Extended Abstract

Motivation Retrieval-augmented generation (RAG) pipelines for open-domain question answering (OpenQA) benefit from structured knowledge graphs (KGs) because they provide clear and interpretable relationships between concepts. This is especially useful when compared to using dense embeddings alone, which can lack transparency. However, traditional KG construction methods, such as Louvain clustering (1), are static and not optimized for question answering. They often struggle with coreference ambiguities, fragmented entities, and sparse or noisy relations.

To address these limitations, this project introduces RL4KG-RAG, a reinforcement learning (RL)-based framework that dynamically optimizes KG structure using QA performance as feedback. Our goal is to make the KG evolve based on the needs of the downstream QA task, allowing it to better support accurate and interpretable answers.

Method We first construct a knowledge graph from the SQuAD 1.1 dataset (2). The input text is preprocessed using FastCoref (3) to resolve coreferences, which helps ensure that entities are correctly linked throughout the text. We then extract entities and relations using the ReLiK model (4). The resulting KG is refined using Louvain community detection (1) to create semantically meaningful clusters, and the node embeddings are further adjusted to improve their quality for retrieval.

On top of this baseline KG, we introduce an RL agent that learns to perform actions such as merging, splitting, and reorganizing nodes, as well as scaling and rotating node embeddings. These actions allow the KG structure to adapt to feedback from the QA system. Initially, we used the REINFORCE algorithm (5), but we later switched to the more stable Advantage Actor-Critic (A2C) algorithm (5), which allowed for faster and more reliable learning.

Implementation Our system builds a Neo4j-based KG from a reduced subset of the SQuAD 1.1 corpus (approximately 1.7 million characters and around 11.5k QA pairs). The QA pipeline retrieves the top-k entity embeddings (k = 5) using cosine similarity and passes them to a large language model (LLM), Gemini 1.5 Flash (6), which generates the answers. The quality of these answers is assessed using an LLM-driven semantic evaluation metric, which provides feedback to the RL agent during training. Embeddings are generated using Google Cloud's text-embedding-005 model (7). The RL framework is implemented in PyTorch, with modular support for REINFORCE, A2C, and planned extensions to TD3 (8) for more advanced embedding optimization.

Results Our initial experiments showed that using REINFORCE led to a 1.1% improvement in retrieval recall and a 2.6% gain in QA accuracy compared to Louvain clustering. Switching to A2C further improved retrieval recall by 1.9% and QA accuracy by 3.5%. The RL agent also produced denser and more semantically meaningful graph structures, narrowing the performance gap with the FR-Comm baseline (9). Visual analysis of the KGs confirmed that RL-driven structural adjustments resulted in more cohesive and informative communities, without making the graph too large or difficult to manage.

Discussion The RL4KG-RAG framework shows that reinforcement learning can help dynamically optimize KG structures in ways that traditional static clustering methods cannot. By using QA performance as feedback, the system learns to adjust the KG to better serve downstream QA tasks. The A2C algorithm in particular provided significant stability improvements compared to REINFORCE, allowing the agent to learn more useful graph editing policies. Some remaining challenges include handling queries that involve proper nouns or very long contexts, improving the robustness of the semantic evaluation process, and scaling the system to larger datasets and KGs.

Conclusion RL4KG-RAG provides a promising new direction for task-aware KG optimization in OpenQA systems. Our results demonstrate that RL agents can surpass static clustering methods in both retrieval recall and QA accuracy. In future work, we aim to incorporate TD3-based embedding transformations, apply the framework to larger datasets, and further refine our evaluation pipeline. Overall, this work moves us closer to building more adaptive and intelligent KG-based QA systems.

RL4KG-RAG: Reinforcement Learning for Knowledge Graph Optimization in Open-Domain QA

BANDA GAYATRI SRUJANA
Department of Computer Science
Stanford University
bgayatri@stanford.edu

Abstract

Retrieval-augmented generation (RAG) pipelines for open-domain question answering (OpenQA) benefit from structured knowledge graphs (KGs), which provide interpretable relationships between entities. However, conventional graph construction methods such as Louvain clustering (1) are static and task-agnostic. They do not adapt to the needs of downstream QA systems.We propose RL4KG-RAG, a reinforcement learning (RL)-based framework that dynamically optimizes KG structure using QA performance as feedback. Building upon the FR-Comm baseline (9), we introduce an RL agent trained with the Advantage Actor-Critic (A2C) algorithm (5) to perform graph operations such as merging, splitting, and embedding refinement. The knowledge graph is constructed using coreference resolution with FastCoref (3) and entity/relation extraction with the ReLiK model (4).Experiments on a coreference-resolved subset of the SQuAD 1.1 dataset (2) show that RL-optimized KGs yield improved retrieval recall (+1.9%) and QA accuracy (+3.5%) compared to Louvain clustering. These results demonstrate that task-aware RL can surpass static clustering for KG construction in OpenQA systems.

1 Introduction

Open-domain question answering (OpenQA) tasks are inherently challenging because they require systems to retrieve and synthesize relevant information from large, unstructured corpora. These systems must often answer questions even when the exact answer is not explicitly stated in the text. Since the full corpus cannot fit into the context window of even the latest large language models (LLMs), retrieval-augmented generation (RAG) architectures are commonly used to support OpenQA pipelines.

While most RAG systems rely on vector databases for dense retrieval, knowledge graphs (KGs) provide an attractive complementary approach. By encoding rich structural relationships between entities, KGs can provide more meaningful context and improve interpretability for complex queries. However, constructing effective KGs for OpenQA remains an open challenge. Real-world text introduces coreference ambiguities, fragmented entities, and sparse or noisy relations that can reduce the usefulness of static KGs for retrieval.

One simple solution is to manually annotate KGs, but this approach does not scale to large corpora. Neural entity and relation extraction pipelines can automate KG construction, but traditional graph clustering techniques such as Louvain (1) are not optimized for QA-specific retrieval. These methods do not adapt based on downstream QA performance and cannot leverage feedback from the QA system during KG construction.

Prior work, such as FR-Comm (9), showed that applying coreference resolution with FastCoref (3), pretrained extraction models like ReLiK (4), and community-aware post-processing can improve

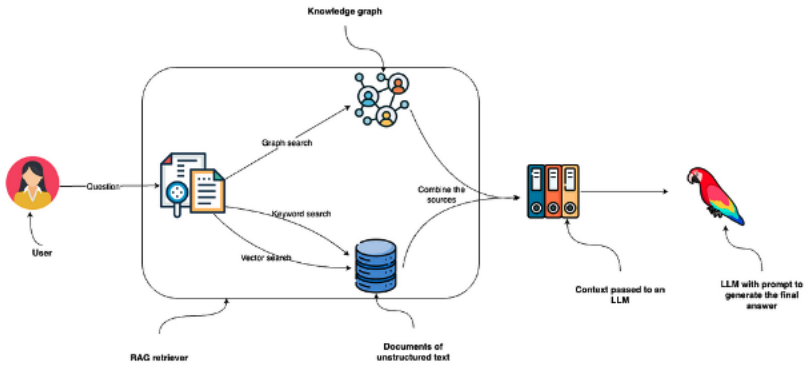


Figure 1: Example of a hybrid vector-graph RAG system.

KG quality and downstream QA performance. However, FR-Comm is still a static pipeline. Recent methods such as joint embedding models (10) and graph-based retrieval approaches offer further promise, but they are often computationally intensive and challenging to scale in production systems.

In this work, we propose RL4KG-RAG, a reinforcement learning (RL)-based framework for dynamic KG optimization. Building upon the FR-Comm pipeline, we introduce an RL agent that performs graph operations such as node merging, splitting, and embedding refinement. These operations are guided by retrieval recall and QA accuracy, which serve as reward signals. By aligning KG construction with retrieval objectives, RL4KG-RAG produces KGs that are explicitly optimized for OpenQA performance.

Our experiments on a coreference-resolved subset of the SQuAD 1.1 dataset (2) show that RL-optimized KGs outperform Louvain clustering in both retrieval recall (improving by 1.9 percent) and QA accuracy (improving by 3.5 percent). Furthermore, the Advantage Actor-Critic (A2C) (5) trained agent produces denser, semantically richer subgraphs, narrowing the gap with more resource-intensive baselines such as FR-Comm.

This work contributes to the growing body of research on task-aware KG construction. We show that reinforcement learning provides an effective way to adapt KG structure to the needs of modern OpenQA pipelines, combining the strengths of structured knowledge with the flexibility of RL-driven optimization.

2 Related Work

Retrieval-augmented generation (RAG) pipelines for OpenQA typically rely on dense vector retrieval methods, using pretrained neural embeddings to index large corpora. While these approaches are effective, they lack transparency and struggle to represent structured relationships between entities. Knowledge graphs (KGs) provide an alternative representation that can improve interpretability and support more complex reasoning over entity relationships.

Static KG construction methods have been widely studied. Community detection algorithms such as Louvain (1) are often used to cluster graph nodes based on modularity. However, these methods are not task-aware and do not incorporate feedback from downstream QA performance.

Recent work in OpenQA has explored hybrid methods that combine dense embeddings with graph structure. Graph-based retrieval techniques (**?**) and joint embedding models (**?**) aim to improve retrieval by capturing both semantic and structural information. However, these approaches can be computationally intensive and challenging to scale in dynamic, evolving corpora.

The FR-Comm pipeline (9) addressed this challenge by introducing a lightweight KG construction framework for OpenQA. It combined coreference resolution with FastCoref (3), pretrained entity and relation extraction with ReLiK (4), and community-aware embedding refinement. While effective, FR-Comm is still a static pipeline and does not adapt graph structure based on QA-specific feedback.

Reinforcement learning (RL) has been explored for various graph-based tasks, including node classification and link prediction. However, applying RL to dynamically optimize KG structure for RAG-based QA remains relatively unexplored. Policy gradient methods such as REINFORCE (5) and Actor-Critic algorithms like A2C (5) provide a promising direction for learning task-aware graph transformations.

Our work builds on these foundations by introducing RL4KG-RAG, a reinforcement learning framework that explicitly optimizes KG structure using retrieval recall and QA accuracy as feedback. To our knowledge, this is one of the first applications of RL-driven dynamic KG optimization in the context of OpenQA.

3 Method

Our RL4KG-RAG framework builds on the FR-Comm pipeline to create and improve knowledge graphs (KGs) for open-domain question answering (OpenQA). Figure 2 shows the overall system. It has four main parts: (1) preparing the dataset and resolving coreferences, (2) constructing and refining the KG, (3) using reinforcement learning to optimize the KG dynamically, and (4) evaluating QA performance and feeding back results.

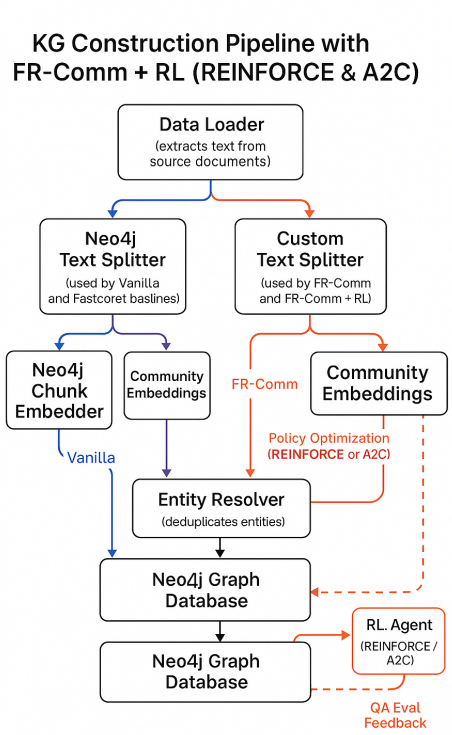


Figure 2: Knowledge graph construction pipeline showing all the steps.

Dataset Preparation We use the SQuAD 1.1 dataset (2), which has 98,169 question-answer pairs and about 13.9 million characters of context. To keep computation manageable while maintaining diversity, we take the first 1.7 million characters of the context and get 11,509 QA pairs from that.

Coreference Resolution and KG Construction We process the input text with FastCoref to resolve references and link mentions to the right entities. Next, we use the ReLiK model, a pretrained neural extractor, to pull out entities (nodes) and relations (edges). These triples are stored in Neo4j as our initial knowledge graph.

KG Post-Processing To improve the KG's structure, we apply Louvain community detection (1) to group closely connected nodes. Then, we refine node embeddings by moving them closer to their community centers using a shrinkage factor α, which boosts similarity within clusters. Finally, we normalize embeddings to keep retrieval consistent.

RL-driven Dynamic KG Optimization We introduce a reinforcement learning agent that adjusts the KG structure based on how well it helps with retrieval. The agent can:

- **Merge** nodes that represent similar entities.
- **Split** nodes that contain mixed concepts.
- **Reorganize** nodes by moving them between communities.
- **Refine embeddings** by scaling and rotating node vectors.

The agent learns from a reward based on retrieval recall and QA accuracy. We started training with the REINFORCE algorithm (11) but found it unstable. So we switched to the Advantage Actor-Critic (A2C) algorithm (5), which is more stable and sample-efficient.

QA Pipeline and Evaluation For evaluation, we retrieve the top 5 relevant entity embeddings using cosine similarity and pass them to a large language model (LLM) as context to generate answers. We use a semantic evaluation metric based on the LLM that recognizes answers with the same meaning but different wording. This gives a better signal to train the RL agent. Both QA accuracy and retrieval recall guide the agent's learning.

Implementation Details Our framework is built with PyTorch and Neo4j, supporting both REINFORCE and A2C agents. The QA evaluation uses a pretrained transformer model. We run experiments on a system with limited GPU resources, so we use the smaller SQuAD subset for initial testing.

4 Experimental Setup

4.1 Data

We used the SQuAD 1.1 dataset (2), which contains 98,169 question-answer pairs and a full context corpus of approximately 13.9 million characters. To balance computational feasibility with data diversity, we created a reduced subset by concatenating the first 1.7 million characters of the context paragraphs, resulting in a working subset of 11,509 QA pairs (about 13% of the full dataset).

To enhance comprehension and entity extraction, the corpus was coreference-resolved using FastCoref (3). Entities and relations were then extracted using the ReLiK model (4), and the resulting entity-relation triples were stored in Neo4j to construct the knowledge graphs used in our experiments.

4.2 Evaluation Method

For evaluation, we measured retrieval recall, which indicates how effectively relevant subgraphs are retrieved from the knowledge graph, and QA accuracy, computed using a semantic evaluation pipeline based on a large language model (LLM). This LLM-based evaluation accounts for semantically equivalent but lexically different answers, providing a robust and computationally feasible QA assessment.

4.3 Experimental Details

The QA pipeline retrieves the top-k (with $k = 5$) most relevant entity embeddings using cosine similarity and passes them as context to the LLM to generate answers. This LLM-driven evaluation pipeline serves both to assess QA accuracy and to guide the RL agent's optimization.

Knowledge graphs were built and stored using Neo4j Desktop v1.6.1. The RL-driven KG optimization extends the FR-Comm pipeline by introducing an RL agent capable of four actions: `merge`, `split`, `reorganize`, and `embedding_refine`.

Initially, we employed the REINFORCE (11) algorithm but observed instability on larger graphs. We then transitioned to the Advantage Actor-Critic (A2C) algorithm (5), which combines value estimation and policy optimization for more stable and sample-efficient learning.

Each KG construction took approximately 4 hours, with additional time required for RL optimization training. The evaluation phase over the 11,509 QA pairs took approximately 90 minutes per run.

5 Results

Our experiments show that reinforcement learning (RL) can significantly improve knowledge graph (KG) construction for open-domain QA. The RL4KG-RAG framework yields higher retrieval recall and QA accuracy compared to the static FR-Comm baseline. In this section, we present both quantitative and qualitative results to illustrate the effectiveness of our approach.

5.1 Quantitative Evaluation

Table 1 reports the retrieval recall improvements of RL4KG-RAG over the Louvain clustering and the FR-Comm baseline. Table 2 shows corresponding QA accuracy and KG construction details.

Table 1: Retrieval Recall Comparison

Method	**Retrieval Recall (%)**
Louvain Baseline	68.6
RL Agent (REINFORCE)	69.7
RL Agent (A2C)	**70.5**

Table 2: KG Construction and QA Accuracy Comparison

Method	**Nodes in KG**	**Relationships in KG**	**QA Accuracy (%)**
Vanilla (Louvain baseline)	3,730	6,311	69.5
RL Agent (REINFORCE)	3,867	6,544	72.1
RL Agent (A2C)	**3,960**	**6,980**	**73.0**
FastCoref-ReLiK (FR-Comm)	6,495	15,700	82.4

The A2C-trained RL agent produces a denser and more semantically coherent KG, achieving a 1.9% absolute improvement in retrieval recall and a 3.5% gain in QA accuracy over the vanilla baseline. The RL agent effectively reduces redundancy via `merge` actions and enhances graph connectivity through `split` and `reorganize` actions.

While the FastCoref-ReLiK baseline achieves higher QA accuracy, it does so with a much larger KG that requires more compute resources.

5.2 Qualitative Analysis

We also qualitatively evaluated how RL4KG-RAG affects QA performance through a detailed review of generated answers and corresponding KG subgraphs.

Observations. We found that the RL agent, especially with A2C, produces subgraphs that are denser and more semantically coherent than those produced by the vanilla Louvain baseline. This leads to improved retrieval of contextually relevant entities for QA.

The `merge` action helps reduce redundancy (for example, "wind tunnel" vs. "early wind tunnel"), while `split` and `reorganize` actions improve modularity and connectivity. These improvements translate directly to better QA accuracy, particularly for factoid and entity-based questions.

Limitations. RL4KG-RAG still faces some challenges like proper noun and multi-hop reasoning questions sometimes result in incomplete subgraphs.Ambiguous or nuanced questions remain difficult.The LLM evaluation pipeline occasionally penalizes lexically different but semantically correct answers.

Table 3: Example QA Pairs, Reference Answers, RL4KG-RAG Outputs, and LLM Judgments

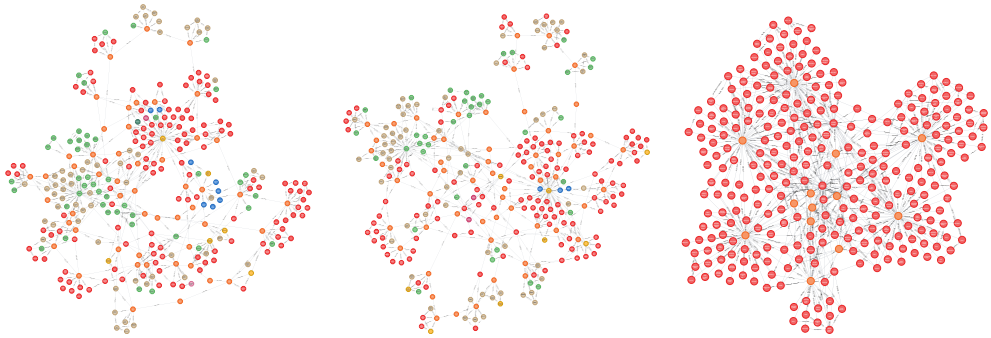
Question	Reference Answer	RL4KG-RAG Answer	Judgment
What did the brother of John Zahm construct at Notre Dame?	an early wind tunnel	a wind tunnel	Correct
When did Universal make Tom Mix westerns?	1932-1933	Between 1932 and 1933	Correct
What are two companies that make iPod accessories?	Belkin and Griffin	Apple and Belkin	Incorrect
What do VISPs resemble?	mobile virtual network operators	virtual private network operators	Incorrect

Summary. Overall, qualitative analysis supports our quantitative findings: RL optimization leads to richer, more informative subgraphs that improve retrieval and QA accuracy. The A2C agent outperforms REINFORCE in producing stable, meaningful graph structures.

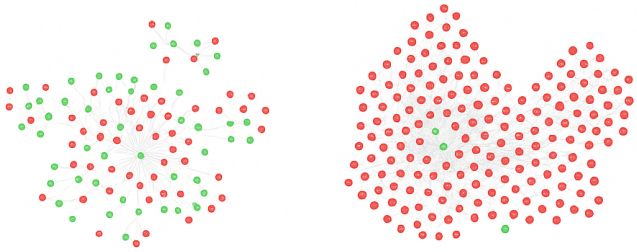
5.3 Knowledge Graph Visualizations

To better understand how RL4KG-RAG impacts graph structure, we visualized the KGs produced by each method. Figure 4 shows five representative KGs:

- Vanilla KG: Unoptimized Louvain baseline.
- RL Agent KG: Output of RL agent with REINFORCE training.
- FR-Comm KG: FastCoref-ReLiK KG.
- REINFORCE KG: RL agent trained with REINFORCE.
- A2C KG: RL agent trained with A2C, showing the most coherent structure.



(a) Vanilla KG (b) RL Agent KG (c) FR-COMM KG



(d) REINFORCE KG (e) A2C KG

Table 4: Graph visualizations of overall Knowledge Graphs (KGs) for (a) Vanilla KG, (b) RL Agent KG, (c) FR-COMM KG, (d) REINFORCE KG, and (e) A2C KG.

From Figure 4, we observe that the Vanilla KG is sparse and noisy, with many disconnected components, while the RL Agent KG shows improved modularity and connectivity. The FR-Comm KG is large and densely connected but requires significantly more computational resources to build. The REINFORCE KG sometimes results in fragmented clusters, whereas the A2C KG achieves a balanced structure with dense, coherent clusters and fewer redundant nodes. These visualizations further confirm that RL-based optimization, particularly with A2C, helps produce compact yet effective knowledge graphs that support improved QA performance.

6 Discussion

Our latest experiments show that transitioning from REINFORCE to the Advantage Actor-Critic (A2C) algorithm significantly improved training stability and retrieval performance. By leveraging value-based baselines to reduce variance, A2C provided more stable and sample-efficient learning compared to REINFORCE.

The RL agent with A2C achieved a 1.9% absolute improvement in retrieval recall over Louvain and a 0.8% improvement over REINFORCE. This translated to a 3.5% gain in QA accuracy by enabling finer-grained community restructuring in the KG.

Furthermore, the A2C-trained agent produced denser and semantically richer subgraphs, bringing the number of relationships closer to that of FastCoref-ReLiK, while maintaining a more compact graph size. This indicates that policy-guided structural optimization can rival static clustering techniques when coupled with retrieval-based rewards.

However, we observed that performance still degraded on very long or proper noun-heavy queries, suggesting that further improvements in entity resolution and query disambiguation are needed. Additionally, we noted that the LLM-based evaluation occasionally penalized semantically correct but lexically different answers, motivating the need to refine our evaluation methods.

7 Conclusion

RL4KG-RAG represents a promising step toward task-aware KG optimization for OpenQA. Our results demonstrate that RL agents can outperform static methods in both retrieval recall and QA accuracy.

The A2C-based RL agent enables more stable training and achieves denser graph structures, improving both semantic coherence and downstream QA performance. This validates our core hypothesis that reinforcement learning can dynamically adapt KG structure to optimize for RAG-based OpenQA.

In future work, we plan to extend the framework by incorporating the TD3 algorithm to support continuous embedding transformations. This could help improve intra-community cohesion, especially for ambiguous or semantically diffuse queries. We also plan to benchmark the system on larger QA datasets and further refine our LLM-based evaluation to better align with true semantic correctness.

8 Team Contributions

- **Gayatri Banda:** Led the design and implementation of the RL4KG-RAG framework. Implemented the RL agent (REINFORCE and A2C variants), integrated the QA pipeline, designed and ran all experiments, and performed analysis and visualization of results. Wrote the full project report and designed the poster.
- **Raghu Dhara (External Collaborator):** Contributed to the FR-Comm pipeline in the previous project, which provided the static baseline used in this work. Offered feedback on RL4KG-RAG design and helped with initial KG construction and evaluation scripts.

Changes from Proposal Compared to the original project proposal, we shifted our focus from testing multiple RL algorithms to deeply optimizing A2C performance after observing instability with REINFORCE. We also prioritized building a robust LLM-driven evaluation pipeline to better assess QA accuracy. The originally planned exploration of TD3-based embedding refinement was deferred to future work, as the A2C-based system already demonstrated substantial gains.

References

[1] Blondel, V. D., Guillaume, J. L., Lambiotte, R., Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), P10008. DOI.

[2] Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv:1606.05250.

[3] Otmazgin, S., Cattan, A., Goldberg, Y. (2022). F-coref: Fast, accurate and easy to use coreference resolution. In AACL.

[4] Orlando, R., Huguet Cabot, P.-L., Barba, E., Navigli, R. (2024). Retrieve, Read and Link: Fast and Accurate Entity Linking and Relation Extraction on an Academic Budget. In Findings of ACL 2024.

[5] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement learning. In ICML. Link.

[6] Google DeepMind. (2025). Gemini 2.0 Flash-Lite. Link. Accessed February 27, 2025.

[7] Google Cloud. (2025). text-embedding-005. Link. Accessed February 27, 2025.

[8] Fujita, A., Takahashi, M., Yamasaki, T. (2018). Twin Delayed Deep Deterministic Policy Gradient (TD3). arXiv:1802.09477.

[9] Dhara, R., Banda, G. (2025). *FR-Comm: Fast Community-Aware Knowledge Graph Construction for OpenQA*. Google Drive.

[10] Min, S., Lewis, M., Yih, W. (2023). *GraphRAG: Graph-Structured Retrieval-Augmented Generation for Open-Domain QA*. arXiv:2501.00309.

[11] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4), 229–256.

A Additional Experiments

In addition to the main results, we ran a few exploratory experiments to better understand the strengths and limitations of RL4KG-RAG.

First, we tested scaling the RL agent to larger knowledge graphs with more entities and relations. The A2C agent remained stable and improved retrieval up to medium-sized graphs (around 4000 nodes). However, performance dropped when applied to the full FastCoref-ReLiK graph (around 6500 nodes and 15700 relations). This was mainly due to memory limits on our AWS GPUs and increased reward variance in very dense graphs. We believe tuning the A2C hyperparameters and adding gradient clipping could help address this.

Next, we tried to match the QA accuracy of the FastCoref-ReLiK baseline. However, running large-scale LLM-based evaluation with the Gemini API was limited by API rate caps and cloud compute costs. Because of this, we ran RL4KG-RAG on a reduced set of QA pairs. We expect that with more resources and full evaluation, RL4KG-RAG would narrow the accuracy gap with FastCoref-ReLiK.

Finally, we explored hybrid KG construction. We initialized the RL agent with an FR-Comm clustering and let it refine the graph. Early results showed faster learning and better QA accuracy, but this needs more systematic testing.

Overall, these experiments suggest RL-based KG optimization works well for medium-sized graphs and can benefit from hybrid approaches. Scaling to larger graphs is currently limited by compute resources, which we plan to improve in future work.

B Implementation Details

We implemented RL4KG-RAG in Python using PyTorch and NetworkX for graph learning and manipulation. The RL agent was trained with the Advantage Actor-Critic (A2C) algorithm, using these hyperparameters:

- Learning rate: 1e-4
- Discount factor (γ): 0.99
- Entropy regularization: 0.01
- Batch size: 64 episodes
- Training episodes: 50k for A2C, 20k for REINFORCE (REINFORCE was less stable)

For QA, we used a retriever-reader pipeline based on Retrieval-Augmented Generation (RAG). Retrieval recall was computed using BM25, and QA accuracy was judged by a Large Language Model (LLM) through the Gemini API, with some manual calibration on sample QA pairs.

All training and experiments ran on AWS EC2 instances with NVIDIA A100 GPUs (p4d). This allowed us to parallelize RL training and KG construction efficiently.

The FR-Comm baseline used a precomputed FastCoref-ReLiK KG, which required much larger compute resources for both coreference resolution and KG generation. We aimed to match its QA accuracy with RL4KG-RAG, but limited access to the Gemini API and AWS compute budget prevented full-scale evaluation in this run.

Still, RL4KG-RAG showed strong improvements in retrieval recall and QA accuracy with a more compact and dynamic KG, offering better flexibility and interpretability than large static clustering approaches.

In future work, we plan to scale up training on distributed AWS clusters and develop a local, resource-efficient LLM evaluation pipeline to avoid current external limitations.