GraphRAG vs. Baseline RAG: Solving Multi-Hop Reasoning in LLMs

Stanley Jovel

Full Stack Developer interested in making lives better through software

Updated Nov 12, 2024

In our previous entry Building a custom RAG pipeline, we discussed how to create our own RAG pipeline using LlamaIndex and its benefits. In this article, I'd like to discuss into its limitations and shortcomings in specific scenarios and introduce how GraphRAG, proposed by Microsoft, addresses these issues. We'll follow this with a hands-on benchmark and comparison of both methods to highlight GraphRAG's advantages over Baseline RAG, using the MultiHop-RAG dataset for evaluating retrieval-augmented generation across documents.

Get the Source Code

The code for this analysis can be found in our project repository. You'll need to provide an OpenAI API key.

Data: For this project, we are using the MultiHop-RAG Dataset, which is designed for evaluating retrieval and reasoning across documents.

Baseline RAG and Its Shortcomings

As we mentioned in previous entries, a RAG application uses an external dataset composed of multiple structured or unstructured documents that serves as the knowledge base for our LLMs. Each document is then chunked and converted into vector embeddings. Retrieving information from this knowledge base relies on performing a similarity check on these vectors to retrieve relevant chunks.

The problem with this approach is that it often struggles with complex queries that require integrating information from multiple sources. Specifically, baseline RAG performs poorly when handling multi-hop queries, which are questions that require reasoning over multiple pieces of evidence or connecting disparate information across documents. For instance, answering a question like: "What is the name of the CEO of the company that acquired Bethesda Game Studios?" would require the system to first identify which company acquired Bethesda Game Studios and then find out who the CEO of that company is.

In such cases, simple vector similarity is not sufficient because the relevant information spans across different documents, and the connection between them isn't captured by embedding similarity alone.

knowledge graphs to enhance the retrieval process in RAG systems. The way it works is quite fascinating. At first glance, an arbitrary corpus of unstructured documents may not have obvious relationships. However, with the use of LLMs, it is possible to construct a knowledge graph where the LLM interprets the data and decides how different pieces of information are related. The result is a graph that is compatible with graph databases like Neo4j.

Technically, GraphRAG enhances the traditional RAG pipeline by incorporating:

1. **Entity Extraction**: The LLM processes text chunks to extract entities (e.g., people, organizations, places).
2. **Relationship Identification**: It identifies relationships between these entities across different documents.
3. **Knowledge Graph Construction**: The entities and relationships are used to build a knowledge graph, where nodes represent entities and edges represent relationships.
4. **Hierarchical Clustering**: The knowledge graph is organized into clusters or communities using algorithms like the Leiden method, grouping related entities together.

Getting started

We will set up these two RAG pipelines to test and compare their performance using the MultiHop-RAG dataset. This dataset comprises a collection of articles on various subjects from different sources and features questions that require the application to consult multiple sources for accurate answers. This presents the perfect scenario for comparing these two approaches to RAG.

If you want to follow along feel free to checkout our project repository

Loading the Corpus of Articles

We'll use the JSON reader since the MultiHop-RAG dataset is a single large JSON file:

```
from llama_index import download_loader

JSONReader = download_loader("JSONReader")

reader = JSONReader()
documents = reader.load_data("./data/corpus.json")
```

Setting Up the Baseline RAG Pipeline

After parsing and loading the articles, we'll set up a baseline RAG pipeline using LlamaIndex, similar to the steps described in our previous article: Indexing the Documents, Setting Up the Vector Index, Querying the index and Generating Answers with the LLM.

Setting Up GraphRAG

In this article, we'll be using LlamaIndex's Property Graph Index, which facilitates the creation and querying of knowledge graphs.

Getting started is very simple:

```
from llama_index import download_loader, PropertyGraphIndex
from llama_index.embeddings.openai import OpenAIEmbedding
from llama_index.llms import OpenAI

index = PropertyGraphIndex.from_documents(
    documents,
    llm=OpenAI(model="gpt-4o-mini"),
    embed_model=OpenAIEmbedding(model_name="text-embedding-3-small"),
    show_progress=True,
)
query_engine_KnowledgeGraph = index.as_query_engine(
    include_text=True
)
```

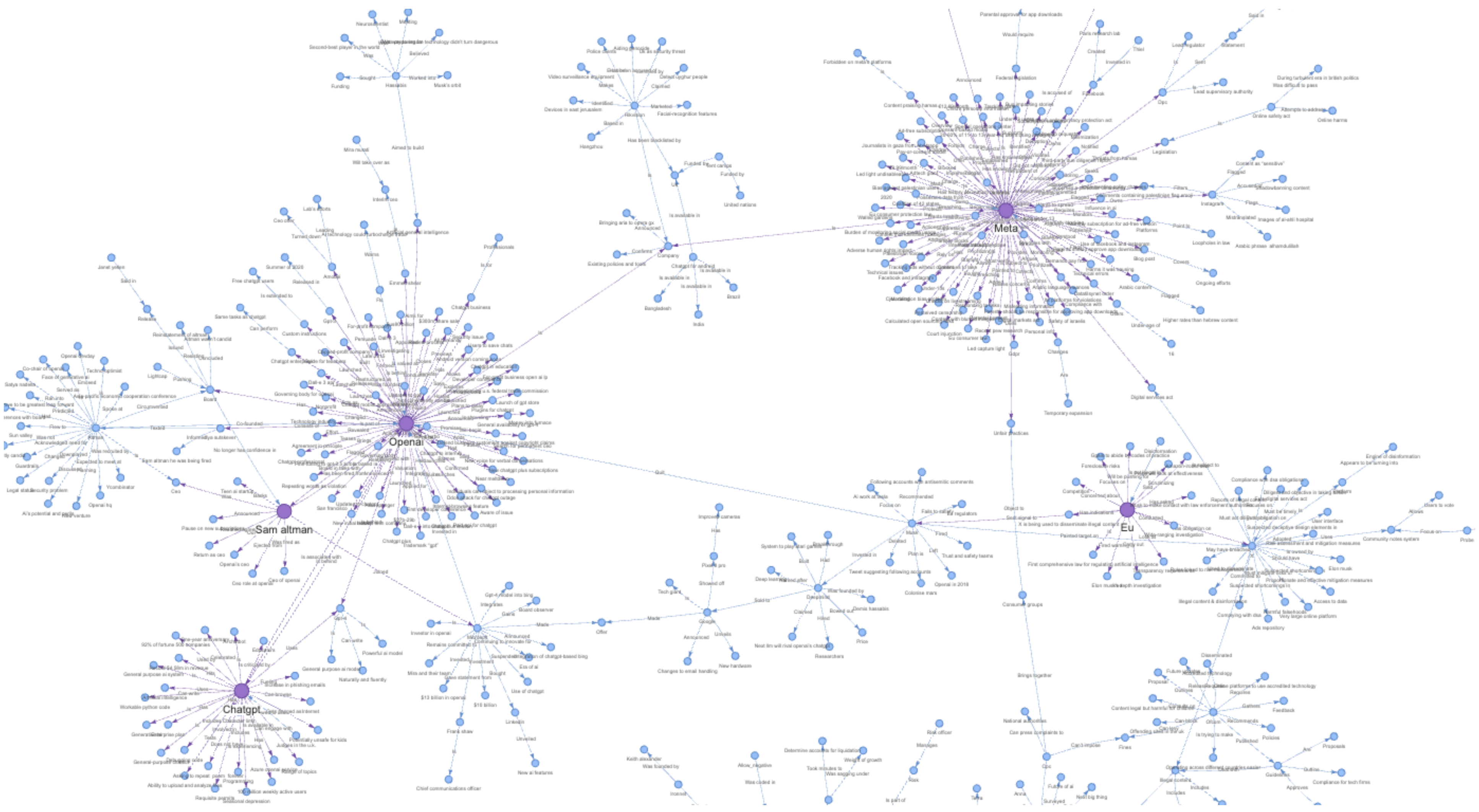
There are multiple ways to extract a knowledge graph from your data using the Property Graph Index. By default, it employs implicit extraction and free-form extraction, but you also have the option for schema-guided extraction. In schema-guided extraction, you define allowed entity types, relationship types, and their connections in a schema, and the LLM will only extract graph data that conforms to this schema. For more information on different extraction methods and combinations visit the detailed usage page.

Visualizing the Knowledge Graph

By running:

```
index.property_graph.save_graph_to_html("knowledge_graph.html")
```

We can visualize our Knowledge Graph. We can appreciate the levels of information we discussed earlier, with entities connected through relationships. Entities like Amazon, Meta, OpenAI, etc., appear in their own clusters and communities.



Comparing Baseline RAG with GraphRAG

Question 1: Who became a prominent figure in generative AI technology, notably with ChatGPT, and was recently the subject of controversy involving his departure from OpenAI, as discussed in both Fortune and TechCrunch articles?

Ground truth answer: Sam Altman

Question Type: Inference Query

Number of sources: 3

	Baseline RAG	GraphRAG
Answer	Dave Willner	Sam Altman
Correct	❌	✅

This is a good example of baseline RAG's limitations when the answer requires searching multiple sources. The query was constructed in a way that pure vector similarity did not produce the correct answer. Baseline RAG retrieved documents based on semantic similarity but failed to connect the necessary pieces of information across different articles.

- **Node Selection:** Uses synonym and keyword expansion with the LLM to generate relevant terms from the query.
- **Vector Retrieval**: Employs embeddings to find nodes in the graph that match the expanded query.
- **Graph Traversal**: Once nodes are found, it traverses adjacent nodes and relationships, effectively connecting disparate pieces of information.

In this example, our property graph focuses on the following nodes:

OpenAI ➔ Ousted ➔ CEO Sam Altman

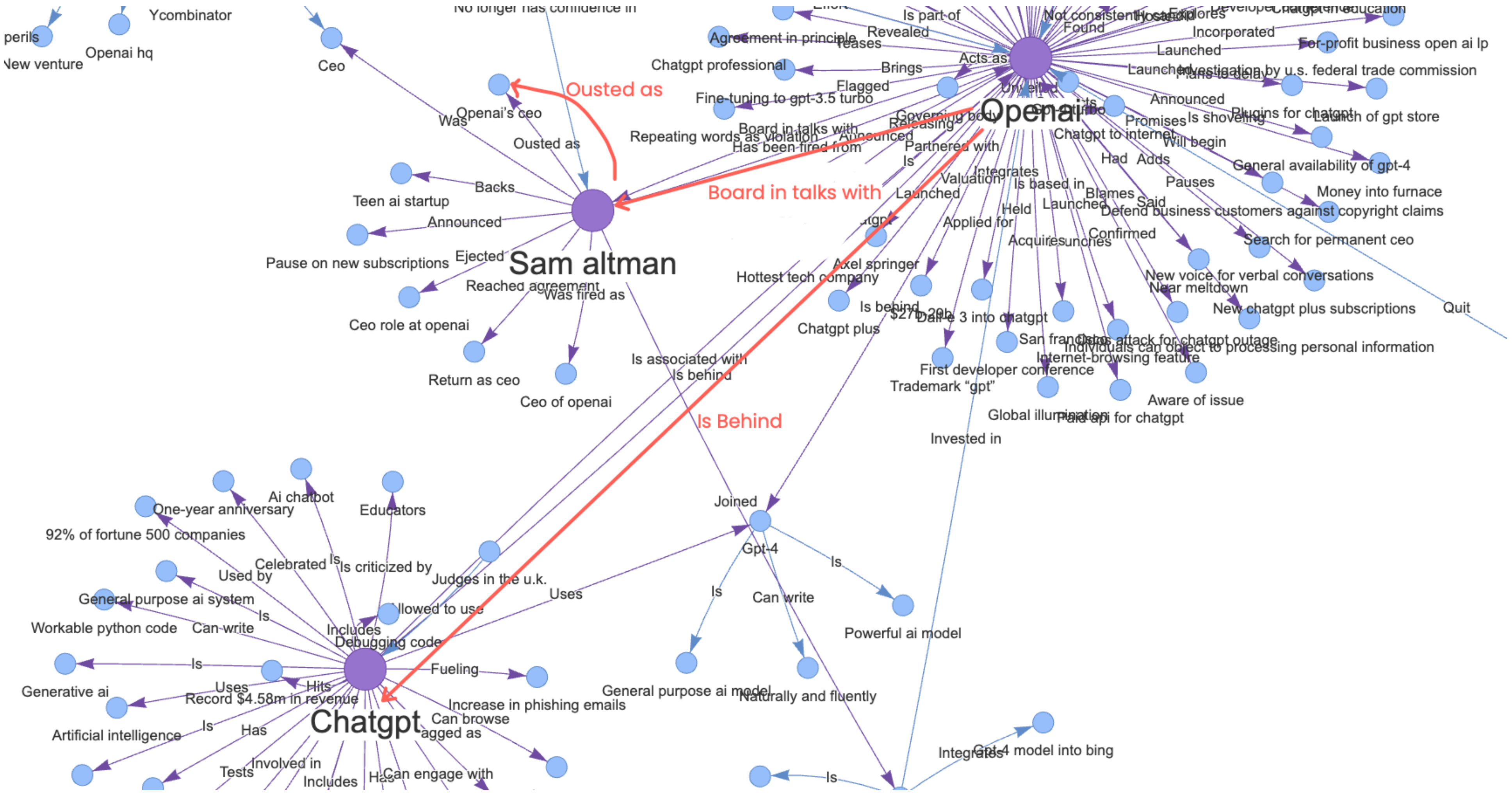
OpenAI ➔ Experienced ➔ Whirlwind four days

OpenAI ➔ Worth ➔ Just under $30 billion

Openai ➔ Pushed out ➔ Sam altman

(See our repository for the full list of retrieved nodes)

The relationship "OpenAI Ousted CEO Sam Altman" directly contains the answer. Since information across multiple articles was parsed and indexed into Entity ➔ Relationship ➔ Entity, our Property Graph could piece together the answer to the question.



Question 2: Does 'The Verge' article suggest that Sam Bankman-Fried set withdrawal permissions based on FTX's total trading revenue, while 'Fortune' and 'TechCrunch' articles focus on the jury's determination of his truthfulness and allegations of committing fraud for personal gain, respectively, without mentioning specific operational practices like withdrawal permissions?

Ground truth answer: Yes

Question Type: Comparison Query

Number of sources: 3

	Baseline RAG	GraphRAG
Answer	The 'The Verge' article does not suggest that (...)	Yes
Correct	❌	✅

In this comparison query, the answer depends on integrating information from three different sources. Baseline RAG, by depending solely on vector search, retrieves documents that are semantically similar to the query. However, even if the retrieved documents contain the answer it does not capture the relationships between the entities mentioned in different articles. Therefore, it cannot effectively determine

Question 3: Which entity is currently engaged with Amazon to address competition concerns, facilitating dialogue with consumer groups against Meta, deploying staff within its AI Office for future regulations, and has previously focused on illegal content and disinformation issues related to the Israel-Hamas war, as reported by TechCrunch?

Ground truth answer: The European Commission

Question Type: Inference Query

Number of sources: 4

	Baseline RAG	GraphRAG
Answer	Microsoft	The European Commission
Correct	❌	✅

This question is a good example of how the text in the prompt will prime our baseline RAG to find information not related to the answer. For example, keywords like “entity,” “Amazon,” “Meta,” “AI Office” are thrown in. The answer will not be found by finding the nodes in the documents that match these words but rather through understanding the semantic relationships of what is being discussed about them, showcasing the benefits of GraphRAG.

Conclusion

Our experiments show that GraphRAG outperforms Baseline RAG, especially for complex, multi-step queries that draw information from various sources. By building a knowledge graph and using entity relationships, GraphRAG handles tasks where traditional RAG models struggle. While Baseline RAG works well for simple, single-document queries, it falls short in more complex scenarios. GraphRAG’s advanced reasoning capabilities make it a strong tool for applications that require integrating and understanding information across a larger dataset.

How can we help?

Can we help you apply these ideas on your project? Send us a message! You'll get to talk with our awesome delivery team on your very first call.

Email*

you@example.com

Name (optional)

First Last

Company (optional)

What would you like to talk about?*

Let's talk!

3417 Evanston Ave N
Suite 529
Seattle, WA 98103

hello@genui.com

(206) 539-0034

Github
LinkedIn