RAG Application Using Knowledge Graph and Vector Search

Prototype Project Report

HackerEarth

MECA Dynamics

Ethan Villalovoz, Molly Iverson, Adam Shtrikman, Chandler Juego

Table of Contents

Introduction

I. Project Introduction

In recent years, there has been an explosion of interest in large language models (LLMs) and their application in natural language processing (NLP), driving innovations across industries, from customer service chatbots to research assistants. One compelling application of LLMs is in Retrieval-Augmented Generation (RAG), where models retrieve relevant information from a dataset and generate contextually accurate responses. This combination of retrieval and generation has positioned RAG systems as an essential tool in question-answering and content generation. However, while RAG models using vector search have shown success, they are often limited by the unstructured nature of the data they rely on.

This project addresses this limitation by incorporating knowledge graphs into the retrieval process, a novel approach that promises to revolutionize RAG systems. Knowledge graphs offer structured, contextual information that enhances the ability of RAG systems to understand and retrieve more accurate, fact-based responses. By integrating knowledge graphs with vector search, this project aims to develop an advanced RAG application that improves traditional methods by utilizing unstructured and structured data. The innovative nature of this project is sure to excite and intrigue those in the field of NLP and technology.

This project will use a large dataset of 10,000 Wikipedia articles to create a knowledge graph and vector search database. The system will handle user queries by retrieving semantically relevant information from the knowledge graph and vector search, enabling the generation of more precise and contextually appropriate responses. The motivation behind this project lies in the potential for knowledge graphs to revolutionize the retrieval aspect of RAG applications, ultimately improving the overall quality and utility of generated responses in various use cases, from conversational agents to research tools.

II. Background and Related Work

To evaluate current leaders in the same domain as our project, we must first define what that domain is. For this document, we have narrowed down this definition to “knowledge-intensive natural language processing tasks.” Our project is at the intersection of information retrieval, natural language generation, and structured knowledge representation through knowledge graphs. Given this definition, our research highlights a representative example of a highly successful existing contribution in the domain.

The most notable and relevant to our work is the research by P. Lewis et al. (2020) from Facebook/Meta on Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [1]. Their model has two main components: a retriever, which fetches relevant documents based on a query, and a generator, which uses those documents to create responses. RAG has proven successful for open-domain question-answering and retrieving accurate information from large datasets. According to this study, dense vector search significantly improves performance compared to standard generation models. However, their method only uses vector search and does not take advantage of the structure and insights provided by knowledge graphs. While pure vector search mechanisms can capture similarity or relatedness between data, they do not provide the logical deduction and robust inferencing that knowledge graphs can, as shown by K. Guu et al. [2]. At the cost of additional complexity due to the structure of knowledge graphs, a combination of vector search and knowledge graphs can enhance both semantic retrieval and

structured reasoning. By leveraging the strengths of vector embeddings for approximate similarity search alongside the explicit relationships in a knowledge graph, this hybrid approach can offer more precise, explainable, and context-aware results. Our project builds on this by integrating knowledge graphs into the RAG pipeline, aiming to improve retrieval accuracy and response quality, especially for complex, multi-step queries.

To successfully execute the technical aspects of this project, our team must master several concepts, tools, and frameworks. A strong understanding of RAG, vector search, and knowledge graphs is critical. Before any coding starts, it's essential to practice building a knowledge graph from a small set of text documents, adding relationships between nodes, adding vector indices and embeddings, implementing vector search to retrieve relevant information from the data, and integrating an LLM (e.g. OpenAI) to generate a response from the search results. To accomplish this, we plan to take several online courses, including Knowledge Graphs for RAG [3], JavaScript RAG Web Apps with LlamaIndex [4], Building and Evaluating Advanced RAG Applications [5], and Generative AI with Large Language Models [6]. These courses will equip us with the necessary skills to effectively integrate vector search and knowledge graphs into our application. Furthermore, we will explore existing LLMs to understand their implementation and how they can be optimized for generating responses based on the retrieved data.

III. Project Overview

We are currently in a time where AI-driven technologies and solutions are rapidly evolving. This has created a new demand for efficient information retrieval and generation systems in various industries. This application will utilize a dataset of 10,000 Wikipedia articles to build a knowledge base using the knowledge graph architecture combined with vector search algorithms, enabling more contextually rich and accurate responses.

While current RAG applications rely on vector search, new research suggests that incorporating knowledge graphs can greatly enhance performance. While vector search can retrieve semantically similar data points, it lacks the contextual relationships offered by knowledge graphs. This project aims to integrate knowledge graphs into RAG architecture to enhance relevance, accuracy, and contextual depth. U

The difficulty is in integrating the benefits of vector search, which can retrieve semantically relevant content from large unstructured datasets, with the structured and relational data inherent to knowledge graphs. This project will focus on querying both vector search and knowledge graphs databases.

The primary objective of this project is to build a RAG application that combines vector search and knowledge graphs. Users will be able to interact with the RAG model through a full-stack web application with a chat-style frontend and a backend supported by JavaScript, featuring data persistence.

The core milestones are such:

- **Initial Research and Learning:** The team will study RAG model architecture (retriever and generator), knowledge graphs (creation, querying, integration into natural language processing tasks), vector search fundamentals, and tools relevant to implementation.
- **Data Collection and Knowledge Graph Construction:** The team will gather a dataset of Wikipedia articles and utilize an existing knowledge graph.

- **Implementation of Vector Search:** The team will generate embeddings for data points using pre-trained models, index them using a vector search library, and implement a retrieval mechanism.
- **Integration of Components:** The team will connect the vector search to the knowledge graph and the vector search to the large language model to create structured responses.
- **Application Development:** The team will implement the logic for query processing, retrieval, and response generation by inputting the query into the command line and outputting the response to a text file.
- **Testing and Optimization:** The team will test various inputs to assess performance and refine the retrieval algorithm, knowledge graph integration, and generator model to optimize output results.
- **Documentation and Final Presentation:** The team will formulate documentation throughout the project, focusing on code, methodologies, and important findings. This will culminate in a final presentation showcasing the application and key takeaways.

IV. Client and Stakeholder Identification and Preferences

Our primary client is HackerEarth, where Vikas Aditya, the company's CEO, will be our main point of contact. We will work closely with him and HackerEarth to ensure that the RAG application we develop meets the company's expectations and serves as a strong example of how knowledge graphs and vector search can enhance retrieval-augmented generation systems.

While the immediate goal is to create a well-functioning RAG application based on thousands of Wikipedia articles, the broader aim is to deliver a solution that HackerEarth can potentially use as inspiration for future projects. In addition to the core RAG application, a full-stack web app will be developed to streamline the process of making queries and receiving clear responses. By integrating an intuitive user interface with the power of knowledge graphs and vector search, the app will enable an effortless interaction for querying and retrieving relevant information. The application will be built to demonstrate best practices in combining knowledge graphs and vector search, providing HackerEarth with a reference point for future AI and data retrieval initiatives.

HackerEarth's research and engineering teams can use this project to explore how such technologies can be applied across different domains. It is important for the project to be designed in such a way that it makes it easier for future teams to experiment with similar technologies or adapt the methodology for other use cases. By maintaining a clear line of communication and focusing on delivering a solution that is both effective and easy to understand, we aim to create a RAG application that not only meets current goals but also opens the door for potential future use in various company projects.

V. Glossary

Knowledge Graph: A knowledge base that uses a graph-structured data model or topology to represent and operate on data.

Large Language Model (LLM): A machine learning model that can perform natural language processing tasks. They are trained on vast amounts of data in order to detect patterns effectively.

Natural Language Processing (NLP): a branch of artificial intelligence that uses machine learning to help computers interpret and generate human language.

Retrieval-Augmented Generation (RAG): A framework that combines the capabilities of large language models with information retrieval systems to improve the accuracy and relevance of AI-generated text.

Vector Search: A method for finding similar items to data points in large collections by using vector representations of items. These representations, or vector embeddings, can be used to quickly locate items in large data sets and allow for searches by meaning, rather than just keywords.

VI. References

[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Curran Associates Inc., Vancouver, BC, Canada, 2020, pp. 9459–9474. doi: https://dl.acm.org/doi/abs/10.5555/3495724.3496517.

[2] K. Guu et al., "Traversing Knowledge Graphs in Vector Space", *Stanford NLP Group – Standford University*, 2015. Available: https://nlp.stanford.edu/pubs/kg_traversal.pdf

[3] A. Kollegger, "Knowledge Graphs for RAG," DeepLearning.AI. [Online]. Available: https://www.deeplearning.ai/short-courses/knowledge-graphs-rag/

[4] L. Voss, "JavaScript RAG Web Apps with LlamaIndex," DeepLearning.AI. [Online]. Available: https://www.deeplearning.ai/short-courses/javascript-rag-web-apps-with-llamaindex/

[5] J. Liu and A. Datta, "Building and Evaluating Advanced RAG Applications," DeepLearning.AI. [Online]. Available: https://www.deeplearning.ai/short-courses/building-evaluating-advanced-rag/

[6] "Generative AI with Large Language Models," Coursera. [Online]. Available: https://www.coursera.org/learn/generative-ai-with-llms