Reducing Hallucinations in Language Model-based SPARQL Query Generation Using Post-Generation Memory Retrieval

Aditya Sharma[1,2], Christopher J. Pal[1,2,3], Amal Zouaq[1,2]

[1]Polytechnique Montréal, [2]Mila - Quebec AI Institute, [3]Canada CIFAR AI Chair

Correspondence: aditya.sharma@mila.quebec

Abstract

The ability to generate SPARQL queries from natural language questions is crucial for ensuring efficient and accurate retrieval of structured data from knowledge graphs (KG). While large language models (LLMs) have been widely adopted for SPARQL query generation, they are often susceptible to hallucinations and out-of-distribution errors when generating KG elements, such as Uniform Resource Identifiers (URIs), based on opaque internal parametric knowledge. We propose PGMR (Post-Generation Memory Retrieval), a modular framework where the LLM produces an intermediate query using natural language placeholders for URIs, and a non-parametric memory module is subsequently employed to retrieve and resolve the correct KG URIs. PGMR significantly enhances query correctness (SQM) across various LLMs, datasets, and distribution shifts, while achieving the near-complete suppression of URI hallucinations. Critically, we demonstrate PGMR's superior safety and robustness: a retrieval confidence threshold enables PGMR to effectively refuse to answer queries that lack support, and the retriever proves highly resilient to memory noise, maintaining strong performance even when the non-parametric memory size is scaled up to 9 times with irrelevant, distracting entities.

1 Introduction

Question answering (QA) is a fundamental task in natural language processing research. Many QA systems leverage knowledge graphs (KGs) (Saxena et al., 2020; Lan et al., 2021; Peng et al., 2023), which represent world knowledge as multi-relational graphs of factual triples *(subject, relation, object)*, for example, *(The Truman Show, nominated for, Academy Award for Best Supporting Actor)*. Answering questions over KGs (KGQA) can typically involve two steps: first, translating a natural language question into its corresponding SPARQL[1] query (discussed in Section 3.2), and second, executing that query against the KG to retrieve an answer (Banerjee et al., 2022).

Large language models (LLMs) have shown promise for this task (Meyer et al., 2024; Diallo et al., 2024; Banerjee et al., 2022; Qi et al., 2024; Brei et al., 2024). However, the process of authoring SPARQL queries remains demanding for non-experts and is non-trivial even for modern LLMs, as it mandates an intimate familiarity with the underlying KG schema and the obscure structure of its Uniform Resource Identifiers (URIs - described in Section 3.1). Specifically, the use of non-semantic identifiers within platforms such as Wikidata (e.g., a prefix followed by arbitrary digits like Q937 for Albert Einstein) forces LLMs to rely solely on their parametric memory for accurate identifier mapping, impeding robust query generation. Furthermore, LLMs often hallucinate and produce plausible-looking but incorrect identifiers, such as URIs that don't exist in the KG (Huang et al., 2023; Ji et al., 2023; Huang et al., 2021). This limitation raises concerns about the reliability of LLMs in real-world information retrieval (IR) applications and has driven extensive research into techniques for detecting and mitigating such errors (Lin et al., 2024; Varshney et al., 2023; Dhuliawala et al., 2023; Chern et al., 2023; Li et al., 2023).

Retrieval-augmented generation (RAG) (Khandelwal et al., 2019; Lewis et al., 2020a; Guu et al., 2020; Lewis et al., 2020b) is a popular paradigm that is often employed for reducing hallucinations (Huang et al., 2023). It uses a non-parametric memory for retrieving information in response to a question. This information is provided in the prompt for the LLM before generation to enhance the grounding of responses. However, since the LLM is not strictly constrained to rely on the retrieved context, it may still generate responses that are not fully

---

[1]https://www.w3.org/TR/rdf-sparql-query/

grounded, thereby failing to eliminate hallucinations (Barnett et al., 2024).

In response to these limitations, we propose PGMR (Post-Generation Memory Retrieval) [2], a new modular architecture featuring non-parametric memory modules with a retriever for managing KG elements. This approach enables LLMs to focus on generating SPARQL query syntax while the retriever ensures accurate retrieval of relevant KG elements. PGMR enables the LLM to generate intermediate queries with a structured query representation that preserves SPARQL syntax while replacing KG URIs with natural language placeholders. Each placeholder is paired with the corresponding URI label and description in a mapping block. This design enables LLMs to focus on generating syntactically correct query structures, while deferring the grounding of placeholders to precise URIs to a post-generation retrieval step.

The core conceptual innovation of PGMR lies in its temporal repositioning of the retrieval phase. While RAG relies on pre-generation retrieval from natural language questions that often lack semantic alignment with URI metadata in the memory, PGMR performs retrieval after generation and leverages the LLM to generate explicit labels and descriptions after the query structure is formed. By producing these placeholders in a format that directly mirrors the non-parametric memory, PGMR achieves significantly higher latent similarity compared to RAG. This architecture effectively decouples structural synthesis from identifier grounding, ensuring that all generated URIs are strictly aligned with existing knowledge graph elements to nearly eliminate hallucinations.

We address the following research questions:

1. Does separating query structural prediction from identifier grounding reduce URI hallucinations while improving query correctness?

2. How effectively does a retrieval confidence threshold enable accurate query refusal under incomplete knowledge conditions?

3. How robust is PGMR's performance when subjected to substantial memory scale-up and irrelevant data injection?

This study introduces several key contributions:

1. We propose PGMR, a modular architecture that enhances LLM-based SPARQL generation by explicitly separating query syntax (generated by the LLM) from URI resolution (handled by a non-parametric retriever).

2. Our results show that PGMR drastically reduces URI hallucinations while significantly improving performance across LLMs and datasets, with most cases showing an almost complete elimination of hallucinated URIs.

3. We demonstrate that a retrieval confidence threshold enables PGMR to correctly refuse to answer queries based on incomplete knowledge, thereby ensuring safety and robustness.

4. PGMR exhibits minimal performance degradation (an approximate 4% SQM drop) when the memory is subjected to a ninefold increase in memory size and irrelevant data.

2   Related Work

SPARQL Query Generation using LLMs. Even though various models have been proposed for the SPARQL query generation task recently, they suffer from one key problem: they require tagged question data (Banerjee et al., 2022; Reyd and Zouaq, 2023; Diallo et al., 2024; Qi et al., 2024). In tagged data, the input to the model contains the natural language question, along with the URIs and their labels, necessary to formulate the corresponding SPARQL query. This requirement presents a challenge due to the high costs and labor-intensive nature of tagging, making it unsuitable for practical, real-world use cases. Therefore, our study emphasizes and assesses the untagged versions of the datasets.

Reyd and Zouaq (2023) and Diallo et al. (2024) propose a copy mechanism for copying URIs from the tagged question while generating the SPARQL query using LLMs. Additionally, they investigate the models' performance in dealing with novel question-query structures and unknown URIs. We incorporate this evaluation method to measure the out-of-distribution robustness of our proposed approach. Banerjee et al. (2022) propose a pointer generator network-based approach for SPARQL query generation over tagged data, utilizing BERT (Kenton and Toutanova, 2019) and a finetuned T5. Their approach generates multiple SPARQL queries and selects the first query that executes and fetches an answer from the knowledge base as the

predicted SPARQL query. This evaluation strategy contrasts with our approach of generating only one SPARQL query per question, which is more efficient. Qi et al. (2024) employ a pre-training method that integrates the proposed Triple Structure Correction (TSC) strategy. This technique, which requires tagged data, involves randomly interchanging the positions of the subject, predicate, and object in SPARQL query triples with a certain probability, alongside a Masked Language Modeling (MLM) objective in a multi-task learning setup, before proceeding with fine-tuning on the downstream tagged dataset. Zahera et al. (2024) explore chain-of-thought prompting (Wei et al., 2022) for SPARQL query generation. RAG for SPARQL query generation has also been explored in recent works (Emonet et al., 2024). Our comparative baseline adopts a comparable RAG approach.

Mitigating hallucinations in LLMs. Despite their widespread adoption, LLMs are vulnerable to generating factually incorrect information through hallucinations, which affects their reliability in real-world applications (Huang et al., 2023; Ji et al., 2023; Huang et al., 2021). This challenge has led to significant research aimed at hallucination detection and mitigation (Lin et al., 2024; Varshney et al., 2023; Dhuliawala et al., 2023; Chern et al., 2023; Li et al., 2023). Varshney et al. (2023) identify potential hallucination candidates using the LLM's logit outputs, validating their correctness, addressing any detected hallucinations, and then proceeding with the generation process. In contrast, PGMR lets the LLM hallucinate between special tags and corrects these hallucinations post-generation using memory retrieval. The Chain-of-Verification (CoVe) (Dhuliawala et al., 2023) method involves the model first producing an initial draft, then formulating verification questions to validate it by independently answering these questions, and generating the final, verified output. Our method instead relies on a non-parametric memory to verify and include factually correct information from a KG.

External memory augmented LLMs. Earlier studies have investigated methods to augment LLMs by retrieving documents from external memory and integrating them into the contextual framework of tasks to provide relevant information. According to Modarressi et al. (2023) and Modarressi et al. (2024), the lack of a specialized memory unit in current LLMs constrains their ability to store and retrieve knowledge relevant to tasks explicitly. This observation supports the central premise of our study. They suggest enhancing LLMs with an API-based memory unit for read-write operations to address this limitation. Interactions with external memory can be implemented using natural language interfaces (Park et al., 2023; Zhou et al., 2023) or formal mechanisms, such as standardized APIs, that allow the model to parse and execute commands (Schick et al., 2024; Hu et al., 2023; Modarressi et al., 2023, 2024). Our approach is akin to the use of a formal mechanism for memory retrieval through our intermediate query format (as described in Section 4.1).

3 Preliminaries

3.1 Uniform Resource Identifiers (URIs)

In a large KG like Wikidata, each entity, relation, and property is assigned a Uniform Resource Identifier (URI), which is associated with a corresponding natural language URI label and other relevant metadata. For example, for an entity with the URI label "The Truman Show," the associated Wikidata URI is "Q214801." There are two types of URIs in the Wikidata KG, namely, "Q-ids," which correspond to the entities (e.g., "Q76" refers to the entity "Barack Obama"), and "P-ids," which correspond to relations, properties, and classes in the KG (e.g., "P26" refers to the relation "spouse").

3.2 SPARQL Queries

SPARQL is the standard declarative query language for Resource Description Framework (RDF) based KGs, such as Wikidata. It enables data retrieval by matching triple patterns corresponding to subject-predicate-object relationships. For example, a natural language question like *"For which film was Sergei Eisenstein the film editor?"* can be translated into the following SPARQL query - "*select distinct ?sbj where { ?sbj wdt:p1040 wd:q8003 . ?sbj wdt:p31 wd:q11424 }*," where p1040, q8003, p31, and q11424 are URIs representing entities, relations, and properties from a KG like Wikidata[3]. However, authoring SPARQL queries is challenging for both non-experts and LLMs, as it requires precise knowledge of the KG's schema and opaque URIs. Generating Wikidata's non-semantic URIs (e.g., a letter followed by arbitrary digits) forces an LLM to rely solely on its parametric memory to recall identifier mappings.

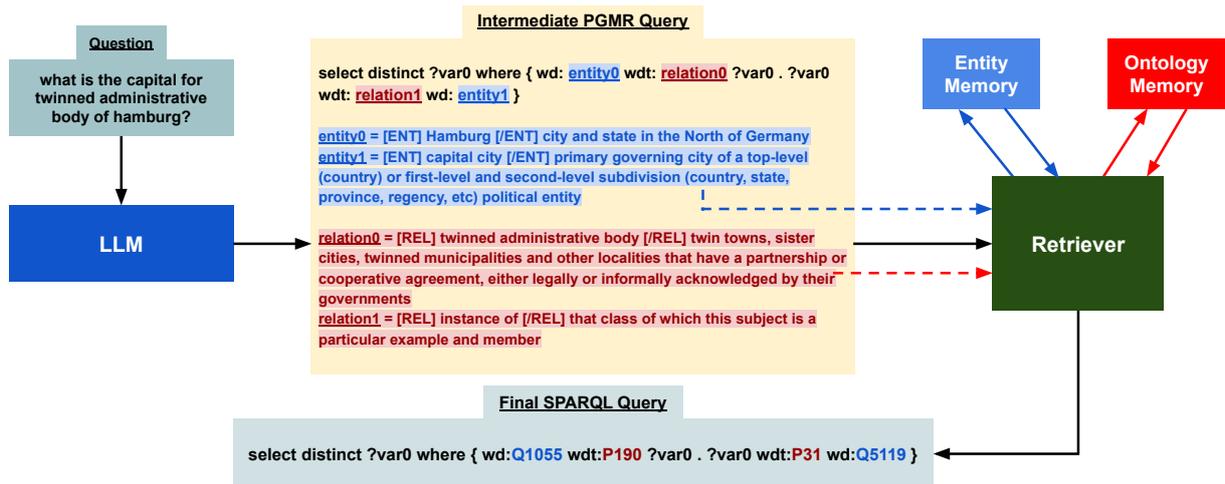
[3] https://www.wikidata.org

Figure 1: As discussed in Section 4, PGMR first employs an LLM to produce an intermediate SPARQL-PGMR query, where all entities and relations are replaced by placeholders which are mapped to natural language labels and definitions, rather than formal URIs (Section 4.1). The retriever (Section 4.2) subsequently fetches and replaces these labels with the most similar URIs from the relevant non-parametric memory to create the final SPARQL query.

4 Post-Generation Memory Retrieval (PGMR)

In this section, we introduce PGMR ("Post-Generation Memory Retrieval"), a novel method for SPARQL query generation utilizing LLMs. In PGMR, we initially convert SPARQL queries into an intermediate SPARQL-PGMR query format. This intermediate query representation preserves the syntax of SPARQL while replacing URIs with placeholders that explicitly encode entities and relations in natural language. Specifically, entities are denoted as entityX = [ENT] Label [/ENT] Description, where Label provides the canonical surface form of the entity and Description gives a concise disambiguating context. Relations follow a parallel scheme, expressed as relationY = [REL] Label [/REL] Description, where the description clarifies the semantics of the property and distinguishes it from closely related relations. These intermediate queries are later grounded in KG URIs using a retriever (discussed in Section 4.2) to produce the final SPARQL query.

As seen in Figure 1, a SPARQL-PGMR intermediate query thus consists of two components: (1) a SPARQL query block that maintains the original query structure but replaces URIs with entity and relation placeholders, and (2) a mapping block that defines the corresponding natural language labels and disambiguating descriptions for each placeholder. This design ensures syntactic fidelity to SPARQL while maintaining human readability, thereby enabling LLMs to generate valid queries

that can be reliably grounded to knowledge graph URIs in a post-processing step. This representation achieves two goals: (i) it constrains generation to a valid SPARQL structure, ensuring compatibility with downstream execution, and (ii) it provides natural language grounding for entities and relations, enabling post-generation alignment with the KG using a non-parametric memory of URIs and a retriever. By decoupling the SPARQL structural generation from the disambiguation of identifiers, SPARQL-PGMR supports accurate query generation and facilitates transparent inspection of model outputs. Unlike LLM-based direct SPARQL query generation, which may produce URIs not present in the KG, PGMR's approach of conducting memory retrieval post-LLM generation greatly mitigates the risk of hallucinating non-existent KG URIs, as discussed in Section 6.

4.1 Training Data Transformation

To train models capable of generating SPARQL-PGMR queries, we design a systematic transformation process that converts standard question-SPARQL query pairs into the PGMR representation. The transformation consists of three stages: entity replacement, relation replacement, and mapping construction. In the first stage, all entity URIs (e.g., Q937) appearing in the query are replaced with symbolic placeholders of the form entityX. A mapping entry is then created for each placeholder, expressed as entityX = [ENT] Label [/ENT] Description, where the label corresponds to the canonical surface form and the description provides a

Model	Type	F1 ↑	SQM ↑	BLEU ↑	Qid EM ↑	Pid EM ↑	URI Hallucination ↓
T5-Small	FT	44.84	37.82 %	81.54	42.93 %	89.69 %	30.02 %
T5-Small RAG	FT	67.30	67.03 %	89.96	73.68 %	95.48 %	14.70 %
T5-Small PGMR (ours)	FT	**86.75**	**86.37 %**	94.63	93.17 %	97.05 %	**0.0 %**
Llama 3.1 8B	PEFT	19.89	0.75 %	66.74	1.52 %	61.23 %	89.84 %
Llama 3.1 8B RAG	PEFT	81.66	81.37 %	94.65	90.19 %	91.83 %	7.23 %
Llama 3.1 8B PGMR (ours)	PEFT	**88.73**	**87.84 %**	95.88	94.26 %	93.17 %	**0.0 %**
GPT 4o	25-shot	17.27	2.82 %	56.65	7.04 %	50.11 %	81.02 %
GPT 4o RAG	25-shot	58.85	27.21 %	69.79	82.27 %	65.86 %	4.88 %
GPT 4o PGMR (ours)	25-shot	**62.23**	**51.24 %**	77.49	79.12 %	68.10 %	**0.01 %**

Table 1: **Results on LCQUAD 2.0 (original split):** Across different LLMs, PGMR demonstrates a substantial reduction in hallucinations while achieving significantly higher SQM and answer-level F1 scores than direct SPARQL generation and RAG, as discussed in Section 6.

short disambiguating definition extracted from the KG (e.g., [ENT] Albert Einstein [/ENT] German-born theoretical physicist, developer of the theory of relativity, Nobel Prize laureate (1921)). The second stage applies the same procedure to relation and property URIs (e.g., P35), replacing them with placeholders such as relationY and constructing mappings of the form relationY = [REL] Label [/REL] Description. The description obtained from the KG provides the necessary semantic information for relation disambiguation (e.g., [REL] head of state [/REL] official with the highest formal authority in a country/state). Finally, a mapping block is appended to each transformed query, listing all entity and relation definitions. In this study, we obtain labels and descriptions from Wikidata using its official Python library[4].

4.2 Memory Retriever

Memories: We utilize two memory modules (one for entities and another for relations) designed as a dictionary that pairs LLM encoder-generated embeddings of URI labels and their descriptions as keys with the corresponding URIs as values. The entity memory module includes such embeddings and URIs for all entity URIs in the dataset. Similarly, the ontology memory module includes these for all the relation URIs in the dataset.

Retriever: Given a natural language input such as "[ENT] Albert Einstein [/ENT] German-born theoretical physicist, developer of the theory of relativity, Nobel Prize laureate (1921)," our memory retriever locates the URI whose label and description are most similar to this text within the relevant

memory. The label is first parsed and matched with the URI labels in the memory. If there is no direct match or multiple matches for the label, then the input text with the label and description is encoded into an embedding by an LLM encoder. The retriever then uses FAISS (Douze et al., 2024; Johnson et al., 2019) to retrieve the URI from the memory whose embedding is nearest to the input text embedding. We use BGE (Chen et al., 2024) as the encoder for our retriever in this study.

4.3 Setup

Training Setup: Using the transformed training data from Section 4.1, we finetune the LLM to generate intermediate queries given the questions. **Few-shot Setup:** Drawing on the transformed training data, we apply few-shot prompting to the LLM, enabling it to generate intermediate queries. **Inference:** During the inference phase, we generate a SPARQL-PGMR query for each question using the LLM. Each URI placeholder is then replaced by its corresponding URI through the retriever, thereby forming the final SPARQL query.

5 Experimental Setup

5.1 Datasets

We evaluate our work on two widely used datasets for SPARQL query generation: LCQUAD 2.0 (Dubey et al., 2019) and QALD-10 (Usbeck et al., 2023). Even though tagged versions of these datasets exist, where the entities and relations in the question are already linked and tagged with associated URIs from the KG, we use the untagged versions. Tagging is expensive, time-consuming, and impractical in the real world. SPARQL query

---

[4]https://pypi.org/project/Wikidata/

Model	Type	F1 ↑	SQM ↑	BLEU ↑	Qid EM ↑	Pid EM ↑	URI Hallucination ↓
T5-Small	FT	22.67	0.16 %	70.44	2.71	84.75 %	61.66 %
T5-Small RAG	FT	67.25	64.14 %	89.85	71.98 %	94.11 %	16.01 %
T5-Small PGMR (ours)	FT	**82.13**	**81.57 %**	94.24	90.4 %	95.93 %	**0.0 %**
Llama 3.1 8B	PEFT	22.19	0.03 %	67.33	0.53 %	65.40 %	91.04 %
Llama 3.1 8B RAG	PEFT	82.35	81.24 %	94.95	89.67 %	91.10 %	7.71 %
Llama 3.1 8B PGMR (ours)	PEFT	**84.47**	**83.33 %**	94.35	92.65 %	90.80 %	**0.0 %**
GPT 4o	25-shot	17.27	1.45 %	55.39	4.07 %	48.92 %	85.87 %
GPT 4o RAG	25-shot	53.79	23.65 %	64.28	83.46 %	63.25 %	3.47 %
GPT 4o PGMR (ours)	25-shot	**69.05**	**58.42 %**	82.77	86.14 %	73.10 %	**0.0 %**

Table 2: **Results on LCQUAD 2.0 (unknown URI split):** Even in an out-of-distribution setting, PGMR significantly mitigates hallucinations while enhancing SQM and answer-level F1 performance over the baselines across various LLMs and settings (see Section 6).

generation becomes a much harder problem when using untagged data (see Section 6).

LCQUAD 2.0 comprises a mix of simple and complex questions with associated SPARQL queries over Wikidata, crafted by human annotators from Amazon Mechanical Turk. The **original split** of LCQUAD 2.0 includes a broad dataset with 21k questions for training, 3k questions for validation, and 6k questions for testing. To specifically test how our models perform in the face of out-of-distribution URIs not seen before in training, we also construct an **"unknown URI" split** of LCQUAD 2.0 in line with Reyd and Zouaq (2023). Every query in the test set of this version of the dataset includes at least one URI not seen during training, thus making accurate query generation a much more complex problem. The "unknown URI" split contains 24k train, 3k validation, and 3k test questions and associated queries.

QALD-10 is a benchmarking dataset that is a part of the Question Answering over Linked Data (QALD) challenge series for KGQA using SPARQL query generation. Owing to the complexity of SPARQL queries that involve various clauses and literals, QALD-10 is recognized as one of the most challenging and practically applicable datasets in the QALD challenge series. Although each question in QALD-10 is translated into eight different languages, we only consider the English versions of the questions for this analysis. The dataset consists of 412 training questions and 394 test questions, which is much smaller than LCQUAD 2.0 but contains more complex queries. Given the limited availability of training data for QALD-10, we employ a two-step approach in our finetuning experiments: pre-training the LLM on LCQUAD 2.0 before finetuning it on QALD-10.

5.2 Metrics

To measure the effectiveness of our models, we utilize the following three metrics:

F1 Score: We execute the gold and the generated SPARQL queries and then calculate the answer-level F1 score based on the responses.

Semantic Query Match (SQM): We introduce the SQM metric, which assesses semantic equivalence rather than literal string match. Recognizing that LLMs may produce valid queries with different variable names than the reference (e.g., ?value vs. ?uri), we normalize all variables to a canonical representation (e.g., ?var0, ?var1), a technique consistent with the evaluation framework of Qi et al. (2024). Additionally, since the ordering of triples within a WHERE clause and of entities within a triple does not alter the query's logical semantics (queries would result in the same outcome when executed), our comparison is insensitive to their permutation. Once the variables are normalized and the WHERE clause is compared in a permutation-invariant manner, the remainder of the query is subjected to an exact-match comparison.

BLEU score (Papineni et al., 2002) is a popular neural machine translation metric that assesses the predicted query against the gold standard query by comparing tokens.

Qid-EM and Pid-EM: To assess the Qid exact match (EM) score, we directly evaluate whether the set of all entity URIs in the predicted query matches exactly with those in the reference query. Pid EM does the same for relations and properties.

Model	Type	F1 ↑	SQM ↑	BLEU ↑	Qid EM ↑	Pid EM ↑	URI Hallucination ↓
T5-Small	FT	11.30	5.07 %	34.37	11.42 %	25.38 %	66.50 %
T5-Small RAG	FT	**23.15**	15.98 %	43.47	45.68 %	27.66 %	4.82 %
T5-Small PGMR (ours)	FT	22.12	**16.75 %**	44.31	42.39 %	27.41 %	**0.0 %**
Llama 3.1 8B	PEFT	3.93	0.25 %	24.35	1.27 %	22.33 %	89.09 %
Llama 3.1 8B RAG	PEFT	35.46	23.35 %	45.91	60.15 %	36.55 %	8.12 %
Llama 3.1 8B PGMR (ours)	PEFT	**39.41**	**25.63 %**	45.36	53.55 %	37.81 %	**0.0 %**
GPT 4o	25-shot	13.20	3.55 %	41.21	11.93 %	46.19 %	83.76 %
GPT 4o RAG	25-shot	46.52	28.68 %	51.66	74.62 %	53.55 %	11.42 %
GPT 4o PGMR (ours)	25-shot	**51.19**	**29.19 %**	54.95	65.48 %	46.19 %	**0.0 %**

Table 3: **Results on QALD-10:** PGMR consistently minimizes hallucinations across LLMs while attaining significantly better SQM and answer-level F1 scores than direct SPARQL generation, even for the more complex QALD-10 dataset, as discussed in Section 6.

URI Hallucination: We define a new metric, called URI Hallucination, which measures the percentage of queries where at least one URI is hallucinated. A URI is considered hallucinated if it does not exist in the knowledge base memory.

Refusal Accuracy is defined as the success rate with which the model identifies and correctly signals a lack of the requisite internal knowledge, thereby declining to answer the unsupported query (discussed further in Section 7).

5.3 Baselines

Direct Generation refers to the standard scenario where the LLM directly generates the SPARQL query given a question.

RAG: We additionally compare our results against the RAG paradigm, a strong baseline, wherein the question is initially processed by a retriever that retrieves k URIs from the URI memory whose labels and descriptions most closely match the question text in latent space, as described in Section A.1.

PGMR: As described in Section 4, we implement the proposed PGMR, where an LLM first generates intermediate queries, followed by a retrieval step to translate them into SPARQL queries.

5.3.1 Language Models

Our evaluation incorporates a diverse range of LLMs spanning several orders of magnitude in parameter size, including models from the smaller T5 architecture (Raffel et al., 2020) (fine-tuning - FT) up to Llama 3.1 8B Instruct (parameter-efficient fine-tuning - PEFT) (Dubey et al., 2024; Mangrulkar et al., 2022) and the foundational GPT-4o (25-shot prompting) (Achiam et al., 2023).

6 Results

PGMR enhances the quality of SPARQL queries. Across all assessed LLMs, PGMR consistently enhances the accuracy and structure of SPARQL queries on both LCQUAD 2.0 and QALD-10 datasets. As seen in Tables 1, 2, and 3, PGMR improves the SQM across all baselines, LLMs, and datasets. On LCQUAD2 (original split), PGMR yields a performance gain of nearly 61.35 % over direct generation and 16.61 % over the strong RAG baseline on average across LLMs. Notably, this advantage continues on the out-of-distribution "unknown URI" split of LCQUAD2, with PGMR improving 73.89 % over direct generation and 18.09 % over RAG across LLMs, showing its robustness to shifts in data distribution. Even on the more challenging QALD-10 dataset, PGMR outperforms both direct generation and RAG across LLMs.

PGMR nearly eliminates hallucinations. From Tables 1, 2, and 3, we see that direct generation hallucinates URIs 75.42 % of the time on average across datasets and models. While RAG reduces these hallucinations to 8.71 % on average, PGMR achieves a near-complete elimination of the URI hallucination problem regardless of the LLM, dataset, and data distribution used.

Our results confirm that PGMR not only minimizes hallucinations to a great extent but also improves the accuracy and reliability of SPARQL query generation across various LLMs, datasets, and data distributions.
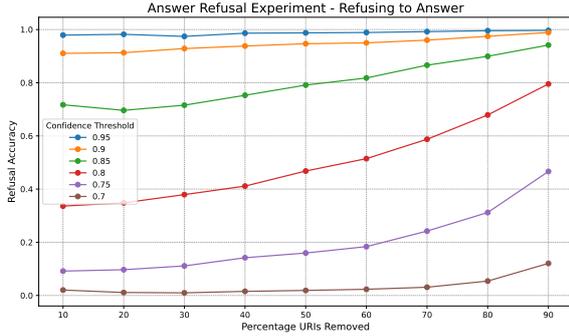
(a)
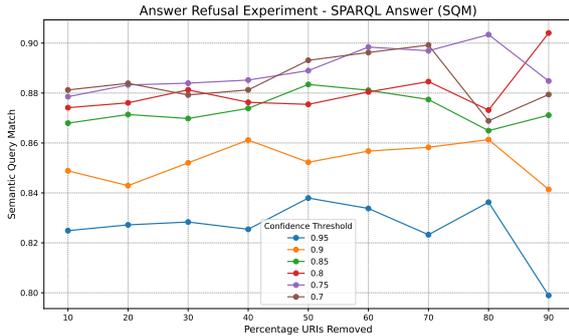
(b)

Figure 2: For LCQUAD2, Figure 2a shows the accuracy of Llama 3.1 8B PGMR on the unanswerable set as a function of the percentage of URIs removed across confidence thresholds, while Figure 2b shows the SQM on the answerable set (described in Section 7).

7   Analysis and discussion

Answer Refusal Experiment. A model's ability to abstain from responding when it lacks the required information is a key determinant of its safety and trustworthiness. This mechanism directly counters model hallucination by signaling uncertainty instead of fabricating content. Such a conservative approach is indispensable in critical applications where a confidently false answer has severe consequences. We assess such robustness by randomly ablating a percentage of entity URIs from the memory, rendering dependent test queries unanswerable. We then divide the set, calculating SQM on the answerable queries and refusal accuracy on the unanswerable queries. The PGMR model architecture is inherently equipped to handle this uncertainty without additional training, thanks to the implementation of a confidence threshold applied to the retrieval similarity score. Specifically, the model executes a refusal action if the similarity score of the top-ranked retrieved URI falls below this predetermined threshold for any URI required in the generated query. Figure 2a demonstrates a clear

positive correlation between the confidence threshold and refusal accuracy for Llama 3.1 8B PGMR. Conversely, as seen in Figure 2b, this increase in threshold leads to a corresponding decline in SQM on the answerable subset, illustrating a direct trade-off in performance. Our analysis indicates that the optimal threshold value for balancing these two metrics is approximately 0.85. Furthermore, this confidence threshold can potentially be learned by adding a small neural network with a sigmoid head to the retriever's encoder, but we leave this exploration for future work.
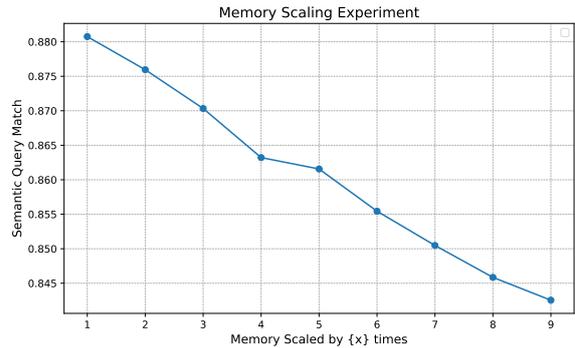

Figure 3: Llama 3.1 8B PGMR maintained robust performance on LCQUAD2, exhibiting only an approximate 4% drop in SQM despite a 9× memory size increase. This demonstrates the retriever's resilience to memory scaling and noise (discussed in Section 7).

Memory Scaling Experiment. We investigate the scaling robustness of the PGMR framework, manipulating the entity memory size from a factor of 2 to 9 relative to the original set. This procedure involves augmenting the memory with irrelevant entities sourced from the Wikidata KG. These entities serve as distractors during the retriever's SPARQL URI resolution. From Figure 3, we see that Llama 3.1 8B PGMR exhibits minimal performance degradation of approximately 4% on LCQUAD2, even at 9 times the memory size, validating the retriever's robustness to expanded, noisy memory.

Inference Latency Analysis. Under equivalent constraints for maximum input and output token lengths, the inference latency of the proposed methods can be characterized by the base generation time (T_{dg}) and the retrieval overhead (T_{ret}). For a model such as Llama 3.1 8B, T_{dg} is approximately 0.9 seconds per response. In contrast, the retrieval component is highly efficient, requiring only 0.0016 seconds (T_{ret}) and a modest 600MB of additional GPU memory. Consequently, while Direct Generation time is simply T_{dg}, RAG latency

is defined as $T_{rag} = T_{dg} + T_{ret}$. The latency for PGMR is expressed as $T_{pgmr} = T_{dg} + (k \cdot T_{ret})$, where k represents the number of URIs identified within the generated intermediate query. Given that $T_{ret} \ll T_{dg}$, the computational overhead introduced by PGMR remains negligible relative to the base model's inference time.

8 Conclusion

We presented PGMR (Post-Generation Memory Retrieval), a modular framework that addresses URI hallucination in LLM-generated SPARQL queries by separating query logic generation from retrieval. LLMs generate intermediate queries with URI placeholders and associated URI natural language mappings, which a non-parametric retriever then resolves into accurate URIs. Despite its simplicity, PGMR consistently achieved superior semantic query match (SQM) and answer-level F1 scores across diverse LLMs and datasets, yielding a near-complete elimination of URI hallucinations. Crucially, the system demonstrates exceptional robustness, both in its native ability to refuse to answer queries based on confidence thresholds and its resilience to memory scaling, maintaining high SQM with only minimal performance degradation even when memory was expanded nine-fold with distracting entities.

Limitations

Even though PGMR exhibits markedly improved performance in generating correct URIs and reducing hallucinations, its effectiveness is tempered by the LLM's ability to construct SPARQL query structures accurately. This is especially evident when looking at the results on the QALD-10 dataset (see Table 3), which features notably complex queries involving diverse clauses and literal expressions compared to LCQUAD 2.0. This limitation may be addressed by using more powerful, large LLMs pre-trained specifically on SPARQL query data, combined with the PGMR generation paradigm, or by combining the advantages of RAG and PGMR, adding a retrieval step both before and after generation. However, we leave that for future work.

Acknowledgements

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. 2022. Modern baselines for sparql semantic parsing. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2260–2265.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199.

Felix Brei, Johannes Frey, and Lars-Peter Meyer. 2024. Leveraging small language models for text2sparql tasks to improve the resilience of ai assistance. *arXiv preprint arXiv:2405.17076*.

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

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

Papa Abdou Karim Karou Diallo, Samuel Reyd, and Amal Zouaq. 2024. A comprehensive evaluation of neural sparql query generation from natural language questions. *IEEE Access*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18*, pages 69–78. Springer.

Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarcisio Mendes de Farias, and Ana Claudia Sima. 2024. Llm-based sparql query generation from natural language over federated knowledge graphs. *arXiv preprint arXiv:2410.06062*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Yichong Huang, Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2021. The factual inconsistency problem in abstractive text summarization: A survey. *arXiv preprint arXiv:2104.14839*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020a. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464.

Zichao Lin, Shuyan Guan, Wending Zhang, Huiyan Zhang, Yugang Li, and Huaping Zhang. 2024. Towards trustworthy llms: a review on debiasing and dehallucinating in large language models. *Artificial Intelligence Review*, 57(9):243.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Lars-Peter Meyer, Johannes Frey, Felix Brei, and Natanael Arndt. 2024. Assessing sparql capabilities of large language models. *arXiv preprint arXiv:2409.05925*.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2023. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2024. Mem-llm: Finetuning llms to use an explicit read-write memory. *arXiv preprint arXiv:2404.11672*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. 2023. Knowledge graphs: Opportunities and challenges. *Artificial Intelligence Review*, 56(11):13071–13102.

Jiexing Qi, Chang Su, Zhixin Guo, Lyuwen Wu, Zanwei Shen, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. 2024. Enhancing sparql query generation for knowledge base question answering systems by learning to correct triplets. *Applied Sciences*, 14(4):1521.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Samuel Reyd and Amal Zouaq. 2023. Assessing the generalization capabilities of neural machine translation models for sparql query generation. In *International Semantic Web Conference*, pages 484–501. Springer.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

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

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, et al. 2023. Qald-10–the 10th challenge on question answering over linked data. *Semantic Web*, (Preprint):1–15.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*.

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

Hamada M Zahera, Manzoor Ali, Mohamed Ahmed Sherif, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2024. Generating sparql from natural language using chain-of-thoughts prompting. In *SEMANTICS*, pages 353–368.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. 2023. Recurrentgpt:

Interactive generation of (arbitrarily) long text. *arXiv preprint arXiv:2305.13304*.

A Appendix A

A.1 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) (Khandelwal et al., 2019; Lewis et al., 2020a; Guu et al., 2020; Lewis et al., 2020b) has become a popular technique for injecting external information into LLMs, helping them generate content grounded in an external knowledge base like a KG. As seen in Figure 5, we propose RAG for SPARQL query generation, where the question first goes to a retriever (discussed in further detail in Section 4.2) which returns k URIs from the memory whose labels are the closest to the question text in latent space, where k is a positive integer. The URIs and their labels are appended to the question to create the prompt for the LLM (see Figure 6). The LLM then uses this information to generate the SPARQL query corresponding to the given question.

B Appendix B

Unlike the evaluation approach utilized by Banerjee et al. (2022) and Qi et al. (2024), which involves generating multiple queries for each question and selecting the first query that successfully executes and returns an answer as the predicted query, our evaluation generates only a single query from our models.

B.1 Hyperparameters

Our T5-small model has 60,506,624 trainable parameters and was finetuned on a single 80GB Nvidia A100 GPU within 14 hours with a batch size of 128. We used a learning rate of 0.0015 for finetuning T5 and trained for 150 epochs on LCQUAD 2.0 and 30 epochs on QALD-10. Due to the limited training data for QALD-10, we finetuned our versions of T5, which were pre-trained on the much larger LCQUAD 2.0 dataset.

For our experiments with Llama 3.1 8B, we finetuned the Llama-3.1-8B-Instruct model, training it for 10 epochs using the AdamW optimizer with a learning rate of 0.0001, no weight decay, and a fixed seed of 42 to ensure reproducibility. Mixed precision training and quantization were enabled to improve efficiency. We employed a parameter-efficient finetuning strategy using LoRA, with a low-rank dimension of 64, a scaling factor (alpha) of 128, and a dropout rate of 0.05. The complete
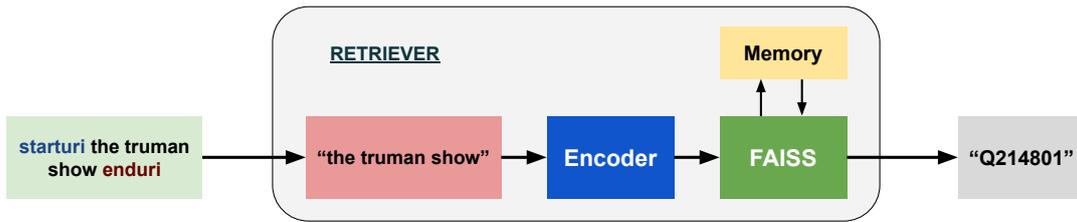
Figure 4: The retriever finds each natural language URI label in the generated intermediate query using the special tokens and retrieves the corresponding URI from memory using similarity search, as discussed in Section 4.2
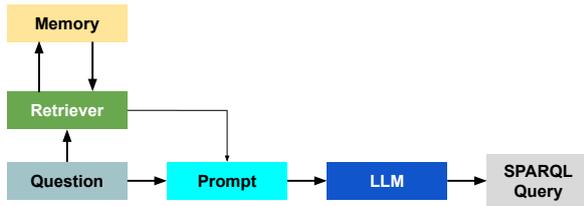
.

Figure 5: RAG-based SPARQL generation workflow. A retriever first fetches k URIs based on latent similarity to the input question, which are then utilized by the LLM to synthesize the final query, as discussed in Section A.1

.

generate the wikidata sparql query to answer the given question using the following information:

p725 = voice actor
p161 = cast member
p5007 = behind the voice actors person id
q4029 = tara strong
q16538 = south park
q181587 = michiru yuimoto
q2405480 = voice actor
q311103 = jeff bennett
[question] what is voice actresses of south park , that has employment is singer?

Figure 6: The prompt for RAG includes k retrieved URIs along with the question as input to the LLM. In our study, we set k to 10 in our RAG baseline.

training process took around 3 hours and 30 minutes. Notably, the model achieved convergence at epoch 5, underscoring its rapid convergence and effectiveness in capturing key aspects of the target data.