An Automated Evaluation Framework for Graph Database Query Generation Leveraging Large Language Models

Bailan He, Yushan Liu, Marcel Hildebrandt, Zifeng Ding, Yaomengxi Han and Volker Tresp
• **Knowledge Graphs**: a collection of triples \( G \subseteq E \times R \times E \)

(Siemens, supplies to, BASF),
(Sumitomo, located in, JP),
......
**Query Generation:** given a user’s natural language request $X = (x_1, x_2, ..., x_m)$, generate a corresponding query $Y = (y_1, y_2, ..., y_n)$, that can retrieve the answer the user wants from a database.

**User:**
How many companies are in Germany?

**Target Query:**
MATCH (n:Supplier) –[:LOCATED_IN] -> (:Country {name:"DE"})
RETURN count(n)
Framework
Overview

• **Process I – Query Dataset Creation**
  A dataset consisting of diverse NL requests and queries.

• **Process II – Query Generation**
  Evaluate the model performance of different prompts based on the generated evaluation dataset.
Query Dataset Generation

Workflow

Initial Query Template -> Placeholder Substitution -> Requests Generation -> Human Evaluation
### Query Dataset Generation

#### Different Prompting

I want you to act as a Neo4j specialist. I will supply you with a Cypher query that needs to be explained in a simple manner, suitable for someone without any knowledge of databases. Please write a user-friendly question for the given query in one sentence, ensuring it is concise, easily understandable, and avoids technical jargon. The question should not start the description with "This query..." or "The query..." and do not mention the database or network.

<table>
<thead>
<tr>
<th>Simple prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The schema is delimited with triple dashes.</strong></td>
</tr>
<tr>
<td><strong>Schema: ```{schema}``````</strong></td>
</tr>
</tbody>
</table>

I will supply you with two examples of questions and answers. Each example contains a [Query] and a [Question]. In the end you need to supply a question for the given query.

<table>
<thead>
<tr>
<th>+Schema prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Query]: <code>MATCH (n)-[r]-&gt;(m:ManufacturerPart) RETURN n, r</code></strong></td>
</tr>
<tr>
<td><strong>[Question]: What are the things connected to manufacturer, how are they related to each other?</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>+In-context prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Query]: <code>MATCH (n:Branch)-[r1]-&gt;(m:Branch) RETURN COUNT(r1)</code></strong></td>
</tr>
<tr>
<td><strong>[Question]: How many connections exist between different branches?</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simple prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Query]: ```{query}``````</strong></td>
</tr>
<tr>
<td><strong>[Question]:</strong></td>
</tr>
</tbody>
</table>
Query Generation with LLMs
Different Prompting

Request: What countries are connected to branches, and what are the names of those branches?
Gold query: MATCH (n:Country)-[r1]->(m:Branch) RETURN n, collect(m.bp_name)

Simple Prompt
Generate a Cypher query corresponding to the following verbalization delimited with triple backticks, the output should be a Cypher query and nothing else.

[no prose]:

[Question]: ```{requests}```

Output: MATCH (n:countries)-[r1]-> (m:branches) RETURN n, m.names

Promp with Schema
Simple Template + Given the Neo4j schema delimited with triple dashes and the property keys delimited with open angle brackets.
Schema: ```{schema}```
Property keys: ```{property}```

Output: MATCH (n:Country)-[r1]-> (m:Branch) RETURN n, m.bp_name

In-context Prompt
Schema Template + Several examples will be provided. Example:

[Question]: ```What are the names of all the countries?
```
[Query]: MATCH (n:Country) RETURN collect(n.name_de)

Output: MATCH (n:Country)-[r1]-> (m:Branch) RETURN n, collect (m.bp_name)
Framework

Metrics

• **Execution Rate (ER)**

\[
ER = \frac{\text{Number of executable queries}}{\text{Total number of output queries}}
\]

• **Gold Query Accuracy (GQA)**

\[
GQA = \frac{1}{N} \sum_{i=1}^{N} \text{BERTScore}(O_i, G_i)
\]

• **Execution Accuracy (EA)**

\[
EA = \frac{1}{N} \sum_{i=1}^{N} \text{BERTScore}(R_i, R_{G_i})
\]
The dataset [1] is constructed with internal information of the company Siemens.

In total, there are 16,910 tier-1 suppliers, 43,759 tier-2 suppliers, and 49,775 tier-3 suppliers of Siemens.
Experiment
Settings: Supply Chain Query Dataset

<table>
<thead>
<tr>
<th>Query Template Type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Matching</td>
<td>10</td>
</tr>
<tr>
<td>Relationship Matching</td>
<td>10</td>
</tr>
<tr>
<td>Aggregation and Analysis</td>
<td>10</td>
</tr>
<tr>
<td>Combining Filters and Aggregation</td>
<td>10</td>
</tr>
<tr>
<td>Complex Queries</td>
<td>15</td>
</tr>
<tr>
<td>Traversal and Paths</td>
<td>5</td>
</tr>
</tbody>
</table>

The dataset includes 825 pairs of query-requests, with a value of 0.72 w.r.t Fleiss’ kappa metric, indicating the soundness of the dataset.
### Experiment

#### Results: Query Generation Results

<table>
<thead>
<tr>
<th>Method</th>
<th>GPT-3.5</th>
<th></th>
<th></th>
<th>GPT-4</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GQA</td>
<td>ER</td>
<td>EA</td>
<td></td>
<td>GQA</td>
<td>ER</td>
</tr>
<tr>
<td>simple</td>
<td>0.47</td>
<td>0.76</td>
<td>0.31</td>
<td>simple</td>
<td>0.55</td>
<td>0.81</td>
</tr>
<tr>
<td>schema</td>
<td>0.62</td>
<td>0.74</td>
<td>0.42</td>
<td>schema</td>
<td>0.71</td>
<td>0.89</td>
</tr>
<tr>
<td>ICL-1 shot</td>
<td>0.63</td>
<td>0.87</td>
<td>0.44</td>
<td>ICL-1 shot</td>
<td>0.69</td>
<td>0.89</td>
</tr>
<tr>
<td>ICL-3 shot</td>
<td>0.70</td>
<td>0.90</td>
<td>0.55</td>
<td>ICL-3 shot</td>
<td>0.73</td>
<td>0.91</td>
</tr>
<tr>
<td>ICL-5 shot</td>
<td>0.72</td>
<td>0.91</td>
<td>0.52</td>
<td>ICL-5 shot</td>
<td>0.75</td>
<td>0.93</td>
</tr>
</tbody>
</table>

**Notation:**

Simple denotes direct model instruction, Schema indicates prompting with schema, and ICL-$k$ shot (in-context learning with $k$ examples) involves instructing the model with in-context demonstrations.

- **GPT-4** outperforms GPT-3.5 across prompting all methods;
- Employing **schema** yields better results;
- **In-context learning** consistently outperforms using direct instructions.
Conclusion
Findings and Limitations

• The proposed **automated QG evaluation framework** tackles domain-specific challenges (SCM).

• The framework involves both **dataset creation** and **model performance evaluation**.

• The work studies how different **prompting** and **models** affect LLMs’ performance of QG.
Future Directions

• **Experimental Configuration**
  - Include other prominent LLMs like Gemini [1] and Llama [2]

• **Query Style**
  - Explore multi-turn dialogue in the workflow.