# Dissecting GraphRAG: A Modular Analysis of Knowledge Structuring for Factoid Question Answering

**Noriki Nishida**[♠] **Rumana Ferdous Munne**[♠] **Shanshan Liu**[♠] **Narumi Tokunaga**[♠]
**Yuki Yamagata**[♠] **Fei Cheng**[♢] **Kouji Kozaki**[♣] **Yuji Matsumoto**[♠]

[♠]RIKEN, Japan [♢]Kyoto University, Japan [♣]Osaka Electro-Communication University, Japan
noriki.nishida@riken.jp

## Abstract

We present a systematic analysis of module-level design choices in GraphRAG, a retrieval-augmented generation framework that integrates structured knowledge graphs into question answering. Focusing on triple extraction, community clustering, and report generation, we evaluate multiple strategies across two knowledge-intensive benchmarks. Our results show that high-quality triple extraction is critical, as the accuracy and coverage of the resulting knowledge graph can become a bottleneck for downstream reasoning. We also find that the granularity of fundamental knowledge units, as determined by community clustering, has a significant impact on downstream performance: Achieving a balance between factual detail and topical coherence within each unit is important to enable precise and comprehensive retrieval and to facilitate effective multi-hop reasoning. In addition, simple template-based reporting outperforms LLM-based summarization in both accuracy and efficiency. These findings provide practical guidance for the structure-aware design of retrieval-augmented systems.

## 1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks (Brown et al., 2020; Kojima et al., 2022; Liu et al., 2023). However, they suffer from inherent limitations in their internalized knowledge, particularly in domains that are highly specialized or rapidly evolving (Dhingra et al., 2022; Luu et al., 2022; Kasai et al., 2023). Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Izacard et al., 2023; Zhao et al., 2024) enhances LLMs by providing access to external documents at inference time, allowing the model to generate responses grounded in up-to-date knowledge.

Recently, GraphRAG (Edge et al., 2024; Wen et al., 2024; Chang and Zhang, 2024; Li et al., 2025; Han et al., 2025a,b) has emerged as a promising variant that incorporates knowledge graphs, constructed from source documents via triple extraction, to generate grounded answers. These graphs organize scattered facts into structured knowledge in the form of `(subject, relation, object)` triples, which capture high-level semantics and make it easier to find relevant information and draw connections between them.

Despite its conceptual appeal, the internal design space of GraphRAG remains under-explored. While prior work (Edge et al., 2024) has demonstrated its utility in tasks such as corpus summarization, its applicability to tasks requiring factual specificity and multi-hop reasoning has not been fully examined. In particular, it is unclear which components of GraphRAG are essential and how their design affects overall performance.

To address this gap, we analyze GraphRAG in a modular manner, examining how different design settings influence its performance on factoid question answering in knowledge-intensive tasks. Rather than asking simply whether graph-based retrieval improves QA, we aim to understand when it helps and which components are essential to its effectiveness. Specifically, we investigate the following questions: (1) *How much does the quality of triple extraction influence downstream QA accuracy?* (2) *What is the optimal granularity for partitioning a knowledge graph into meaningful units?* (3) *Is it necessary to use large language models for report generation, or can simpler alternatives suffice?* To this end, we systematically vary the implementation of each core module in the GraphRAG pipeline, namely, triple extraction, community clustering, and report generation, and evaluate their impact on QA performance. Experiments are conducted



Transactions of the Association for Computational Linguistics, vol. 14, pp. 627–655, 2026. https://doi.org/10.1162/TACL.a.615
Action Editor: Jingbo Shang. Submission batch: 6/2025; Revision batch: 9/2025; Published 5/2026.

on two benchmarks, CDR-QA and DocRED-QA, constructed from biomedical and encyclopedic corpora, respectively.

Our experiments show that the effectiveness of GraphRAG depends not only on the use of a knowledge graph, but also on how the graph is constructed, partitioned, and verbalized. (1) The quality of extracted triples is critical: fine-tuned small language models yield cleaner and more precise triples than LLMs with in-context prompting, leading to more reliable QA performance. (2) The granularity of fundamental knowledge units, which are determined by community clustering, strongly affects the GraphRAG effectiveness: segmenting the graph into moderately fine-grained, topically coherent communities enables both precise and comprehensive retrieval of relevant facts and facilitates effective multi-hop reasoning. (3) For report generation, template-based methods outperform LLM-based counterpart by ensuring complete factual coverage, avoiding omissions or hallucinations, and offering much higher efficiency. To facilitate further research, we release our implementation and two QA benchmarks.[1]

## 2 Related Work

**GraphRAG:** Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Izacard et al., 2023; Zhao et al., 2024; Gao et al., 2024) faces challenges such as sensitivity to noisy retrievals (Cuconasu et al., 2024; Yang et al., 2024b; Soudani et al., 2024; Chen et al., 2024a; Wang et al., 2025a), imcomplete retrievals (Leto et al., 2024), and limited reasoning over retrieved contexts (Yang et al., 2024b; Chen et al., 2024a; Kim et al., 2025). GraphRAG mitigates these issues by leveraging knowledge graphs (Edge et al., 2024; Wen et al., 2024; Chang and Zhang, 2024; Li et al., 2025; Han et al., 2025a,b). While prior work shows its effectiveness, most studies focus on specific components or domains, e.g., graph-based retrieval (Huang et al., 2023; Xu et al., 2025; Zhu et al., 2025), graph-based reasoning (Jin et al., 2024; Han et al., 2025c), recommendation (Wang et al., 2025b), or biomedical applications (Wu et al., 2024). This work fills the gap by conducting a modular analysis of GraphRAG, examining how knowledge graph quality, graph partitioning granularity, and verbalization methods affect factoid QA performance.

**Triple Extraction:** Triple extraction, comprising named entity recognition (Yu et al., 2020), entity disambiguation (Wu et al., 2020), and relation extraction (Zhou et al., 2020), has been widely studied with a focus on accuracy or model architecture (Ye et al., 2022; Zhong et al., 2023). Recent work often adopts LLM-based extraction, using few-shot prompting to derive triples from raw text (Wadhwa et al., 2023; Ozyurt et al., 2024; Bhattarai et al., 2024). We aim to shift focus from extraction accuracy alone to its downstream impact on factoid QA. We compare extractors with different precision-coverage trade-offs and analyze their effects on GraphRAG in knowledge-intensive QA.

**Community Clustering:** GraphRAG systems typically adopt two main strategies for detecting communities (subgraphs) within knowledge graphs. The first extracts entities from the user query, links them to concept nodes, and traverses the corresponding subgraphs (Feng et al., 2020; Xia et al., 2025). This method works well when extraction and linking are accurate and the knowledge graph is complete, but degrades when extraction fails, a suitable linker is unavailable, or the graph is incomplete (Huang et al., 2023; Xu et al., 2025). The second strategy detects communities in advance using algorithms (Traag et al., 2019; Chang and Zhang, 2024; Yu et al., 2023; Li et al., 2024), thereby avoiding reliance on linking. However, prior work in Document-RAG (Chen et al., 2024b; Sarthi et al., 2024; Zhong et al., 2025) shows that community granularity significantly affects response quality, and naive partitioning can be suboptimal. Building on these insights, we focus on the latter strategy and evaluate how different graph partitioning and subgraph extraction methods influence QA performance, identifying granularity levels that best support accurate answering.

**Report Generation:** The task of verbalizing knowledge graphs, or converting structured graph data into natural language, has long been studied as graph-to-text generation (Ke et al., 2021; Zhao et al., 2023), predating the rise of LLMs. There are two main strategies. The first is template-based generation, which uses predefined patterns to

[1] https://github.com/norikinishida/kapipe.

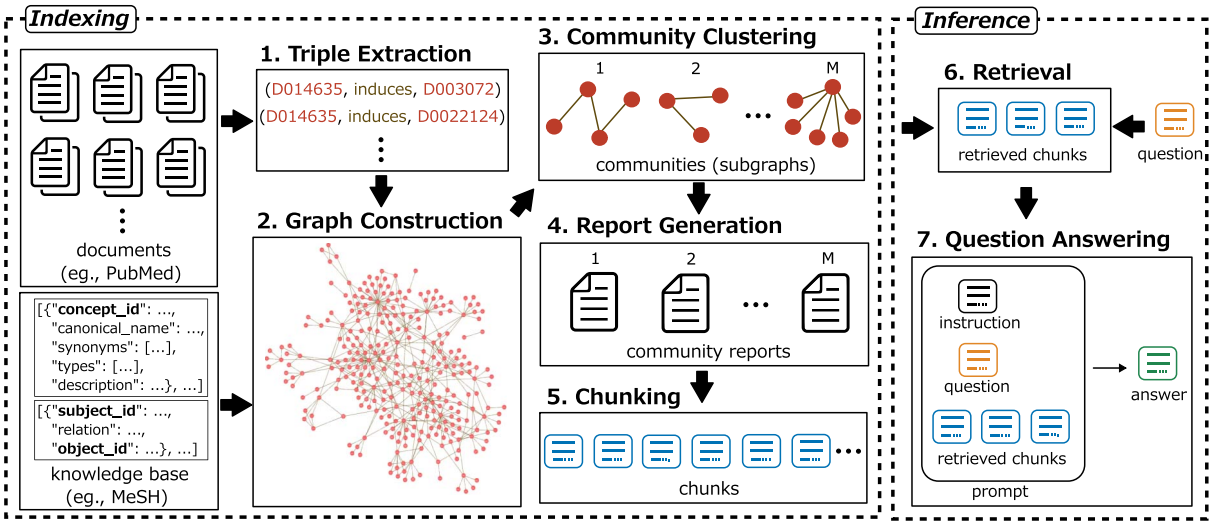



Figure 1: Overview of our GraphRAG pipeline. The study focuses on module-level variations in triple extraction, community clustering, and report generation.

transform triples into text (Huang et al., 2023; Yu et al., 2023; Li et al., 2024), The second strategy uses pretrained language models (Edge et al., 2024; Yuan and Faerber, 2023), which generate text from graph structures but often struggle to capture graph topology (Chang and Zhang, 2024) and require substantial computational and financial resources. In this work, we evaluate how LLM-based and template-based verbalization strategies impact downstream QA performance, aiming to clarify their effect on overall GraphRAG effectiveness.

## 3 Methods

### 3.1 GraphRAG Framework

GraphRAG is a retrieval-augmented generation framework that constructs structured knowledge from source documents and leverages it to answer user queries. Our implementation follows a seven-stage pipeline, outlined in Figure 1. The pipeline is conceptually divided into two phases: (i) *Offline knowledge indexing* (Stages 1–5), where documents are transformed into a knowledge graph, then partitioned, verbalized, chunked, and indexed to support efficient retrieval; (ii) *Online question answering* (Stages 6–7), where relevant chunks are retrieved and used to generate answers.

The overall process is as follows: (1) **Triple Extraction** extracts `(subject, relation, object)` triples from raw text. This step defines the semantic backbone of the knowledge graph. A relation refers to a semantic connection between entities, categorized in a knowledge base (e.g., `located_in`). (2) **Graph Construction** builds a directed knowledge graph by merging entity mentions that share the same knowledge base concept ID (e.g., MeSH term,[2] DBpedia label (Bizer et al., 2009)[3]). (3) **Community Clustering** partitions the graph into semantically coherent subgraphs (*communities*), each of which serves as a fundamental unit of knowledge. The choice of clustering strategy determines the granularity (i.e., the size and specificity of each unit) and cohesion of these units. (4) **Report Generation** converts each community into a natural language report (or summary), making structured knowledge accessible to language models. (5) **Chunking** splits each community report into non-overlapping text chunks, capped at 100 tokens, and prepends the community title to preserve context. These chunks are then indexed to support dense retrieval. (6) **Retrieval** selects top-$k$ relevant chunks using a pre-trained dense retriever, which scores similarity between queries and candidate chunks via inner product. (7) **Question Answering** uses GPT-4o (OpenAI, 2024) to generate an answer based on the retrieved chunks and the user query.

Although all stages contribute to the overall system, we focus our empirical analysis on the three core modules: triple extraction, community clustering, and report generation, as they directly

[2] `https://www.nlm.nih.gov/mesh/meshhome.html`.

[3] `https://www.dbpedia.org/`.

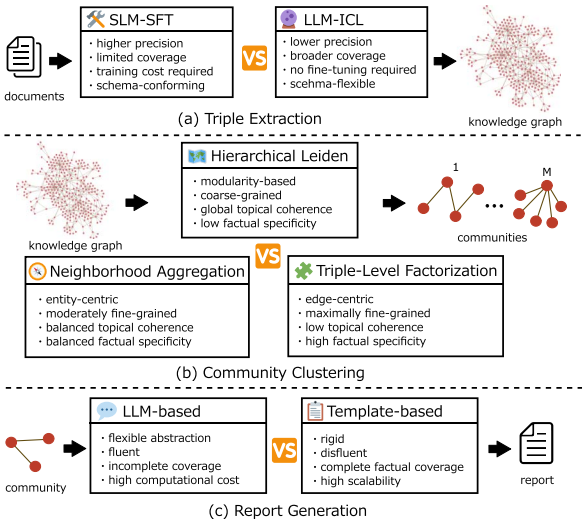



Figure 2: Variants of core modules in the GraphRAG pipeline. Each block illustrates alternative strategies for a key module: Triple Extraction, Community Clustering, and Report Generation.

shape how knowledge is extracted, organized, and represented for retrieval and question answering. Other components are held constant across configurations to isolate the effects of these design choices. We now detail the alternative strategies evaluated for each core module.

### 3.2 Triple Extraction Strategies

The triple extraction process consists of three subtasks. First, Named Entity Recognition (NER) identifies entity mentions in text and assigns predefined types such as `Chemical`, `Disease`, or `Person` to them. Second, Entity Disambiguation (ED) links each mention to a canonical concept in an external knowledge base (e.g., MeSH or DBpedia), merging mentions that refer to the same concept ID. In practice, ED itself involves two steps: (i) retrieving candidate concepts for each mention, and (ii) reranking these candidates to select the best match. Finally, Document-level Relation Extraction (DocRE) identifies semantic relationships between entity pairs that co-occur within a document.

To assess how extraction quality affects downstream QA performance, we compare two strategies for this pipeline, as illustrated in Figure 2(a). These differ in their degree of schema control and reliance on supervised training data.

**SLM-SFT Triple Extraction:** The first strategy uses a pipeline of smaller language models (SLMs), each fine-tuned for a specific subtask via supervised fine-tuning (SFT). For NER, we use Biaffine-NER (Yu et al., 2020) to identify entity mentions and assign predefined types. ED is performed using BLINK (Wu et al., 2020), which adopts a two-stage architecture: a Bi-Encoder for retrieving candidate concepts from the knowledge base, followed by a Cross-Encoder for reranking these candidates to select the most appropriate concept for each mention. DocRE is handled by ATLOP (Zhou et al., 2020), trained to predict relations between entity pairs. All components are trained on gold-annotated datasets such as CDR (Li et al., 2016) and Linked-DocRED (Genest et al., 2023). This strategy yields high-precision triples that conform to a fixed schema defined by the underlying dataset. However, its coverage is inherently constrained by the predefined entity and relation types available in the training.

**LLM-ICL Triple Extraction:** In the second strategy, a large language model (LLM) is applied to multiple subtasks. It performs NER and DocRE through in-context learning (ICL) (Wadhwa et al., 2023; Ozyurt et al., 2024; Bhattarai et al., 2024), and also contributes to ED by reranking candidate concepts. Candidate retrieval for ED is still performed by the BLINK Bi-Encoder, as the number of concepts in large knowledge bases (e.g., MeSH, DBpedia) is prohibitively large and cannot be reasonably included in the LLM prompt. We provide few-shot demonstrations to prompt the model to (i) extract entity mentions and assign types, (ii) rerank candidate concepts for each mention, and (iii) identify relations directly from raw text (see Figures 6, 7, and 8 in Appendix A for the prompt templates). This strategy allows for schema flexibility, enabling the model to produce entity and relation types beyond those defined in the training schema. As a result, it has the potential to discover long-tail relations that may be missed by supervised fine-tuning models. However, as the model is not supervised and operates without schema constraints, the outputs are often noisy, including hallucinated relation types. Rather than filtering these outputs, we retain them in the graph.

### 3.3 Community Clustering Strategies

To examine how community granularity influences retrieval and QA performance, we compare three clustering strategies that vary in structural assumptions and level of detail. As shown in Figure 2(b), these methods range from broad,

topic-based groupings to fine-grained, triple-level partitions. Evaluating them within the same GraphRAG framework allows us to assess which granularity is most effective for factoid QA.

**Hierarchical Leiden:** Hierarchical Leiden recursively applies the Leiden algorithm (Traag et al., 2019) to partition the knowledge graph into communities. At each level, it optimizes modularity based on node connectivity and attempts to further subdivide any community that exceeds a predefined size threshold. Subdivision continues until no additional splits are accepted by the algorithm or all resulting communities satisfy the size constraint. Even the smallest communities tend to be coarser-grained than those produced by more localized strategies, such as Neighborhood Aggregation or Triple-Level Factorization. Despite this, Hierarchical Leiden provides global coverage and a hierarchy-aware decomposition, and has been adopted in prior work (Edge et al., 2024), making it a strong structural baseline for graph partitioning. In our experiments, we fixed the hyperparameters rather than tuning them: specifically, the threshold was set to 10 across all datasets to isolate the effect of community granularity.

**Neighborhood Aggregation:** Neighborhood Aggregation forms communities by grouping each node with its immediate in- and out-neighbors, producing fine-grained and highly localized subgraphs. These clusters preserve entity-centric semantics and reflect the immediate relational context of each entity, in contrast to the coarser-grained communities produced by Hierarchical Leiden. This strategy has no tunable hyperparameters: the neighborhood size is fixed to one hop and applied deterministically for every node.

**Triple-Level Factorization:** Triple-Level Factorization takes an edge-centric approach, assigning each individual `(subject, relation, object)` triple to its own atomic community. This design yields maximally fine-grained knowledge units, each representing a single atomic fact, and eliminates intra-community noise. However, the lack of surrounding context makes it challenging to support multi-hop reasoning in cases that require integrating multiple related facts. This strategy is parameter-free by design, as each triple deterministically forms its own atomic community.

### 3.4 Report Generation Strategies

Each community (i.e., subgraph) is converted into a natural language report, which serves as the interface between structured knowledge and the language model. We compare two strategies for report generation: prompting an LLM to produce structured summaries, and using templates to generate fixed-format text (Figure 2(c)). The former offers fluent, adaptive descriptions, while the latter ensures efficiency and full factual coverage. We evaluate how these choices affect answer accuracy and scalability in GraphRAG.

**LLM-Based Report Generation:** In this strategy, a large language model is prompted to generate a report for each community. First, the LLM takes as input the community's entities, the relationships between them, and, if available, the reports of its child communities, and produces a JSON-style object using the prompt template shown in Figure 9 (Appendix A). This object contains a title describing the community, a summary of its content, and a list of several findings. The JSON-style object is then converted into a plain-text report based on a deterministic template (see Figure 10 in Appendix A). Specifically, the title appears as the report heading, followed by the summary as an introductory paragraph, and then the findings are listed sequentially.

**Template-Based Report Generation:** This strategy provides a lightweight alternative to LLM-based report generation by constructing each report deterministically from the graph structure. For each community, we identify its most central entities (top three by degree) to form the report title. The body text is then generated using a fixed template that enumerates all entities in the format ''`{name} | {type} | {definition}`,'' where entity names, types, and definitions are derived from external dictionaries such as MeSH or DBpedia, followed by a list of all relations in the form ''`{head} | {relation} | {tail}`.'' The template used to render these components into a plain-text report is shown in Figure 11 in Appendix A. This deterministic formatting guarantees full coverage of the community content and efficiency, at the cost of reduced fluency compared to LLM-based generation.

| Dataset | Question Type | Question-Answer Pair |
|---|---|---|
| CDR-QA | Neighborhood | **Question**: *What chemical compounds induce Defects, Intraventricular Septal?*<br>**Answer**: *Sulfasalazine*; *Aspirin*; *flumioxazin* |
| | Intersection | **Question**: *What diseases are commonly induced by both Oral Contraceptives and Unfractionated Heparin?*<br>**Answer**: *Venous Thromboembolism*; *Venous Thrombosis* |
| | Multi-hop | **Question**: *What chemical compounds induce the diseases that are also induced by Clonazepam?*<br>**Answer**: *Amiodarone*; *Alprazolan*; *Ammonia*; . . . |
| DocRED-QA | Neighborhood | **Question**: *On which platform was Shadowrun Returns released?*<br>**Answer**: *Microsoft Windows*; *Linux* |
| | Intersection | **Question**: *Which sports team do both Davy Jones (baseball) and Bo McLaughlin play for?*<br>**Answer**: *Chicago Cubs*; *Baltimore Orioles* |
| | Multi-hop | **Question**: *What did the person who has Emily Brontë as a sibling write?*<br>**Answer**: *Villete (novel)*; *Jane Eyre* |

Table 1: Examples of the three question types in CDR-QA and DocRED-QA. Each question expects a set of entity-level answers. Underlines indicate entities instantiated into the question template placeholders.

### 3.5 Chunking, Retrieval, and QA

Each community report, once generated, is treated as a standalone document and segmented into fixed-length chunks. These chunks are stored in the vector database and serve as candidates during query-time retrieval. At inference time, the retriever selects the top-$k$ relevant chunks. Then, the top-$k$ chunks are concatenated with the query and provided to the LLM for answer generation. We did not apply any fine-tuning to the LLM. The prompt template used for answer generation is shown in Figure 12 in Appendix A.

## 4 Experimental Setup

### 4.1 Datasets

Most existing QA datasets offer only independent QA pairs, or at best link questions to a single document or an existing knowledge graph. In contrast, few datasets provide a complete package that combines documents, manually annotated triples, a multi-document knowledge graph constructed from them, and reasoning-based questions with multiple correct answers.

Thus, we introduce two new QA datasets, **CDR-QA** and **DocRED-QA**, to enable a systematic analysis of GraphRAG's performance. CDR-QA and DocRED-QA were constructed based on biomedical (CDR (Li et al., 2016)) and encyclopedic (Linked-DocRED (Genest et al., 2023)) corpora with manual annotations. They pair multiple documents with gold triples and reasoning-oriented questions that are fully answerable on the constructed knowledge graphs, often requiring multi-hop reasoning and admitting multiple correct answers. Each question is annotated with a ground-truth answer set, represented as canonical concept IDs accompanied by multiple surface-form synonyms, and the answer set always contains at least two elements. This design supports both (i) evaluating the effect of automatic triple extraction, and (ii) conducting controlled experiments under an ''ideal extraction'' condition, where gold triples are provided so that subsequent modules can be assessed independently.

**Knowledge Graph Construction:** We now detail the construction procedures for CDR-QA and DocRED-QA. First, for each QA dataset, we constructed a directed knowledge graph from manually annotated triples provided in existing corpora. For CDR-QA, we used the CDR corpus (Li et al., 2016), which contains 1,500 PubMed abstracts annotated with chemical and disease mentions linked to MeSH terms, along with causal relations (`Chemical-Induce-Disease`) between entities. For DocRED-QA, we used the Linked-DocRED corpus (Genest et al., 2023), which includes 4,051 Wikipedia articles annotated with named entities linked to DBpedia and inter-entity relations such as `located_in`.

**Question Generation:** Then, from each knowledge graph, we generate three types of factoid questions designed to probe different reasoning patterns: *neighborhood*, *intersection*, and *multi-hop*. These question types vary in the number and structure of graph traversals required to reach the answer: (1) **Neighborhood Questions** ask for all entities directly connected to a given

| Dataset | Type | # Cases | # Answers | | |
|---|---|---|---|---|---|
| | | | Tot. | Avg. | Max. |
| CDR-QA | Neighborhood | 128 | 798 | 6.23 | 60 |
| | Intersection | 128 | 310 | 2.42 | 6 |
| | Multi-Hop | 128 | 5089 | 39.76 | 191 |
| DocRED-QA | Neighborhood | 128 | 412 | 3.22 | 18 |
| | Intersection | 128 | 304 | 2.38 | 7 |
| | Multi-Hop | 128 | 8,441 | 65.95 | 2,264 |

Table 2: Statistics of question types and answer counts in CDR-QA and DocRED-QA (min. = 2 in all cases).

node via a specific relation. For example, given a chemical and the CHEMICAL-INDUCE-DISEASE relation, the question asks which diseases are induced by that chemical. These questions test the system's ability to retrieve facts through single-hop relations. (2) **Intersection Questions** ask for entities that are connected to two different nodes via the same relation. For example, a question might ask which diseases are induced by both Drug A and Drug B. These questions test the model's ability to identify overlapping facts under a shared relational constraint. (3) **Multi-Hop Questions** ask for entities that are connected to the source entity through a chain of two relations. For example, given a chemical, the question might ask which other chemicals are connected via an intermediate disease (e.g., chemical $\rightarrow$ disease $\rightarrow$ chemical). These questions evaluate the model's ability to perform compositional reasoning and traverse multiple relational steps. For each question type, we generate 128 question–answer pairs per dataset using predefined templates (Yang et al., 2018; Sciavolino et al., 2021; Mallen et al., 2023). We show question-answer examples in Table 1.

**Statistics:** Table 2 shows the statistics of CDR-QA and DocRED-QA. Both datasets include three types of reasoning-oriented questions: Neighborhood, Intersection, and Multi-Hop. In CDR-QA, Neighborhood and Intersection questions yield relatively small answer sets (average 6.23 and 2.42, respectively), while Multi-Hop questions produce substantially larger answer spaces (average 39.76 answers, with a maximum of 191). This distribution indicates that even relatively short reasoning chains in the biomedical knowledge graph can lead to a large number of possible answers, which makes multi-hop reasoning particularly challenging. In DocRED-QA, a similar pattern emerges on a larger scale. Multi-Hop questions yield much larger answer sets, with an average of 65.95 answers and extreme cases exceeding 2K entities. This reflects the dense relational structure of the Linked-DocRED dataset, which is constructed from encyclopedic documents. By contrast, Neighborhood and Intersection questions remain more constrained (average 3.22 and 2.38 answers, respectively).

**Triple Extraction Datasets:** We used annotated corpora to support supervised training and in-context prompting for the triple extraction modules, including named entity recognition (NER), entity disambiguation (ED), and document-level relation extraction (DocRE). For CDR-QA, we used the CDR dataset, which provides 500 training examples. For DocRED-QA, we used the training split of Linked-DocRED. For LLM-based extraction, we sampled the 3 most densely annotated examples from each training set as few-shot examples.

### 4.2 Evaluation Metrics

We evaluate QA performance using a recall-based metric that measures how completely a model recovers the correct answer entities. Let $G_q$ be the set of gold-standard entities for question $q$, and let $a_q$ be the model-generated response. Each gold entity is associated with a set of known surface-form synonyms, and is considered correctly predicted if any of its synonyms appears as a substring in $a_q$. Formally, **Answer Recall** is defined as $\frac{\sum_q \text{NUMCOVERED}(a_q, G_q)}{\sum_q |G_q|}$, where $\text{NUMCOVERED}(a_q, G_q)$ returns the number of gold entities in $G_q$ that are covered by the model's answer $a_q$.[4]

In addition to Answer Recall, we measure how completely the retrieval stage recovers the necessary supporting triples within the top-$k$ results. Formally, **Evidence Recall@k** is defined as $\frac{\sum_q |\{t \in T_q : t \in U_q^k\}|}{\sum_q |T_q|}$, where $q$ denotes a question, $T_q$ is the set of gold support triples required to answer $q$, $R_q^k$ is the set of top-$k$ retrieved chunks, and $U_q^k$ is the union of triples contained in those chunks:

[4]While recall can be directly measured against the gold annotations, a precision-based metric cannot be meaningfully defined in our setting. This is because plausible entities absent from the knowledge graph (or annotations in CDR or Linked-DocRED) would be counted as false positives, even though they may in fact be correct. For this reason, we report recall as the primary metric, which emphasizes coverage of gold-standard entities without penalizing plausible extra outputs.

$U_q^k = \bigcup_{c \in R_q^k} \text{Triples}(c)$. In practice, $\text{Triples}(c)$ is obtained by mapping a retrieved chunk $c$ back to its source report and collecting the triples associated with that report,[5]

### 4.3 Default Configuration

Unless otherwise specified, all experiments follows a default configuration of GraphRAG (Edge et al., 2024) in which only the core modules are varied, while all other components are held constant. By default, triple extraction uses manually annotated triples to provide an upper-bound input for downstream modules. Community clustering is performed using the Hierarchical Leiden, and report generation uses an LLM-based strategy. Chunking is applied by splitting each report into non-overlapping spans of up to 100 tokens, with the corresponding community title prepended. Retrieval is conducted using Contriever (Izacard et al., 2021), which ranks chunks by inner product similarity and selects the top 10 chunks per question. Answer generation is performed using GPT-4o (OpenAI, 2024), conditioned on the retrieved chunks.

## 5 Results and Discussion

### 5.1 How Much Do GraphRAG Design Choices Matter?

We begin by asking whether different design configurations of GraphRAG lead to significant differences in QA performance. We compare the following systems: (1) Direct LLM, where the LLM generates an answer without any retrieved context; (2) Document-RAG, which retrieves passage chunks from source documents; (3) GraphRAG (default), which uses knowledge graphs constructed from the same sources and mainly follows common design choices in prior work (Edge et al., 2024); and (4) GraphRAG (best), which employs the best-performing configuration identified through module-level analysis (see subsequent sections). In all cases, the final answer is generated by GPT-4o conditioned on the selected context.

[5]As a result, there is no guarantee that the chunk itself explicitly describes those triples, particularly for LLM-based report generation, where not all community triples are faithfully expressed, and for chunking, where the alignment between chunks and triples is imprecise. Evidence Recall@k should therefore be interpreted as an approximate upper bound on retrieval quality.

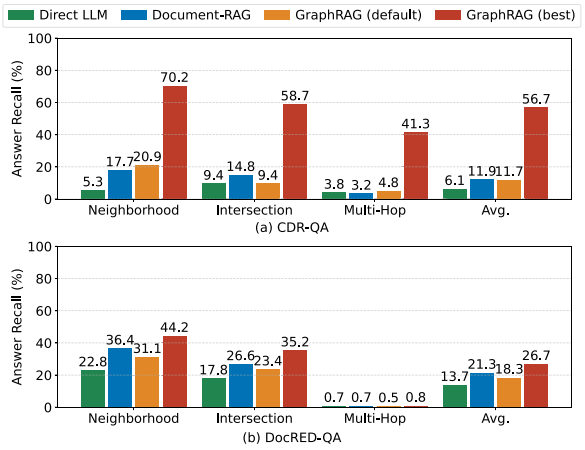



Figure 3: Answer Recall for four system configurations on CDR-QA (top) and DocRED-QA (bottom). GraphRAG achieves substantially higher performance when all modules are carefully configured, highlighting the importance of design choices across benchmarks.

As shown in Figure 3, GraphRAG in its default configuration slightly underperformed Document-RAG, consistent with prior findings (Edge et al., 2024) that graph-based RAG often falls short on factoid queries. This suggests that without careful design, structured knowledge integration alone does not guarantee improved QA performance. More notably, the best-performing GraphRAG configuration, which combines SLM-SFT Triple Extractor, Neighborhood Aggregation, and Template-based Report Generator, yielded substantial further improvements, achieving 56.7% Answer Recall on CDR-QA and 26.7% on DocRED-QA. These results indicate that **the effectiveness of GraphRAG hinges not only on the use of a knowledge graph, but also on how that graph is constructed, clustered, and verbalized**. Carefully chosen module configurations significantly enhance the utility of graph-structured knowledge and directly impact the quality of downstream answers.

### 5.2 How Much Does Triple Extraction Quality Affect Performance?

Triple extraction forms the foundation of the knowledge graph, yet its quality can vary substantially depending on the extraction strategy. We compare three configurations: (1) Manual, which uses gold-standard triples from CDR and Linked-DocRED; (2) SLM-SFT, which uses task-specific smaller language models fine-tuned on those annotations; and (3) LLM-ICL, which

| Dataset | Triple Extraction | Ans. Recall | Ev. Recall@10 | NER | | | ED | DocRE | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | P | R | F1 | Acc. | P | R | F1 |
| CDR-QA | Manual | **11.7** | **26.3** | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| | SLM-SFT | 7.7 | 14.5 | 87.7 | 89.1 | 88.4 | 90.7 | 66.0 | 70.2 | 68.0 |
| | LLM-ICL | 4.8 | 8.4 | 60.6 | 80.8 | 69.3 | 89.5 | 38.2 | 90.2 | 53.7 |
| DocRED-QA | Manual | **18.3** | **45.1** | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| | SLM-SFT | 18.2 | 33.0 | 88.2 | 88.0 | 88.1 | 85.5 | 66.5 | 58.9 | 62.5 |
| | LLM-ICL | 14.5 | 5.6 | 70.7 | 76.1 | 73.3 | 87.5 | 10.8 | 18.7 | 13.7 |

Table 3: Comparison of triple extraction strategies on CDR-QA and DocRED-QA (averaged over the three question types), and performance on NER, ED, and DocRE on the CDR test set and the Linked-DocRED development set. Each score is reported with precision (P), recall (R), F1, and In-KB normalized accuracy (Wu et al., 2020).

| Community Clustering | Answer Recall | | | Evidence Recall@10 | | | Community Statistics | |
|---|---|---|---|---|---|---|---|---|
| | Manual | SLM-SFT | LLM-ICL | Manual | SLM-SFT | LLM-ICL | # Comm. | Avg. Size |
| **CDR-QA** | | | | | | | | |
| Hierarchical Leiden | 11.7 | 7.7 | 4.8 | 26.3 | 14.5 | 8.4 | 330 | 9.6 |
| Neighborhood Aggregation | **33.5** | **23.6** | **16.9** | **58.8** | **42.9** | **37.2** | 1,263 | 4.9 |
| Triple-Level Factorization | 24.7 | 17.5 | 11.0 | 27.2 | 17.8 | 9.7 | 2,435 | 2.0 |
| **DocRED-QA** | | | | | | | | |
| Hierarchical Leiden | 18.3 | 18.2 | 14.5 | 45.1 | 33.0 | 5.6 | 3,099 | 13.7 |
| Neighborhood Aggregation | **26.5** | **25.1** | **19.7** | **74.3** | **56.4** | **10.4** | 17,859 | 3.7 |
| Triple-Level Factorization | 22.4 | 20.4 | 19.7 | 54.8 | 39.9 | 7.7 | 29,205 | 2.0 |

Table 4: Comparison of community clustering strategies on CDR-QA and DocRED-QA (averaged over the three question types). Columns under ''Answer Recall'' and ''Evidence Recall@10'' correspond to triple extraction methods used to construct the input graph. We also report community statistics (number of communities and their average size) for the manually constructed graph.

uses in-context learning with GPT-4o-mini to perform triple extraction without additional training. Other modules are fixed to their default settings (Hierarchical Leiden for community clustering, LLM-based report generation).

As shown in Table 3, manually annotated triples unsurprisingly achieved the best results, establishing an upper bound. Among automatic methods, SLM-SFT consistently outperformed LLM-ICL, with gains of +2.9 Answer Recall and +6.1 Evidence Recall@10 on CDR-QA, and +3.7 Answer Recall and +27.4 Evidence Recall@10 on DocRED-QA. This advantage is largely explained by higher DocRE precision (66.0 vs. 38.2 on CDR, 66.5 vs. 10.8 on DocRED). These results indicate that **the quality of extracted triples critically determines GraphRAG performance**. Even when the source corpus is identical, differences in knowledge graph quality can lead to significant variation in GraphRAG's final performance. Moreover, **not only broad coverage but also precision in triple extraction is important**. Noisy or hallucinated relations (common in LLM-ICL) can undermine both answer and evidence recall, while precise extraction substantially improves downstream QA. Results of the error analysis of the LLM-ICL strategy are also provided in Appendix B.1.

### 5.3 What Is the Optimal Granularity for Community Clustering?

To evaluate how the granularity of knowledge graph partitioning affects QA performance, we compare three clustering strategies: (1) Hierarchical Leiden, a global modularity-based method that produces coarse-grained clusters; (2) Neighborhood Aggregation, a localized method that forms communities from each node and its immediate neighbors; and (3) Triple-Level Factorization, an edge-centric method that treats each triple as an independent community.

As shown in Table 4, Neighborhood Aggregation consistently outperformed both Hierarchical Leiden and Triple-Level Factorization, yielding the highest Answer Recall and Evidence Recall@10 across datasets and extraction settings.

| Dataset | Report Generation | Answer Recall | | | Evidence Recall@10 | | | Total Time [min.] |
|---|---|---|---|---|---|---|---|---|
| | | HL | NA | TF | HL | NA | TF | |
| CDR-QA | LLM-based | 11.7 | 33.5 | 24.7 | 26.3 | 58.8 | 27.2 | 46.129 |
| | Template-based | **24.4** | **56.7** | **28.6** | **41.1** | **70.4** | **32.0** | 0.005 |
| DocRED-QA | LLM-based | 18.3 | 26.5 | 22.4 | 45.1 | **74.3** | 54.8 | 590.905 |
| | Template-based | **22.9** | **26.7** | **25.3** | **62.4** | 74.1 | **59.3** | 0.070 |

Table 5: Comparison of report generation strategies on CDR-QA and DocRED-QA (averaged over the three question types). Columns under ''Answer Recall'' and ''Evidence Recall@10'' correspond to community clustering methods used to create the input communities: Hierarchical Leiden (HL), Neighborhood Aggregation (NA), and Triple-Level Factorization (TF). We also report the total processing time (in minutes) required to generate reports for all communities obtained by applying Hierarchical Leiden to the manually annotated graph.

Hierarchical Leiden produced the fewest but largest communities (e.g., 330 with an average of 9.6 nodes on CDR-QA), while Triple-Level Factorization produced the most numerous and smallest ones (e.g., 2,435 communities with 2.0 nodes). Neighborhood Aggregation offered moderate granularity (1,263 communities, 4.9 nodes). These results highlight **the importance of choosing an appropriate level of granularity when partitioning the knowledge graph**, which is roughly consistent with prior findings on Document-RAG (Chen et al., 2024b). If the communities are too coarse (as in Hierarchical Leiden), they may help with multi-hop reasoning, but important facts can get lost in broad and less focused contexts, making them harder to retrieve accurately. On the other hand, if the communities are too fine-grained (as in Triple-Level Factorization), they keep individual facts clear, but split related information into separate parts, which makes it hard to gather enough context for reasoning. **Neighborhood Aggregation strikes a balance**: by preserving local semantic cohesion while avoiding excessive breadth or fragmentation, it improves the precision and recall of retrieval and facilitates multi-hop reasoning. This suggests that granularity is not a mere structural detail but a key design variable that directly shapes how well structured knowledge can support complex information needs. Results of the sensitivity analysis on hyperparameters are presented in Appendix B.2.

### 5.4 Does Template-Based Report Generation Outperform LLMs?

Once the knowledge graph is partitioned into communities, these must be transformed into a textual report that serves as a retrieval unit. To evaluate how the report generation strategy affects downstream QA, we compare two systems: (1) LLM-based, which uses GPT-4o-mini to generate structured summaries (including a title, abstract, and key findings) that are then converted into natural language; and (2) Template-based, which deterministically formats entities and triples in each community using predefined templates.

As shown in Table 5, the Template-based generator achieved higher Answer Recall and Evidence Recall@10 than the LLM-based alternative across nearly all community clustering settings and datasets, with only one exception. Template-based generation also required substantially less processing time (e.g., 0.005 minutes vs. 46.129 minutes). These findings indicate that in fact-centric QA, **fully capturing key details is more important than producing natural-sounding text**. LLM-based reports, despite their fluency, often omitted important facts or introduced irrelevant or hallucinated content, which degraded retrieval quality and answer accuracy. In contrast, the template-based approach ensured full coverage of entities and relations in each community, providing consistent input for retrieval and improving both Answer Recall and Evidence Recall@10. In addition, **Template-based generation offers a more efficient and scalable solution for structured retrieval and factoid QA**, as it requires orders of magnitude less processing time compared to LLM-based generation. Results of the analysis of LLM-based report generation variations are shown in Appendix B.3.

### 5.5 Ablation Study

While the previous sections analyzed each module of GraphRAG in isolation, we now turn to

| TE | CC | RG | CDR-QA | DocRED-QA |
|---|---|---|---|---|
| Manual | NA | Temp. | **56.7** | **26.7** |
| LLM | NA | Temp. | 28.0 (−28.7) | 21.1 (−5.6) |
| Manual | HL | Temp. | 24.4 (−32.3) | 22.9 (−3.8) |
| Manual | NA | LLM | 33.5 (−23.2) | 26.5 (−0.2) |
| LLM | HL | LLM | 4.8 (−51.9) | 14.5 (−12.2) |

Table 6: Ablation results on CDR-QA and DocRED-QA (averaged over three question types), showing Answer Recall (%) under different module settings. Each configuration replaces one module from the best-performing setup while keeping others fixed. Parentheses indicate recall drop relative to the best. Abbreviations: TE = Triple Extraction; CC = Community Clustering; RG = Report Generation; HL = Hierarchical Leiden; NA = Neighborhood Aggregation; Temp. = Template-based.

a more holistic question: Among triple extraction (TE), community clustering (CC), and report generation (RG), which module contributes most to overall QA performance? To answer this, we perform a controlled ablation study. Starting from the best-performing configuration, we vary one module at a time by replacing it with its least effective alternative, while holding the other two fixed. This allows us to isolate the impact of each module.

As shown in Table 6, replacing triple extraction or community clustering with their weakest alternatives caused substantial drops in Answer Recall on CDR-QA (−28.7 and −32.3, respectively) and moderate drops on DocRED-QA (−5.6 and −3.8). Changing the report generation method led to smaller decreases (−23.2 on CDR-QA and −0.2 on DocRED-QA). These findings indicate that all three modules contribute to GraphRAG's effectiveness, but **high-quality triple extraction and appropriate community granularity are especially critical**. Errors in either can compromise the structure and retrievability of the knowledge graph, leading to significant losses in QA performance.

### 5.6 Retrieval Scale Analysis

The effectiveness of retrieval-augmented QA systems depends not only on the quality of retrieved content, but also on how well the system scales with increased retrieval depth. In this section, we investigate how retrieval scale, i.e., the number of retrieved chunks ($k$), impacts the QA performance of two contrasting approaches: Document-RAG, which retrieves unstructured raw text, and GraphRAG, which retrieves structured symbolic summaries. By varying $k \in \{1, 5, 10, 15, 20, 25, 30\}$, we analyze how each system responds to additional context and whether they benefit equally from larger retrieval budgets. We report Answer Recall (%) as the evaluation metric. All results are based on CDR-QA (similar trends were observed on DocRED-QA). For GraphRAG, we use the best-performing configuration identified in previous sections (manual triple extraction, Neighborhood Aggregation, and Template-based report generation).

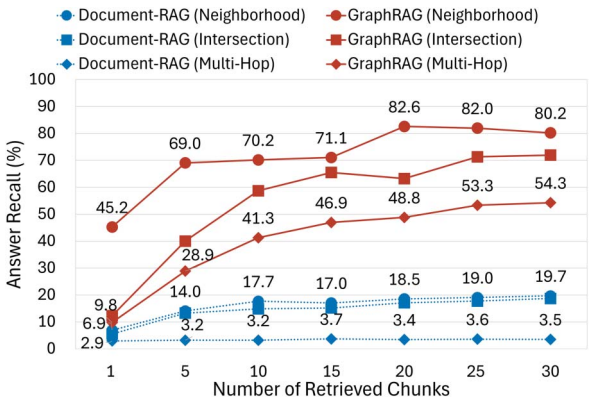



Figure 4: Answer Recall on CDR-QA across different retrieval sizes.

As shown in Figure 4, GraphRAG shows clear gains as $k$ increases, with Answer Recall peaking at $k = 20$, with a slight decline to $k = 30$. Document-RAG, in contrast, exhibits only marginal improvements across all question types, with Multi-Hop performance remaining nearly flat. These results highlight the benefit of increasing the retrieval size in GraphRAG, where **structured knowledge allows for effective accumulation of relevant evidence from multiple communities or graph clusters**. Performance generally saturates around $k = 25$, with a minor drop at $k = 30$, possibly due to redundancy or mild information overload. In contrast, the limited gains in Document-RAG suggest that simply retrieving more unstructured text does not substantially improve performance. Raw text chunks may lack cohesion and contextual salience, making them less useful for answering complex queries. This highlights a key strength of GraphRAG: **it encodes relevant knowledge into compact, semantically coherent units that scale well with larger retrieval budgets**.

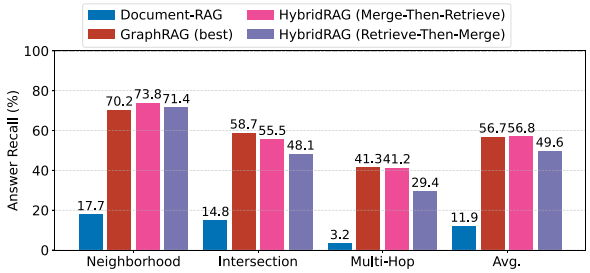



Figure 5: Answer Recall of hybrid settings on CDR-QA (averaged over the three question types).

### 5.7 Complementarity Analysis

While GraphRAG has been shown to outperform Document-RAG in isolation, it remains unclear whether the two offer complementary information. We explore whether combining chunks retrieved from symbolic graphs and raw documents improves QA performance beyond either source alone. We compare four configurations: (1) Document-RAG, retrieving $k$ chunks from unstructured documents; (2) GraphRAG, retrieving $k$ chunks from structured community reports; (3) HybridRAG (Merge-Then-Retrieve), merging all chunks into one pool and retrieving the top $k$ based on Contriever similarity; (4) HybridRAG (Retrieve-Then-Merge), retrieving $k/2$ chunks from each source and concatenating them. We fix $k = 10$ for all settings.

As shown in Figure 5, Document-RAG yields substantially lower Answer Recall across all question types. GraphRAG achieves the highest Answer Recall on Intersection and Multi-Hop questions and performs well on Neighborhood questions. HybridRAG (Merge-Then-Retrieve) matches GraphRAG on average and slightly surpasses it for Neighborhood questions, while HybridRAG (Retrieve-Then-Merge) performs worst overall. These results highlight several important observations. First, **GraphRAG consistently excels in complex reasoning settings**, underscoring the advantage of structured knowledge for integrating dispersed evidence. Second, **HybridRAG (Merge-Then-Retrieve) demonstrates partial complementarity**: By incorporating raw document chunks into the retrieval pool, it can capture highly localized information that is sometimes underrepresented in community summaries, leading to improved performance on Neighborhood questions. However, the overall benefit of merging sources remains marginal, as GraphRAG alone is competitive or superior for most question types. Finally, HybridRAG (Retrieve-Then-Merge) consistently underperforms, likely because splitting the retrieval budget across sources reduces the retriever's ability to prioritize the most relevant evidence.

| Backbone LLM | Ans. Recall | Ev. Recall@10 |
|---|---|---|
| GPT-4o-mini | 16.9 | 37.2 |
| Qwen-2.5-7B-Instruct | 13.8 | 38.4 |
| Llama-3.1-8B-Instruct | 18.9 | 38.4 |
| Llama-3.1-70B-Instruct | 21.1 | 41.3 |
| GraphRAG (best) (w/o LLM) | 56.7 | 70.4 |

Table 7: Comparison of backbone LLMs on CDR-QA (averaged over the three question types).

### 5.8 Sensitivity to Backbone LLMs

In our main experiments, GPT-4o-mini was used for LLM-ICL triple extraction and LLM-based report generation. We conducted additional experiments by replacing these LLM-based modules with open-source models: Qwen-2.5-7B-Instruct (Yang et al., 2024a; Qwen-Team, 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Llama-3.1-70B-Instruct. Table 7 reports QA performance under these settings, where the same LLM was used consistently for both triple extraction and report generation. Neighborhood Aggregation was used for community clustering. Answer Recall was 16.9% with GPT-4o-mini, 13.8% with Qwen-2.5 (7B), 18.9% with Llama-3.1 (8B), and 21.1% with Llama-3.1 (70B). Although proprietary models often outperform open-source ones, this was not always the case here: GPT-4o-mini produced triples of comparable or lower quality (e.g., 53.7 vs. 58.7 F1 for Llama-3.1 70B), reducing its overall performance. Importantly, this drop does not alter our conclusions: even with stronger backbones, LLM-ICL triple extraction and LLM-based reporting remain weaker than non-LLM configurations (GraphRAG (best) in Table 7). Thus, backbone choice shifts absolute scores but preserves the relative ranking of GraphRAG strategies.

## 6 Conclusion

We presented a systematic analysis of module-level design choices in GraphRAG for factoid question answering. By evaluating

multiple strategies for triple extraction, community clustering, and report generation, we identified how each component contributes to overall performance. Our results show that high-quality triple extraction is critical; the granularity of fundamental knowledge units, as determined by community clustering, has a significant impact on downstream performance; in addition, simple template-based reporting outperforms LLM-based summarization in both accuracy and efficiency. Our evaluation focused on biomedical abstracts and encyclopedic articles. Generalization to domains with different levels of triple extraction difficulty, knowledge graph definition, knowledge complexity, and retriever availability remains an open question. Future work will expand evaluation across diverse domains to examine the broader applicability of our findings.

## Acknowledgments

We would like to thank the reviewers and the action editor for their thoughtful and insightful feedback, which greatly helped us improve the paper. This work was supported by JSPS KAKENHI grant numbers 21K17815.

## References

Kriti Bhattarai, Inez Y. Oh, Zachary B. Abrams, and Albert M. Lai. 2024. Document-level clinical entity and relation extraction via knowledge base-guided generation. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 318–327, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.bionlp-1.24

Christian Bizer, Jens Lehmann, Georgi Kobilarov, S. Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - A crystallization point for the web of data. *Journal Web Semantics*, 7:154–165. https://doi.org/10.1016/j.websem.2009.07.002

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Red Hook, NY, USA. Curran Associates Inc.

Rong-Ching Chang and Jiawei Zhang. 2024. Communitykg-rag: Leveraging community structures in knowledge graphs for advanced retrieval-augmented generation in fact-checking.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024a. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024b. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.emnlp-main.845

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pages 719–729, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3626772.3657834

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases.

*Transactions of the Association for Computational Linguistics*, 10:257–273. https://doi.org/10.1162/tacl_a_00459

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

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.99

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Pierre-Yves Genest, Pierre-Edouard Portier, Elöd Egyed-Zsigmond, and Martino Lovisetto. 2023. Linked-docred - enhancing docred with entity-linking to evaluate end-to-end document-level information extraction pipelines. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pages 3064–3074, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3539618.3591912

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun

Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U., Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A., Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh

Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. 2025a. Rag vs. graphrag: A systematic evaluation and key insights.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. 2025b. Retrieval-augmented generation with graphs (graphrag).

Haoyu Han, Yaochen Xie, Hui Liu, Xianfeng Tang, Sreyashi Nag, William Headden, Yang Li, Chen Luo, Shuiwang Ji, Qi He, and Jiliang Tang. 2025c. Reasoning with graphs: Structuring implicit knowledge to enhance LLMs reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25698–25714, Vienna, Austria. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.findings-acl.1319

Yongfeng Huang, Yanyang Li, Yichong Xu, Lin Zhang, Ruyi Gan, Jiaxing Zhang, and Liwei Wang. 2023. MVP-tuning: Multi-view knowledge retrieval with prompt tuning for commonsense reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13417–13432, Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.750

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. https://doi.org/10.48550/ARXIV.2112.09118

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

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 163–184, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-acl.11

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. Realtime qa: What's the answer right now?In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc. https://doi.org/10.52202/075280-2130

Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu, and Minlie Huang. 2021. JointGT: Graph-text joint representation learning for text generation from knowledge graphs. In *Findings of*

*the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2526–2538, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.223

Yunsoo Kim, Yusuf Abdulle, and Honghan Wu. 2025. BioHopR: A benchmark for multi-hop, multi-answer reasoning in biomedical domain. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12894–12908, Vienna, Austria. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.findings-acl.668

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Alexandria Leto, Cecilia Aguerrebere, Ishwar Bhati, Ted Willke, Mariano Tepper, and Vy Ai Vo. 2024. Toward optimal search and retrieval for rag.

Patrick 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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Red Hook, NY, USA. Curran Associates Inc.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative V CDR task corpus: A resource for chemical disease relation extraction. *Database: Journal of Biology Databases and Curation*, 2016. https://doi.org/10.1093/database/baw068, PubMed: 27161011

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation.

Zhuoyang Li, Liran Deng, Hui Liu, Qiaoqiao Liu, and Junzhao Du. 2024. Unioqa: A unified framework for knowledge graph question answering with large language models.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9). https://doi.org/10.1145/3560815

Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith. 2022. Time waits for no one! Analysis and challenges of temporal misalignment. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5944–5958, Seattle, United States. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.435

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.546

OpenAI. 2024. Gpt-4 technical report.

Yilmazcan Ozyurt, Stefan Feuerriegel, and Ce Zhang. 2024. Document-level in-context few-shot relation extraction via pre-trained language models.

Qwen-Team. 2024. Qwen2.5: A party of foundation models.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple

entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.496

Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine tuning vs. retrieval augmented generation for less popular knowledge. pages 12–22. https://doi.org/10.1145/3673791.3698415

V. Traag, L. Waltman, and Nees Jan van Eck. 2019. From Louvain to Leiden: Guaranteeing well-connected communities. *Scientific Reports*, 9:5233. https://doi.org/10.1038/s41598-019-41695-z, PubMed: 30914743

Somin Wadhwa, Silvio Amir, and Byron Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15566–15589, Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.868, PubMed: 37674787

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan O Arik. 2025a. Astute RAG: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30553–30571, Vienna, Austria. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.acl-long.1476

Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025b. Knowledge graph retrieval-augmented generation for LLM-based recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27152–27168, Vienna, Austria. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.acl-long.1317

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10370–10388, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.558

Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, and Vicente Grau. 2024. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.519

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian McAuley. 2025. Knowledge-aware query expansion with large language models for textual and relational retrieval. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4275–4286, Albuquerque, New Mexico. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.naacl-long.216

Derong Xu, Xinhang Li, Ziheng Zhang, Zhenxi Lin, Zhihong Zhu, Zhi Zheng, Xian Wu, Xiangyu Zhao, Tong Xu, and Enhong Chen. 2025. Harnessing large language models for knowledge graph question answering via adaptive multi-aspect retrieval-augmentation.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin

Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Gui, Ziran 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 Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. 2024b. CRAG - comprehensive RAG benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1259

Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. 2022. Generative knowledge graph construction: A review. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1–17, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.emnlp-main.1

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.577

Shuzhou Yuan and Michael Faerber. 2023. Evaluating generative models for graph-to-text generation. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 1256–1264, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria. https://doi.org/10.26615/978-954-452-092-2_133

Feng Zhao, Hongzhi Zou, and Cheng Yan. 2023. Structure-aware knowledge graph-to-text generation with planning selection and similarity distinction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8693–8703, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.537

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K. Qiu, and Lili Qiu. 2024. Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4). https://doi.org/10.1145/3618295

Zijie Zhong, Hanwen Liu, Xiaoya Cui, Xiaofan Zhang, and Zengchang Qin. 2025. Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5756–5774, Abu Dhabi, UAE. Association for Computational Linguistics.

Wenxuan Zhou, Kevin Huang, Tengyu Ma, and Jing Huang. 2020. Document-level relation extraction with adaptive thresholding and localized context pooling. *CoRR*, abs/2010.11304.

Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. 2025. Knowledge graph-guided

retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8912–8924, Albuquerque, New Mexico. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.naacl-long.449

# A Prompts

Figures 6, 7, 8, 9, and 12 show the prompts used for NER, ED (reranking), DocRE, report generation, and QA, respectively. Figures 10 and 11 show the templates to generate plain-text community reports.

# B Complementary Results

## B.1 LLM-ICL Triple Extraction Analysis

Figures 13, 14, 15, and 16 show the results of the error analysis for the LLM-ICL triple extraction.

## B.2 Community Clustering Hyperparameter Analysis

Table 8 and Figure 17 show the results of the hyperparameter-sensitivity analysis, varying the maximum cluster size for Hierarchical Leiden and the neighborhood size for Neighborhood Aggregation, respectively.

## B.3 LLM-Based Report Generation Analysis

Table 9 and Figure 18 show the results of the analysis of LLM-based report generation variations, including the freeform design and different number of findings per report.

| Max. Cluster Size | Ans. Recall | Community Statistics | |
|---|---|---|---|
| | | # Comm. | Avg. Size |
| 5 | 25.1 | 515 | 7.2 |
| 10 | 24.4 | 330 | 9.6 |
| 15 | 26.0 | 225 | 12.2 |
| 20 | 25.8 | 163 | 15.4 |

Table 8: We investigated the sensitivity of the maximum cluster size parameter (i.e., threshold) in the Hierarchical Leiden strategy. For each setting, we constructed communities and then evaluated QA performance using the Answer Recall (%) on CDR-QA (averaged over three questions). Manually extracted triples and Template-based report generation were used consistently across all settings. The results indicate that the performance of Hierarchical Leiden is not highly sensitive to the choice of the maximum cluster size, as Answer Recall varied only modestly across settings.

| Report Type | Answer Recall | Evidence Recall@10 |
|---|---|---|
| Structured | 33.5 | 58.8 |
| Freeform | 31.3 | 56.5 |

Table 9: To assess the impact of report design, we evaluated a freeform variant in which the LLM was instructed to produce plain text with only ‘‘TITLE’’ and ‘‘REPORT’’ sections, removing explicit structural constraints. These results indicate that structured reports provide clearer retrieval cues by explicitly separating salient information, whereas freeform reports disperse relevant content and introduce variability that reduces retrieval effectiveness. We therefore adopt the structured setting as our default configuration.

```
### Instruction

Given the text of a document, extract all entity mentions and classify them into one
of the *predefined entity types* specified below. Each entity mention must be exact
copies of their corresponding original text spans, without any modifications or
alternations. Provide the output in bullet-point format, with each line containing
the entity mention and its corresponding entity type, separated by a pipe (`|`)
symbol.

Predefined Entity Types:
- {entity_type_1}: {definition_1}
- {entity_type_2}: {definition_2}
...

Output Format:
- Entity Mention 1 | Entity Type of Entity Mention 1
- Entity Mention 2 | Entity Type of Entity Mention 2

### Examples

Example 1:
Text: {example_text_1}
Output:
- {example_mention_1_1} | {example_type_1_1}
- {example_mention_1_2} | {example_type_1_2}

Example 2:
Text: {example_text_2}
Output:
- {example_mention_2_1} | {example_type_2_1}
- {example_mention_2_2} | {example_type_2_2}
...

### Test Case

Now, extract entity mentions along with their predefined types for the following text.

Text: {input_document_text}

Provide the output in the bullet-point format specified above.
```

Figure 6: Prompt used for few-shot named entity recognition (NER) using an LLM. Curly braces {} indicate variable substitution for entity types, examples, and test input.

```
### Instruction

Given the text of a document, a list of entity mentions (extracted through Named
Entity Recognition), and candidate {knowledge_base_name_prompt} concepts, assign the
most appropriate Concept ID from the provided candidates to each entity mention based
on the context of the document. If none of the candidates is appropriate for a
mention, simply assign "NA" for it. Provide the output in bullet-point format, with
each line containing the entity mention and its corresponding concept ID, separated
by a pipe (`|`) symbol.

Output Format:
- Entity Mention 1 | Concept ID of Entity Mention 1
- Entity Mention 2 | Concept ID of Entity Mention 2

### Examples

Example 1:
Text: {example_text_1}
Mention 1: {example_mention_1}
Candidate Concept IDs for Mention 1:
- ID: {example_id_1_1} | Name: {example_name_1_1} | Description: {example_desc_1_1}
- ID: {example_id_1_2} | Name: {example_name_1_2} | Description: {example_desc_1_2}
...
Mention 2: {example_mention_2}
Candidate Concept IDs for Mention 2:
- ID: {example_id_2_1} | Name: {example_name_2_1} | Description: {example_desc_2_1}
- ID: {example_id_2_2} | Name: {example_name_2_2} | Description: {example_desc_2_2}
...
Output:
- {example_mention_1} | {gold_concept_id_1}
- {example_mention_2} | {gold_concept_id_2}
...

Example 2:
...

### Test Case

Now, assign the most appropriate Concept ID to each entity mention in the following
text.

Text: {input_document_text}
Mention 1: {input_mention_1}
Candidate Concept IDs for Mention 1:
- ID: {cand_id_1_1} | Name: {cand_name_1_1} | Description: {cand_desc_1_1}
- ID: {cand_id_1_2} | Name: {cand_name_1_2} | Description: {cand_desc_1_2}
...
Mention 2: {input_mention_2}
Candidate Concept IDs for Mention 2:
- ID: {cand_id_2_1} | Name: {cand_name_2_1} | Description: {cand_desc_2_1}
- ID: {cand_id_2_2} | Name: {cand_name_2_2} | Description: {cand_desc_2_2}
...

Provide the output in the bullet-point format specified above.
```

Figure 7: Prompt used for few-shot entity disambiguation (ED) reranking using an LLM. Curly braces {} indicate variable substitution for candidate concepts, examples, and test input.

```
### Instruction

Given the text of a document and a list of detected {knowledge_base_name_prompt}
entities (associated with their corresponding mentions in the text), identify the
relationships between these entities. Each relationship must be selected from the
*predefined set of relationship labels* provided below. Provide the output in
bullet-point format, with each line containing the subject entity, the relationship
label, and the object entity, separated by a pipe (`|`) symbol.

Predefined Relationship Labels:
- {relation_label_1}: {relation_definition_1}
- {relation_label_2}: {relation_definition_2}
...

Output Format:
- Subject Entity | Relationship Label | Object Entity
- Subject Entity | Relationship Label | Object Entity

### Examples

Example 1:
Text: {example_text_1}
Entities:
- Entity0: {example_entity_0_mentions} ({example_entity_0_type})
- Entity1: {example_entity_1_mentions} ({example_entity_1_type})
...
Output:
- {example_subject_1} | {example_relation_1} | {example_object_1}
- {example_subject_2} | {example_relation_2} | {example_object_2}
...

Example 2:
...

### Test Case

Now, identify the relationships for the following text and entities. Each
relationship must be selected from the *predefined set of relationship labels*
provided above.

Text: {input_document_text}
Entities:
- Entity0: {input_entity_0_mentions} ({input_entity_0_type})
- Entity1: {input_entity_1_mentions} ({input_entity_1_type})
...

Provide the output in the bullet-point format specified above.
```

Figure 8: Prompt used for few-shot document-level relation extraction (DocRE) using an LLM. Curly braces {} indicate variable substitution for relation labels, examples, and test input.

```
### Instruction

Write a comprehensive report of a community, given a list of entities that belong to the
community as well as their relationships. The report will be used to inform
decision-makers about information associated with the community and their potential
impact.

The report should include the following sections:
- TITLE: community's name that represents its key entities - title should be short but
specific. When possible, include representative named entities in the title.
- SUMMARY: An executive summary of the community's overall structure, how its entities
are related to each other, and significant information associated with its entities.
- DETAILED FINDINGS: A list of 1-3 key insights about the community. Each insight should
have a short summary followed by multiple paragraphs of explanatory text. Be
comprehensive.

Return output as a well-formed JSON-formatted string with the following format:
    {{
        "title": <report_title>,
        "summary": <executive_summary>,
        "findings": [
            {{
                "summary":<insight_1_summary>,
                "explanation": <insight_1_explanation>
            }},
            {{
                "summary":<insight_2_summary>,
                "explanation": <insight_2_explanation>
            }}
        ]
    }}

### Real Data

Use the following tables for your answer. Do not make anything up in your answer.

Entities:
- {entity_name_1} | {entity_type_1} | {entity_definition_1}
- {entity_name_2} | {entity_type_2} | {entity_definition_2}
...

Relationships:
- {head_entity_1} | {relation_1} | {tail_entity_1}
- {head_entity_2} | {relation_2} | {tail_entity_2}
...

Sub-Communities' Reports:
[Sub-Community 1]
Title: {subcommunity_title_1}
{subcommunity_text_1}
[Sub-Community 2]
Title: {subcommunity_title_2}
{subcommunity_text_2}
...

Output:
```

Figure 9: Prompt used for LLM-based report generation. Curly braces {} indicate variable substitution.

```
{title}
{summary}
[Finding 1] {finding_summary_1}: {finding_explanation_1}
[Finding 2] {finding_summary_2}: {finding_explanation_2}
...
```

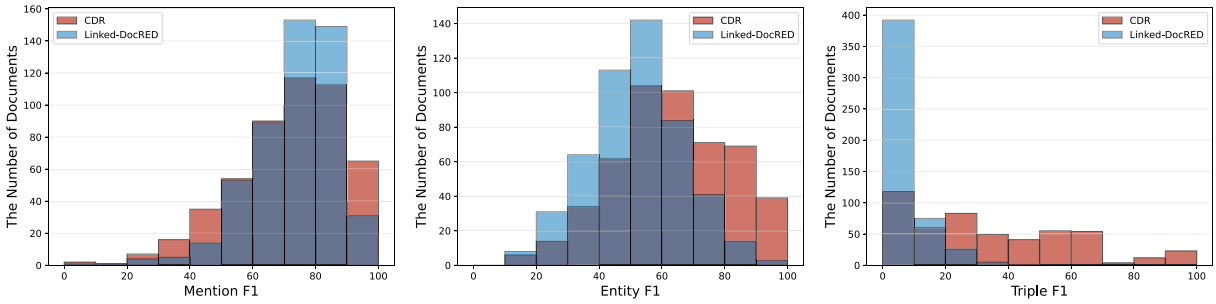
Figure 10: Template used for LLM-based report generation. This template is applied after parsing the LLM's response into a structured object.

```
The primary entities in this community are: {entity_1}, {entity_2}, {entity_3}

This community contains the following entities:
- {entity_name_1} | {entity_type_1} | {entity_definition_1}
- {entity_name_2} | {entity_type_2} | {entity_definition_2}
...

The relationships between the entities are as follows:
- {head_entity_1} | {relation_1} | {tail_entity_1}
- {head_entity_2} | {relation_2} | {tail_entity_2}
...
```

Figure 11: Template used for Template-based report generation. Curly braces {} indicate variable substitution.

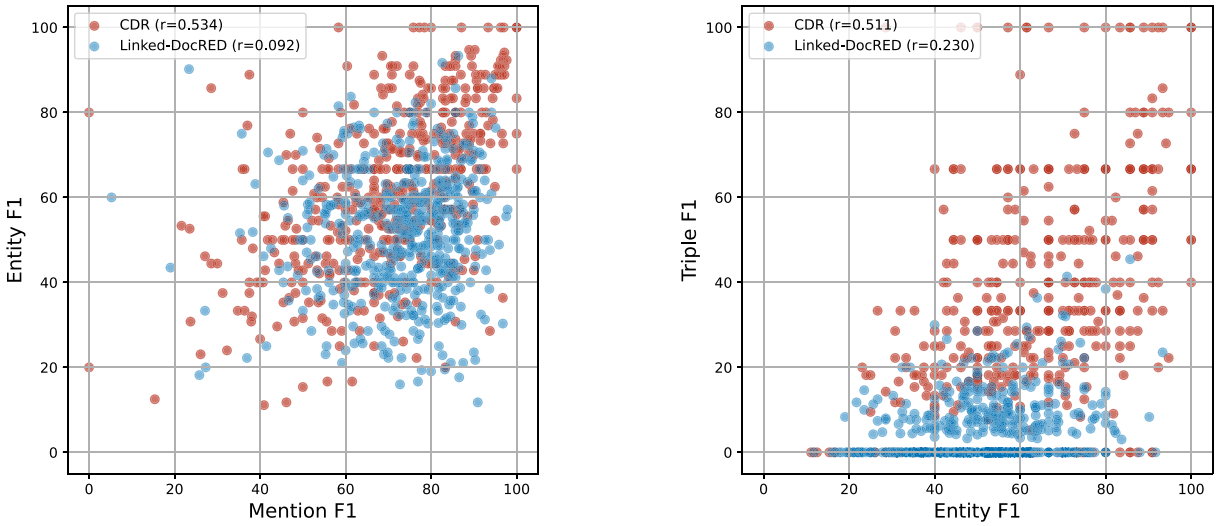
```
### Instruction

Answer a given question based on the provided context. Additionally, provide a score
between 0.0 and 1.0, indicating how well your answer addresses the question. Provide
the output in the following format.

Output Format:
Answer: [Your answer here, a single line without line breaks]
Score: [Score between 0.0 and 1.0]

### Context

[1] {chunk_title_1}: {chunk_text_1}

[2] {chunk_title_2}: {chunk_text_2}

...

[10] {chunk_title_10}: {chunk_text_10}

### Test

Now, based on the context, answer the following question and provide a score.

Question: {input_question}

Provide the output in the format specified above.
```

Figure 12: Prompt used for zero-shot question answering with supporting context. Curly braces {} indicate variable substitution for retrieved chunks, examples, and test input.




(a) Distribution of Mention F1 (b) Distribution of Entity F1 (c) Distribution of Triple F1

Figure 13: Distribution of F1 scores for each stage in LLM-ICL triple extraction. These results suggest that while mention detection is generally reliable, performance sharply degrades by the time the pipeline outputs triples. Importantly, these triple-level errors do not solely stem from the DocRE model (LLM-ICL). Since DocRE operates on the entities provided by earlier stages, errors in NER or ED can propagate and compound, ultimately lowering the final triple F1 scores. Thus, the low triple F1 reflects both intrinsic DocRE errors and accumulated upstream errors.




(a) Mention F1 vs. Entity F1 (b) Entity F1 vs. Triple F1

Figure 14: Scatter plots showing pairwise relationships between stages in LLM-ICL triple extraction. Overall, these results highlight DocRE as a fundamental bottleneck, introducing substantial additional errors beyond those inherited from earlier stages. At the same time, the scatter plots also show evidence of error propagation from upstream components (NER and ED), indicating that low triple-level performance can stem from both intrinsic DocRE failures and accumulated upstream errors.

**PMID:** 12739036

**Text**:
Differential diagnosis of high serum creatine kinase levels in systemic lupus erythematosus . We report the clinical and bioptic findings for a 57 - year - old woman with severe **chloroquine** - induced **myopathy** . Since 1989 , she had been suffering from systemic lupus erythematosus ( SLE ) with renal involvement and undergone periods of treatment with azathioprine and cyclophosphamide . Additional therapy with **chloroquine** ( CQ ) was started because of arthralgia . At the same time , slightly increased creatine kinase ( CK ) levels were noted . Myositis was suspected , and the patient was treated with steroids . The CK increase persisted , however , and she developed progressive **muscular weakness** and **muscular atrophy** . Routine controls revealed markedly elevated CK levels of 1 , 700 U / l . The neurological and electrophysiological findings were not typical of myositis . Thus , muscle biopsy of the deltoid muscle was performed in order to exclude polymyositis or toxic myopathy . As it revealed chloroquine - induced myopathy , medication was stopped . Discriminating between primary SLE - induced affection of the musculoskeletal system and drug - induced side effects is important for appropriate treatment of SLE patients .

**Predicted Entity Mentions (listed in the order they appear in the text, with type and ID):**
creatine kinase(Chemical,D003402); systemic lupus erythematosus(Disease,D008180);
**chloroquine(Chemical,D002738)**; **myopathy(Disease,D009135)**;
systemic lupus erythematosus(Disease,D008180); azathioprine(Chemical,D001379);
cyclophosphamide(Chemical,D003520); **chloroquine(Chemical,D002738)**; arthralgia(Disease,D018771);
creatine kinase(Chemical,D003402); CK(Chemical,D003401); Myositis(Disease,D009220);
steroids(Chemical,D013256); CK(Chemical,D052276); **muscular weakness(Disease,D018908)**;
**muscular atrophy(Disease,D009133)**; CK(Chemical,D003401); myositis(Disease,D009220);
polymyositis(Disease,D017285); toxic myopathy(Disease,D009135); myopathy(Disease,D009135);
chloroquine(Chemical,D002738); myopathy(Disease,D009135)

**Gold Entity Mentions (listed in the order they appear in the text, with type and ID):**
creatine(Chemical,D003401); systemic lupus erythematosus(Disease,D008180);
**chloroquine(Chemical,D002738)**; **myopathy(Disease,D009135)**;
systemic lupus erythematosus(Disease,D008180); SLE(Disease,D008180);
renal involvement(Disease,D007674); azathioprine(Chemical,D001379);
cyclophosphamide(Chemical,D003520); **chloroquine(Chemical,D002738)**; CQ(Chemical,D002738);
arthralgia(Disease,D018771); creatine(Chemical,D003401); Myositis(Disease,D009220);
steroids(Chemical,D013256); **muscular weakness(Disease,D018908)**;
**muscular atrophy(Disease,D009133)**; myositis(Disease,D009220); polymyositis(Disease,D017285);
myopathy(Disease,D009135); chloroquine(Chemical,D002738); myopathy(Disease,D009135);
SLE(Disease,D008180); affection of the musculoskeletal system(Disease,D009140); SLE(Disease,D008180)

**Predicted Triples:**
D003402 - CID - D018908; D003402 - CID - D009133; **D002738** - CID - **D009135**;
D001379 - CID - D009135; D003520 - CID - D009135; D013256 - CID - D009135;
D052276 - CID - D018908; D052276 - CID - D009133

**Gold Triples:**
**D002738** - CID - **D018908**; **D002738** - CID - **D009133**

**Mention F1:** 70.8 / **Entity F1:** 85.7 / **Triple F1:** 0.0

Figure 15: Case study where all entities required for the gold triples were recognized but relation extraction failed (PMID:12793036). The model failed to extract the gold triples `(chloroquine: D002738, CID, muscular weakness: D018908)` and `(chloroquine: D002738, CID, muscular atrophy: D009133)`, resulting in a triple F1 of 0, because the causal link was only implied by temporal and clinical context. Although it correctly predicted `(chloroquine: D002738, CID, myopathy: D009135)`, this higher-level relation was excluded from the gold triples, penalizing the prediction as a false positive.

**PMID:** 19642243

**Text**:
**Acute renal failure** in patients with AIDS on **tenofovir** while receiving prolonged **vancomycin** course for osteomyelitis . **Renal failure** developed after a prolonged course of **vancomycin** therapy in 2 patients who were receiving **tenofovir disoproxil fumarate** as part of an antiretroviral regimen . Tenofovir has been implicated in the development of Fanconi syndrome and renal insufficiency because of its effects on the proximal renal tubule . **Vancomycin** nephrotoxicity is infrequent but may result from coadministration with a nephrotoxic agent . Clinicians should be aware that **tenofovir** may raise the risk of **renal failure** during prolonged administration of **vancomycin** .

**Predicted Entity Mentions (listed in the order they appear in the text, with type and ID):**
**renal failure(Disease,D051437)**; **tenofovir(Chemical,C096918)**; **vancomycin(Chemical,D014640)**; osteomyelitis(Disease,D010019); **Renal failure(Disease,D051437)**; **vancomycin(Chemical,D014640)**; tenofovir(Chemical,C096918); **tenofovir disoproxil fumarate(Chemical,C418563)**; Tenofovir(Chemical,C096918); Fanconi syndrome(Disease,D005198); renal insufficiency(Disease,D051437); **Vancomycin(Chemical,D014640)**; nephrotoxicity(Disease,D007674); **tenofovir(Chemical,C096918)**; **renal failure(Disease,D051437)**; **vancomycin(Chemical,D014640)**

**Gold Entity Mentions (listed in the order they appear in the text, with type and ID):**
**Acute renal failure(Disease,D058186)**; AIDS(Disease,D000163); **tenofovir(Chemical,C096918)**; **vancomycin(Chemical,D014640)**; osteomyelitis(Disease,D010019); **Renal failure(Disease,D051437)**; **vancomycin(Chemical,D014640)**; **tenofovir disoproxil fumarate(Chemical,C418563)**; Tenofovir(Chemical,C096918); Fanconi syndrome(Disease,D005198); renal insufficiency(Disease,D051437); **Vancomycin(Chemical,D014640)**; nephrotoxicity(Disease,D007674); nephrotoxic(Disease,D007674); **tenofovir(Chemical,C096918)**; **renal failure(Disease,D051437)**; **vancomycin(Chemical,D014640)**

**Predicted Triples:**
C096918 - CID - D051437; C096918 - CID - D005198; C096918 - CID - D007674; **D014640** - CID - **D051437**; D014640 - CID - D007674; **C418563** - CID - **D051437**; C418563 - CID - D005198

**Gold Triples:**
**D014640** - CID - **D058186**; **C418563** - CID - **D058186**

**Mention F1:** 84.8 / **Entity F1:** 87.5 / **Triple F1:** 0.0

Figure 16: Case study where triple extraction failed due to upstream entity recognition errors (PMID:19642243). The model failed to extract the gold triples `(vancomycin: D014640, CID, acute renal failure: D058186)` and `(tenofovir disoproxil fumarate: C418563, CID, acute renal failure: D058186)`, yielding a triple F1 of 0. Although the text explicitly mentions ‘‘*Acute renal failure*’’ (`D058186`), it was incorrectly linked to `renal failure` (`D051437`), omitting the critical modifier ‘‘acute.’’ This case shows how even minor upstream entity errors can cascade into complete triple extraction failure despite relatively high mention- and entity-level scores.

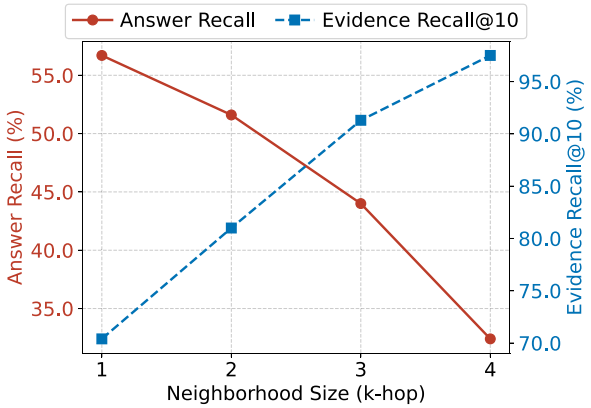



Figure 17: Neighborhood Aggregation forms communities from each node and its neighbors, and we examined its effect on CDR-QA by varying the neighborhood size $k$ from 1 to 4. The opposite trends of Answer Recall and Evidence Recall can be explained by the growing scope of communities as neighborhood size increases. With larger neighborhood sizes, communities include more neighboring nodes. As a result, nodes that are far from the evidence triples but irrelevant to them are also grouped into the same community. This means that a retrieved chunk may be counted as a correct one simply because it comes from a community containing the evidence triples, even if the chunk only describes irrelevant triples. In addition, the growing presence of irrelevant or noisy triples within communities with evidence triples makes it harder for the QA model to combine the scattered evidence and identify the correct answers, leading to the observed decline in Answer Recall.

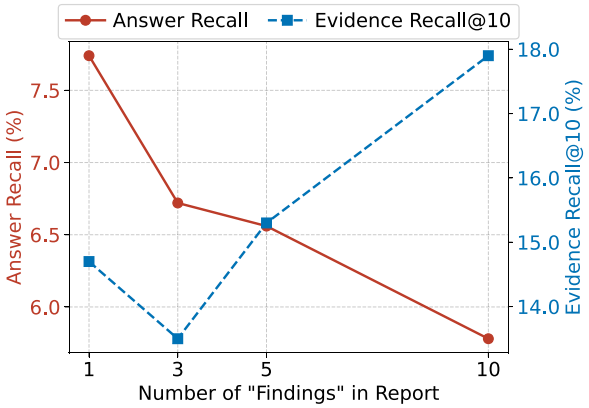



Figure 18: Within the structured design, we further varied the number of findings per report. Adding more findings increases the likelihood that relevant evidence is preserved in the report, which directly benefits retrieval and improves Evidence Recall. At the same time, additional findings inevitably introduce less salient or irrelevant details. This effect is particularly harmful for multi-hop questions, which require precise reasoning across multiple pieces of evidence. In such cases, extraneous content acts as noise, making it harder for the QA model to identify and combine the correct evidence even when it is present in the provided context.