FRI Natural language processing course 2025
Project report

University of Ljubljana
Faculty of Computer and
Information Science

Integrating Structured Knowledge into Large Language Models

Konstantin Bojchevski, Matic Conradi, and Matjaž Kumin

Abstract

This project explores integrating structured knowledge into Large Language Models (LLMs) to improve reasoning. We focus on incorporating knowledge graphs (KGs), particularly focusing on a subset of Wikidata's data about the Slovenian counties, cities, castles, rivers and mountain peaks, to enhance question answering. We explore direct input augmentation, model finetuning and LoRA based model augmentation. This report presents our approaches and methods, as well as evaluation of their impact on question answering performance.

Keywords
LLMs, Knowledge Graphs, Wikidata, RAG, Finetuning, LoRA, Reasoning

Advisor: Slavko Žitnik

Introduction

Large Language Models (LLMs) have significantly advanced natural language understanding and generation. However, these models often struggle with complex reasoning tasks that require structured knowledge. This project explores techniques for integrating knowledge graphs (KGs) into LLMs to enhance their ability to accurately answer complex questions.

Our focus is on incorporating structured data about the Slovenian counties, cities, castles, rivers and mountain peaks into LLMs via KGs. By leveraging semantic relationships between aforementioned entities, we aim to improve the model's performance in accuracy and reasoning. This project will evaluate different integration techniques and their impact.

Related Work

Several previous studies have explored how to combine knowledge graphs and LLMs to improve reasoning and language understanding. Key works include:

- **GraphGPT: Graph Instruction Tuning for Large Language Models** [1]: Tang et al. (2024) explored tuning LLMs with graph-structured data, demonstrating the benefits of structured knowledge in language generation.
- **Combining Knowledge Graphs and Large Language Models** [2]: Kau et al. (2024) investigated hybrid approaches for combining KGs and LLMs.
- **Deep Bidirectional Language-Knowledge Graph Pretraining** [3]: Yasunaga et al. (2022) introduced pretraining methods integrating bidirectional knowledge graphs into LLMs.
- **Graph Language Models** [4]: Plenz and Frank (2024) overview the use of graph-structured information with LLMs.
- **Chain-of-Knowledge: Integrating Knowledge Reasoning into Large Language Models by Learning from Knowledge Graphs** [5]: Zhang et al. (2024) explored integrating knowledge reasoning into LLMs by learning from knowledge graphs.
- **InfuserKI: Enhancing Large Language Models with Knowledge Graphs via Infuser-Guided Knowledge Integration** [6]: Wang et al. (2024) proposed enhancing LLMs with KGs via infuser-guided knowledge integration.

Methodology

Dataset preparation

Our methodology for data preparation and knowledge graph construction began with programmatic data acquisition from Wikidata. A proprietary client was implemented to run our SPARQL queries to retrieve information related to Slovenian municipalities, mountain peaks, castles, their properties, and their relationships. These queries targeted specific attributes such as population counts, geographical area, elevation, and cultural heritage classifications.

The client was engineered to parse the structured JSON responses from the Wikidata Query Service, performing necessary data type conversions and managing absent data points, ultimately organizing the information into a directed graph. Entities such as municipalities, regions, peaks and castles, along with their respective attributes (e.g., population, area, elevation, heritage status), were modeled as nodes within this graph. In particular, numerical attributes were also represented as distinct nodes, including nodes for their approximate rounded values to support more flexible querying. The relationships between these entities, for instance, "is located in," "has population," or "belongs to heritage," were encoded as directed edges. This meticulously constructed knowledge graph provides a structured representation of the domain, enabling the subsequent generation of question-answer pairs through a system that utilizes predefined templates to interrogate the graph's relational structure and content.

RAG

Our approach to Retrieval Augmented Generation (RAG) centers on using a structured knowledge graph to improve how LMMs answer questions. The process can be outlined as follows:

First, we establish a knowledge base in the form of a graph, following the procedure described in the previous section. From this graph, we automatically create question-answer pairs. Importantly, each of these pairs also includes a set of relevant facts, drawn directly from the graph, which provide the necessary context for each question.

Next, we evaluate the RAG system by giving these questions to an LLM in two different ways. In the first scenario, the LLM answers the questions using only its internal knowledge. In the second, the LLM receives the same questions, but this time, its input is enriched with the contextual facts we extracted earlier from our knowledge graph. We then check how accurate the LLM's answers are in both situations by comparing them to the known correct answers. This checking process is handled in two steps. Firstly, a separate LLM, which acts as an unbiased judge to see if the generated answer matches the correct one, evaluates the answers. Then answers are evaluated manually to double check everything.

Lastly, we gauge the improvement by comparing the LLM's accuracy when it answers with the extra context versus without it. This comparison helps us measure how much the retrieved facts from the knowledge graph actually help the LLM, showing how well our RAG strategy works.

Finetuning

To improve the ability of a pretrained language model to reason over structured information specific to our domain, we attempted supervised finetuning on the `cjvt/GaMS-9B` model. This large Slovenian causal language model was selected due to its linguistic compatibility and public availability.

We constructed a dataset of 10,000 examples in the form of instruction pairs, where each sample consists of a natural language question (prompt) and a corresponding structured answer (response), typically derived from our domain-specific knowledge graph. Each example was serialized in JSON format and tokenized using the GaMS tokenizer, with a maximum sequence length of 256 tokens.

Finetuning was implemented using HuggingFace's Trainer API with half-precision floating point (`fp16`) and gradient accumulation in an attempt to manage the substantial memory requirements of the 9B-parameter model. Despite careful tuning, we encountered persistent CUDA out-of-memory and initialization errors across multiple configurations (including reduced batch sizes and single/multi-GPU variants). These issues made it unable to successfully complete the finetuning process within the available hardware constraints.

While this component of the pipeline could not be finalized, we include it in the report to demonstrate the feasibility of the approach and to document the encountered limitations. In theory, finetuning offers the model a chance to learn domain-specific patterns and phrasing present in our generated graph-based examples. In contrast to RAG, which augments retrieval at inference time, finetuning aims to internalize this reasoning capability and is expected to generalize better on structured queries grounded in our knowledge graph.

LoRA Fine-tuning

Due to the computational constraints encountered with full fine-tuning of the 9B parameter model, we implemented an alternative approach using Low-Rank Adaptation (LoRA) on the smaller `cjvt/GaMS-1B` model. LoRA enables parameter-efficient fine-tuning by decomposing weight updates into low-rank matrices, significantly reducing memory requirements while maintaining the potential for effective adaptation.

We configured LoRA with a rank of 8, alpha value of 16, and targeted the query and value projection layers (`q_proj`, `v_proj`) with a dropout rate of 0.05. The training dataset consisted of 5,000 question-answer pairs automatically generated from our knowledge graph, formatted as direct question-answer pairs without explicit contextual information. This "knowledge integration" approach aims to internalize factual knowledge directly into the model parameters rather than relying on external retrieval.

The training process utilized 4-bit quantization and half-precision floating point (`fp16`) to optimize memory usage, with a batch size of 2 and gradient accumulation over 4 steps. Training was conducted for 3 epochs using the Hugging Face Trainer API, with models saved at each epoch to monitor progress and prevent overfitting.

Results

RAG

The integration of a knowledge graph into our Retrieval Augmented Generation (RAG) system demonstrably enhanced performance by providing a structured and contextually rich source for information retrieval. As a result, the relevance and factuality of the generated outputs saw a significant increase, generally achieving a near perfect score of 100% in accuracy, though we should note that the validation set is quite small, only on the order of hundreds of examples, due to the fact that evaluation has been done manually in the second step of our evaluation procedure. A perfect score is a massive jump compared to sub 10% accuracy of the model on its own.

Furthermore, the knowledge graph enabled more sophisticated reasoning capabilities, allowing the RAG model to synthesize information from multiple, interconnected data points to answer complex queries that previously yielded incomplete or superficial answers. This was particularly evident in scenarios requiring an understanding of multi-hop relationships or hierarchical structures between entities provided in the prompt. Initial observations therefore point towards a marked improvement in overall answer quality and correctness, indicated by our results.

LoRA Fine-tuning

Our LoRA implementation utilized a rank of 8, alpha value of 16, and targeted the query and value projection layers (`q_proj`, `v_proj`) with a dropout rate of 0.05. The training dataset consisted of 5,000 question-answer pairs automatically generated from our knowledge graph using predefined templates, where each example contained only a direct question (e.g., ”V kateri regiji se nahaja Ljubljana?”) paired with its corresponding answer, without explicit contextual facts in the prompt.

This approach, termed ”knowledge integration,” aims to internalize factual knowledge directly into the model's parameters rather than relying on external retrieval at inference time. The model was trained for 3 epochs using 4-bit quantization, gradient accumulation, and half-precision floating point (`fp16`) to optimize memory usage on available hardware. Training completed after 1875 steps with a batch size of 2, achieving a final training loss that fluctuated between 0.75-0.83 in the final epochs.

Evaluation through our testing revealed that the LoRA fine-tuned model showed modest improvement over the base model. While the model demonstrated some learning capacity and occasionally provided more relevant responses compared to the base model, it frequently failed to provide correct answers when tested on geographical questions about Slovenia without explicit context. The limited success suggests that while the knowledge integration approach shows promise, the training dataset of 5,000 question-answer pairs may have been insufficient for effective knowledge internalization. We suspect that significantly more training data would be required to achieve robust performance on factual geographical information. The model's performance indicated that simply fine-tuning on question-answer pairs without explicit context, particularly with limited training examples, may not be sufficient for comprehensive knowledge internalization. Despite the computational efficiency of the LoRA approach, the results suggest that either larger training datasets or alternative methods such as retrieval-augmented generation (RAG) may be more effective for incorporating structured knowledge into language models for domain-specific question answering tasks.

References

[1] Jiabin Tang, Qinkai Zheng, Yuchen Yan, Hongliang Fei, Jianfei Cai, and Li Guo. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

[2] A. Kau, X. He, A. Nambissan, A. Astudillo, H. Yin, and A. Aryani. Combining knowledge graphs and large language models. *arXiv preprint arXiv:2407.06564*, 2024.

[3] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323, 2022.

[4] Moritz Plenz and Anette Frank. Graph language models. *arXiv preprint arXiv:2401.07105*, 2024.

[5] Y. Zhang, X. Wang, J. Liang, S. Xia, L. Chen, and Y. Xiao. Chain-of-knowledge: Integrating knowledge reasoning into large language models by learning from knowledge graphs. *arXiv preprint arXiv:2407.00653*, 2024.

[6] F. Wang, R. Bao, S. Wang, W. Yu, Y. Liu, W. Cheng, and H. Chen. Infuserki: Enhancing large language models with knowledge graphs via infuser-guided knowledge integration. *arXiv preprint arXiv:2402.11441*, 2024.