Knowledge Graph Builder – Constructing a Graph from Arbitrary Text Using an LLM

Andreas Benno Kollegger
Neo4j Inc, Cambridge, UK

Alexander Erdl
Neo4j Inc, Munich, DE

Michael Hunger
Neo4j Inc, Dresden, DE

Abstract

Knowledge graphs improve many information retrieval tasks over structured and unstructured data. However, knowledge graph construction can be challenging even for domain experts. The Knowledge Graph Builder is an application incorporating advanced techniques for deriving a knowledge graph from unstructured data using an LLM.

2012 ACM Subject Classification Information systems → Network data models

Keywords and phrases Knowledge Graph, Lexical Graph

Digital Object Identifier 10.4230/LIPIcs.GD.2024.61

Category Software Abstract

Supplementary Material *Software (Source Code)*: `https://github.com/neo4j-labs/llm-graph-builder`

Acknowledgements We want to thank the engineering team at Neo4j for their pioneering work on knowledge graph construction using an LLM.

1 Knowledge graph builder – Summary

A knowledge graph is a self-descriptive data structure that represents real-world information using the elements of a graph by storing values on both nodes and relationships. Deriving a knowledge graph from arbitrary text produces a graph with two complementary sections: a lexical graph that preserves the structure of the source text and a domain graph that distills the entities and relationships described by the source text. The Knowledge graph builder is an application which constructs a viable knowledge graph from unstructured data.

2 Knowledge graph construction

Knowledge graph construction follows a 3 phase process.

Content decomposition

1. A document node is created to represent the text source.
2. The text content is split into chunks, which are stored as graph nodes.
3. Chunks are connected to each other and the document node.

Content cross-linking

1. Highly similar chunks are connected with a SIMILAR relationship to form a K-Nearest Neighbors graph.
2. Embeddings are computed, stored as properties on the chunk nodes, and backed by a vector index.

Entity extraction

1. An LLM is used to process each chunk, to identify entities and their inter-relationships.
2. Entities are stored in the graph and connected to the originating chunks.
3. Entity relationships are stored in the graph, creating a rich network of concepts.

The entity extraction can be unbounded, allowing the LLM to extract and classify any entities and relationships it finds, or it can be constrained to conform with a pre-determined collection of entity and relationship types.

3 Knowledge graph model

The resulting knowledge graph is amenable to multiple GraphRAG access patterns.

3.1 Lexical graph

The lexical graph section preserves the implicit structure of the original source text, including the sequential flow of text and also any hierarchies up to the enclosing document itself.

Chunks are part of a document

```
(a:Chunk)-[:PART_OF]->(d:Document)
```

Chunks form a linked list of sequential text

```
(a:Chunk)-[:NEXT_CHUNK]->(b:Chunk)
```

Neighborhoods of similar chunks

```
(a:Chunk)-[:SIMILAR]->(b:Chunk)
```

3.2 Domain graph

Entities are connected to chunk that mentions them

```
(a:Chunk)-[:HAS_ENTITY]->(e1)
```

The `(e1)` node matches any entities which have been extracted, with corresponding labels determined by the LLM.

Entities related to each other

```
(e1)-[r]->(e2)
```

The `[r]` relationship will be determined by the LLM.

4 Conclusion

This brief description of the operation of the Knowledge Graph Builder application is just an introduction to the construction of knowledge graphs. More techniques are available to continue elaborating upon and enriching the graph, and connecting to structured data. The broad topic of GraphRAG then leverages the resulting knowledge graph for use with Generative AI applications.