# Knowledge Graphs and Retrieval-Augmented Generation

Adrienn Molnár

# Introduction

This project builds upon the foundations established in the previous semester's work, which evaluated Retrieval-Augmented Generation (RAG) models under OCR-induced noise. The current focus shifts towards the integration and implementation of knowledge graphs within the RAG framework to enhance retrieval accuracy and contextual understanding. Specifically, this semester's objective is to explore knowledge graphs as a standalone enhancement to RAG and assess whether their integration can independently improve retrieval and generation results.

By leveraging the structured and interconnected nature of knowledge graphs, the project aims to overcome limitations associated with vector stores, particularly in handling queries that require reasoning across multiple documents. Through this approach, the project seeks to demonstrate the unique benefits of knowledge graphs, such as semantic enrichment and cross-document reasoning.

### Problem Statement

RAG systems rely on retrieving relevant documents from a database to provide context for generating answers to user queries. In the previous semester's project, vector stores were employed for this retrieval process and tested under OCR-induced noise. While effective for local retrieval tasks, vector stores face significant challenges when addressing complex queries that require insights spanning multiple documents. Additionally, vector stores lack the capacity to represent semantic relationships between entities, limiting their ability to provide rich and interconnected contexts.

### Objectives

1. Improve retrieval accuracy by leveraging structured and semantic connections between entities.
2. Provide richer contextual understanding for queries, especially for global and multi-document questions.
3. Evaluate the independent impact of knowledge graphs on RAG performance.

The project implements a Python-based pipeline to construct, query, and evaluate knowledge graphs using the Neo4j database and OpenAI's GPT models. By integrating these elements, the study aims to determine whether knowledge graphs can enhance RAG functionality without relying on additional retrieval mechanisms.

# Knowledge Graph

A knowledge graph is a structured representation of information where entities (nodes) and their relationships (edges) are modeled to reflect real-world data. Entities can include people, places, objects, or concepts, while edges define the semantic connections between them. For example, a knowledge graph about movies might include entities such as actors, directors, and films, with edges representing relationships like "acted in" or "directed by." Knowledge graphs are widely used in various domains for reasoning, search, and natural language understanding.

### How Knowledge Graphs Work?

1. **Entity Extraction**: Entities are identified and classified from raw sources such as text, databases or APIs.
2. **Relationship Identification**: Semantic links between entities are established based on patterns, rules, or ontologies.
3. **Graph Construction**: Nodes and edges are organized into a graph, often enriched with attributes or metadata.
4. **Community Detection**: Graph algorithms group strongly connected entities, revealing patterns or clusters.
5. **Query Processing**: Traversal techniques (e.g. Cypher) enable complex queries, including subgraph extraction and inferencing.

### Techniques for Enhancing RAG with Knowledge Graphs

The following techniques are presented in Edge et al. [2024] and demonstrate how knowledge graphs can improve RAG systems:

1. **Graph Indexing for Retrieval**: Knowledge graphs serve as advanced indexes for RAG. Using vector representations of nodes and edges, retrieval becomes more accurate as graph-based similarity metrics align well with semantic relevance.
2. **Community-Based Summarization**: Modular community detection algorithms partition the graph into closely connected entities, enabling modular summarization. Summarizing these partitions ensures that the retrieved context is coherent and thematically aligned with the query.
3. **Graph-Driven Embedding Alignment**: Techniques like Node2Vec and GraphSAGE generate graph embeddings, which can be combined with text embeddings. This hybrid representation allows RAG systems to seamlessly integrate structured and unstructured data.
4. **Query Expansion via Traversal**: Graph traversal algorithms expand user queries to include related entities and relationships, enriching the context provided to the RAG model. Traversal-based expansions are particularly useful for multi-hop questions where reasoning spans several nodes and edges in the graph.

5. **Context Prioritization and Filtering**: Contexts retrieved via knowledge graphs are ranked based on graph metrics like edge weight, node centrality, and community modularity. This ensures that the most relevant and impactful data is fed into the RAG pipeline.

6. **Hierarchical Summarization**: Summaries are generated at multiple levels of granularity, from individual nodes and edges to entire communities. The hierarchical nature allows for scalable and efficient query resolution, especially for global questions requiring comprehensive coverage.

7. **Global Query-Specific Optimization**: The system can dynamically adjust graph traversal depth and context fusion strategies based on the nature of the user query, improving both precision and recall.

**How Knowledge Graphs Enhance RAG Models**

- **Improved Retrieval Accuracy**: Structured navigation improves grounding by reducing irrelevant data.
- **Contextual Enrichment**: Semantic relationships provide richer contexts for language models.
- **Cross-Domain Reasoning**: Connect disparate data sources to enable complex reasoning.
- **Noise Resilience**: The structured nature makes knowledge graphs robust against incomplete data.

# Knowledge Graph Construction

We build directly on the SQuAD dataset (Stanford Question Answering Dataset), which was also used in the previous semester's project. By reusing SQuAD as a base, this project maintains consistency while introducing knowledge graphs to enhance RAG functionality.

The SQuAD dataset contains Wikipedia articles, including context paragraphs paired with related questions and answers. To unlock the potential of this data for advanced question-answering tasks, we transform it into a graph structure stored in Neo4j. The graph not only represents the content but also encodes semantic relationships between entities and their contextual links, enabling hybrid retrieval methods that combine traditional and vector-based search techniques.

This section explains how we create the graph from the documents in the database and utilize Neo4j for storing and querying the graph.

**Data Preparation and Transformation**

The dataset is prepared for graph construction by treating each context paragraph as an individual document. Each document is assigned a unique identifier through hashing, ensuring consistent referencing.
The extracted information is transformed into a graph-compatible format, where entities are represented as nodes and their relationships as edges.

- **Document Transformation**: This process focuses on converting unstructured text from the dataset into graph-ready structures. Each document is analyzed to extract key entities, such as names, locations, or technical terms, leveraging an LLM-based transformer. These entities are represented as nodes in the graph, while semantic relationships, such as "is part of" or "related to," are identified using dependency parsing and contextual analysis to form edges between nodes. Each node and edge is enriched with metadata, such as labels and relevance scores, to capture the underlying semantics and structure of the document.

- **Embedding Generation**: Semantic embeddings are generated for each document and entity using a pre-trained language model. These embeddings capture the contextual meaning of the text and are stored as node properties, enabling similarity-based queries.

- **Neo4j Integration**: We use Neo4j to organize and retrieve data within the knowledge graph. Nodes represent entities or document segments, while edges capture relationships between them, with both enriched by properties such as semantic embeddings and metadata for advanced querying. Full-text indexing on entity text enables fast keyword-based searches across the graph, allowing to locate relevant nodes and relationships efficiently using textual queries. By combining full-text indexing with the graph's structured relationships and semantic embeddings, Neo4j delivers a robust hybrid retrieval system that supports both precise and semantically rich queries, enhancing question-answering capabilities.

The figure 1 on the next page demonstrates how a subgraph from the knowledge graph can be visualized using Neo4j Bloom. Neo4j Bloom provides an intuitive interface to explore and interact with the graph, where **purple nodes** represent **documents**, and **yellow nodes** represent **entities** such as *"Rollo"* and *"Normandy"*. The edges denote semantic relationships between these entities, such as *"ARRIVAL"* or *"RAIDED AND SETTLED"*.

# Experimental Setup

This section outlines three initial approaches to integrating structured knowledge graph retrieval and unstructured vector-based document retrieval into the RAG framework. Both approaches explore how combining these methodologies can enhance contextual understanding and answer accuracy.

For each question in the SQuAD database, a specific document (in our case paragraph) is marked as the source from which the answer should be derived.

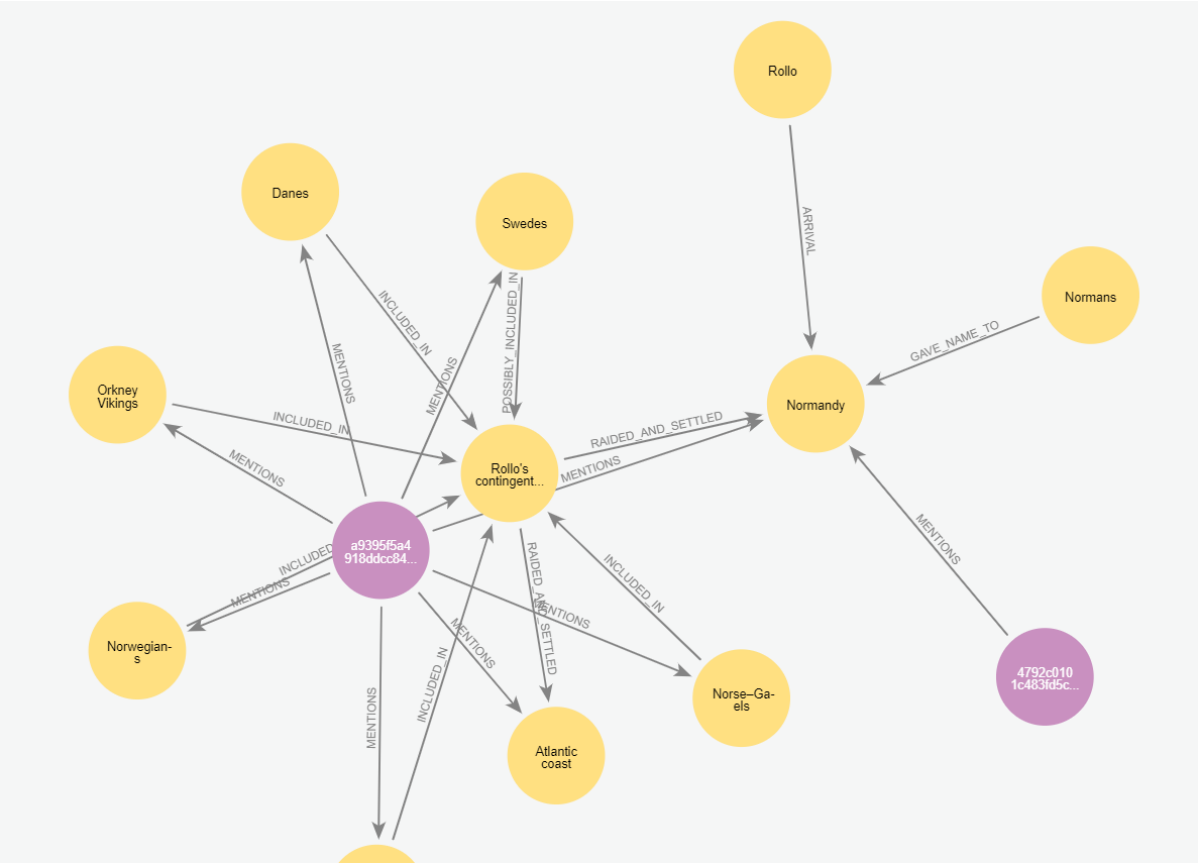


Figure 1: Example of a knowledge graph subgraph visualized in Neo4j Bloom. The nodes represent entities and documents, while the edges denote relationships.

**Evaluation Metric**
The primary evaluation metric is recall, defined as follows:

- **Recall = 1**: If the marked document is among the top 3 documents retrieved.
- **Recall = 0**: If the marked document is not included in the top 3 retrieved documents.

The metric assesses the system's ability to include the correct context in its retrieved results, which is critical for accurate answer generation.

The methodology combines **structured retrieval** from a Neo4j knowledge graph and **unstructured retrieval** using a Neo4j vector store to enhance the RAG pipeline. This **hybrid approach** leverages the semantic relationships captured in the knowledge graph and text-based similarity to retrieve relevant contextual information for a given query.

**Previous Results**
The results from last semester's project, using a vector store-based retrieval framework, serve as a baseline for evaluating the current retrieval-augmented generation system. The evaluation was conducted on a set of 237 questions, and the outcomes were as follows:

- **Correct Answer and Correct Context:** 169 instances (71%)
- **Correct Answer but Incorrect Context:** 14 instances (6%)
- **Correct Context, Incorrect Answer:** 33 instances (14%)
- **Both Answer and Context Incorrect:** 21 instances (9%)

The goal of the current experiment is to outperform these results by implementing a hybrid retrieval approach.

## First Approach

In this approach, retrieval is performed sequentially, prioritizing structured knowledge graph data before incorporating unstructured document retrieval.

1. **Structured Retrieval**:
   The structured retrieval process aims to retrieve the most relevant documents associated with entities identified in the user query.

   - **Entity Extraction:** Key entities, such as people, locations, or organizations, are extracted from the query.
   - **Node Matching:** These entities are matched against the knowledge graph to locate corresponding nodes.
   - **Relationship Traversal:** Relationships, such as *"MENTIONS"*, are traversed to find connected document nodes that reference the entities.

   The results include document nodes and their contextual connections to the query's entities. The rationale behind this process is to ensure that the most relevant documents to the query are retrieved by leveraging their direct associations with the extracted entities. See Appendix A for a detailed example.

2. **Unstructured Retrieval**: After structured retrieval the query is processed through a Neo4j vector store, which performs similarity-based searches on document embeddings generated using OpenAI's text-embedding-ada-002 model. Document embeddings are compared to the query embedding to retrieve the top-k documents ranked by semantic similarity.

3. **Result Integration**: The final context contains 3 documents, with priority given to those retrieved through structured retrieval. Unstructured data complements the structured results by providing additional textual details.

4. **Answer Generation**: The final context, consisting of the top 3 relevant documents, is formatted into a predefined prompt that instructs the GPT model (gpt-3.5-turbo) to generate answers based solely on the provided context. The generated answers are evaluated by comparing them to the given answers in the SQuAD database and verifying whether the correct document ID is included in the retrieved context.

### Results

The results of the first approach, evaluated using 237 questions, are as follows:

- **Correct Answer and Correct Context:** 139 (59%)
- **Correct Answer but Incorrect Context:** 37 (16%)
- **Correct Context but Incorrect Answer:** 23 (10%)
- **Both Answer and Context are Incorrect:** 38 (16%)

These results fall below expectations when compared to the previous semester's baseline. The documents retrieved from the structured retrieval might not always be the most relevant, as they are simply the documents that mention the given entity and may not contain the most critical information related to the query. In the second approach, we plan to address this issue by dropping these documents.

## Second Approach

The second approach builds on the limitations identified in the first approach and introduces a refinement to improve the relevance of the retrieved context. Unlike the first approach, which prioritized documents retrieved from structured retrieval, this approach focuses on leveraging relationships from the knowledge graph and unstructured vector-based documents. By excluding the structured documents, the goal is to enhance the quality of the retrieved context and, consequently, the accuracy of generated answers.

**Retrieval Adjustments**: Structured retrieval is limited to extracting relationships from the knowledge graph instead of retrieving full documents associated with the entities. The unstructured retrieval process remains the same.

The final context consists of:

- **Retrieved relationships** from the structured retrieval.
- **Top 3 documents** retrieved through vector-based similarity search in the unstructured retrieval step.

### Results

The results of the second approach, evaluated using 237 questions, are as follows:

- **Correct Answer and Correct Context:** 186 (78%)
- **Correct Answer but Incorrect Context:** 4 (2%)
- **Correct Context but Incorrect Answer:** 30 (13%)
- **Both Answer and Context are Incorrect:** 17 (7%)

This approach outperforms the first by focusing on the most relevant elements of the query. By excluding documents from structured retrieval, which may only loosely reference the entities, and relying on relationships from the knowledge graph, the retrieved context becomes more targeted. Combining these relationships with vector-based documents ensures that the final context captures both semantic connections and detailed textual information. This refinement minimizes irrelevant data and improves the relevance of the retrieved context, leading to higher answer accuracy and recall.

## Third Approach

The third approach introduces a hybrid retrieval methodology that combines vector-based retrieval and full-text search within the Neo4j knowledge graph. This approach leverages the strengths of both techniques to ensure that the retrieved context is both semantically relevant and textually precise.

1. **Embedding-Based Retrieval:** The query is converted into an embedding using OpenAI's embedding model. A vector similarity search is performed using the Neo4j vector index to retrieve nodes with high semantic similarity to the query. This ensures that nodes related to the meaning of the query are prioritized. This is similar to the unstructured retrieval in the previous approaches.

2. **Full-Text Search:** The query is also processed using a full-text index within Neo4j, which retrieves nodes matching the query text semantically or via keywords. This complements the embedding-based retrieval by capturing relevant nodes that may not have strong embedding similarity but are textually aligned with the query.

3. **Combining Results:** Results from the vector-based and full-text searches are combined into a single list. Scores from both retrieval methods are normalized to ensure consistency and comparability. The combined list is ranked by relevance scores, with the top-3 nodes selected for inclusion in the final context.

4. **Context Integration:** The selected best 3 results are combined into a single, cohesive context. Additional semantic relationships between entities (retrieved using graph traversal) are included to enrich the context further.

### Results and Analysis

The results of the third approach, evaluated using 237 questions, are as follows:

- **Correct Answer and Correct Context:** 192 (81%)
- **Correct Answer but Incorrect Context:** 5 (2%)
- **Correct Context but Incorrect Answer:** 29 (12%)
- **Both Answer and Context are Incorrect:** 11 (5%)

The third approach demonstrates the best performance among all tested methods, achieving a significant improvement in the rate of correct answers with correct context (81%). When combining this with the cases of incorrect answers but correct context (12%), the overall correct context retrieval rate reaches 93%, highlighting the effectiveness of the retrieval strategy.

This improvement can be attributed to the use of **comprehensive retrieval**, which combines embedding-based similarity and full-text search. This hybrid approach ensures that the most relevant nodes are identified by leveraging semantic and textual relevance. The normalization of scores enables a fair comparison between the two methods, leading to a more precise and contextually relevant selection.

# Conclusion

This project successfully addressed the initial problem of improving context retrieval for Retrieval-Augmented Generation systems, specifically for queries requiring reasoning across multiple documents. By integrating knowledge graphs, the project demonstrated significant advancements over traditional vector-based retrieval methods, which lacked the capacity to capture semantic relationships and struggled with cross-document reasoning.

Through three iterative approaches, the project showcased how knowledge graphs enhance the context retrieval process. The first approach, which combined structured retrieval from the knowledge graph with vector-based unstructured retrieval, provided a foundation but revealed limitations in relying solely on documents mentioning the query entities.

Building on this, the second approach prioritized semantic relationships from the knowledge graph over entity-referenced documents, combining them with vector-based retrieval. This refinement significantly improved the relevance and quality of the retrieved context.

The third approach achieved the best results by unifying vector-based retrieval and full-text search into a comprehensive hybrid framework. This approach combined the strengths of embedding-based similarity, textual keyword relevance, and enriched semantic relationships. By normalizing and ranking results, it delivered an overall correct context retrieval rate of 93% and improved answer accuracy.

The project accomplished a substantial improvement in context retrieval by integrating knowledge graph semantics and hybrid retrieval methodologies. This solution directly addresses the initial problem statement, overcoming the challenges of traditional vector stores in handling complex, multi-document queries. By providing interconnected knowledge alongside relevant texts, the system ensures not only more accurate retrieval but also enriched contextual understanding, ultimately enhancing the overall performance of RAG systems.

These findings underscore the transformative potential of hybrid retrieval frameworks combining knowledge graphs and vector-based methods. The outcomes of this study provide a robust foundation for future research into context retrieval techniques and their broader applications in domains requiring deep semantic reasoning and contextual precision.

# Appendix

## A Structured Retrieval Example

**Query**: *"Who did Rollo sign the treaty of Saint-Clair-sur-Epte with?"*

Extracted terms:

- Entity: `'Rollo'`
- Entity: `'Saint-Clair-sur-Epte'`

Retrieved relationships:

- `"Rollo - ARRIVAL -> Normandy"`
- `"treaty of Saint-Clair-sur-Epte - BETWEEN -> King Charles III of West Francia"`
- `"treaty of Saint-Clair-sur-Epte - BETWEEN -> famed Viking ruler Rollo"`

## References

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.