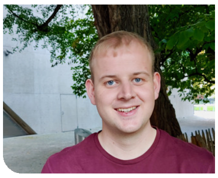
Natural Language to GraphQL

Leveraging the power of LLMs to build a natural language interface for GraphQL schemas

Graduate



Florian Rohrer

Lucien Zimmermann

Advisor
Prof. Stefan F. Keller

Co-Examiner
Claude Eisenhut, Burgdorf, BE

Subject Area
Software, Artificial Intelligence

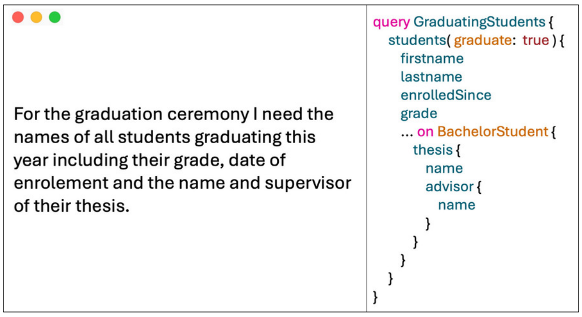
Project Partner
ChilliCream, Zürich

Introduction: In-context learning enables large language models (LLMs) to comprehend and respond based on the context of the input, allowing them to adapt to a wide range of domain-specific tasks without additional training. However, they have notable limitations in accessing real-time or domain-specific data because they operate primarily on the knowledge they were trained on. Natural language-to-SQL (NL-to-SQL) systems offer a practical solution, enabling LLMs to transform natural language into SQL commands. This makes data accessible to people without technical expertise. GraphQL, having emerged as a flexible alternative to REST, enables software clients to specify the precise data they require from an API based on a schema. Based on this research, the project aimed to evaluate the best practices for building a natural language interface for GraphQL by adapting the concepts of NL-to-SQL to the GraphQL domain. By understanding the concepts and evaluating a few optimal strategies, a strong foundation was laid for future development.

Approach: Upon starting the project, in-depth research was conducted to understand the current state-of-the-art, focusing on NL-to-SQL due to its similarities and substantial literature available. Key findings were documented, and possible strategies were defined. After a proof-of-concept, two strategies, with few sub-variants, were established for implementation. The first strategy solely relies on the capabilities of LLMs to directly follow instructions and, using in-context learning, enhance the prompt with relevant samples. The second strategy focuses on using entity extraction to identify entities in the user's question, match them to the schema, and then build the operation based on an abstract syntax tree (Fig 1). During implementation, a third strategy combining the benefits of both previous approaches evolved that overcomes the context size limitation of LLMs. In order to measure the strategies' performance in various metrics, an evaluator was built to efficiently test different implementations against a test set (Fig 3). The latter was inspired by the Spider dataset, a widely used benchmark for NL-to-SQL solutions.

Result: The evaluation has seen 7 different LLMs tested against the 4 most mature variants. The results were analyzed to determine the best-performing combinations. While the first strategy showed promising results for simple test cases, it demonstrated limitations in terms of quality and consistency for more complex ones. The best-performing combination uses entity extraction and algorithmic query generation, which is capable of correcting intermediate errors and always produces valid output, making it reliable enough to be used in experimental environments. In general, OpenAI models (GPT-4) are reliable in following instructions, while open source models (Llama3, Mistral) have trouble consistently generating valid structures such as JSON. Hallucinations, though occurring on both OpenAI and open source models, can be drastically reduced with prompt engineering, making the output more consistent. However, both strategies are limited by the context size of the LLM used, making them cost-inefficient or even non-processable for large schemas. To overcome this limitation, future research should focus on an iterative entity extraction approach, enabling large schemas to be processed. Additionally, the shot sampling process can be optimized to further benefit from the LLM's in-context learning capabilities.

Fig 1: A demonstration of how a user prompt is transformed into a valid GraphQL operation
Own presentment



For the graduation ceremony I need the names of all students graduating this year including their grade, date of enrolement and the name and supervisor of their thesis.

```
query GraduatingStudents {
  students( graduate: true ) {
    firstname
    lastname
    enrolledSince
    grade
    ... on BachelorStudent {
      thesis {
        name
        advisor {
          name
        }
      }
    }
  }
}
```

Fig 2: Entity Extraction-based solution strategy, using few-shot in-context learning, creating an Abstract Syntax Tree
Own presentment

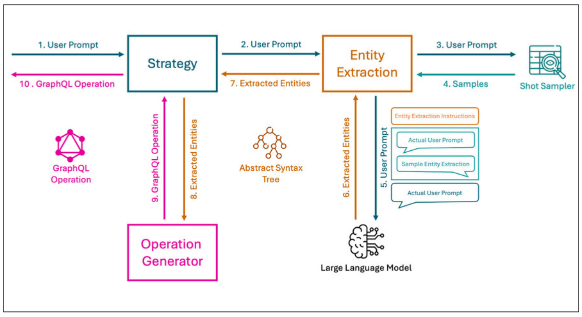


Fig 3: Evaluation results of the Entity Extraction-based strategy on a selection of test cases (lower numbers = better)
Own presentment

Metric	TestCase Results			
	Simple 1	Medium 1	Medium 2	Complex 1
Valid Syntax	✓	✓	✗	✓
Valid on Schema	✓	✓	✗	✓
Equivalent	✓	✓	✗	✗
Overfetched Fields	0	0	-	1
Underfetched Fields	0	0	-	1
Argument Missmatch	0	0	-	2
LLM Roundtrips	1	1	1	1
Tokens Used	4100 / 222	5621 / 157	4638 / 287	5644 / 489