Bachelor Practical Course – Data Engineering
TUM Campus Heilbronn – TUM School of Computation, Information and Technology (CIT)
Technical University of Munich

LLMs for Structured Constraint Generation

Jaden Rotter ✉ and Saumil Savani ✉

Motivation

As the internet continues to grow exponentially, ensuring the quality, consistency, and correctness of web data becomes increasingly vital, especially for machine readable formats like the resource description framework (RDF), on which the semantic web is built. Manual validation of such data is impractical at scale, and while the shapes constraint language (SHACL) provides a means of defining validation rules for RDF graphs, writing SHACL shapes remains a non trivial and error prone task due to its technical complexity.

Consider the following complexity involved in validating a simple RDF Graph.

Example RDF Graph

```
:Bob a :Person ;
:Bob :age "Twenty-five" .
```

Figure 1 Simple RDF graph defining a person and their age

SHACL Shape to validate the Person's datatype

```
:PersonShape a sh:NodeShape ;
  sh:targetClass :Person ;
  sh:property [
    sh:path :age ;
    sh:datatype xsd:integer ;
  ] .
```

Figure 2 Simple SHACL shape validating Figure 1

To ease the task of manually creating complex syntax heavy shapes, we construct a SHACL shape generation pipeline, which takes only Natural Lanaguage as input. Consider the ease of writing the following example to validate Figure 1 instead of using Figure 2.

The SHACL shape :PersonShape applies to all instances of the class :Person, specifying that the property :age must be an integer.

Our model can then take the NL input and return the generated SHACL shape.

Link to Paper

Figure 3 Link to Github and Paper

Method and Model

Our solution enables users to construct natural language prompts, which are then passed to a finetuned model that generates the corresponding SHACL shapes, significantly simplifying the process of creating these complex, syntax heavy specifications. To train the model to recognize structured SHACL patterns, we developed a data generation pipeline that produces pairs of natural language descriptions and their SHACL equivalents. Using this synthetic dataset, we fine tuned several open source models from Hugging Face and validated their outputs against a predefined ground truth. For evaluation, we implemented an automated validation pipeline that applies natural language processing (NLP) techniques to assess both syntactic and semantic similarity to the ground truth. Additionally, we performed manual evaluation to ensure accuracy and completeness.

Method Overview

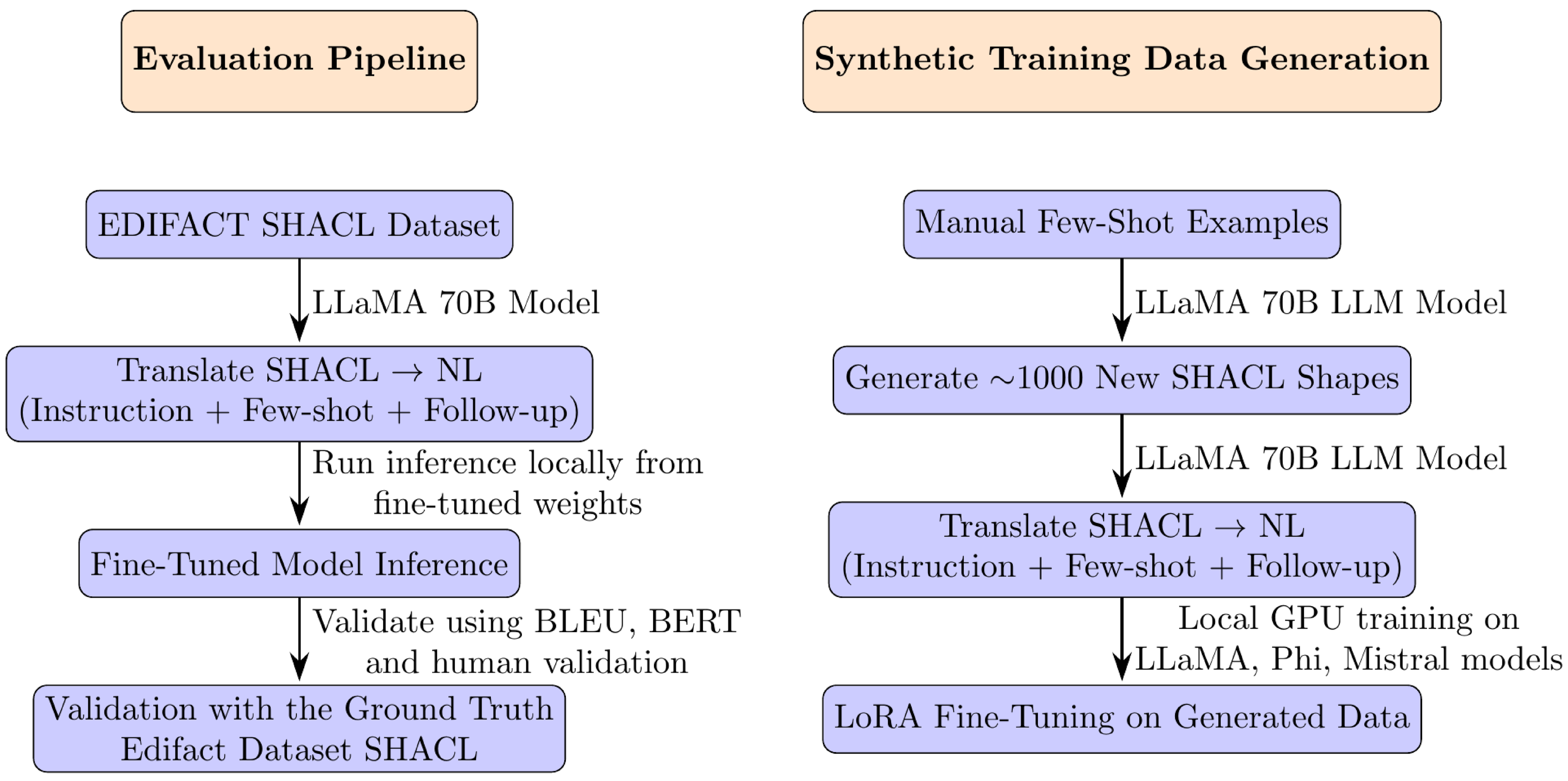

Figure 4 Method Schema

Results

Model Evaluation Results

We evaluated three LLMs:Groq70B, Mistral7B,and Qwen7B using BLEU and BERTScore. BLEU checks word overlap, while BERTScore captures semantic similarity to reference translations.

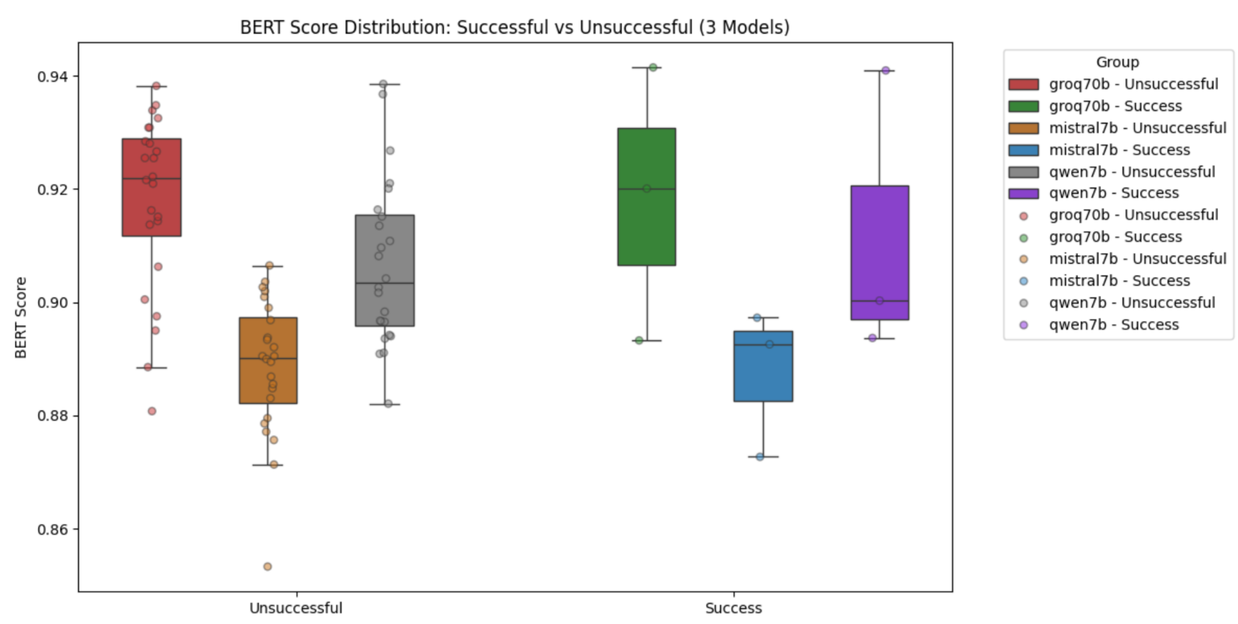

Figure 5 BERT Score Distribution by Model and Success Status

This plot shows BERTScore distributions for successful and unsuccessful outputs across three LLMs. Successful outputs consistently score higher, indicating better semantic alignment with the reference translations. Groq70B and Qwen7B show strong performance, while Mistral7B has a larger drop in unsuccessful cases. This result validated BERTScore as a reliable automatic metric for identifying high-quality SHACL translations in our pipeline.

This graph compares BLEU and BERT scores across the three models: Groq70B, Mistral7B, and Qwen7B. While all models achieve high BERT scores (semantic quality), their BLEU scores (lexical overlap) vary significantly.Groq70B performs best in both metrics, indicating its outputs are both structurally accurate and semantically faithful. We conducted this comparison to identify the most reliable model for generating accurate SHACL shapes from natural language.This result helped us choose Groq70B as the most balanced model for SHACL translation, aligning well with both human judgment and automatic metrics.

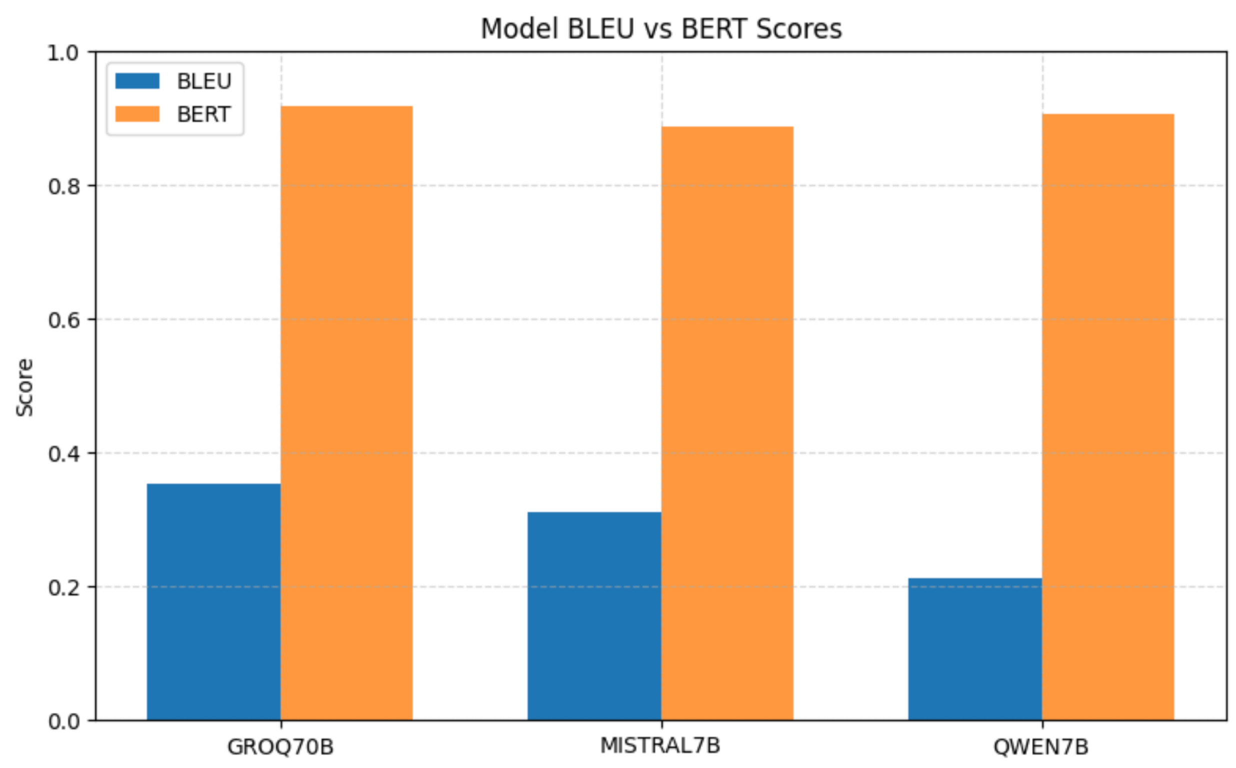

Figure 6 Model-wise Comparison of BLEU and BERT Scores

TUM Campus Heilbronn – TUM School of Computation, Information and Technology (CIT), Technical University of Munich

We would like to thank our advisor, Tim Schwabe, for his guidance and support throughout this project.