LLMs for Structured Constraint Generation

Jaden Rotter ✉ **and Saumil Savani** ✉

✉ jaden.rotter@tum.de
✉ go69jal@mytum.de

July 8, 2025

Abstract — This paper explores the use of large language models in generating structured language, SHACL shapes, from natural language. Using a LLaMA 70B model, we built a data generation pipeline from a SHACL dataset, producing 1089 natural language to SHACL pairs via few shot prompting. Using this synthetic dataset, we fine tuned smaller 7B models and used the generated output to test the inference ability of our trained models. The generated shapes along with the original ground truth were evaluated in our validation pipeline using both manual and NLP techniques. The results of this validation highlights the feasibility of SHACL generation with LLMs, especially under constrained resources.

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 [7]. 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.

NL is simple for humans to understand and generate, but allow for many ambiguous interpretations, which make set rule based translation strategies nonviable. The task of translating human generated NL to a machine readable format requires a more flexible mechanism, in order to correctly interpret the semantic meaning in the text. The recent explosion of OpenAI and its GPT models in popularity for NL chatting and question and answer tasks, highlights the ability of LLMs to perform NL reasoning and understanding. Following this reasoning and related work, [4] [1] [8], we construct a similar pipeline using pretrained and our own fine tuned LLMs.

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 Identifiers (URI)s, a 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 schemes. This flexibility restricts the enforcing of rules and conditions, which are essential for data consistency and reliable reasoning in RDF based systems. Without these rules, RDF data can

become incompatible between systems or propagate errors such as missing values, wrong cardinalates 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 propagate 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. RDF structure already requires a very specific syntax, but the validation of RDF is far more complicated. Already the complexity difference between the SHACL shape and the RDF data is significant and there are no cardinality or length requirements. For humans, a simpler validation input, in the form of NL would greatly save time and effort. 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

LLMs are massive deep neural networks trained on vast volumes of text from sources such as books, articles, and websites. This extensive training enables them to develop strong linguistic understanding, pattern recognition, and contextual reasoning capabilities [5]. During training, these models learn by adjusting billions of internal weights, often exceeding 100 billion parameters in larger models, through computationally intensive processes that demand significant storage and processing power. While general purpose LLMs are capable across a wide range of tasks, they can be fine tuned on domain specific data to improve performance on particular tasks. Fine tuning helps the model adapt to more specialized patterns, allowing it to perform more accurately in targeted applications.

Unfortunately, the compute and storage requirements for both training and inference grow super linearly as model size increases [3]. While scaling up models generally leads to better accuracy and generalization, the improvements follow a law of diminishing returns, where increasingly large resource investments yield only modest gains in performance. These substantial demands placed by the training and deployment of large or even medium sized LLMs is far beyond the scope of our current infrastructure, requiring data centers, months of time, and huge volumes of training data.

For this project, we are limited to a single NVIDIA Tesla V100 GPU with 16 GB of VRAM, which restricts both the size of models we can use and the feasibility of training entire models. Under these constraints, we rely on fine tuning small open source models using parameter efficient techniques such as LoRA (Low Rank Adaptation) [2]. LoRA significantly re-

duces memory and compute overhead by freezing the original model weights and injecting low rank trainable matrices. In practice, this approach allows us to train only about 0.3% to 1.5% of the model's parameters, making fine tuning viable within our hardware limitations.

Despite their strengths in NL tasks, LLMs remain fundamentally probabilistic generative models. They interpret ambiguous inputs by estimating the most likely next tokens based on learned patterns in the training data. This means that, while LLMs excel at matching linguistic patterns, they do not perform true reasoning. As a result, they can produce outputs that appear coherent but are factually incorrect, called hallucinations. In translation tasks, such hallucinations may take the form of invented URIs or non existent namespace prefixes, posing challenges for tasks requiring strict semantic accuracy.

3 Methods

We used the LLaMA 70B parameter model via API to establish a data generation pipeline based on an existing EDIFACT dataset[1] containing 89 SHACL shapes. From this dataset, we manually selected a subset of representative shapes to create few shot prompts with human written translations. These prompts were designed to expose the model to a wide variety of SHACL syntax patterns. Following past research, we generated prompts constructed out of an initial context statement, 4 few shot examples, the given SHACL shape to translate, and a follow up prompt.

To improve the quality of translations, we used the same 70 billion parameter LLaMA model to generate further examples to help our fine tuning process avoid overfitting on the same data. We developed few shot prompts using the edifact SHACL shapes to generate new SHACL shapes, which were then fed into our original pipeline to generate further translation pairs. This approach allowed us to generate 1089 examples on which to train our model on. Unfortunately, a limitation of this data generation approach is the lack of diversity in the generated data. All outputs use the same static few shot prompting template, potentially leading to overfitting or syntactic bias in the output SHACL shapes. This method however, still produced more variety in training data than increasing the epochs of our original dataset would have.

[1] https://github.com/DE-TUM/EDIFACT-VAL/blob/main/example/ProcessExample.ttl

For model fine tuning, we used the Unsloth library to train the largest models possible within our hardware constraints, specifically the memory limitation of 16 GB of VRAM. We fine tuned several 7B parameter open source models, Mistral and Qwen, and compared their performance to the model used for data generation. Due to shared hardware availability, fine-tuning was constrained to approximately 9–12 hours per model, resulting in 100–300 epochs depending on the model and batch size.

Initially, the fine tuned models struggled to learn certain structural patterns in SHACL, such as correctly distinguishing between `sh:minCount` and `sh:maxCount` instead of hallucinating properties like `sh:cardinality`, which do not exist in the SHACL specification. To address this, we developed a set of manually defined transformation rules that targeted the core subset of SHACL constructs. These rules were introduced to help the model infer correct syntax and improve output accuracy. While effective for our selected subset, future work involving SHACL generation via LLMs may benefit from an expanded and more comprehensive set of domain specific rules.

4 Evaluation

Assessing the effectiveness of LLMs when translating between SHACL and natural language necessitates a more in depth analysis than standard NLP tasks like summarization. SHACL encapsulates data restrictions such as essential attributes, datatype regulations, or logical groupings like `sh:or`, all of which must be accurately maintained. A translation that sounds fluent could still introduce logical inaccuracies, resulting in invalid SHACL shapes. Thus, evaluation for this task must extend beyond superficial fluency to ensure both semantic and structural accuracy.

We implemented an evaluation approach: BLEU [6], BERTScore [9], and human assessment. BLEU measures n-gram overlap and evaluates lexical similarity between the generated output and the reference, yet it penalizes even legitimate paraphrases, making it overly rigid for this area. BERTScore assesses semantic similarity through contextual embeddings, which more effectively captures meaning preserving variations. Lastly, human assessment was used to confirm and evaluate the accuracy of both the BLEU and BERT scores, as well as to provide a baseline model inference accuracy. The human assessment furthermore allowed for a more holistic understanding of the models ability to understand SHACL rules without introducing in-

valid constraints. The performance of each model on 89 examples from the Edifact dataset were compared using all three defined evaluation metrics.

Table 1 presents a summary of the outcomes. LLaMA 70B achieved the highest score with a BLEU score of 0.34, a BERT score of 0.92, and a human accuracy rate of 85.3. Mistral7B closely trailed in terms of semantic similarity with a BERT score of 0.90 but demonstrated slightly lower overall performance. Qwen7B recorded the lowest BLEU score of 0.22, though it had a solid BERT score of 0.91 and an elevated human rating. This indicates that while Qwen7B tends to paraphrase more freely, it frequently maintains accurate meaning. LLaMA 70B likely performed the best due to its significantly larger parameter count, which enabled the model to capture the complex patterns better. Potentially, despite the fine tuning of the 7B models, their smaller capacity limited their ability to model the complex nature of the data.

Table 1 Evaluation scores across three LLMs on SHACL ↔ Natural Language translation.

Model	BLEU	BERTScore	Human Accuracy
Groq70B	0.34	0.92	85.3%
Mistral7B	0.30	0.90	83.1%
Qwen7B	0.22	0.91	84.8%

4.1 Graph Result And Analysis

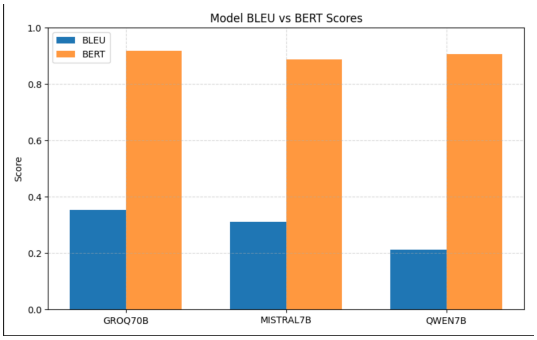

Figure 3 Comparison of lexical and semantic similarity scores across models using BLEU and BERTScore.

Figure 3 illustrates the evaluation of the three large language models, Groq70B (built on LLaMA 70B), Mistral7B, and Qwen7B, based on their BLEU and BERTScore metrics. These standard NLP metrics were chosen to capture both the syntactic and semantic quality of generated translations. Groq70B leads with the highest BLEU score of 0.34 and BERT score of 0.92, meaning it produced outputs that closely follow both the wording and meaning of the ground truth SHACL shapes. Mistral7B follows with similar scores in both metrics, with BLEU scoring 0.30 and BERT scoring 0.90, showing consistent performance but slightly lower alignment. Qwen7B scored the lowest on BLEU with 0.22, while still achieving a high BERT score of 0.91, suggesting its translations often use varied phrasing yet successfully retain the intended SHACL semantics. This is especially important for our task, where maintaining meaning, rather than exact syntax, is essential for correct rule interpretation.

These results highlight a key insight in our SHACL ↔ NL translation task: relying solely on BLEU scores can unfairly penalize models like Qwen7B that produce semantically and syntactically valid shape, but are penalized due to the different structure. Our evaluation approach benefits from combining BLEU with BERTScore, allowing us to capture both literal overlap and deeper meaning preservation. LLaMA 70B demonstrates the strongest performance, making it ideal for general broad focus tasks. With 70B parameters, LLaMA 70B requires multiple GPUs to even run inference, meaning these models are only available through rate limited APIs. For local training with limited resources, the smaller models performed very well, rivaling the model 10x larger. With greater memory, compute, training data, and time the smaller models have the potential to outperform even the larger LLaMA 70B. Overall, this dual-metric analysis helps us make informed decisions in selecting LLMs based on the nature of SHACL rules and the expected variation in natural language output.

4.2 Semantic Reliability and Human Evaluation Patterns

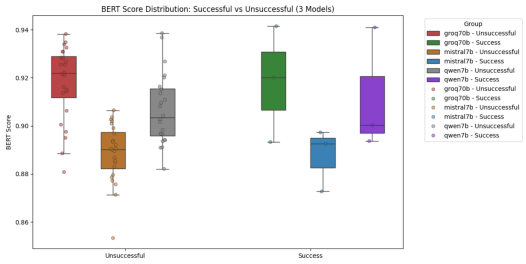

Figure 4 BERTScore distribution for human-evaluated outputs (Correct = 1, Incorrect = 0).

In our SHACL ↔ Natural Language translation project, Figure 4 visualizes the distribution of BERTScores for each model, grouped by whether the output was judged correct or incorrect by human an-

notators. Across all three models—Groq70B, Mistral7B, and Qwen7B—we observed a consistent trend: translations rated as correct by humans had significantly higher BERTScores than incorrect ones. Notably, Qwen7B exhibited the most distinct separation between correct and incorrect groups, with its accurate outputs clustering near the top end of the BERTScore range and incorrect outputs falling far below. This tight separation suggests high semantic reliability: when Qwen7B is correct, it is also consistently close to the intended meaning, allowing BERTScore to serve as a strong signal for correctness.

In contrast, Groq70B, while achieving the highest average performance overall, showed a wider spread of BERTScores even among correct outputs. Some of its correct translations received only moderate scores, and a few incorrect ones still scored high—indicating that surface-level similarity (captured by BERTScore) doesn't always guarantee logical validity in SHACL rules. This highlights a critical insight for our task: while BERTScore is a helpful metric for detecting alignment with the reference description, it should not be used in isolation. Human validation remains essential, especially to detect subtle errors in constraint logic. The boxplot results emphasize that evaluating SHACL translations requires both semantic similarity and domain-specific correctness checks to ensure outputs are structurally and logically sound.

5 Summary

In this project, we developed a robust pipeline to translate between SHACL and NL using large language models (LLMs). We generated 1,089 high-quality synthetic SHACL–NL pairs covering diverse domains and constraints to enable effective training. Fine-tuning was conducted on a 7B Qwen model using LoRA with 4-bit quantization for efficiency. Evaluation included automatic metrics consisting of BLEU for surface similarity, BERTScore for semantic alignment and human validation for logical consistency and correctness. Groq70B outperformed other models, achieving the highest BLEU score of 0.34 and the BERT score of 0.92, indicating excellent structural and semantic translation quality.

For future work, we aim to expand the pipeline to support multilingual SHACL–NL translation, incorporate Levenshtein distance metrics for detecting structural deviations, and develop an interactive tool for real-time SHACL editing with LLM-assisted feedback. Furthermore, increasing compute, storage, training and time has the potential to further improve translation quality. These enhancements will support broader adoption and integration into data governance and semantic web applications.

References

[1] Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarcisio Mendes de Farias, and Ana Claudia Sima. LLM-based SPARQL query generation from natural language over federated knowledge graphs. October 2024.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank adaptation of large language models. June 2021.

[3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. January 2020.

[4] Mihai Nadas, Laura Diosan, and Andreea Tomescu. Synthetic data generation using large language models: Advances in text and code. March 2025.

[5] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. March 2022.

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[7] Ansgar Scherp, Gerd Gröner, Petr Skoda, Katja Hose, and Maria-Esther Vidal. Semantic web: Past, present, and future. *TGDK*, 2(1):3:1–3:37, 2024.

[8] Tim Schwabe, Louisa Siebel, Patrik Valach, and Maribel Acosta. Q-NL verifier: Leveraging synthetic data for robust knowledge graph question answering. March 2025.

[9] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *International Conference on Learning Representations (ICLR)*, 2020.