L'Oreal Capstone Knowledge Graph Construction Midterm Report

Brandon Keung[1], Jiaheng Dai[1], Jiayi Dong[1], and Pei Tian[1]

[1] *Columbia University, Fu Foundation School of Engineering and Applied Science*

Course:
ENGI E4800 Data Science Capstone & Ethics

Date:
March 17, 2025

Contents

Abstract

Large Language Models (LLMs) have transformed the way that people go about their daily lives. However, LLMs continue to face challenges such as hallucinations, factual accuracy, and domain-specific query resolution. Retrieval Augmented Generation (RAG) systems mitigate some of these issues by incorporating external domain knowledge. Classical RAG applications utilize vector based information to provide LLMs with meaningful information. However, graph-based RAG (GraphRAG) emerged as a promising alternative to the traditional vector based retreival methods. Research into GraphRAG shows promising results but challenges lie in constructing knowledge graphs (KG) and querying from them.

Our project investigates the impact of various KG construction techniques on GraphRAG performance, focusing on retrieval accuracy, response relevance, and inference speed. In this report we explore different KG construction ideas such as entity relationships and hierarchical structures to optimize graph formation. By evaluating KG methodologies, this research aim to advance GraphRAG's application in the e-commerce industry.

1 Introduction

1.1 Previous Work & Motivation

In recent years, Large Language Models (LLMs) have revolutionized natural language processing with their ability to generate coherent, contextually relevant responses. However, these models still face significant challenges related to hallucinations, factual accuracy, and contextual relevance when dealing with domain-specific queries. To address these limitations, Retrieval Augmented Generation (RAG) has emerged as a promising solution by augmenting LLMs with external knowledge sources. First introduced in the paper ”Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks” (Lewis et al., 2020), RAG systems enhance model outputs by retrieving and incorporating relevant information from external databases during the generation process.

Last semester's capstone project conducted a comprehensive comparison between traditional Vector-based RAG systems and Graph-based RAG (GraphRAG) systems for L'Oréal's product recommendation and description use cases. Their findings demonstrated that while Vector RAG outperformed GraphRAG slightly in terms of faithfulness and semantic similarity metrics, GraphRAG exhibited significantly faster retrieval times, approximately 1 second compared to Vector RAG's 47 seconds. This performance advantage positions GraphRAG as a potentially superior option for real-time applications where response latency is critical.

Despite these promising results, several limitations were identified in the previous implementation. The knowledge graph construction process was largely unstructured, with node and edge formation lacking systematic organization. Additionally, the free Neo4j instance used for testing did not support efficient augmentation of the existing knowledge graph with new documents, necessitating complete graph reconstruction when adding information. Perhaps most significantly, the study revealed a disparity in performance between description-based queries (where GraphRAG scored 8.9/10 in human evaluation) and recommendation-based queries (scoring 7.6/10), suggesting that the knowledge graph structure may have been better optimized for certain query types.

Our project extends the previous work by focusing specifically on knowledge graph construction methods and their impact on GraphRAG system performance. We explore various techniques for entity extraction, relationship identification, and graph structuring, evaluating how these methodologies affect both the quality and efficiency of responses. By systematically analyzing these approaches, we aim to develop guidelines for optimizing GraphRAG implementations based on specific use cases, query types, and performance requirements.

1.2 Background

GraphRAG systems represent a specialized approach to retrieval augmentation by organizing information in graph-based structures where entities serve as nodes and relationships as edges. This approach builds upon decades of knowledge representation research, from early semantic networks to modern knowledge graphs like Google's Knowledge Graph (Singhal, 2012) and Neo4j's graph database technologies. Unlike vector stores that rely primarily on similarity metrics, knowledge graphs capture explicit relationships between entities, enabling more nuanced navigation through complex information spaces.

The current landscape of GraphRAG research and implementation is rapidly evolving. Recent work by Sarmah et al. (2023) demonstrated that GraphRAG systems excel at handling complex, multi-hop reasoning tasks where information must be connected across multiple entities. Yasunaga et al. (2022) showed how knowledge graphs can improve LLM reasoning through structured exploration of knowledge spaces. Meanwhile, industry implementations from companies like Microsoft have demonstrated that GraphRAG approaches particularly shine for global queries about general categories rather than specific instances (Edge et al., 2023).

However, a critical gap exists in understanding how different knowledge graph construction methodologies impact the performance of GraphRAG systems. Current approaches range from fully automated LLM-based construction (as used in the previous project with Neo4j's Knowledge Graph Builder), to rule-based extraction, to hybrid human-in-the-loop approaches. Each method presents different tradeoffs in terms of precision, recall, scalability, and domain adaptation capabilities, yet systematic evaluation of these tradeoffs remains limited.

Knowledge graph construction techniques have evolved substantially in recent years. Traditional methods relied heavily on manual curation or rule-based systems that were difficult to scale (Paulheim, 2017). More recent approaches leverage transformer models for named entity recognition and relationship extraction (Heinzerling et al., 2021). Some cutting-edge methods employ LLMs themselves to identify entities and relationships through few-shot prompting (West et al., 2022), while others use specialized graph neural networks to iteratively refine graph structures (Ji et al., 2021).

The performance of GraphRAG systems is intimately tied to the quality and structure of the underlying knowledge graph. Factors such as graph density, relationship diversity, entity granularity, and contextual embedding all influence the system's ability to retrieve relevant information efficiently. Understanding these relationships requires systematic investigation of how different construction methodologies impact downstream performance across various metrics and use cases.

2 Research Questions

This project seeks to address the following key questions:

1. What are the existing ways to construct knowledge graphs, and how do they differ in methodology, complexity, and resource requirements?

2. How do different knowledge graph construction methods affect the retrieval accuracy, response relevance, and inference speed of GraphRAG systems?

3. What graph structures and relationship patterns are most effective for different types of queries?

4. What techniques can improve GraphRAG performance for recommendation tasks where the previous implementation showed lower performance?

By addressing these questions, our research aims to advance the understanding of GraphRAG systems and provide practical insights for their implementation in industrial applications, particularly in e-commerce and product information management contexts.

3 Experiment Methodology

3.1 Datasets

Currently, we are experimenting with a dataset found on Hugging Face: milistu/AMAZON-Products-2023. With 117,243 rows and 15 columns, we have a wide range of products and brands. Currently, we are working with the entire dataset but plan to work on a subset of data. In particular we are choosing products that fall under the beauty and personal care category due to the relevancy to L'Oreal. In doing so, we hope that our project can more directly relate to L'Oreal as specific entity relationships can be transferred to L'Oreal products. This results in about 3,000 products that will be implemented in our knowledge graph. Due to the nature of our project being primarily experimental, we want to also experiment with products not under the beauty/personal health category to get a better idea of how to better construct knowledge graphs slightly different domains.

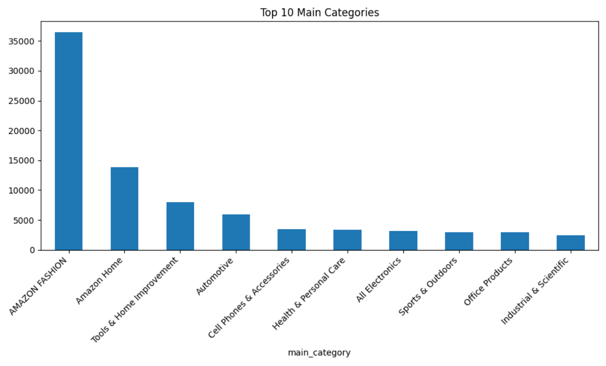


Figure 1: Overall distribution of main category for the entire dataset

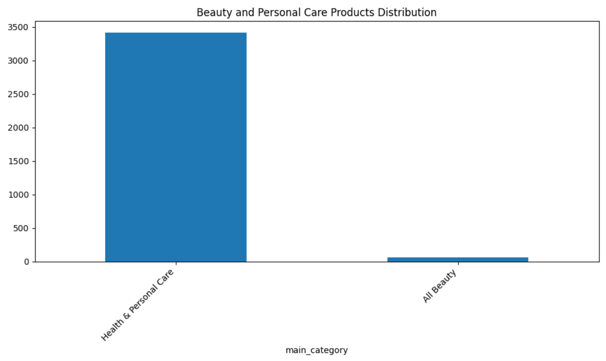


Figure 2: main category distribution for sample of data we are working with (beauty and health products)

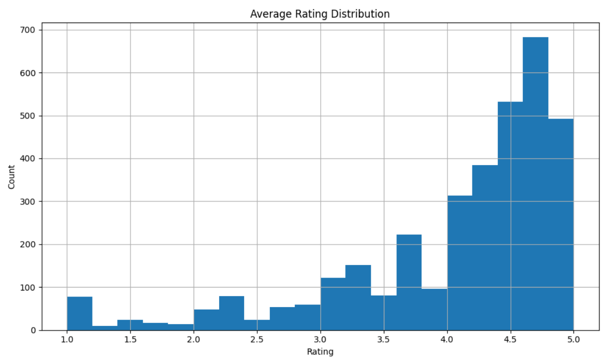


Figure 3: Average rating among products that fall under the beauty and health products

The graphs provide valuable information about the dataset. Amazon Fashion is the largest category in the entire dataset, followed by Amazon Home and Tools & Home Improvement. The category we're interested in, Health & Personal Care, is ranked 6th, indicating a decent representation. However, our other focus category, Fashion, has very few data points, which might interfere with domain-specific knowledge extraction.

We analyze the rating feature using a histogram, revealing a right-skewed distribution where most products receive positive reviews. Lower ratings (1.0 - 3.0) appear significantly less frequent than higher ratings (3.0 - 5.0), suggesting a generally positive customer experience or potential biases in the rating/reviewing mechanism.

3.2 RAG

Retrieval-Augmented Generation (**RAG**) combines retrieval-based methods with generative models to improve knowledge-grounded text generation. The methodology consists of two key components:

- **Retriever:** Given a query, the retriever fetches relevant documents from a knowledge source (e.g., a vector database).
- **Generator:** The retrieved documents are then provided as context to a pre-trained generative model that generates a response based on both the input query and the retrieved knowledge.

The process follows these steps:

1. Encode the query into a vector space and retrieve the top-k relevant documents.
2. Concatenate the retrieved documents with the input query.
3. Feed the combined input into a sequence-to-sequence model to generate the final response.

RAG enhances model responses by incorporating external knowledge dynamically, reducing hallucinations compared to standard language models while maintaining flexibility for various applications, such as product recommendations and customer support in e-commerce.

In our project, **ChromaDB** is used as a vector database to store and retrieve document embeddings in a RAG pipline. Each document is embedded into a high-dimensional vector space using a pre-trained embedding model (e.g., OpenAI embeddings). This approach, referred to as **Vector RAG**, enhances the retrieval process by leveraging dense vector representations to find the most relevant documents for query augmentation.

3.3 Graph RAG

Graph RAG extends the Vector RAG framework by incorporating structured knowledge graphs into the retrieval process. This approach enhances the model's ability to generate contextually relevant and semantically rich responses leveraging explicitly defined relationships between entities.

For example, consider two entities **Apple AirPods Pro** and **Apple AirPods Pro Case** in the **Amazon Product Dataset**. They might be connected through a relationship, such as:

Apple AirPods Pro Case —[**ACCESSORY FOR**]— Apple AirPods Pro

At the implementation level, Graph RAG enhances the retrieval and generation pipeline by integrating a graph database, such as **Neo4j**. The system typically consists of three core components:

1. **Retriever**: Traverses the knowledge graph to identify relevant nodes and edges, leveraging structured relationships to improve retrieval accuracy.

2. **Ranker**: Prioritizes retrieved subgraphs based on contextual relevance, ensuring the most informative and meaningful knowledge is passed to the model.

3. **Generator**: Conditions the LLM on both graph-based insights and retrieved text passages, allowing for responses that are both contextually rich and factually grounded.

Neo4j's Cypher queries allow efficient extraction of interconnected concepts, making Graph RAG particularly effective for domains requiring reasoning over structured knowledge.

3.4 Evaluation Metrics

To evaluate the quality of Graph RAG and compare it with the Vanilla RAG baseline, we used RAGAS (Retrieval-Augmented Generation Assessment). The following metrics are used for assessment:

- **Context Recall**: Measures how many relevant documents or information pieces are retrieved.

- **Context Precision**: Measures the signal-to-noise ratio, which means counting how many sentences from the context are necessary to answer the question compared to the total number of sentences retrieved.

- **Faithfulness**: Measures how well the generated answer aligns with the given context, ensuring that claims can be inferred from the retrieved text.

- **Answer Relevancy**: Measures how well the generated answer aligns with the given prompt, assigning lower scores to incomplete or redundant.

In addition to RAGAS, the inference time of models is also valued. Faster inference ensures smoother user experiences, particularly in interactive settings like customer support and and recommendation systems.

4 Results

4.1 Current Progress

In the initial phase of this project, we have focused on setting up the fundamental evaluation pipeline for both Vector RAG and Graph RAG approaches in the context of Question Answering (QA) tasks. This involved designing a systematic framework that allows for a fair and comprehensive comparison of these two retrieval strategies.

We successfully completed the Vector RAG pipeline using ChromaDB as the vector database, and evaluated it on 1000 random samples of the entire Amazon Product dataset. The evaluation results for Vector RAG are depicted in Figure 4.

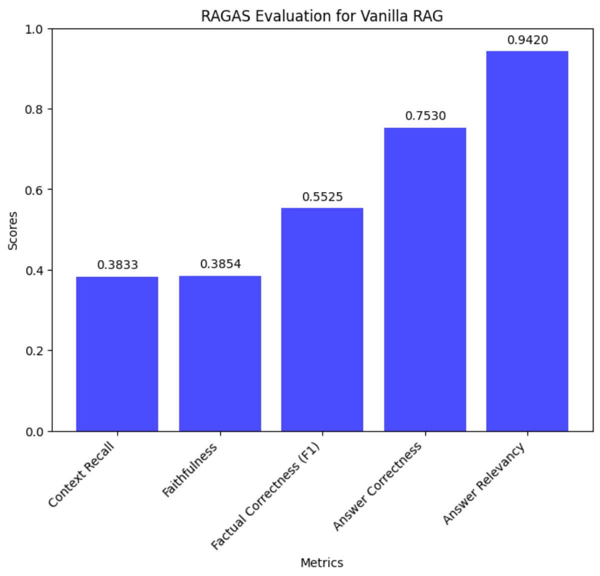
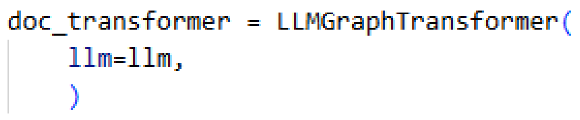


Figure 4: RAGAS Evaluation of Vector RAG

Additionally, we have curated and refined an Amazon Product database, ensuring data cleanliness and consistency. A basic sample database has been created, which serves as the foundation for testing retrieval mechanisms and evaluating different RAG pipelines. The preprocessing steps involved data normalization, deduplication, and schema standardization, making the dataset more structured and suitable for downstream tasks.

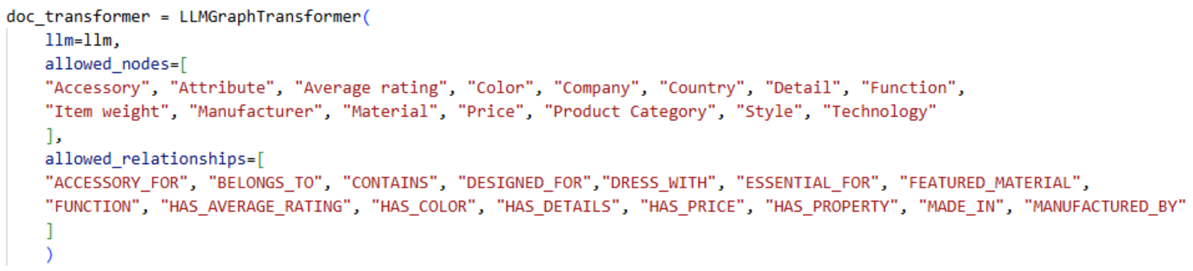
Previously, we ran into issues with passing large amounts of data into OpenAI API, leading to problems generating our knowledge graph. To experiment with a smaller data size, we chose 50 samples from the Amazon Product dataset and successfully constructed a knowledge graph. We utilized LLMGraphTransformer, a framework empowered by LangChain that leverages large language models (in our case, GPT-Turbo 3.5 via API call) to extract structured knowledge from unstructured text, and generate our knowledge graph. We explored two different approaches to construct the graph: the default LLMGraphTransformer function call (in Figure 5) and a customized version with predefined nodes and relationships (in Figure 6). The default approach produced 2260 nodes and

5803 relationships, while the predefined approach generated 2545 nodes and 6548 relationships, indicating that using predefined parameters results in a more detailed knowledge graph. Our graph, visualized in Figure 7, captures key relationships (e.g., belongs to, contains, has color) between specified nodes (e.g., attributes, accessory, company). However, this is only a displayable subset of the full graph. This situation emphasizes the need for effective filtering and reduction of nodes, as we scale up to the entire dataset.



```
doc_transformer = LLMGraphTransformer(
    llm=llm,
    )
```

Figure 5: Code Snippet for Default Function Call



```
doc_transformer = LLMGraphTransformer(
    llm=llm,
    allowed_nodes=[
    "Accessory", "Attribute", "Average rating", "Color", "Company", "Country", "Detail", "Function",
    "Item weight", "Manufacturer", "Material", "Price", "Product Category", "Style", "Technology"
    ],
    allowed_relationships=[
    "ACCESSORY_FOR", "BELONGS_TO", "CONTAINS", "DESIGNED_FOR","DRESS_WITH", "ESSENTIAL_FOR", "FEATURED_MATERIAL",
    "FUNCTION", "HAS_AVERAGE_RATING", "HAS_COLOR", "HAS_DETAILS", "HAS_PRICE", "HAS_PROPERTY", "MADE_IN", "MANUFACTURED_BY"
    ]
    )
```

Figure 6: Code Snippet for Predefined Nodes and Relationships Function Call

Figure 7: Knowledge Graph generated from 50 selected samples

5 Future work

As the project progresses, we plan to enhance and expand our current implementation through the following key areas of development:

5.1 Knowledge Graph Construction

- Investigate various techniques for constructing knowledge graphs, ranging from classical ontology-based methods to more advanced LLM-powered graph generation approaches.
- Compare the effectiveness of handcrafted knowledge graphs versus automatically extracted knowledge graphs using deep learning techniques.

5.2 Scalability and Deployment

- Migrate from a local graph database setup to an online scalable solution using Neo4j's AuraDB cloud platform. This transition will facilitate handling larger datasets and improve performance in real-world applications.

5.3 Enhanced Information Extraction for Knowledge Graphs

- Improve the knowledge extraction methodology by shifting from keyword-based Cypher queries to ML-enhanced techniques for better entity recognition, relationship extraction, and contextual understanding.
- Leverage NLP models for semantic parsing and automated graph enrichment, ensuring more accurate and meaningful connections in the knowledge graph.

5.4 Embedding Model Optimization

- Transition from OpenAI's proprietary embedding models to open-source Hugging Face embedding models, enhancing transparency, flexibility, and cost efficiency.
- Experiment with different embedding architectures to optimize retrieval performance in both Vector and Graph RAG pipelines.

5.5 Performance Visualization and Comparative Analysis

- Develop interactive visualizations to compare various Graph RAG pipelines, including:
  - Traditional knowledge graphs (Classical KG)
  - LLM-based knowledge graphs (LLM-enhanced KG)
  - Machine learning-augmented KG retrieval techniques
- Conduct quantitative evaluations to assess the effectiveness of different retrieval mechanisms in terms of accuracy, response quality, and efficiency.

By systematically implementing these improvements, we aim to refine the retrieval process, enhance knowledge representation, and ultimately improve the quality of answers generated by the RAG system.

6 Contributions

- **Brandon Keung:** Team Captain, Neo4j Implementation, Neo4j Research, Data Management, Generate node/relationship queries, Generating Knowledge Graph using OpenAI API
- **Jiaheng Dai:** Vector RAG Pipeline, Evaluation Pipeline, Clean and Preprocess Data, Report Writing
- **Jiayi Dong:** Knowledge Graph Construction & LangChain API Research, KG construction, Report Writing
- **Pei Tian:** Evaluation Pipeline, Neo4j Implementation, Generate node/relationship queries, Generate Knowledge Graph

7 Reference

[1] Lewis, Patrick, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401, arXiv, 12 Apr. 2021. arXiv.org, https://doi.org/10.48550/arXiv.2005.11401.

[2] Singhal, A. (2012, May 16). Introducing the knowledge graph: Things, not strings. Google.
https://blog.google/products/search/introducing-knowledge-graph-things-not/

[3] Sarmah, Bhaskarjit, et al. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction. arXiv:2408.04948, arXiv, 9 Aug. 2024. arXiv.org, https://doi.org/10.48550/arXiv.2408.04948.

[4] Yasunaga, Michihiro, et al. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. arXiv:2104.06378, arXiv, 13 Dec. 2022. arXiv.org, https://doi.org/10.48550/arXiv.2104.06378.

[5] Edge, Darren, et al. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130, arXiv, 19 Feb. 2025. arXiv.org, https://doi.org/10.48550/arXiv.2404.16130.

[6] Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web, 8(3), 489-508.

[7] Heinzerling, Benjamin, and Kentaro Inui. Language Models as Knowledge Bases: On Entity Representations, Storage Capacity, and Paraphrased Queries. arXiv:2008.09036, arXiv, 21 Apr. 2021. arXiv.org, https://doi.org/10.48550/arXiv.2008.09036.

[8] West, Peter, et al. Symbolic Knowledge Distillation: From General Language Models to Commonsense Models. arXiv:2110.07178, arXiv, 28 Nov. 2022. arXiv.org, https://doi.org/10.48550/arXiv.2110.07178.

[9] Ji, Shaoxiong, et al. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. arXiv:2002.00388, arXiv, 1 Apr. 2021. arXiv.org, https://doi.org/10.48550/arXiv.2002.00388.