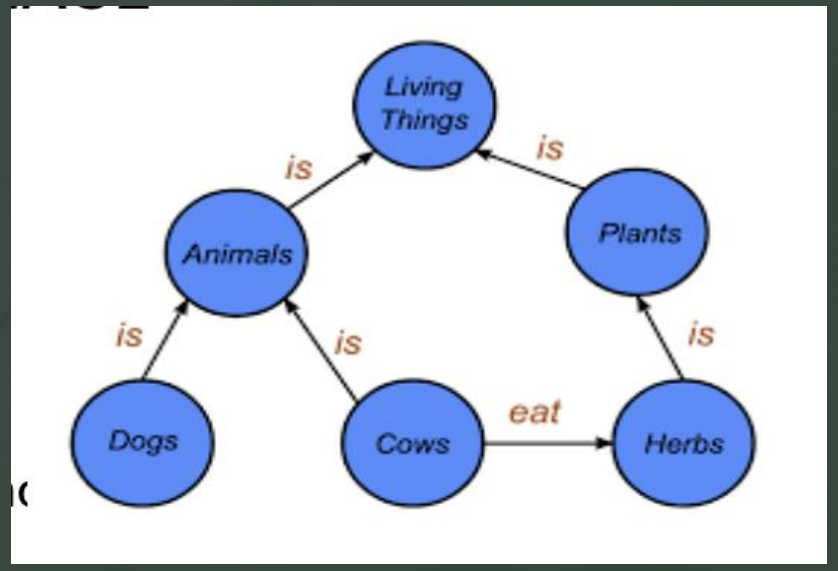
Generating SHACL shapes from Documentations using LLMs

- Jaden Rotter & Saumil Savani
- Technische Universität München
- TUM School of Computation, Information, and Technology
- Data Engineering Research Group

- Munich, 11 May 2025

Background—Knowledge Graphs & SHACL



Knowledge Graphs (KGs)

- Linked Data, representing relationships between entities
- Enhances decision-making, supports NLP, and semantic search
- Machine readable format to enable automated analysis and inference

SHACL (Shapes Constraint Language)

- Validates data structure and integrity
- Machine readable

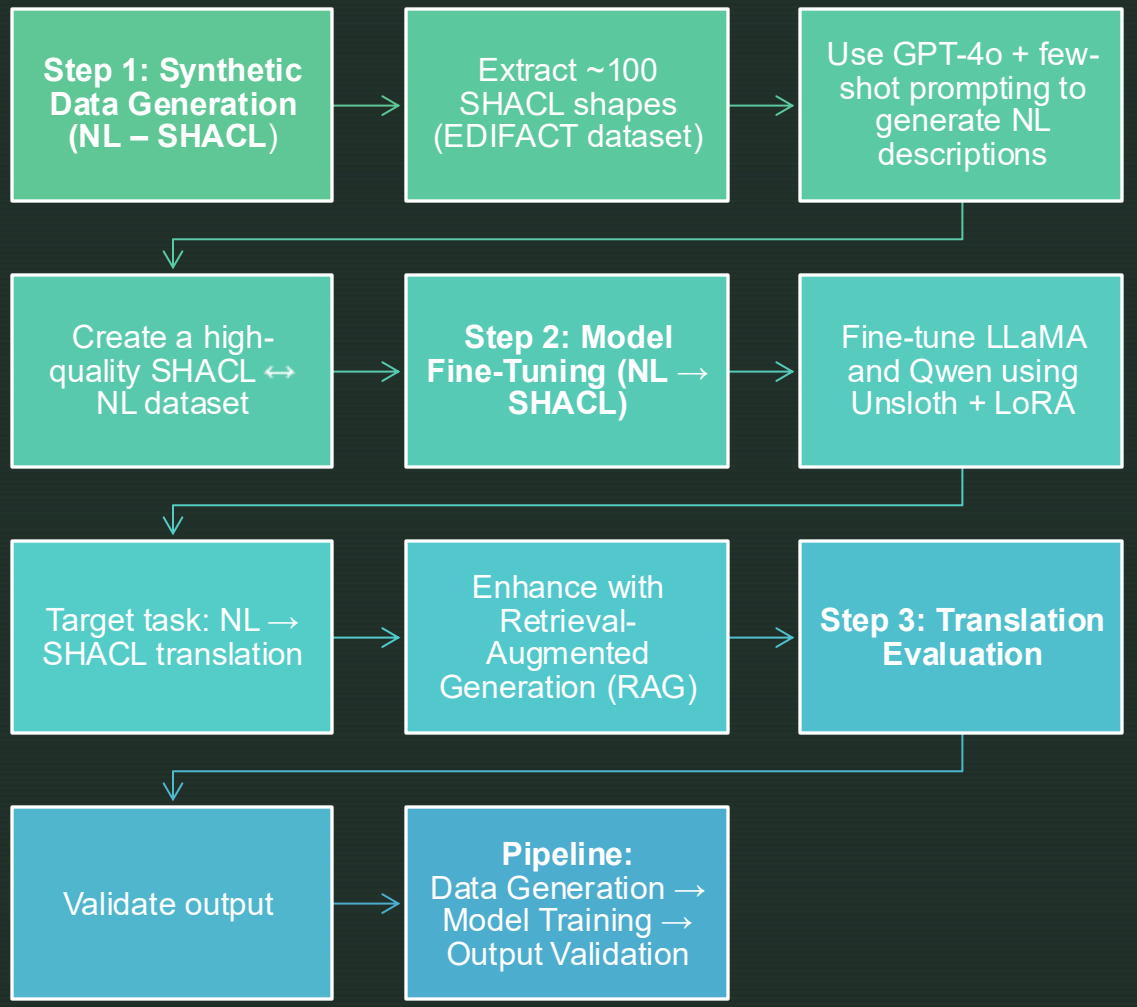
Research Questions & Motivation

Research Questions (RQs)	Can SHACL shapes be generated from NL using LLMs?
	Which LLM performs best for generating SHACL from NL?
	Can LLMs be used to generate synthetic data for validation?
Motivation	Documentation currently written in NL for clarity
	Validating KGs requires SHACL, a machine-readable format
	SHACL syntax is difficult and slow for humans to produce and learn
	Past research[1] has shown LLMs have succeeded in converting NL to machine-readable formats like SPARQL

1: Q-NL Verifier: Leveraging synthetic data for robust knowledge graph question answering

1: LLM-based SPARQL Query Generation from natural Language over Federated Knowledge Graphs

Our Approach



SHACL→Natural Language Dataset Generation (Using Gpt-4o)

We first created a synthetic dataset by extracting ~100 SHACL shapes from the **EDIFACT dataset.** We used **GPT-4o** (OpenAI's latest LLM) to translate these shapes into natural language descriptions. To ensure high quality, we applied

Few-shot prompting → we gave examples before new tasks

Reflection prompt: We asked GPT-4o to review and improve its own answers.

Potentially, depending on the accuracy of our model in step 2, we will need to generate more SHACL-NL pairs using similar prompt engineering to improve the model training.

We will use the following SHACLs : https://github.com/DE-TUM/EDIFACT-VAL/blob/main/example/ProcessExample.ttl

https://arxiv.org/pdf/2503.01385

Synthetic Dataset Generation with GPT-4o

Translation Prompt : You are an AI assistant. Your task is to convert the following SHACL shape into a clear natural language documentation sentence for a human reader.

```
:SteuerpflichtigerBetrag
 a sh:NodeShape;
 sh:targetClass edifact-o:InvoiceDetails;
 sh:property [
   sh:path edifact-o:hasTaxableAmount ;
   sh:minCount 1 ;
   sh:maxCount 1 ;
   sh:message "The segment MOA+125 is missing, i.e., the
indication of the taxable amount."
 ].
```

 # GPT-4o output: Defines a constraint for InvoiceDetails. The property hasAmount must be a decimal ≥ 0.

 # Why GPT-4o?

1. High accuracy in structured → unstructured translation
2. Fast + highly generalizable model for synthetic dataset generation

Expected GPT-4o Output: "The SteuerpflichtigerBetrag shape defines a constraint for InvoiceDetails. The property has hasTaxableAmount must be present exactly once. If missing, the message 'The segment MOA+125 is missing, i.e., the indication of the taxable amount.' is displayed

Reflection Prompt: – Review your answer. Is the meaning clear and concise? Are all important constraints (target class, property name, cardinality, message) mentioned? If you find any missing or unclear information, rewrite and improve your answer. Only return the improved natural language sentence

Using Synthetic Data to Train LLM to translate NL to SHACL

Challenge :

- **NL → SHACL is more difficult than SHACL → NL**
- Natural language is ambiguous; SHACL has rigid syntax
- Requires correct inference of class/property names
- Needs domain-specific knowledge

Our Approach :

- We plan to fine-tune open-source LLMs like **LLaMA** and **Qwen** and benchmark their performance against **GPT-4**.
- To make training efficient, we will use **LoRA (Low-Rank Adaptation)**, which reduces the number of trainable parameters and lowers GPU memory requirements.
- Additionally, we will enhance the models with **Retrieval-Augmented Generation (RAG)** by providing relevant **entity URIs and class/property definitions** in the prompt to help the model generate accurate SHACL structures

Validation Strategy (Translation)

 # How will we check model quality?

The **goal** of validation is to ensure that the SHACL shapes generated by the LLM are both syntactically and semantically correct.

1. **Syntactic validation**: We will use the Python libraries **pySHACL** and **rdflib** to verify that the SHACL output conforms to RDF and SHACL syntax standards.
2. **Semantic validation: We** will compare the generated SHACL shapes to the original dataset using the **BLEU score** for token-level comparison and **BERTScore** to measure similarity based on language model embeddings

-> Additionally, selected examples will be reviewed manually to further ensure translation accuracy.

Weekly Timeline & Team Responsibility

Week (2025)	Dates	Activities
Week 1	April 28 – May 4	Project Kick-off.
Week 2	May 5 – May 11	Understand SHACL, KGs, and prior work (papers). **Preliminary Presentation** (May 12)
Week 3	May 12 – May 18	Extract ~100 SHACL shapes from the EDIFACT dataset. Prepare initial prompts for SHACL → NL generation. Generate a SHACL → NL dataset using GPT-4o + manual review. Start documenting the pipeline.
Week 4	May 19 – May 25	Improve prompt engineering + generate more SHACL → NL data if needed, review quality, and improve prompt instructions. **Preliminary Presentation** (May 19/21)
Week 5	May 26 – June 1	Progress stand-up. Clean and finalize the synthetic dataset. Prepare training dataset (format, split)

May = Synthetic dataset generation,Model training,
June = fine-tuning, retrieval-augmented generation
July = Validation, final results, prepare poster

Weekly Timeline & Team Responsibility

Week (2025)	Dates	Activities
Week 6	June 2 – June 8	Attend the poster course. Setup fine-tuning pipeline (LLaMA and Qwen models with Unsloth + QLoRA)
Week 7	June 9 – June 15	Start fine-tuning LLM models on the dataset. Implement RAG retrieval setup
Week 8	June 16 – June 22	Evaluate initial results. Adjust parameters/retrain if necessary. **Mid-term Presentation 1** (June 16)
Week 9	June 23 – June 29	Continue fine-tuning and evaluate new results. Test RAG and compare against the GPT-4o baseline. Prepare the next presentation. **Mid-term Presentation 2** (June 23)
Week 10	June 30 – July 6	Finalise fine-tuning. Complete model selection. **Mid-term Presentation 3** (June 30)
Week 11	July - July 13	Start the validation phase. Syntactic validation (pySHACL). Similarity evaluation (BLEU, BERTScore). **Stand-up + Q&A** (July 7/9)
Week 12	July 14	Final poster design + presentation preparation. Submit poster + results. **Final Poster Presentation** (July 14)

Thank you + Question ??

We are happy to answer any questions about our plan, methodology, or timeline