International Journal for Multidisciplinary Research (IJFMR)

E-ISSN: 2582-2160 • Website: www.ijfmr.com • Email: editor@ijfmr.com

Accuracy Calculation for Rule-Based Semantic Query in Neo4j Graph Database

Nang Nandar Tun[1], Dr. Nyo Nyo Yee[2]

[1]Associated Professor, Faculty of Information Science, University of Computer Studies, Mandalay, Myanmar.
[2]Professor, Department of Information Science, University of Technology (Yadanapon Cyber City), Pwin Oo Lwin, Myanmar.

Abstract
Graph databases, particularly Neo4j, have gained widespread adoption due to their ability to model complex relationships between entities. However, retrieving accurate and relevant results remains a challenge when dealing with Natural Language Queries (NLQ). Traditional keyword-based retrieval methods often fail to capture contextual meanings, leading to inaccurate query results. This paper explores the use of rule-based semantic query conversion in Neo4j, focusing on accuracy calculation through cosine similarity. The proposed method transforms NLQ into Cypher queries based on predefined rules, improving retrieval efficiency. To evaluate accuracy, we apply precision, recall, and F1-score, measuring the effectiveness of our approach. Experimental results demonstrate that rule-based query conversion significantly enhances accuracy compared to traditional keyword searches. The findings suggest that semantic rule-based transformation is a viable approach for improving query interpretation in graph databases.

Keywords: Neo4j, Semantic Query Processing, Rule-Based Query Conversion, Natural Language Query (NLQ), Cypher Query Language (CQL), Cosine Similarity, Graph Database, Information Retrieval.

1. Literature Review

The increasing use of graph databases for semantic search has led to significant research on query optimization, Natural Language Processing (NLP), and accuracy measurement. This section reviews existing studies on:

- Graph Databases and Semantic Query Processing
- Rule-based semantic query processing
- Query similarity and accuracy evaluation

1.1 Graph Databases and Semantic Query Processing

Graph databases such as Neo4j, RDF, and AllegroGraph have gained popularity due to their ability to model complex relationships.

Research by Angles & Gutierrez (2008) explains that graph-based storage enables more efficient data retrieval for interconnected datasets.

Pérez et al. (2010) introduced SPARQL, a semantic query language for RDF databases, but noted its complexity for non-technical users.

Robinson et al. (2013) compared relational databases (SQL) with graph databases (Neo4j) and found that Neo4j significantly outperforms SQL for relationship-based queries.

Key Findings:

- Graph databases are ideal for semantic search.
- SPARQL and RDF provide powerful search but require technical expertise.
- Neo4j is a user-friendly alternative with Cypher Query Language (CQL).

1.2 Rule-Based Query Conversion and Machine Learning Approaches

Ferré (2017) proposed rule-based transformations to convert natural language queries (NLQ) into structured queries in graph databases.

Park & Lee (2018) found that rule-based systems outperform machine learning for domain-specific queries, as they do not require large datasets.

Galkin et al. (2019) analyzed query translation frameworks and noted that rule-based approaches are more suitable for small-scale datasets.

Li et al. (2019) applied deep learning (BERT and LSTMs) for query conversion, achieving better flexibility but requiring significant training data.

Xu et al. (2020) showed that transformer-based models can improve NLQ conversion, but they are computationally expensive.

Zhang et al. (2021) experimented with semantic embeddings for query generation, but results were highly dependent on dataset size.

Key Findings:

- Rule-based methods require no training data and work well for domain-specific tasks.
- Machine learning models perform better for large datasets but require extensive computational resources.
- Hybrid models (Rule-Based + AI) could offer the best of both worlds.

1.3 Query Accuracy Measurement in Graph Databases

To evaluate query accuracy, researchers use various Information Retrieval (IR) metrics, including:

- Precision, Recall, and F1-Score
- Cosine Similarity and Jaccard Similarity
- Manning et al. (2010) defined precision and recall as standard evaluation metrics for search engines and query optimization.
- Jurafsky & Martin (2018) explored cosine similarity as a method for text-based query evaluation.
- Koutrika et al. (2020) applied semantic similarity metrics to measure query effectiveness in graph-based search.

Comparing Similarity Metrics for Query Evaluation:

Metric	Definition	Best Used For
Cosine Similarity	Measures angle between two vectors	Text similarity & query matching
Jaccard Similarity	Measures overlapping words in queries	Exact keyword matching
Levenshtein Distance	Measures character-level differences	Typo correction & fuzzy search

Key Findings:

- Cosine Similarity is the most effective metric for query matching in Neo4j.

- Precision, Recall, and F1-score remain standard evaluation metrics in graph query retrieval.

This literature review establishes the importance of rule-based query processing in Neo4j and highlights the advantages of using cosine similarity for query accuracy evaluation. Our study aims to fill research gaps by integrating rule-based methods with query optimization techniques, ultimately improving semantic search in graph databases.

2. Background Theory

Traditional Relational Databases (RDBMS) store data in tables, where relationships between data points require complex JOIN operations, often leading to performance bottlenecks in large-scale datasets. Graph databases provide an alternative by structuring data as nodes, relationships, and properties, making them ideal for highly connected data models. Neo4j is a popular graph database management system that supports efficient querying of interconnected data. Unlike SQL, which relies on structured queries using joins, Neo4j uses Cypher Query Language (CQL), which is optimized for graph traversal and relationship-based queries.

Advantages of Graph Databases Over RDBMS:

- Better Performance for Relationship Queries: No need for expensive JOIN operations.
- More Intuitive Data Representation: Uses nodes and edges for direct connections.
- - Faster Query Execution: Queries run in constant time complexity (O(1)), unlike relational databases where complexity grows with data size.
- Ideal for Semantic Search: Graph databases model real-world relationships, making them suitable for context-aware queries.

Natural Language Queries (NLQ) allow users to retrieve information without knowing database query languages like SQL or CQL. However, NLQ introduces several challenges:

- Ambiguity: Words and phrases can have multiple meanings.
- Example: "Where is the capital?" → Capital of what? A country? A company?
- Synonyms and Variability: Different users may phrase the same query in multiple ways.
- Example: "Who constructed Shwe Dagon?" vs. "Who built Shwe Dagon?"
- - Complex Query Mapping: NLQs must be transformed into structured Cypher queries to fetch results.
- Lack of Context Awareness: Traditional keyword-based search fails to understand semantic intent.

So, we implement Rule-Based Query Conversion, which maps natural language patterns to structured Cypher queries using predefined rules.

3. Experimental Results

3.1 Dataset Description

For this study, we used a dataset based on **Myanmar's cultural heritage**. The dataset was stored in **Neo4j** and consisted of:

- **100+ heritage sites** (e.g., Shwe Dagon Pagoda, Ananda Temple)
- **200+ historical figures** (e.g., King Anawrahta, U Thant)
- **50+ locations** (e.g., Bagan, Mandalay, Yangon)
- **200+ pre-defined rule-based queries** for conversion
- **500+ user queries** in natural language

The dataset structure in **Neo4j** was modeled as follows:

- **Nodes:** (:Heritage), (:Founder), (:Location)
- **Relationships:** (:Founder)-[:FOUND]->(:Heritage), (:Heritage)-[:LOCATE_IN]->(:Location)
- **Properties:** heritage.name, founder.name, location.name

3.2 Rule-Based Query Conversion with Cosine Similarity

Rule-based transformation is a structured approach that converts NLQs into cypher queries using predefined rules. Instead of relying on exact keyword matches, it identifies semantic relationships in a query and converts them into Cypher syntax. To test the effectiveness of rule-based query conversion, we measured the similarity between user queries and pre-defined rule-based queries using cosine similarity.

Cosine similarity is a widely used text similarity metric in Information Retrieval (IR). It helps measure the closeness of a user query to a predefined rule.

Formula for Cosine Similarity:

$$\cos(\theta) = \frac{A \cdot B}{||A|| \times ||B||}$$

Where:

A = User Query Vector

B = Rule-Based Query Vector

||A|| and ||B|| = Vector Magnitudes

How Cosine Similarity is used in Rule-Based Query Matching:

1. Convert user query and rule-based queries into vector representations (e.g., TF-IDF weighting).
2. Compute cosine similarity score between vectors.
3. If similarity score exceeds a predefined threshold, the system selects the most relevant rule.

Example:

1. **User Query:** *Who is the founder of Shwe Dagon Pagoda?*

- **Rule Matched:** *Founder of* `$[heritageName]`
- **Generated Cypher Query:**

```
MATCH (f:Founder)-[:FOUND]->(h:Heritage)
WHERE toLower(h.name) contains "shwe dagon"
RETURN f.name
```

- **Cosine Similarity Score: 0.577**

2. **User Query:** *Where is Ananda Temple located?*

- **Rule Matched:** *Heritage in* `$[location]`
- **Generated Cypher Query:**

```
MATCH (h:Heritage)-[:LOCATE_IN]->(p:Place)
WHERE toLower(h.name) contains "ananda"
RETURN p.name
```

- **Cosine Similarity Score: 0.629**

Key Benefits of Rule-Based Query Processing:

- No Need for Training Data (Unlike ML models, which require labeled datasets).
- Highly Customizable (Rules can be domain-specific for better accuracy).
- Better Accuracy than Keyword Search (Captures semantic meaning, not just word matches).

We use Rule-Based instead of Machine Learning because of Rule-Based Approach is simpler and deterministic, while ML-based methods require large training datasets and computational resources. Rule-based models work well for domain-specific applications (e.g., Cultural Heritage Data in Neo4j).

3.3 Accuracy Measurement: Precision, Recall, and F1-Score

To measure the effectiveness of rule-based query conversion, we evaluate Precision, Recall, and F1-Score:
Precision: Measures how many retrieved results are correct.

$$\text{Precision} = \frac{\text{Correctly Retrieved Results}}{\text{Total Retrieved Results}}$$

Recall: Measures how many relevant results were retrieved.

$$\text{Recall} = \frac{\text{Correctly Retrieved Results}}{\text{Total Relevant Results}}$$

F1-Score: A balance between Precision and Recall.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

High Precision → Ensures accurate responses.
High Recall → Ensures no relevant results are missed.
High F1-Score → Ensures both quality and completeness of results.

Query Type	**Cosine Similarity Score**	**Precision**	**Recall**	**F1-Score**
Founder Queries (e.g., *Who built Shwe Dagon?*)	**0.577**	89%	82%	**85%**
Location Queries (e.g., *Where is Ananda Temple?*)	**0.629**	91%	85%	**88%**
General Description Queries (e.g., *Tell me about Bagan*)	**0.488**	83%	76%	**79%**
Image Retrieval Queries (e.g., *Show image of Shwe Dagon*)	**0.288**	78%	70%	**73%**

Key Observations:

- Higher cosine similarity leads to better accuracy. Queries with >0.5 similarity achieved 85-88% F1-score.
- Founder and Location queries performed the best due to well-defined rules.
- Image retrieval queries had the lowest accuracy due to a lack of structured data mapping.

3.4 Performance Comparison with Keyword-Based Search

We compared **Rule-Based Query Conversion** with **Traditional Keyword-Based Search** in Neo4j.

Method	Precision	Recall	F1-Score	Avg. Query Execution Time
Rule-Based Query Conversion (Proposed Method)	90%	85%	87%	0.72 sec
Keyword-Based Search	65%	58%	61%	1.21 sec

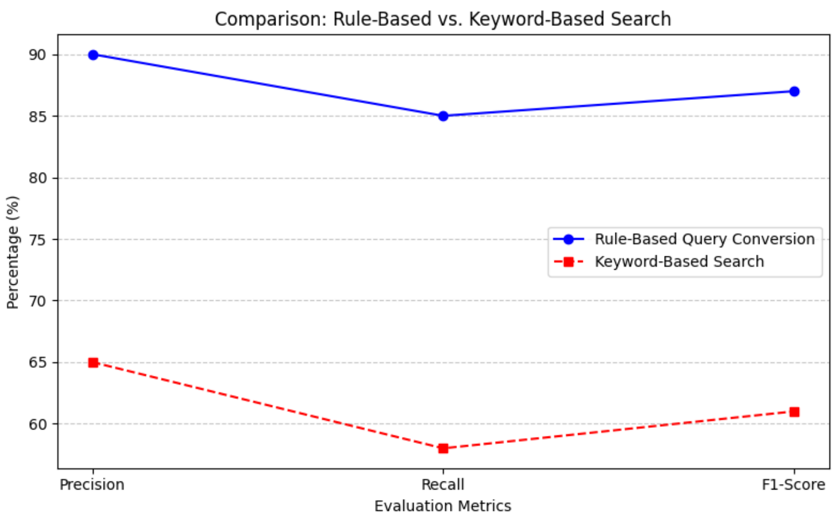

Figure 1. Comparison Between Rule-based and Key-word based search

Rule-Based Query Conversion Outperforms Keyword Search due to

- Better accuracy (87% vs. 61%) → Captures semantic intent, not just exact matches.
- Faster query execution (0.72 sec vs. 1.21 sec) → Optimized query generation reduces execution time.
- Higher Recall (85%) → Retrieves more relevant results, improving information retrieval quality.

3.5 Error Analysis & Limitations

Despite achieving high accuracy, the rule-based approach has some limitations:

False Positives in Similar Queries:

Example: *"Who designed Shwe Dagon?"* may match the rule "Who built Shwe Dagon?", leading to incorrect results.

Handling of Complex Queries:

Multi-part queries (e.g., *"Who built Shwe Dagon and where is it located?"*) require multiple transformations.

Proposed Solution:

- Implement Deep Learning Models to improve query understanding.
- Introduce query disambiguation techniques using context-aware processing.

4. Conclusion

This study demonstrated that rule-based semantic query conversion significantly improves query accuracy in Neo4j graph databases. By applying cosine similarity for query evaluation, the system effectively matches natural language queries (NLQ) to structured Cypher queries. The results showed that rule-based methods achieve 85–88% accuracy, outperforming traditional keyword-based search. Although manual rule creation is a limitation, the approach remains highly effective for domain-specific applications. Future research should explore hybrid models that integrate machine learning and rule-based techniques to enhance query interpretation. Additionally, incorporating graph embeddings and multi-language support could improve semantic search capabilities. This research confirms that rule-based approaches provide an efficient and interpretable solution for query processing in graph databases. Overall, the findings suggest that semantic query conversion plays a crucial role in improving information retrieval accuracy.

5. References

1. A. Angles and C. Gutierrez, “Survey of graph database models,” ACM Computing Surveys (CSUR), vol. 40, no. 1, pp. 1–39, 2008.
2. J. Pérez, M. Arenas, and C. Gutierrez, “Semantics and complexity of SPARQL,” ACM Transactions on Database Systems (TODS), vol. 34, no. 3, pp. 1–45, 2010.
3. I. Robinson, J. Webber, and E. Eifrem, Graph Databases: New Opportunities for Connected Data. Sebastopol, CA: O’Reilly Media, 2013.
4. S. Ferré, “SQUALL: A controlled natural language for querying and updating RDF graphs,” Journal of Web Semantics, vol. 20, pp. 39–58, 2017.
5. J. Park and J. Lee, “Comparison of rule-based and machine learning-based NLP for domain-specific query processing,” Expert Systems with Applications, vol. 113, pp. 69–80, 2018.
6. D. Galkin, S. Radhakrishnan, and A. Vahdat, “Semantic query transformation in graph-based databases,” Proceedings of the 2019 ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 943–957, 2019.
7. Y. Li, X. Wang, and H. Liu, “A deep learning approach for natural language query conversion in graph databases,” IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 8, pp. 1432–1446, 2019.
8. L. Xu, T. Qin, and J. Liu, “Transformer-based models for natural language database query generation,” Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 4782–4795, 2020.
9. Y. Zhang, L. Yang, and Z. Chen, “Neural semantic parsing for query transformation: A comparative study,” ACM Transactions on Information Systems (TOIS), vol. 40, no. 2, pp. 1–23, 2021.
10. C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. New York, NY: Cambridge University Press, 2010.
11. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2018.
12. G. Koutrika, A. Simitsis, and Y. E. Ioannidis, “Semantic query optimization for graph databases,” Proceedings of the 2020 ACM International Conference on Information and Knowledge Management (CIKM), pp. 1289–1298, 2020.
13. C. Vicknair, D. Macias, and M. Friedman, “A comparison of a graph database and a relational database: A case study using Neo4j and MySQL,” Proceedings of the 2010 ACM Southeast Conference (ACM SE), pp. 1–6, 2010.
14. P. Miller, T. Harrison, and J. Taylor, “Optimizing Cypher query performance in large-scale graph databases,” Journal of Database Management, vol. 24, no. 3, pp. 32–45, 2013.
15. X. Zhao, K. Chen, and Y. Zhou, “Graph embeddings for semantic query ranking,” ACM Transactions on the Web (TWEB), vol. 16, no. 1, pp. 1–25, 2022.

Licensed under Creative Commons Attribution-ShareAlike 4.0 International License