

# Building and Analyzing a Knowledge Graph from Wikipedia Data

Sami Taider  
84401097

December 2, 2024

## 1 Introduction

This report documents the process of constructing a knowledge graph using data from Wikipedia. The goal was to extract entities and relationships from the Wikipedia page on *Machine Learning* and store them in a graph database for analysis. The project explores various approaches to content extraction and the challenges faced when working with large textual datasets.

## 2 Process Followed

The process followed for generating the knowledge graph can be broken down into several key steps:

1. **Data Extraction:** The content of the Wikipedia page titled "Machine Learning" was retrieved using the `wikipedia-api` Python package.
2. **Text Preprocessing:** The content was processed using `spaCy` to remove stop words and perform sentence segmentation.
3. **Entity and Relationship Extraction:** The preprocessed text was fed into a custom prompt used with the GPT-3 model. The model was tasked with identifying entities and relationships from the content.
4. **Graph Database Population:** The resulting entities and relationships were converted into Cypher queries, which were then executed against a Neo4j graph database to create the nodes and relationships.

### 3 Approaches Tried

Two approaches were tested for processing the Wikipedia content before feeding it into the entity-extraction model.

- **Approach 1:** The content was split into individual sentences, and stop words were removed. This approach aimed to isolate key pieces of information and improve clarity.
- **Approach 2:** The content was processed as a whole without sentence splitting or stop word removal, allowing the model to consider context from longer passages.

Upon analysis, the second approach (processing the content without splitting into sentences or removing stop words) yielded better results. This was likely because it allowed the model to capture more context and relationships between entities across sentences. The first approach, while effective for shorter pieces of information, resulted in fragmented extraction with lower accuracy.

### 4 Challenges Faced and Solutions

Several challenges were encountered during the project, and solutions were implemented as follows:

#### 4.1 Token Limits for Large Wikipedia Pages

Wikipedia pages can be very large, and the token limits imposed by the GPT-3 model meant that it was not possible to process the entire page at once. To overcome this, the content was split into smaller parts, each of which was processed individually. The `split_content` function was used to divide the content into parts, ensuring that no part exceeded the token limit.

#### 4.2 Hallucinations from the LLM

Another challenge was that the language model sometimes generated hallucinated or inaccurate relationships between entities. To mitigate this, prompt engineering was employed. The context prompt was carefully crafted to ensure that the model understood the task of extracting accurate entities and relationships.

## 5 Key Findings from the Knowledge Graph

The knowledge graph constructed from the Wikipedia page on *Machine Learning* contained various entities and relationships, including:

- Nodes representing key concepts such as algorithms, models, and applications.
- Relationships indicating how these concepts were related, such as `is_a`, `used_in`, and `developed_by`.

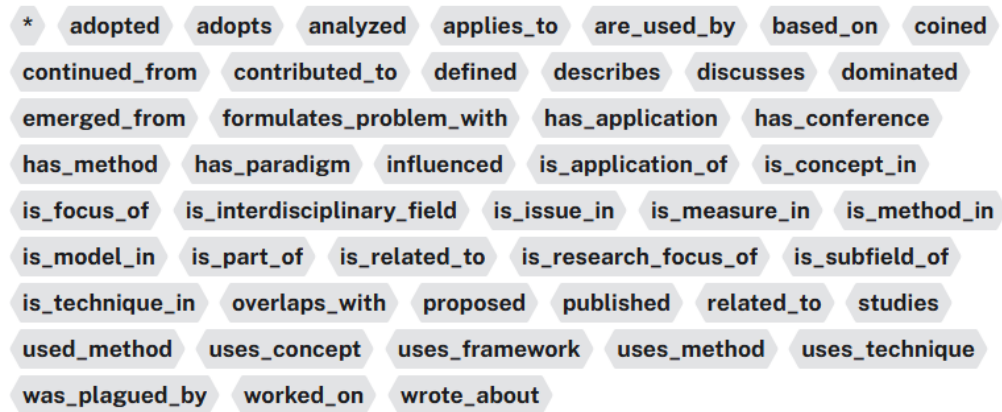
The final graph provides a rich representation of the field of machine learning, showing connections between various subdomains and real-world applications.

### Database information

Nodes (87)



Relationships (89)



Property keys

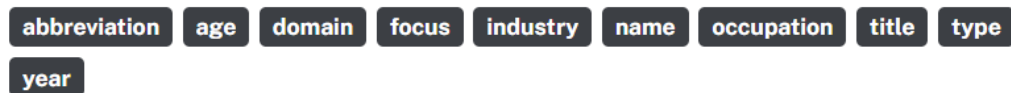


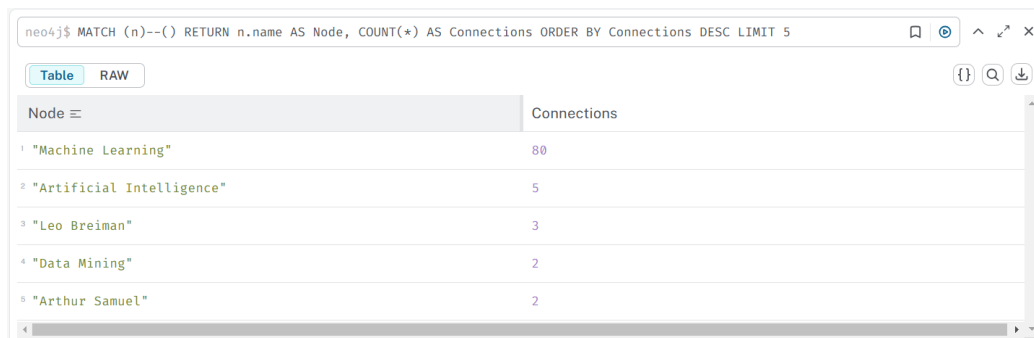
Figure 1: Graph database information.

## 6 Cypher Query Results

Several Cypher queries were executed to analyze the graph:

### 6.1 Most Connected Entities

The query to find the most connected entities returned the following results:



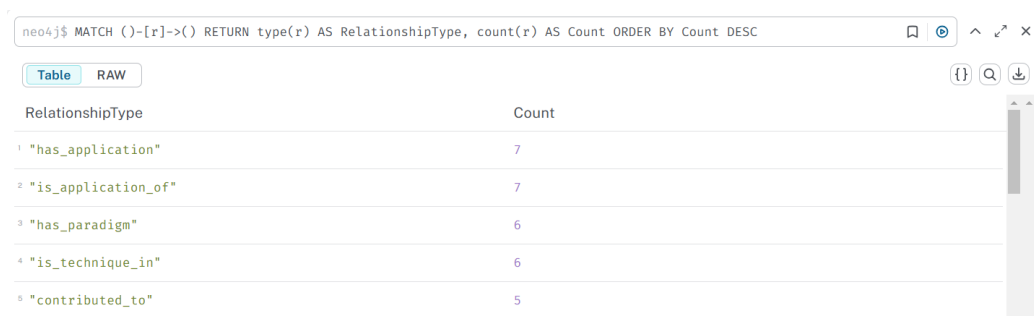
```
neo4j$ MATCH (n)--() RETURN n.name AS Node, COUNT(*) AS Connections ORDER BY Connections DESC LIMIT 5
```

Node	Connections
1 "Machine Learning"	80
2 "Artificial Intelligence"	5
3 "Leo Breiman"	3
4 "Data Mining"	2
5 "Arthur Samuel"	2

Figure 2: Result of Cypher query to find the most connected entities.

### 6.2 Count Relationships by Type

The query to count the relationships by type returned the following results:



```
neo4j$ MATCH ()-[r]->() RETURN type(r) AS RelationshipType, count(r) AS Count ORDER BY Count DESC
```

RelationshipType	Count
1 "has_application"	7
2 "is_application_of"	7
3 "has_paradigm"	6
4 "is_technique_in"	6
5 "contributed_to"	5

Figure 3: Result of Cypher query to count relationships by type.

### 6.3 Count Nodes by Type

The query to count nodes by type returned the following results:

```
neo4j$ MATCH (n) RETURN labels(n)[0] AS NodeType, count(n) AS Count ORDER BY Count DESC
```

	NodeType	Count
1	"Field"	24
2	"Concept"	12
3	"Person"	11
4	"Application"	8
5	"Method"	7
6	"Approach"	7
7	"Technique"	6
8	"Publication"	3
9	"Paradigm"	2

Figure 4: Result of Cypher query to count nodes by type.

## 7 Final Graph Visualization

The visualization of the final knowledge graph is shown below:

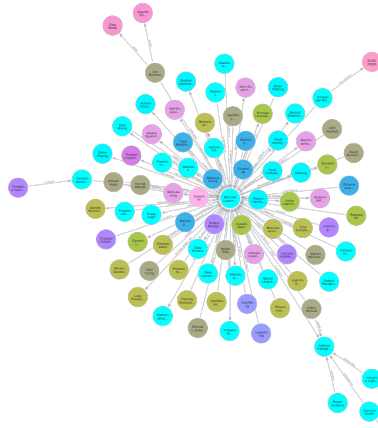


Figure 5: Visualization of the final knowledge graph.

## 8 Conclusion

This project demonstrates the potential of using large language models and graph databases to automatically extract and represent knowledge from unstructured text. The challenges of token limits, hallucinations, and con-

tent processing were addressed through careful engineering and testing. The resulting knowledge graph provides valuable insights into the structure of machine learning, showing how different concepts and techniques are interconnected.