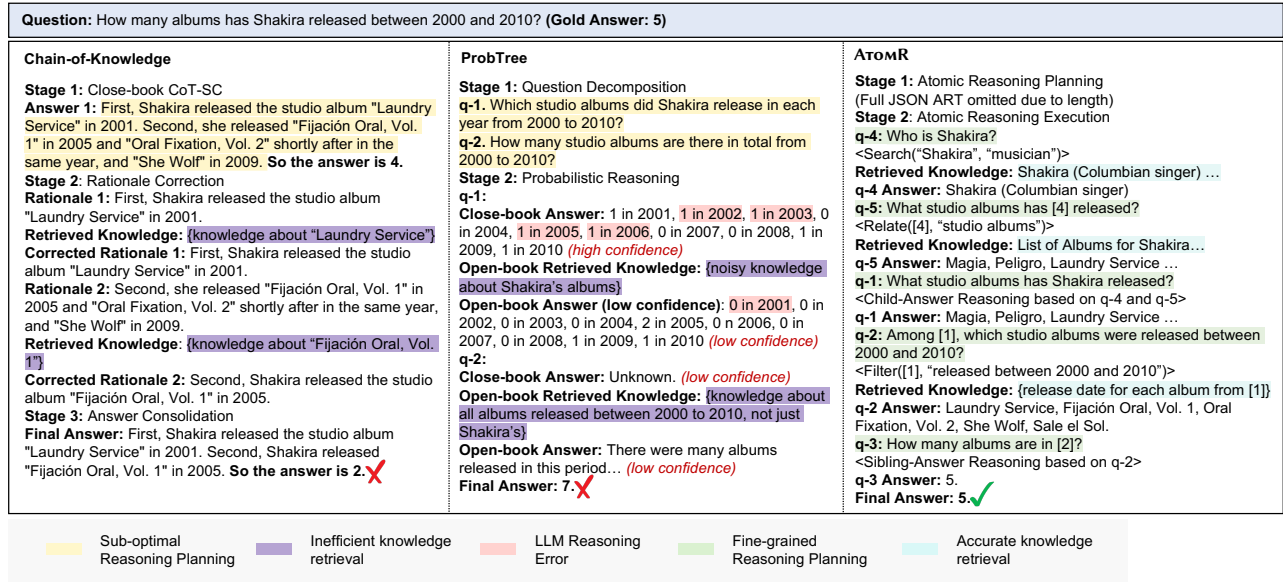


Figure 3: Comparative case study of AtOMR with previous SOTA Chain-of-Knowledge and ProbTree.

[4] Shulin Cao, Jiajie Zhang, Jiaxin Shi, Xin Lv, Zijun Yao, Qi Tian, Lei Hou, and Juanzi Li. 2023. Probabilistic Tree-of-thought Reasoning for Answering Knowledge-intensive Complex Questions. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 12541-12560. <https://doi.org/10.18653/v1/2023.FINDINGS-EMNLP.835>

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 1870-1879. <https://doi.org/10.18653/v1/P17-1171>

[6] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. 2024. CompMix: A Benchmark for Heterogeneous Question Answering. In *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Roy Ka-Wei Lee, Ravi Kumar, and Hady W. Lauw (Eds.). ACM, 1091-1094. <https://doi.org/10.1145/3589335.3651444>

[7] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. <https://openreview.net/forum?id=r1l73iRqKm>

[8] Shangbin Feng, Weijia Shi, Yuyang Bai, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024. Knowledge Card: Filling LLMs' Knowledge Gaps with Plug-in Specialized Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. <https://openreview.net/forum?id=WbWtOYIzIK>

[9] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Linddaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 1433-1445. <https://doi.org/10.1145/3183713.3190657>

[10] Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 3477-3488. <https://doi.org/10.1145/3442381.3449992>

[11] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. *CoRR* abs/2002.08909 (2020). [arXiv:2002.08909](https://arxiv.org/abs/2002.08909)

[12] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 8154-8173. <https://doi.org/10.18653/v1/2023.EMNLP-MAIN.507>

[13] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060* (2020).

[14] Yushi Hu, Hang Hua, Zhengyuan Yang, Weijia Shi, Noah A. Smith, and Jiebo Luo. 2022. PromptCap: Prompt-Guided Task-Aware Image Captioning. *CoRR* abs/2211.09699 (2022). <https://doi.org/10.48550/ARXIV.2211.09699>

[15] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *CoRR* abs/2311.05232 (2023). <https://doi.org/10.48550/ARXIV.2311.05232>

[16] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, 874-880. <https://doi.org/10.18653/v1/2021.EACL-MAIN.74>

[17] Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot Learning with Retrieval Augmented Language Models. *CoRR* abs/2208.03299 (2022). <https://doi.org/10.48550/ARXIV.2208.03299>

[18] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a Benchmark for Question Answering Research. *Trans. Assoc. Comput. Linguistics* 7 (2019), 452-466. [https://doi.org/10.1162/TACL\\_A\\_00276](https://doi.org/10.1162/TACL_A_00276)

[19] Jens Lehmann, Dhananjay Bhandiwad, Preetam Gattogi, and Sahar Vahdati. 2024. Beyond Boundaries: A Human-like Approach for Question Answering over Structured and Unstructured Information Sources. *Trans. Assoc. Comput. Linguistics* 12 (2024), 786-802. [https://doi.org/10.1162/TACL\\_A\\_00671](https://doi.org/10.1162/TACL_A_00671)

[20] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). <https://proceedings.neurips.cc/paper/2020/hash/>

- 6b493230205f780e1bc26945df7481e5-Abstract.html
- [21] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. <https://openreview.net/forum?id=cPgh4gWZlz>
- [22] Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniKQA: Unified Representations of Structured and Unstructured Knowledge for Open-Domain Question Answering. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (Eds.). Association for Computational Linguistics, 1535–1546. <https://doi.org/10.18653/V1/2022.FINDINGS-NAACL.115>
- [23] OpenAI. 2023. GPT-4 Technical Report. *CoRR abs/2303.08774* (2023). <https://doi.org/10.48550/arXiv.2303.08774>
- [24] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. 2006. Semantics and Complexity of SPARQL. In *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4273)*, Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo (Eds.). Springer, 30–43. [https://doi.org/10.1007/11926078\\_3](https://doi.org/10.1007/11926078_3)
- [25] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a Benchmark for Knowledge Intensive Language Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, 2523–2544. <https://doi.org/10.18653/V1/2021.NAACL-MAIN.200>
- [26] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 5687–5711. <https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.378>
- [27] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 9248–9274. <https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.620>
- [28] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-Augmented Black-Box Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (Eds.). Association for Computational Linguistics, 8371–8384. <https://doi.org/10.18653/V1/2024.NAACL-LONG.463>
- [29] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra (Eds.). 2009. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. Lecture Notes in Computer Science, Vol. 5445. Springer. <https://doi.org/10.1007/978-3-642-01907-4>
- [30] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, 809–819. <https://doi.org/10.18653/V1/N18-1074>
- [31] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Trans. Assoc. Comput. Linguistics* 10 (2022), 539–554. [https://doi.org/10.1162/tacl\\_a\\_00475](https://doi.org/10.1162/tacl_a_00475)
- [32] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.
- [33] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 10014–10037. <https://doi.org/10.18653/V1/2023.ACL-LONG.557>
- [34] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85. <https://doi.org/10.1145/2629489>
- [35] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 7534–7550. <https://doi.org/10.18653/V1/2020.EMNLP-MAIN.609>
- [36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*. [http://papers.nips.cc/paper\\_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html)
- [37] Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2022. Match-Prompt: Improving Multi-task Generalization Ability for Neural Text Matching via Prompt Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 2290–2300. <https://doi.org/10.1145/3511808.3557388>
- [38] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-Chain: Interactively Enhancing Large Language Models with Search for Knowledge-intensive Tasks. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 1362–1373. <https://doi.org/10.1145/3589334.3645363>
- [39] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, et al. 2024. CRAG - Comprehensive RAG Benchmark. *arXiv preprint arXiv:2406.04744* (2024).
- [40] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wang, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. 2024. CRAG - Comprehensive RAG Benchmark. *CoRR abs/2406.04744* (2024). <https://doi.org/10.48550/ARXIV.2406.04744>
- [41] 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* (2018).
- [42] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). [http://papers.nips.cc/paper\\_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html)
- [43] Zijun Yao, Yantao Liu, Xin Lv, Shulin Cao, Jifan Yu, Juanzi Li, and Lei Hou. 2023. KoRC: Knowledge Oriented Reading Comprehension Benchmark for Deep Text Understanding. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 11689–11707. <https://doi.org/10.18653/V1/2023.FINDINGS-ACL.743>
- [44] Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation Augmented Iterative Ranking for Knowledge Base Question Answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 6032–6043. <https://doi.org/10.18653/V1/2022.ACL-LONG.417>
- [45] Da Yin, Li Dong, Hao Cheng, Xiaodong Liu, Kai-Wei Chang, Furu Wei, and Jianfeng Gao. 2022. A Survey of Knowledge-Intensive NLP with Pre-Trained Language Models. *CoRR abs/2202.08772* (2022). [arXiv:2202.08772](https://arxiv.org/abs/2202.08772)
- [46] Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, Chunyang Li, Zheyuan Zhang, Yushi Bai, Yantao Liu, Amy Xin, Kaifeng Yun, Linlu Gong, Nianyi Lin, Jianhui Chen, Zhili Wu, Yunjia Qi, Weikai Li, Yong Guan, Kaisheng Zeng, Ji Qi, Hailong Jin, Jinxin Liu, Yu Gu, Yuan Yao, Ning Ding, Lei Hou, Zhiyuan Liu, Bin Xu, Jie Tang, and Juanzi Li. 2024. KoLA: Carefully Benchmarking World Knowledge of Large Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. <https://openreview.net/forum?id=AqN23oqraW>
- [47] Jifan Yu, Xiaohan Zhang, Yifan Xu, Xuanyu Lei, Xinyu Guan, Jing Zhang, Lei Hou, Juanzi Li, and Jie Tang. 2022. XDAL: A Tuning-free Framework for Exploiting Pre-trained Language Models in Knowledge Grounded Dialogue Generation. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data*

Mining, Washington, DC, USA, August 14 - 18, 2022. Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 4422–4432. <https://doi.org/10.1145/3534678.3539135>

[48] Heidi C. Zhang, Sina J. Semnani, Farhad Ghassemi, Jialiang Xu, Shicheng Liu, and Monica S. Lam. 2024. SPAGHETTI: Open-Domain Question Answering from Heterogeneous Data Sources with Retrieval and Semantic Parsing. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 1663–1678. <https://doi.org/10.18653/v1/2024.FINDINGS-ACL.96>

[49] Wenting Zhao, Ye Liu, Tong Niu, Yao Wan, Philip S. Yu, Shafiq Joty, Yingbo Zhou, and Semih Yavuz. 2024. DIVKNOWQA: Assessing the Reasoning Ability of LLMs via Open-Domain Question Answering over Knowledge Base and Text. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (Eds.). Association for Computational Linguistics, 51–68. <https://doi.org/10.18653/v1/2024.FINDINGS-NAACL.5>

[50] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *CoRR* abs/2303.18223 (2023). <https://doi.org/10.48550/ARXIV.2303.18223> arXiv:2303.18223

## A Distribution of Selected Knowledge Sources

In this section, we display the actual selection rate distribution of the three knowledge sources in the reasoning process of ATOMR in Figure 4

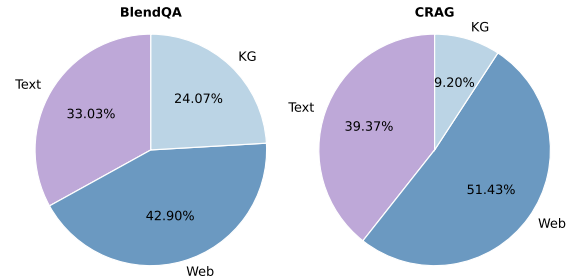


Figure 4: Distribution of selected knowledge sources.

## B In-context Learning Prompts

All prompts for the different operators in ATOMR are shown in Figures 5 to 11.

You are given 3 atomic functions to help you retrieve and operate knowledge from Google, Wikipedia, or Wikidata:

1. Search(). Input: (name, [optional] descriptor). Output: list[entities]. This function helps you find and disambiguate an entity given its name and optional descriptor. If no descriptor is provided, the most popular entity will be returned.
2. Relate(). Input: there are 2 input possibilities, (head\_entity, relation), or (head\_entity, tail\_entity). Output: list[tail\_entities], or list[relations]. This function helps you find the tail\_entities given a head\_entity and relation, or relations given a head\_entity and tail\_entity. You may also search attribute relations using Relate() by treating attributes as tail entities.
3. Filter(). Input: (list[entities], condition). Output: list[entities]. This function helps you filter out entities that satisfy a factual attribute condition.

Construct a hierarchical question decomposition tree in JSON format for the following complex question: "{question}". The tree starts with the original complex question as the root node, and each non-root node is a sub-question of its parent. Continue decomposing until a sub-question cannot be further decomposed and could either be: (1) directly answered by calling one of the three atomic functions Search(), Relate(), Filter(), or (2) directly answered by analyzing the answers of previously answered sibling questions, such as comparing, judging, intersecting, counting, etc. In case (1), write this sub-question with its corresponding function call as a leaf node. In case (2), write this sub-question with a [SIB] mark as a leaf node.

{Examples}  
 Question: {question}  
 Decomposition Tree:

Figure 5: Prompt for Atomic Reasoning Tree generation

Using the retrieved knowledge from Google, Wikipedia, or Wikidata, please answer the following entity disambiguation question "{question}". Answer the question with (1) a Paraphrase Answer that repeats the question, and (2) an Answer List that is a clean entity list. You may not answer with an empty entity list: if the provided information is not enough to answer the question, answer based on your own knowledge. Strictly follow the format of the examples below, ending your answer with "So the answer is: (1) Paraphrase Answer: {{paraphrase\_answer}}; (2) Answer List: [entity\_list]"

Examples.  
Retrieved Knowledge:  
Wikipedia Passages: [1] Kuhle Wampe | Kuhle Wampe is a 1932 German feature film about unemployment, homelessness and left wing politics in the Weimar Republic produced by Prometheus Film.  
Google Results: [1] To Whom Does the World Belong? | Anni Bönike has a badly paid job in a factory ... [2] Kuhle Wampe | Kuhle Wampe is a 1932 German feature film about unemployment, homelessness and left wing politics in the Weimar Republic produced by Prometheus Film.  
Question: What is Kuhle Wampe?  
Answer: Kuhle Wampe is a 1932 German feature film. So the answer is: (1) Paraphrase Answer: Kuhle Wampe is a 1932 German feature film; (2) Answer List: ["Kuhle Wampe (German film)"]  
{more examples}

Your Question.  
Retrieved Knowledge: {retrieved knowledge}  
Question: {question}  
Answer:

Figure 8: Prompt for the Search operator

Please answer the question "{question}" using the retrieved knowledge from Wikipedia, Google, or Wikidata. Answer the question with (1) a Paraphrase Answer that repeats the question, and (2) a clean python Answer List. When answering long lists such as album titles, directly write the list in your answer formulation. If the provided information is not enough to answer the question, answer based on your own knowledge. If neither the provided knowledge nor your own knowledge can answer the question, end your answer with (1) Paraphrase Answer: Unknown; (2) Answer List: []. When answering with numbers, always use arabic numbers i.e. 1,2,3. When answering to "Yes" or "No" questions, simply formulate your Answer list as ["Yes"] or ["No"]. Strictly follow the format of the examples below, ending your answer with "So the answer is: (1) Paraphrase Answer: {{paraphrase\_answer}}; (2) Answer List: [answer\_list]"

Examples.  
Retrieved Knowledge:  
Wikipedia Passages: [1] Daniel Pudil | Daniel Pudil (born 27 September 1985) is a Czech professional footballer who plays for Viktoria Žižkov and the Czech Republic national team as a left back or left winger.  
Google Results: [1] ... 'personal\_information': {'(place\_of\_birth)': 'Prague, Czechoslovakia', 'height': '1.83 m (6 ft 0 in)', 'position\_s': 'Left back, left winger'} [2] Daniel Pudil - Wikidata | Daniel Pudil (2014) (Czech). 1 reference. imported from Wikimedia project ... place of birth - Prague. 1 reference. stated in ...  
Question: In what region of the Czech was Daniel Pudil born?  
Answer: Daniel Pudil was born in Prague. So the answer is: (1) Paraphrase Answer: Daniel Pudil was born in Prague; (2) Answer List: ["Prague"]  
{more examples}

Your Question.  
Retrieved Knowledge: {retrieved knowledge}  
Question: {question}  
Answer:

Figure 9: Prompt for the Relate operator

Using the retrieved knowledge from Wikipedia, Google, or Wikidata, please answer the following filter question "{question}". Formulate a list of entities as your final answer. If the provided passages does not provide helpful information, answer based on your own knowledge. Answer the question with (1) a Paraphrase Answer that repeats the question's filter condition "{condition}", and (2) a clean python Answer List. Strictly follow the format of the examples below, ending your answer with "So the answer is: (1) Paraphrase Answer: {{paraphrase\_answer}}; (2) Answer List: [answer\_list]"

Examples.  
Retrieved Knowledge:  
Wikipedia Passages: [1] Pulitzer Prize for Music | History: guidelines and jury membership will serve that end." Subsequently, in 2006, a posthumous "Special Citation" was given to jazz composer Thelonious Monk, and in 2007 the prize went to Ornette Coleman, a free jazz composer, who won the prize for his disc Sound Grammar, a recording of a 2005 concert, making it the first time a recording won the music Pulitzer, and a first for purely improvised music. In 2018, rapper Kendrick Lamar won the award for his 2017 hip hop album Damn. The recording was the first musical work not in the jazz or classical genres to win the prize. ... [2] Collected Poems of Robert Frost | Reception: Frost received a Pulitzer prize in 1931 for the collection. One of the books in the collection, New Hampshire, had received the Pulitzer Prize in 1924...  
Question: Which of Damn, Damn. Collector's Edition won the Pulitzer Prize?  
Answer: Wikipedia passage [1] suggests that "in 2018, rapper Kendrick Lamar won the award for his 2017 hip hop album Damn", so the album Damn won the Pulitzer Prize. So the answer is: (1) Paraphrase Answer: Damn won the Pulitzer Prize; (2) Answer List: ["Damn"]

Your Question.  
Retrieved Knowledge: {retrieved knowledge}  
Question: {question}  
Answer:

Figure 10: Prompt for the Filter operator

The complex question "{question}" has been divided into child questions. Based on the child questions and answers, answer the complex question. Answer the question with (1) a Paraphrase Answer that repeats the question, and (2) a clean python Answer List. If the provided information is not enough to answer the question, answer based on your own knowledge. If neither the provided knowledge nor your own knowledge can answer the question, end your answer with (1) Paraphrase Answer: Unknown; (2) Answer List: []. When answering with numbers, always use arabic numbers i.e. 1,2,3. When answering to "Yes" or "No" questions, simply formulate your Answer list as ["Yes"] or ["No"]. Strictly follow the format of the examples below, ending your answer with "So the answer is: (1) Paraphrase Answer: {{paraphrase\_answer}}; (2) Answer List: [answer\_list]"

Examples.  
Child questions and answers:  
Q: 1. What is the television series that is a notable work of Christian Lee Navarro? A: ["13 Reasons Why"]  
Q: 2. Who has 2 tapes in [1]? A: Justin Foley has 2 tapes in 13 Reasons Why. ["Justin Foley"]  
Question: Who has 2 tapes in the television series that is a notable work of Christian Lee Navarro?  
Answer: Justin Foley has 2 tapes in the television series that is a notable work of Christian Lee Navarro, 13 Reasons Why. So the answer is: (1) Paraphrase Answer: Justin Foley has 2 tapes in the television series that is a notable work of Christian Lee Navarro, 13 Reasons Why; (2) Answer List: ["Justin Foley"]  
{more examples}

Your Question.  
Retrieved Knowledge: {retrieved knowledge}  
Question: {question}  
Answer:

Figure 11: Prompt for Child and Sibling Answer Reasoning