Bachelor- / Master Thesis Proposal

Natural Language Querying Interface for Knowledge Graphs in Secured Environment

Background and Motivation

In various domains of research and development, findability, accessibility, interoperability, and reusability (FAIR) have been set as the industry standard for data handling. The FAIR principle not only reduces the cost of funding and time of conducting research in the long run but is also a practical approach of data management that ensures validations and transparencies of research.

A common approach in FAIRification is to convert map the data to an ontology and then convert the data into a knowledge graph, which is materialized in the Resource Description Framework (RDF) format. The data that is stored in the knowledge graph can be queried though SPARQL query language.

A challenge that researchers in the field have faced is that the SPARQL query language is a niche tool that only a few are fluent in. This has become a bottleneck that slows down the retrieval of data. To overcome this limitation, previous efforts have already focused on manually translating a user query written in natural language into a valid SPARQL query in order to query the knowledge graph. Examples include the general-purposed SPARQL translation tool like K-Extractor (Tatu, Balakrishna, Werner, Erekhinskaya, & Moldovan, 2016), SPARK (Ferré, 2017), CASIA@12 (He, Zhang, Liu, & Zhao, 2014). Those tools function well only under limited context and uncomplex queries.

The Aim and Approaches

Since the translation of SPARQL queries is still error prone, this thesis tries to follow a different methodology. Instead of translating natural language queries into SPARQL by using ChatGPT, the goal of this thesis is to modify an existing open-source large language model like Hugging Face or Llama to understand the knowledge graph and develop software that researchers can use to query the information within the knowledge graph using natural language. Some existing packages and frameworks like Ollama and Langchain could play a major role in this project.

Below are the two approaches that we would like to compare the performances with:

Vectorize Knowledge Graph And using RAG:

Knowledge graphs are a structured way of storing information as triples of subject, predicate, and object, thus preserving their semantic interpretation. On the other hand, LLMs are trained on a large corpus of text, and the acquired knowledge is fused inside the model, unstructured, and hardly reproducible. In the suggested approach, an LLM is used to navigate the knowledge graph, thus maintaining the deterministic nature of the latter. Because of the input window limitations of LLMs (a couple of thousand tokens for open-source LLMs), access to the knowledge graph can be gained through RAG pipelining. Following this process line, the triples are first converted to sentences and later stored in

Contact Person

Junda Huang, Ali Bahja | **E-Mail:** huang@uni-wuppertal.de, bahja@uni-wuppertal.de

a vector database. The vector store can be queried by similarity search methodologies, and the acquired output is fed as a prompt to the LLM, which would then reformulate an answer accordingly. Following this method, one can study the limitations of vectorizing large knowledge graphs, and ways to overcome such complexities. An example would be studying the effects of vectorizing the semantic model of the knowledge base or applying a graph-based RAG approach.

Periodical Fine-tuning of LLMs:

The other approach is to update the LLM model by retraining on the Knowledge graph, and querying would then be subject to the probabilistic nature of the language model. In the case of updates to the knowledge graph, the model would then be periodically retrained to include the latest version of knowledge. Following this approach, the triple representation must again be converted to form sentences before training.

Project plan

Consider the scope of the contents and the time limitation of the thesis durations; students can choose one or a combination of 2 of the below sub-projects.

Benchmarking current situation (Bachelor Thesis):

Before modification of the large language models and adapting to the project needs, the performances of the models need to be benchmarks. Some of the key benchmarking points are time and computational power needed for LLM finetuning as well as vectorize/tokenize the knowledge graphs.

Proof of concept (Master Thesis):

Using benchmarking data, we need to compare the above-mentioned approaches to find out which one is more efficient in different industrial settings.
E.g. time consumption is more tolerant in academic settings, while computational powers are more abundant in industries.

Interface development (Bachelor Thesis):

The ultimate goal is to produce a user-friendly interface, with which researchers across all domains can make use of querying their data.

Knowledge requirements and supervisions

The ideal candidate(s) for thesis 1 and 2 should have adequate knowledge of how a pre-trained large language model works as well as hands-on knowledge of LLM finetuning and RAG.

For sub-project 3, front-end development plays an important role. We would like you to have the capabilities of GUI development as well as the interaction with backend language models.

Ali Bahja will supervise language model modifications and software development.

Junda Huang will serve as the domain user for the language models and provide general feedback on data management and user experiences.

Contact Person

Junda Huang, Ali Bahja | E-Mail: huang@uni-wuppertal.de, bahja@uni-wuppertal.de