Data Engineering Research Group
TUM School of Computation, Information, and Technology
Technical University of Munich

LLMs for Structured Constraint Generation

Jaden Rotter ✉ **and Saumil Savani** ✉

✉ jaden.rotter@tum.de
✉ saumil.savani@tum.de

July 8, 2025

Abstract — Abstract

1 Introduction

As the internet grows exponentially, guaranteeing the reliability and consistency of web data for use in applications has become increasingly important. However, the sheer volume of data makes manual validation infeasible, driving the need for automated, machine oriented solutions. The Semantic Web, an extension of the internet, addresses this by enabling machines to read and understand data through structured languages defined by standard ontologies. On this structured data, further languages such as Shapes Constraint Language (SHACL) allow parameters and constraints to be defined in order to validate the underlying data. Unfortunately, as both the volume and complexity of data increase, validation requirements become more sophisticated, making the design of structured validation rules increasingly challenging. To simplify human validation tasks, we propose a solution involving large language models (LLMs) to help convert natural language (NL) into structured SHACL shapes.

Our solution enables users to construct natural language prompts, which are then passed to a fine-tuned 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.

2 Background

2.1 Semantic Web and SHACL

At the core of the Semantic Web lies the Resource Description Framework (RDF), a graph based data model designed to express how data entities are connected. RDFs represent information as triples, each consisting of a subject, object and predicate, allowing resources identified by Uniform Resource Identifies (URI)s, a universal mechanism to name and locate information on the internet, to be semantically linked. This set RDF structure enables machines to perform logical inference, interpret, and interconnect data from various sources. However, given the complexity of the internet, RDFs use very flexible language without any standard global schemas, which does not allow for the enforcing of rules and conditions. Without these rules, RDF data can become incompatible between systems or propogate errors such as missing values, wrong cardinalities or wrong data types.

To enforce the RDF data structure, SHACL allows the definition of set rules and conditions, called shapes, to help detect mistakes in RDF graphs. Without SHACL, there would be no formal way to validate or invalidate RDF graphs. Developers would need to write custom scripts, which are error-prone and time-consuming, to ensure the consistency of their RDF graphs. Instead, SHACL uses predefined prefixes and paths to express constraints on the objects, subject and predicates in RDF. Consider the following simple example below.

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

Figure 1 Simple RDF graph defining a person and their age

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

Figure 2 Simple SHACL shape validating Figure 1

As seen in Figure 1, two RDF values are defined for the subject `Bob` using the standard triple format, without any syntactic sugar. These RDF triples declare `Bob` to be of type `Person`, using the keyword predicate `a`, and specify the predicate `age` with the literal value `"Twenty-five"`. The exact prefixes and prefix namespace for `Bob`, `Person`, etc. have been omitted for brevity.

For applications using this RDF data and expecting `Bob`'s age to be an integer, trying to read in the string literal would result in an error. This simple example would likely be found very quickly; however, as the RDF data grows in size and complexity, the need for SHACL becomes increasingly more important. Errors in the RDF data can propogate and inference or analysis on this incorrect data could have drastic consequences. As such, SHACL defines a way to standardize the validation of this RDF data. In Figure 2, an example SHACL shape is defined to verify that the age is indeed of type `integer`. Both the `xsd` and `sh` define standard shacl prefixes to define the `PersonShape` SHACL shape.

Structured languages, require rigid syntax that demands greater input from humans to learn and generate, especially for complex validation languages such as SHACL. Already, the complexity of such a simple SHACL shape is fairly high. For humans, a simpler validation input, in the form of NL would greatly save time and effort. For example, consider the difference between the following natural language prompt in comparison to the SHACL shape in Figure 2.

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

2.2 LLMs and Fine Tuning

2.3 First subsection

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

2.4 Second subsection

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

3 Methods

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

4 Outlook

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.