LEUPHANA
UNIVERSITY LÜNEBURG

Text2SHACL: LLM-Driven Generation of Validation Graphs for Automatic Assessment of Social Benefit Eligibility

Author:
Seike Hanna Theresia Appold

First Supervisor:
Prof. Dr. Ricardo Usbeck

Second Supervisor:
Dr. Debayan Banerjee

A thesis submitted in fulfillment of the requirements for the degree of Master of Science (M.Sc.) in

Management & Data Science

Lüneburg, June 16, 2025

Acknowledgements

Special thanks to the founders of **Förderfunke** - Ben Gläser[1] and Benjamin Degenhardt[2] - for supporting the annotation process with their practical expertise and providing the idea to use SHACL validation to assess social benefit eligibility, which inspired and shaped this project. They provide a functional prototype[3] of their service, and open-source the entire project on GitHub[4].

[1] https://github.com/wbglaeser
[2] https://github.com/benjaminaaron
[3] https://foerderfunke.org/
[4] https://github.com/Citizen-Knowledge-Graph

Abstract

To increase social benefit take-up, applications like FörderFunke automatically assess eligibility based on user data and provide personalized information about relevant benefits. FörderFunke encodes eligibility requirements in the Shapes Constraint Language (SHACL), supporting the German government's efforts to promote open standards in public IT and facilitating their solution's maintainability, modularity, and interoperability. However, like other SHACL-driven approaches to automatic compliance checking, they face the challenge that converting natural language rules to SHACL requires significant manual effort and slows down development. Against this background, this work formally defines the Text2SHACL task and extends existing approaches in two key ways: First, we establish a principled foundation for utilizing SHACL in the formerly unexplored domain of social benefit eligibility assessment. Specifically, we introduce a domain-specific annotated dataset and a schema guiding through the Text2SHACL task alongside a qualitative analysis of critical SHACL- and domain-specific formalization challenges. Second, we explore an end-to-end approach to automating Text2SHACL driven by Large Language Models (LLMs), which overcomes limitations of prior work that struggles with complex constraints and diverse linguistic input. Adopting a prompt engineering methodology, we establish a Zeroshot baseline, analyze reasons for an overall poor initial performance, and demonstrate the positive effects of Fewshot and Chain-of-Thought prompting on the syntactic and semantic quality of generated shapes graphs for a selection of LLMs.

Acronyms

AEC Architecture, Engineering, and Construction

API Application Programming Interface

BIM Building Information Model

CoT Chain-of-Thought

CWA Closed World Assumption

DeepSeek deepseek-r1-distill-llama-70b

EGovG E-Government Gesetz

FN False Negatives

FP False Positives

FS Fewshot

G-BS G-BERTScore

GDPR General Data Protection Regulation

GED Graph Edit Distance

GWDG Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

HPC high-performance computing

HTTP Hypertext Transfer Protocol

IRI Internationalized Resource Identifier

KB Knowledge Base

KFCV K-Fold Cross-Validation

Llama 70B llama-3.3-70b-instruct

Llama 8B llama-3.1-8b-instruct

LLM Large Language Model

Mistral mistral-large-instruct

NL natural language

NLP Natural Language Processing

OWA Open World Assumption

OWL Ontology Web Language

PVOG Portalverbund Online Gateway

Qwen qwen-2.5-72b-instruct

QwQ qwen-qwq-32b

RDF Resource Description Framework

RDFS RDF Schema

Sauerkraut llama-3.1-sauerkrautlm-70b-instruct

SD standard deviation

SGB Sozialgesetzbuch

SHACL Shapes Constraint Language

ShEx Shape Expressions

SOTA state-of-the-art

SSC SHACL Syntax Conformance

T-F1 Triple-F1

T-P Triple-Precision

T-R Triple-Recall

TN True Negatives

TP True Positives

TSC Turtle Syntax Conformance

Turtle Terse RDF Triple Language

V-Acc Validation-Accuracy

V-F1 Validation-F1

V-P Validation-Precision

V-R Validation-Recall

W3C World Wide Web Consortium

ZS Zeroshot

List of Tables

List of Figures

Listings

Contents

1 Introduction

Social benefit systems aim to alleviate poverty and enhance socioeconomic stability. However, non-take-up often undermines their effectiveness, where eligible individuals do not receive the benefits they are entitled to [65, 109, 7, 100]. Research consistently identifies limited awareness of available benefits, entitlements, and application procedures as a major barrier to take-up [7, 31, 113]. To address this challenge in the German context, digital platforms such as *FörderFunke*[5], *Sozialkompass*[6], and *Sozialplattform*[7] seek to provide users with personalized recommendations for potentially eligible social benefits based on their living situation.

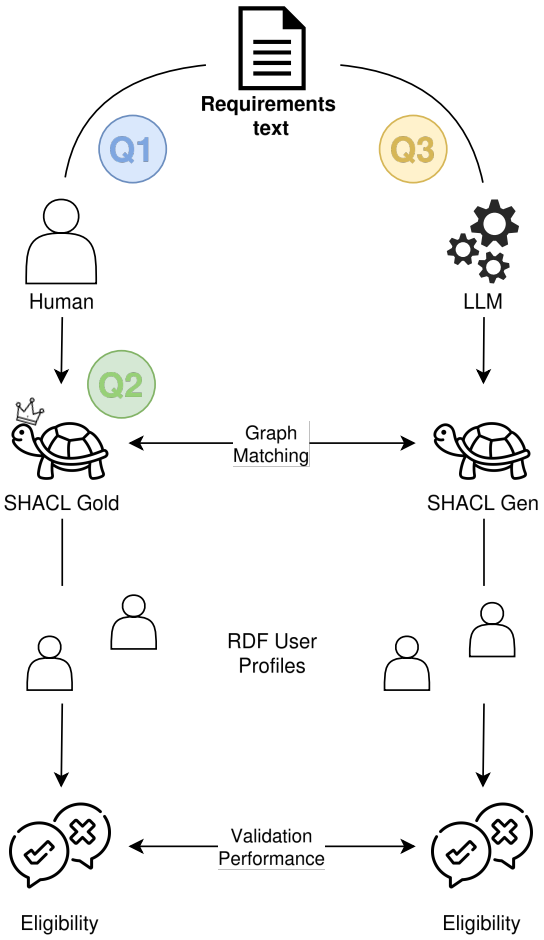


Figure 1: Overview of the overall approach.

However, a key technical challenge is translating eligibility criteria from natural language (NL) into formal representations amenable to automatic validation. *FörderFunke*[8] uniquely addresses this by employing World Wide Web Consortium (W3C) [120] standards, developed to realize the Semantic Web - a vision of a Web of Data based on human- and machine-understandable knowledge representations [9]. Specifically, the social start-up represents user data with the *Resource Description Framework (RDF)* [20] and eligibility requirements with the *Shapes Constraint Language (SHACL)* [64] for data validation. This approach aligns with the German government's efforts to promote open-source software and open

[5] https://foerderfunke.org/
[6] https://sozialkompass.info/de
[7] https://sozialplattform.de/
[8] https://foerderfunke.org/

standards in public IT (§16a E-Government Gesetz [EGovG] [24]). It also ensures interoperability with the broader Semantic Web technology stack and facilitates maintainability and modular development by decoupling knowledge representation from the code base.

While FörderFunke demonstrates the practical feasibility of this approach with a functional prototype[9], the laborious conversion of text to SHACL, a task we term *Text2SHACL*, presents a bottleneck to scaling the scope of supported benefits. Notably, this challenge extends beyond social benefit assessment. For example, SHACL is widely used in the Architecture, Engineering, and Construction (AEC) industry to represent regulatory [80, 28] or technical [44, 45] standards and has also been applied to encode research software best practices [53] and petroleum industry regulations [2].

Despite this cross-domain interest, existing work on Text2SHACL lacks scalability because it depends on either labor-intensive manual modeling [2, 93, 80, 44, 53] or deterministic Natural Language Processing (NLP) pipelines, which handle only low-complexity rules and struggle with NL variability [28, 98]. Moreover, none of these works investigate SHACL's suitability for capturing social benefit rules. To address these limitations and unlock the potential of SHACL for accessible social benefit eligibility assessment, we pose three research questions:

(Q1) *How can eligibility requirements for social benefits be systematically translated from natural language text into SHACL shapes graphs?*

(Q2) *What challenges hinder the faithful representation of eligibility requirements for social benefits with SHACL Core?*

(Q3) *How do different prompting techniques affect the syntactic and semantic quality of SHACL shapes graphs automatically generated by LLMs from eligibility requirement texts?*

Figure 1 illustrates our overall approach, demonstrating where each question is situated within the process.

The objective of Q1 and Q2 is to establish a principled foundation for formalizing social benefit eligibility rules using SHACL. To this end, we first contribute a formal definition of the Text2SHACL task. We then extract requirements texts from the central German platform for administrative services, the Portalverbund Online

[9] https://foerderfunke.org/

Gateway (PVOG) [55], and manually translate them into SHACL shapes graphs. The graphs are validated through systematic testing on human-authored synthetic user data and independent annotation by two domain experts.

In answering Q1, this methodology yields two key contributions: First, a schema for converting NL text into SHACL that captures domain-agnostic challenges as well as domain-specific solutions. Second, an annotated Text2SHACL dataset for social benefit assessment, consisting of 21 requirements texts and human-authored ground truth SHACL shapes graphs, an original task-specific ontology, and a set of 314 manually constructed synthetic RDF user profiles.

Moreover, in response to Q2, we conduct a qualitative analysis of the persistent challenges encountered during the annotation process. This analysis contributes a discussion of key limitations of SHACL Core in representing complex eligibility requirements, as well as domain-specific characteristics that inherently hinder full formalization.

Subsequently, Q3 investigates a scalable, end-to-end approach to automating Text2SHACL using LLMs - Transformer-based models [111] that have fundamentally advanced NLP in recent years [116]. We adopt a prompt engineering methodology, evaluating three prompting techniques across a selection of LLMs, ranging from *Zeroshot (ZS)* prompting with minimal guidance as a baseline, to *Fewshot (FS)* prompting with an exemplary input-output pair, to *Chain-of-Thought (CoT)* prompting based on the previously introduced Text2SHACL schema.

Model outputs are assessed based on their syntactic and semantic quality using three sets of metrics: syntax conformance scores, graph matching metrics [46, 94, 41], and validation performance scores. The latter two sets evaluate the generated shapes graphs against the structural and functional characteristics of the human-authored ground truth, as illustrated in Figure 1. This framework provides empirical insights regarding the capabilities of different types of LLMs to perform Text2SHACL, the specific task elements that pose the greatest difficulty, and the prompting techniques that enhance performance.

We provide the dataset and code openly at: https://github.com/semantic-systems/text-to-SHACL.git

2 Foundations

2.1 Semantic Web Technologies

The Semantic Web is a vision to evolve the traditional Web from a collection of interconnected documents designed to be human-readable to a Web of Data based on standardized knowledge representations that are equally accessible to machines. It conceives of the Web as a global, decentralized database where software can automatically interpret and process meaning, enabling richer human-machine collaboration [9]. To realize this vision, the W3C develops standards that provide concrete technological solutions to the abstract conceptual requirements of the Semantic Web [120].

2.1.1 RDF: Representing Knowledge

The *Resource Description Framework (RDF)* [20] is the W3C standard for representing resources on the Web and their interrelations. Virtually anything from physical objects over digital files to abstract concepts can be a resource and is described in RDF with two core structures: triples and graphs. A triple consists of a subject, a predicate, and an object, corresponding to the statement that "some relationship, indicated by the predicate, holds between the resources denoted by the subject and object" [20]. This relation is called a property, and the object it points to is its value [20]. A set of triples forms a directed, edge-labeled graph called an RDF graph [50].

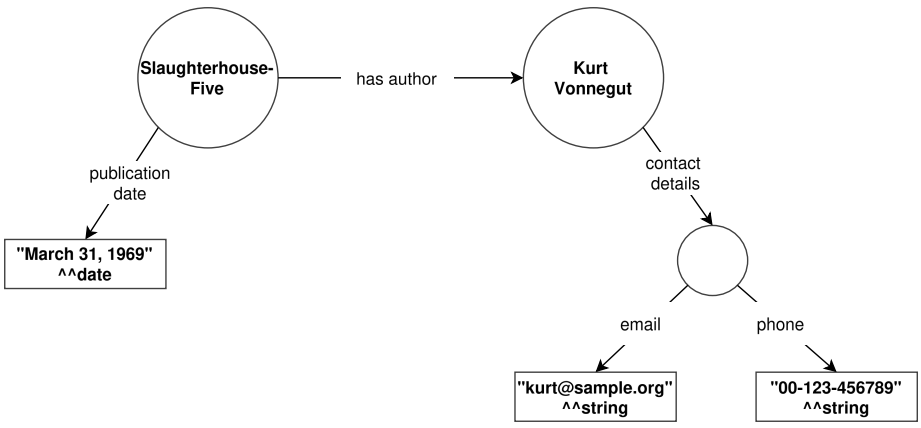


Figure 2: Visualization of a sample RDF graph.

Table 1: Overview of namespace prefix bindings.

Prefix	Namespace
`ex:`	`http://example.com/books#`
`ff:`	`https://foerderfunke.org/default#`
`owl:`	`http://www.w3.org/2002/07/owl#`
`rdf:`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs:`	`http://www.w3.org/2000/01/rdf-schema#`
`sh:`	`http://www.w3.org/ns/shacl#`
`xsd:`	`http://www.w3.org/2001/XMLSchema#`

The schematic visualization of an example RDF graph in Figure 2 illustrates these components. One of its triples consists of the entities *Slaughterhouse-Five* (subject), *has author* (predicate), and *Kurt Vonnegut* (object), which intuitively asserts that "Slaughterhouse-Five was written by Kurt Vonnegut".

Although this statement seems straightforward, its interpretation is contextual. For instance, in a cinematic database *Slaughterhouse-Five* might denote the film of the same name rather than the novel. To disambiguate the identity of resources, RDF employs globally unique Unicode strings called Internationalized Resource Identifiers (IRIs) [32]. For example, using Hypertext Transfer Protocol (HTTP) IRIs resolvable by a web-server, the book might be named `http://example.org/books#SlaughterhouseFive`, and the film `http://example.org/films#SlaughterhouseFive`. For conciseness, IRIs are commonly represented using local namespace prefix bindings. As anyone can register a prefix to establish a namespace, this method helps avoid naming conflicts without central coordination [50]. Table 1 specifies the prefixes used throughout this work.

Additionally, RDF uses two other types of data: literals and blank nodes [96]. Literals are basic values associated with a datatype that appear only in the object position such as the publication date in Figure 2. Blank nodes refer to an anonymous object or subject resource without explicitly identifying it. For instance, the blank node in Figure 2 (empty circle) groups the author's contact details. This ability to describe information with a multi-component structure is most relevant to our discussion of SHACL (§ 2.1.3), although blank nodes may serve other purposes like censoring sensitive information, reifying RDF statements, or including unknown resources [15].

To encode RDF data, this thesis adopts the Terse RDF Triple Language (Turtle) [8].

Another W3C recommendation, this serialization format was chosen for its close resemblance to NL and simultaneous ease of parsing. Listing 1 demonstrates how the RDF graph from Figure 2 can be written down using Turtle, illustrating some of the language's key features to concisely express the concepts described above: the declaration of namespace prefix bindings (lines 1-2), separation by semicolons to combine predicates for the same subject (lines 4–5, 7-8), and square-bracket notation for blank nodes (lines 6-8).

```
1 @prefix ex: <http://example.org/books#> .
2 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
3
4 ex:SlaughtergouseFive ex:hasAuthor ex:KurtVonnegut ;
5                       ex:publicationDate "1969-03-31"^^xsd:date .
6 ex:KurtVonnegut ex:contactDetails [
7                     ex:email "kurt@example.org"^^xsd:string ;
8                     ex:phone "00-123-456789"^^xsd:string
9                 ] .
```

Listing 1: Turtle serialization of a sample RDF graph.

Further reading: For readers seeking a deeper understanding of RDF, W3C offers specifications of its conceptual foundations [20] and practical use [96]. Details on IRIs are available in the specification [32] and the registration procedure [107]. Other relevant RDF serialization formats omitted here include N-Triples [14], RDF/XML [35], and JSON-LD [62].

2.1.2 Ontology: Towards Machine-Understandability

So far, the running example presupposed a shared interpretation of the database terms. However, consider merging data from a distributor that used `ex:writtenBy` in place of `ex:hasAuthor`. Humans effortlessly recognize the semantic equivalence and can even deduce unstated information - e.g., that Kurt Vonnegut is an author. Machines, in contrast, cannot access the implicit assumptions that support such conclusions. To facilitate data exchange and automated reasoning, the meaning of terms and their relationships must be formally defined.

An *ontology* addresses precisely this need. The term originates from philosophy, where ontology as a discipline explores the nature of existence [49]. In computer science, it is an "explicit specification of a conceptualization" [43] that formally defines the representable (i.e., "existing") terms in a given knowledge system. The main

building blocks are classes defining general concepts (e.g., *book*), and properties that describe and connect them (e.g., *has author*). An ontology paired with individual class instances (e.g., *Slaughterhouse-Five*) forms a Knowledge Base (KB), though the transition between the two is often fluid [79].

In practice, ontologies vary in complexity Lassila and McGuinness [67]. describe a range from simple collections of terms to richly axiomatized systems supporting arbitrary logical relationships. Here, we denote by *ontology* a conceptualization tailored to our concrete application that occupies a middle ground on this spectrum of semantic expressivity. This includes a finite set of terms with informal NL definitions, a strict subclass hierarchy where instances of a subclass are also instances of its superclass, and properties with domain and range constraints.

Listing 2 shows how the property `ex:hasAuthor` can be defined using RDF Schema (RDFS) and Ontology Web Language (OWL). RDFS [11] provides basic semantics like class and property hierarchies. For example, lines 1 and 2 link `ex:Book` to `ex:Author` instances, allowing to infer that if Vonnegut wrote a *book*, he is an *author*. OWL [114] adds more expressive semantics, such as declaring `ex:writtenBy` equivalent to `ex:hasAuthor` (line 3), helping align different vocabularies. Ontologies thus formalize meaning, enabling a shift from machine-readable to machine-understandable data [106].

```
1 ex:hasAuthor rdfs:domain ex:Book ;
2              rdfs:range ex:Author ;
3              owl:equivalentProperty ex:writtenBy .
```

Listing 2: Example of defining a property with RDFS and OWL.

Further reading: For a detailed account of the mentioned standards, see the specifications for RDFS [11] and OWL [114]. For readers curious about the logical formalisms of ontologies, Baader et al. [5] provide a thorough treatment of formal knowledge representation with description logics.

2.1.3 SHACL: Validating RDF Data

An ontology formalizes conceptual assumptions about the data, but it does not guarantee that the data adheres to them. For example, the KB in the running example may include the triple in Figure 3. In this case, the conceptualization from Listing 2 implies that *Gandalf* is an *author* due to his relationship with *Slaughterhouse-Five* - an implausible conclusion presuming that *Slaughterhouse-Five* is a real-world book

and *Gandalf* a fictional character. Such discrepancies between the assumed and actual data structure can adversely affect downstream applications powered by the KB, like a digital library catalog.

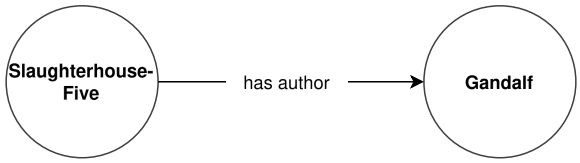
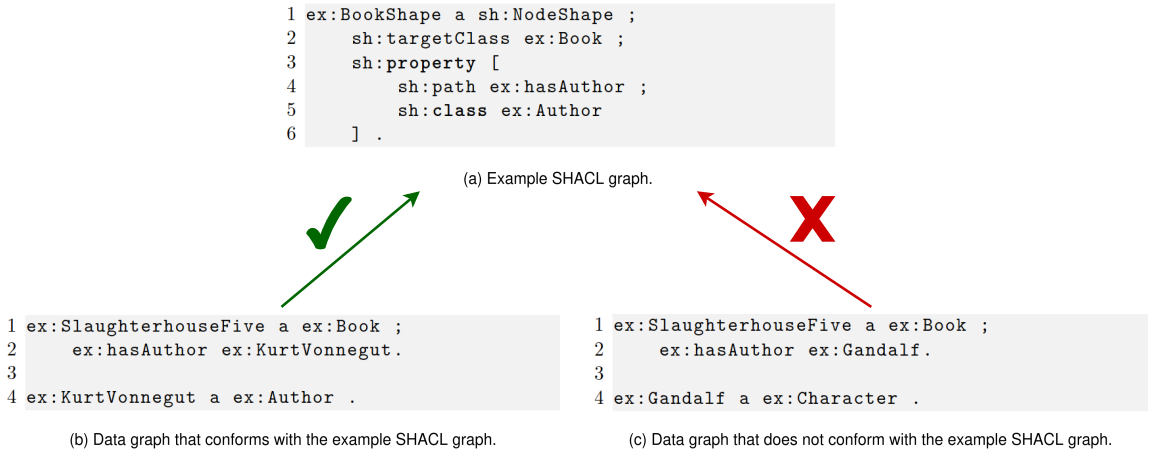


Figure 3: Visualization of an implausible RDF triple.

To validate that RDF data conforms to certain expectations, the W3C introduced the *Shapes Constraint Language (SHACL)* [64]. SHACL's core building blocks are shapes, which are composed of targets and constraints. Constraints define conditions that may apply to any RDF term using the parameters of constraint components. Target definitions are the most direct way to identify concrete nodes in an RDF graph on which a constraint will be evaluated (focus nodes) [86]. For example, the shape in Figure 4 targets all instances of the class `ex:Book` with the constraint that all values of the `ex:hasAuthor` property must by of type `ex:Author` using `sh:class`, a parameter of the constraint component `sh:ClassConstraintComponent`. Intuitively, this may be interpreted as "all books must be written by an author".

SHACL offers various other constraint components and methods for generating focus nodes. This work concentrates on the standardized constraint components defined by SHACL Core, supported by all SHACL implementations, and excludes the SHACL-SPARQL extension. Focus nodes can also be specified by directly passing a node to a validator or emerge from a shape-expecting constraint parameter like `sh:node`, which requires certain property values to conform to a shape [64].

SHACL distinguishes node shapes that apply constraints directly to focus nodes, and property shapes that restrict the nodes reachable via a specified property path. Both may be expressly typed or not and defined as IRIs or blank nodes - an example of SHACL's ability to express the same semantics in different ways, akin to NL paraphrasing. Figure 4 captures this syntactic flexibility: it presents a node shape explicitly named (`ex:BookShape`) and typed (`sh:NodeShape`), which references an anonymous property shape. The constraint in the property shape applies not directly to `ex:Book` instances, but to the objects of `ex:hasAuthor`, highlighting the conceptual distinction between focus nodes, that anchor a shape in the data graph, and value nodes, to which a constraint applies [64].



(a) Example SHACL graph.

(b) Data graph that conforms with the example SHACL graph.

(c) Data graph that does not conform with the example SHACL graph.

Figure 4: Example SHACL validation. (a) A shapes graph consisting of a node shape targeting *books*, with a property shape requiring that each *book* is written by an *author*. (b) and (c) show data graphs validated against this shape. (b) conforms; (c) violates the constraint because the *book* Slaughterhouse-Five (focus node) is written by Gandalf (value node), who is not an *author*.

SHACL validation takes two inputs: a set of SHACL shapes (shapes graph) and an RDF graph that is being validated (data graph) [64]. To conform with a shapes graph, the data graph must conform with each individual shape, which requires that all focus nodes satisfy all constraints [86]. This is the case in Figure 4b, where the value node `ex:KurtVonnegut` is of type `ex:Author` as expected by the only constraint in 4a. If a focus node does not satisfy a constraint, a validation result is generated [64]. For example, the non-conforming data graph in Figure 4c would produce the result that focus node `ex:SlaughterhouseFive` violates the property shape in 4a because it does not satisfy the constraint component `sh:ClassConstraintComponent`. The final output is a validation report, which is an RDF graph reporting the conformance of the data graph and the set of validation results [64].

An important premise of validation is the Closed World Assumption (CWA), which treats a graph as complete and interprets missing relations as nonexistent. Its counterpart, the Open World Assumption (OWA), treats absent information as unknown [63]. In SHACL, CWA applies in two ways. First, the shapes graph is considered exhaustive: only explicitly defined constraints are evaluated, and all nodes trivially conform to a shape without constraints [86]. For instance, the data graphs in Figure 4 would both conform if `ex:hasAuthor` was absent because the shapes

graph does not impose a cardinality constraint on the property. Second, if a relation is missing from the data graph it is assumed not to exist [64]. For example, the data graph in Figure 4b would not conform if Kurt Vonnegut was not explicitly typed as *author*.

A notable use case of SHACL is SHACL-SHACL, a shapes graph that encodes the main syntactic rules of SHACL Core. The meta shapes graph can be used to validate whether a given shapes graph is well-formed, meaning that all of its shapes conform to the SHACL syntax, or ill-formed otherwise [64].

In sum, SHACL shapes graphs make a separate yet deeply integrated contribution to the Semantic Web's ambition of enabling software agents to meaningfully analyze human knowledge [9]. RDF data graphs (§ 2.1.1) accommodate domain-agnostic and machine-readable knowledge representation. Ontologies (§ 2.1.1) additionally enable deductive knowledge discovery. Shapes graphs complement this framework by verifying whether the existing data conforms to certain expectations, fostering interoperability, data quality, security, and usability [86]. Despite their distinct functions, data graphs, ontologies, and shapes graphs are all grounded in the RDF data model and tightly interconnected: ontologies define the intended semantics, data graphs represent actual instances, and shapes graphs serve as a bridge by rendering structural expectations about the data, rooted in formal semantics, verifiable. Figure 5 revisits the library KB one last time to illustrate this interrelation.

Further reading: The SHACL specification [64] offers a comprehensive formal account of SHACL, including SHACL-SPARQL. Steyskal and Coyle [104] discuss possible use cases and requirements from a more practical perspective. Another language for RDF data validation, Shape Expressions (ShEx), was published as a W3C Community Group Report [88]. Gayo et al. [38] discuss its features and relation to SHACL in depth.

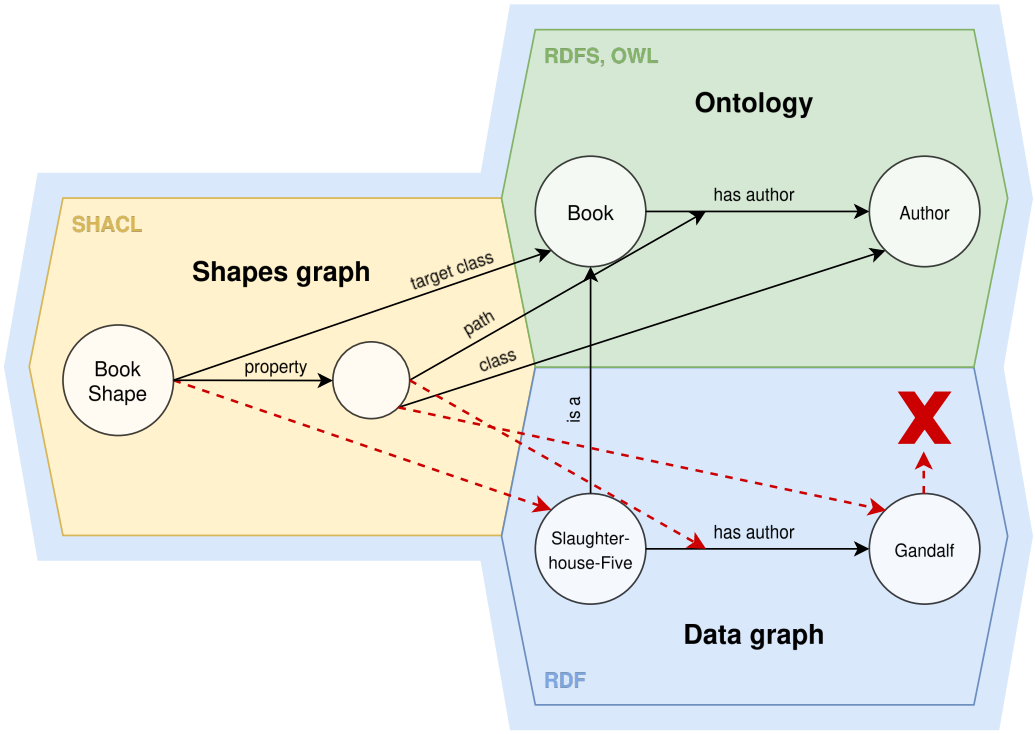


Figure 5: Interrelation of ontology, data graph, and shapes graph. Each hexagon represents one of the structures, with a schematic example and W3C standards. The blue frame highlights the use of the RDF data model throughout. The data graph states that *Slaughterhouse-Five* is a *book* written by *Gandalf*. The ontology defines that *has author* links *books* to *authors*. The *book shape* in the validation graph enforces this constraint by verifying that the *has author*-path for all *books* ends at an *author*. The red cross indicates that the data graph does not conform with the shape because *Gandalf* is not an *author*.

2.2 Large Language Models

Large Language Models (LLM) represent the current pinnacle of language modeling aimed at developing software agents with human-like conversational abilities [70, 116]. They have supplanted recurrent [47, 75] and convolutional neural networks [68], capitalizing on the vast, diverse textual data available on the web, advances in computational abilities, and algorithmic innovation [116]. In particular, the introduction of the Transformer by Vaswani et al. [111] has led to remarkable progress across various NLP tasks (12, 117, 26, *i.a.*).

2.2.1 From Transformers to LLMs

The Transformer was originally developed by Vaswani et al. [111] for machine translation, where accurately modeling the influence of surrounding words and sentence structure on word meaning is crucial. Its fundamental innovation is the reliance on self-attention mechanisms to address this challenge. In contrast to earlier recurrent models [47, 17], which process tokens sequentially and whose computational complexity increases linearly with token distance, self-attention computes dependencies between any two positions in constant time and supports efficient parallelization [111].

Self-attention transforms the input embedding $\mathcal{X}$ by weighing information from all sequence tokens according to their contextual relevance. Concretely, learned weights linearly project $\mathcal{X}$ into matrices $\mathcal{Q}$ (queries), $\mathcal{K}$ (keys), and $\mathcal{V}$ (values). Attention scores, calculated as the scaled dot-product between $\mathcal{Q}$ and $\mathcal{K}$ and normalized via softmax, are used to compute a weighted sum of the values, yielding token representations that are selectively influenced by all other tokens in the sequence. Multi-head self-attention runs multiple self-attention functions in parallel to capture diverse representations [111].

The Tranformer integrates self-attention throughout its encoder-decoder structure. The encoder transforms the source sequence (e.g., an English sentence) into rich contextual representations, while the decoder auto-regressively generates the target sequence token-by-token (e.g., a Dutch translation). Both blocks consist of $\mathcal{N}$ identical layers, each with a multi-head self-attention layer and a feed-forward network. The decoder additionally prepends a masked multi-head self-attention layer, restricting the attention mechanism to the current and past outputs. Moreover, by applying multi-head attention to the encoder's output it accesses contextual information from the entire input sequence [111].

LLMs have advanced the original Transformer along several dimensions. First, Vaswani et al. [111] relied on supervised learning over a task-specific dataset, whereas modern LLMs typically combine self-supervised pre-training on massive corpora with targeted fine-tuning - often in the form of instruction tuning, where models are trained on question-answer pairs to better follow human instructions [70, 127]. Second, with advances in data availability and hardware, model sizes scaled from the initial 65-213 million [111] to hundreds of billions learnable parameters (e.g., 12, 16). Third, model architecture diversified with encoder-only models like BERT [26] and decoder-only models like GPT-3 [12] emerging alongside encoder-decoder variants like T5 [92].

LLM development continues to evolve. Recent innovations include reasoning models like DeepSeek-R1 [22] and QwQ-32B [90] that target performance on logically demanding tasks and use reinforcement learning to achieve competitive results with relative resource-efficiency [10]. Both models are employed in this work and will be described in more detail in Section 6.1.2 on model selection. Moreover, Liu et al. [70] describe a more recent training paradigm that they coin "pre-train, prompt, and predict". Instead of fine-tuning the large, pre-trained foundation models with task-specific training objectives, this approach focuses on prompt engineering, i.e. how to present downstream tasks to pre-trained LLMs, which will be discussed in more depth in the next section.

Further reading: For an introduction to NLP, see Jurafsky and Martin [60]; a new edition also covering Transformers and LLMs is in progress at the time of writing[10]. For technical details on the original Transformer, refer to Vaswani et al. [111].

2.2.2 Prompt Engineering

This work adopts the "pre-train, prompt, and predict" [70] paradigm, which relies on *prompt engineering* - the process of selecting a model input (prompt) that yields the best performance on the downstream task. Because terminology in this area remains unsettled, we define key terms as used in this work in Table 2.

[10]Draft available at https://web.stanford.edu/~jurafsky/slp3/

Table 2: Key prompting terminology (adapted from Schulhoff et al. [97]). The prompt components (Directive, Input, Example, Additional Information) were selected based on their relevance to the present work and are neither required nor exhaustive for building prompts.

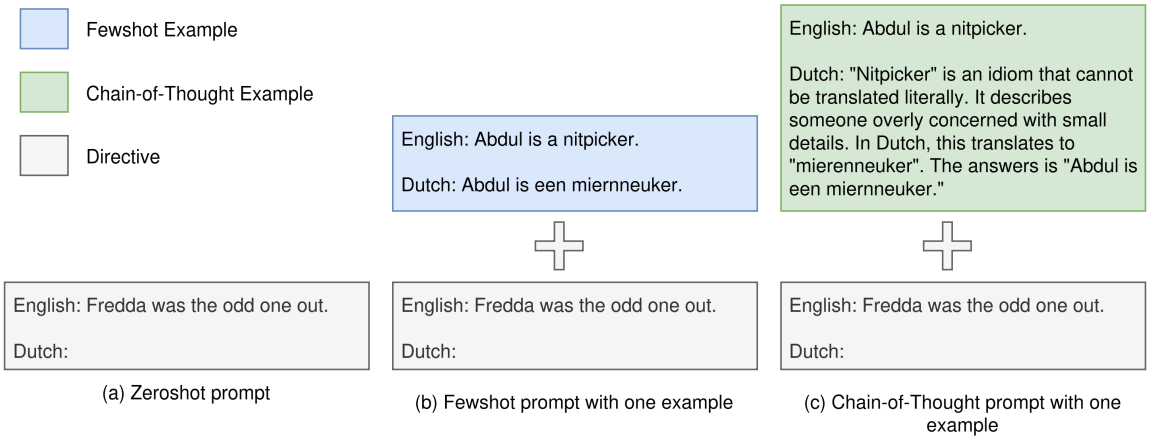
Term	Description	Example
Prompting Technique	A structured approach to formulating prompts.	Few-shot prompting
Prompt Template	A function with one or more variables that transforms an input into a structured prompt.	Translate to [lang]: [input]
Prompt	The complete input provided to an LLM to direct its output.	Translate to Dutch: Fredda was the odd one out.
Directive	An NL instruction or question that expresses the task's core intent.	Translate to Dutch:
Input	The main task-specific content to process or respond to.	Fredda was the odd one out.
Example	A demonstration of the expected output given an example input.	Input: Abdul is a nitpicker. Output: Abdul is een mierenneuker.
Additional Information	Supplementary content relevant to solving the task.	A list of Dutch idioms

Prompt engineering combines computational efficiency - avoiding costly parameter updates, storage of multiple specialized checkpoints, and dataset annotation [95] - with task flexibility. By leveraging the broad linguistic features learned by pretrained LLMs [70], a single model can attain competitive performance across diverse tasks [12, 87, 91] and adapt rapidly to low-resource domains lacking labeled data [95].

One practical approach to prompt engineering is to design task-specific templates by manually adapting an existing prompting technique. A prompting technique may dictate the components to be included (e.g., examples versus no examples [12]), the prompt format (e.g., discrete versus continuous [126]), or the execution schema (e.g., an agentic structure [124]). Here, we focus on three foundational techniques, illus-

trated in Figure 6 based on the familiar example from Table 2. They are characterized by their varying prompt components:

- **Zeroshot (ZS) prompting:** Presents only a directive and the input. While requiring no examples, its performance hinges on the model's pre-trained capacity to generalize from minimal information [12].
- **Fewshot (FS) prompting [12]:** Augments the template with a set of input–output demonstrations. Outperforms ZS across various NLP tasks [12] but results are sensitive to several factors like example order [72], label accuracy [76], and formatting [58]. More examples [12, 69] and examples that are close to the input in the embedding space [36, 69] tend to improve performance.
- **Chain-of-Thought (CoT) prompting [118]:** Incorporates examples with step-by-step solutions to tackle complex, multi-step tasks, yielding marked gains over standard FS [16, 105, 118]. For instance, it may explain idiomatic meaning before translating "nitpicker" into Dutch (Figure 6). Kojima et al. [66] introduce a ZS variant where deliberation is induced with a thought-provoking cue instead of examples. Moreover, several variants explore diverse solution paths, e.g., *Self-Consistency* [115] aggregates outputs from multiple sampled reasoning chains via majority voting, and *Tree of Thoughts* [123] iteratively builds a coherent reasoning trajectory by generating and evaluating alternative thought steps. In this work, CoT refers exclusively to example-based CoT.



(a) Zeroshot prompt

(b) Fewshot prompt with one example

(c) Chain-of-Thought prompt with one example

Figure 6: Prompting technique examples. Zeroshot (a) includes only the directive, fewshot (b) adds input-output pairs, and chain-of-thought (c) extends the example output with reasoning steps. In practice, components are concatenated into a single prompt string.

Further reading: Readers seeking a formal introduction to prompting may refer to Liu et al. [70] who propose a set of mathematical notations and comprehensive typology of prompt-based methods.

3 Related Work

3.1 Automatic Social Benefit Assessment

Several initiatives in Germany highlight that private and public stakeholders alike have recognized the demand for a digital, user-centric service for social benefit eligibility assessment. Although they share the ambition of simplifying access to benefits, they differ in interaction design, transparency of implementation, and underlying data models, as the examples below illustrate. We briefly present each system and then describe how our work departs from - and advances - the current approaches.

Sozialplattform[11]**.** Sponsored by the Ministry of Labor, Health and Social Affairs of North Rhine-Westphalia, Sozialplattform offers the "Sozialleistungsfinder" - a fixed, linear questionnaire that returns non-binding eligibility indications for up to 13 benefits. While municipalities can contact the providers to integrate the service via one of three deployment variants [102], neither the code nor the data schema is publicly available. Thus, technical details remain opaque at the time of writing but informal communications with project consultants indicate that the system relies on rule-based mappings from user responses to benefit categories.

Sozialkompass[12]**.** Originating from an Information Systems seminar in 2022 at the University of Münster and showcased at the 2022 Münsterhack[13] Hackathon, Sozialkompass is maintained by an interdisciplinary group of volunteers. It presents a dynamically branching questionnaire culminating in personalized assessments for ten benefits. The source code and data model are not linked on the project's web presence, nor publicly available on its GitHub[14] or the Münsterhack archives [34]. Based on informal consultations with team members, the system appears to represent eligibility conditions as Boolean expressions evaluated at runtime.

FörderFunke[15]**.** Like Sozialkompass, FörderFunke guides users through a dynamically branching questionnaire that adapts question paths based on prior answers. They encode eligibility rules as SHACL shapes graphs and model user inputs in RDF. A custom SHACL engine validates the assembled user graph against these shapes and classifies each of up to ten benefits on a graded scale from "insufficient

[11] https://sozialplattform.de/
[12] https://sozialkompass.info/de
[13] https://www.muensterhack.de/
[14] https://github.com/Sozialkompass
[15] https://foerderfunke.org/

information" to "probable eligibility." The developers actively encourage community contributions, and the development steps - including the application's source code and knowledge base - are continuously updated under an open-source license on their GitHub repository[16].

The research in my thesis builds on the work of FörderFunke, deliberately adopting two foundational design choices that distinguish it from the previous examples. First, as opposed to both, Sozialkompass and Sozialplattform, we release the source code and dataset under an open-source license on GitHub[17] to promote reproducibility and transparency. Foundational to scientific progress [42], these principles also meet rising expectations for accountability and innovative spirit in public IT, reflected in §16a of the German E-Government Act (E-Government Gesetz, EGovG) [24] that mandates prioritizing open-source software and open standards in federal software procurement. Second, we encode eligibility rules as SHACL shapes, contrasting with what we know about the data models underlying Sozialkompass and Sozialplattform. As an open W3C standard, this choice again aligns with §16a EGovG, ensures interoperability with the wider Semantic Web stack (§ 2.1), and separates requirements representation from application logic. The latter allows legal amendments to be effected simply by revising SHACL shapes - without touching the codebase - while also enabling domain experts to contribute benefit rules without programming expertise.

Finally, we extend FörderFunke's manual modeling workflow by investigating the potential of LLMs to automatize the process after a systematic inspection of the underlying Text2SHACL task. The narrow scope of supported benefits across the above applications underscore the need to accelerate the encoding of eligibility rules into machine-readable formats. By exploring this path, this work contributes to a foundation for scalable assessment systems capable of adapting to the diverse and evolving landscape of social benefit programs.

3.2 Text2SHACL

This section surveys the existing research on generating SHACL shapes from NL text, setting the stage for our own contribution and the formal definition of Text2SHACL presented in Section 4. We conducted keyword queries in LUX and Google Scholar with search keywords constructed from the following terms: *text-to-shacl*, *text2shacl*, *generating SHACL with [large language models | prompt engineering]*, *shacl validation*

[16] https://github.com/Citizen-Knowledge-Graph
[17] https://github.com/semantic-systems/text-to-SHACL.git

of [requirements | rules | natural language text]. We prioritized the top twenty results per query and expanded our scope by following pertinent citation trails.

A two-stage selection process was employed to ensure the relevance and quality of the literature. First, we removed duplicates and applied strict inclusion criteria: publications had to be dated after July 2017 (the release date of the SHACL standard [64]), appear in peer-reviewed scientific journals or conference proceedings, and offer an English full-text version accessible either freely or through Leuphana University's institutional access. Second, we manually screened titles and abstracts of the remaining publications for thematic relevance. For example, we excluded works generating SHACL from structured data such as RDF graphs [73, 81], RDF mapping languages [23, 57], or ontologies [18, 85], as these approaches benefit from pre-existing structural cues in the input. This resulted in a corpus of seven publications directly addressing the conversion of NL to SHACL shapes, presented below.

Most existing work relies on the manual creation of SHACL shapes from text across diverse domains. This includes translating FAIR best practices for research software [54] to assess the quality of GitHub repositories [53], as well as encoding legal norms – like Ghanaian petroleum industry regulations [2] or licensing terms [93] – for automated compliance checking. Furthermore, in the AEC domain, Nuyts et al. [80] demonstrate a SHACL-based validator for a building regulation on accessibility, and Hagedorn and König [44] use the standard to validate adherence of Building Information Models (BIMs) to technical requirements.

By focusing on eligibility criteria for social benefits, we introduce a novel domain to this body of work. This distinction is significant because application-specific characteristics fundamentally shape the manual workflow. For instance, Hummel et al. [53] must operationalize the broad FAIR principles into verifiable criteria, while studies focusing on technical and legal regulations [2, 80, 93, 44] face a narrow interpretive scope, which they navigate with domain expertise and application-specific methods, such as the RASE mark-up language for AEC regulations [80]. Given this diversity, our novel use case warrants a tailored methodology and examination of the associated challenges, which we address throughout Section 5 in answer to Q1 and Q2.

Moreover, we extend the previous studies by investigating the automation of the Text2SHACL task to reduce modeling overhead and foster scalability. In contrast to Senkýr [98] and Donkers and Petrova [28], who map individual sentences to predefined SHACL templates based on syntactic analyses with established NLP methods, we propose an LLM-driven end-to-end approach. We share with those works

the focus on the standardized SHACL Core constraint components that are more amenable to automation than the custom SHACL-SPARQL constraints featured in most of the manual methods [2, 80, 93, 44]. However, with the prompt engineering approach presented in Section 6 we overcome the limitations of template-based methods, which require detailed prior knowledge of the rules and struggle with the inherent variability of NL [28]. This is particularly important because the social benefit requirements we investigate are often verbose with minimal structure, unlike the more predictable hypothetical descriptions [98] and fire safety regulations [28] previously explored.

SHACL shapes graphs represent a syntactically constrained subset of RDF graphs [64] — a close correspondence that we leverage by building on the text-to-knowledge graph (Text2KG) literature in several respects:

- **Task Definition:** Our definition of the Text2SHACL task (§ 4) is informed by Mihindukulasooriya et al. [74], who present a benchmark to evaluate LLMs' abilities to construct ontology-compliant knowledge graphs from NL.
- **Evaluation Metrics:** We employ graph matching metrics validated in prior Text2KG research [4, 41, 46] to evaluate the quality of LLM-generated shapes graphs against a human-generated ground truth.
- **Prompt-Based Methods:** The use of LLMs in knowledge graph construction is comparatively mature [84], and the promising application of prompt-based methods [41, 46, 94] motivates our exploration of prompt engineering for automating Text2SHACL.

While Text2KG research has laid important groundwork, the more restrictive syntax and greater logical intricacy of SHACL as opposed to unconstrained RDF, warrants a distinct examination of LLMs' abilities to perform Text2SHACL with tailored prompts (**Q3**) in Section 6.

4 Task Definition

So far, we described Text2SHACL as the task to convert a NL text into a SHACL shapes graph. The related work presented in section 3.2 highlights several important nuances. First, input texts can contain requirements, rules, or criteria at varying conceptual levels, from general principles [53] to specific legal rules [2]. Second, to support downstream tasks, the generated shapes must conform to the target ontology - whether custom-built [2] or pre-existing, such as the Software Description Ontology [53] - by adopting the data-graph terminology in line with its formal semantics. Third, the output may range from a restricted set of SHACL Core constraint components [98] to the full expressiveness of SHACL, including SHACL-SPARQL constraints [2].

To consolidate this diversity, we build on Mihindukulasooriya et al. [74] who frame Text2KG as a "fact extraction task guided by an ontology". Analogously, we understand Text2SHACL as a *constraints extraction task guided by an ontology*. This task takes two primary inputs: (1) a natural language text containing normative statements (e.g., rules, requirements, or criteria), and (2) an ontology that defines the concepts and properties needed to express the normative statements from (1) that are relevant to the downstream validation task. While not the focus of this work, we note that the notion of ontology could, in principle, be interpreted more broadly to include other representations of the target terminology, such as illustrative data graph examples. The desired output is a set of well-formed SHACL shapes that accurately reflect the constraints expressed in the text and conform to the ontology. Ultimately, all quality criteria converge on the functional requirement that the generated SHACL shapes should validate the target data graphs as intended by the input text.

While inspired by Mihindukulasooriya et al. [74], this implies several important distinctions from their formulation of Text2KG. First, our definition targets normative rather than descriptive input texts. Second, illustrative input-output pairs are no mandatory input component of Text2SHACL to keep the task specification method-agnostic. Finally, we expect the output to consists of syntactically valid SHACL rather than simplified triple verbalizations - reflecting our emphasis on executable output that is directly usable in downstream validation workflows.

We formally express Text2SHACL as follows:

Let $\mathcal{T}$ be the set of all possible texts, $\mathcal{S}$ the set of all well-formed SHACL shapes graphs, and $\mathcal{G}$ the set of all possible RDF graphs that can be constructed from an ontology O.

The goal of Text2SHACL is to learn a multi-valued function

$$f : \mathcal{T} \longrightarrow 2^{\mathcal{S}} \tag{1}$$

that maps any text t, given O, to the set of SHACL shapes graphs $S \subseteq \mathcal{S}$, such that for all $g \in \mathcal{G}$, g conforms with S if and only if it satisfies the constraints expressed in t as interpretable under O.

In this work, we restrict $\mathcal{T}$ to a finite set of NL descriptions of eligibility criteria for social benefits, and $\mathcal{G}$ to a collection of synthetic RDF graphs simulating user input for automated eligibility assessment. O is a task-specific, custom-built ontology. The remainder of this work explores two approaches to finding f: human annotation (Section 5) and prompt engineering (Section 6).

5 Dataset

The initial phase of our research investigates a manual methodology for representing social benefit eligibility requirements using SHACL. The objective is to develop a nuanced understanding of the necessary steps and specific challenges involved in performing the Text2SHACL task in this domain.

5.1 Methodology

5.1.1 Requirements Texts

The goal is to collect NL descriptions of eligibility criteria for German social benefits, which we term "requirements texts" for brevity. To define the analytical scope of our corpus, we adopt the definition of "social benefit" as outlined in §11 of the First Book of the German Social Code (Sozialgesetzbuch I, SGB I), which encompasses the services, benefits in kind, and monetary payments granted under the SGB, including the laws referenced in §68 SGB I [13].

Guided by this definition, we assembled requirements texts using the Suchdienst Application Programming Interface (API)[18] provided by the PVOG. The PVOG is a platform operated under the auspices of the German IT Planning Council that aggregates and distributes metadata and descriptions of administrative services [55]. Seventeen editorial systems - one per federal state and one at the national level - autonomously author and maintain the service descriptions, which are periodically synchronized with the PVOG database [21].

Because the API does not support direct filtering for social benefits, we first collected the full set of publicly available administrative services using two endpoints. The first endpoint returns a catalog of all services associated with a given administrative region [103]. Each entry is assigned a unique code consisting of an editorial system prefix and a local numeric identifier. For example, a national-level service might have the code `B100019LB123456`, where 123456 is its identifier. These so-called IDLBs served as input to the second endpoint, which returns complete service descriptions in JSON format. Using this approach, we downloaded 17,166 descriptions on January 20, 2025, and subsequently identified a relevant subset of social benefits in a three-step selection procedure.

[18] https://anbindung.pvog.cloud-bdc.dataport.de/api/suchdienst

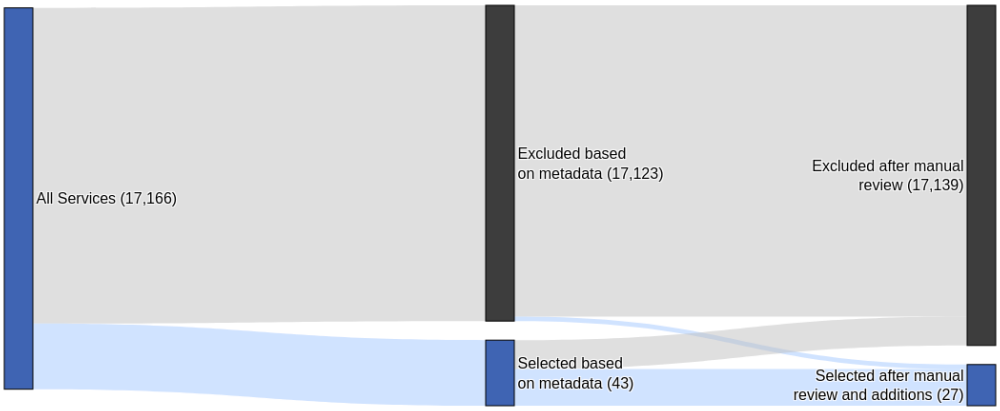


Figure 7: Benefit selection process. This schematic overview visually downscales the substantially larger flow widths of the initial set and excluded services for clarity, while preserving accurate numeric labels. 21 out of the 27 selected services were considered for annotation.

First, we applied a series of rule-based filters using the metadata fields. Specifically, we selected services labeled as "Sozialleistungen" (social benefits) to ensure domain relevance, limited the scope to entries from the national-level editorial system to avoid duplication and regional inconsistencies, and retained only the services targeting natural citizens because they are the intended end users. Additionally, we required a non-empty requirements text field, which serves as the primary input for the Text2SHACL task. This process resulted in 43 candidate services.

Second, we manually reviewed the pre-filtered services to ensure their legal relevance and textual quality. During this step, we excluded duplicate entries, services no longer in effect as of January 27, 2025, and those lacking a legal basis in the SGB or laws referenced by §68 SGB I. Additionally, only services whose requirements texts included clear normative statements amenable to structured representation in SHACL were retained. This process reduced the candidate set to 24 services.

Third, we manually added three services that were not identified in the initial automated selection but are commonly featured on social benefit assessment platforms (Section 3.1), demonstrating their practical relevance. *Child benefit*, although being grounded in a law referenced in §68 SGB I [13] (Bundeskindergeldgesetz), was initially excluded because the metadata lacked the "social benefits" label. *Education and participation*, as well as *basic income support for old age and reduced earn-*

ing capacity, had been removed because they were unavailable through the national editorial system, but could subsequently be sourced from the Lower Saxony system.

Figure 7 gives a schematic overview of this selection process, yielding 27 legally and practically suitable social benefits. Resource constraints necessitated prioritizing a representative subset of 21 social benefits for manual annotation.

5.1.2 Annotation

While the process of constructing faithful representations of the requirements texts in SHACL was not linear, it involved iterating between five conceptually distinct tasks, presented in this section. To guide the reader through this process, we refer throughout to the schematic overview in Figure 8. Our explanations are illustrated using the example of *basic training for people with visual or hearing impairments (basic training)* alongside excerpts from other real-world social benefits drawn from our dataset.

Obtain requirements text. The natural starting point for the Text2SHACL task is obtaining the input texts. The reviewed literature illustrates the variety of conceivable sources depending on the domain of interest such as research papers [53], laws and regulations [2, 80], or construction schedules [28]. Here, requirements texts with social benefit eligibility criteria were sourced form the PVOG as described in the previous section (5.1.1). Procedurally, the requirements text serves as input to the constraints extraction task (Figure 8).

Extract constraints. Prior work frequently includes an intermediate formalization step, such as translating high-level principles into concrete quality criteria [53] or structuring contractual terms as conditional statements [93]. However, the ambiguity inherent in NL typically requires annotators to choose among multiple plausible interpretations, each corresponding to a different level of abstraction. For instance, the requirement for *basic training* to "have a severe visual or hearing impairment or be at risk of such a disability" (Figure 8) can be interpreted as either a single, aggregated constraint, or split into multiple, disjunctive constraints.

To enhance transparency in this process, we propose to explicitly articulate the intended granularity of individual constraints and iteratively refining this definition based on emerging examples. We define a constraint as a *condition specifying that a person is eligible for a benefit only if a distinctive physical or legal fact is present or absent prior to application.* This definition captures four aspects: (1) constraints

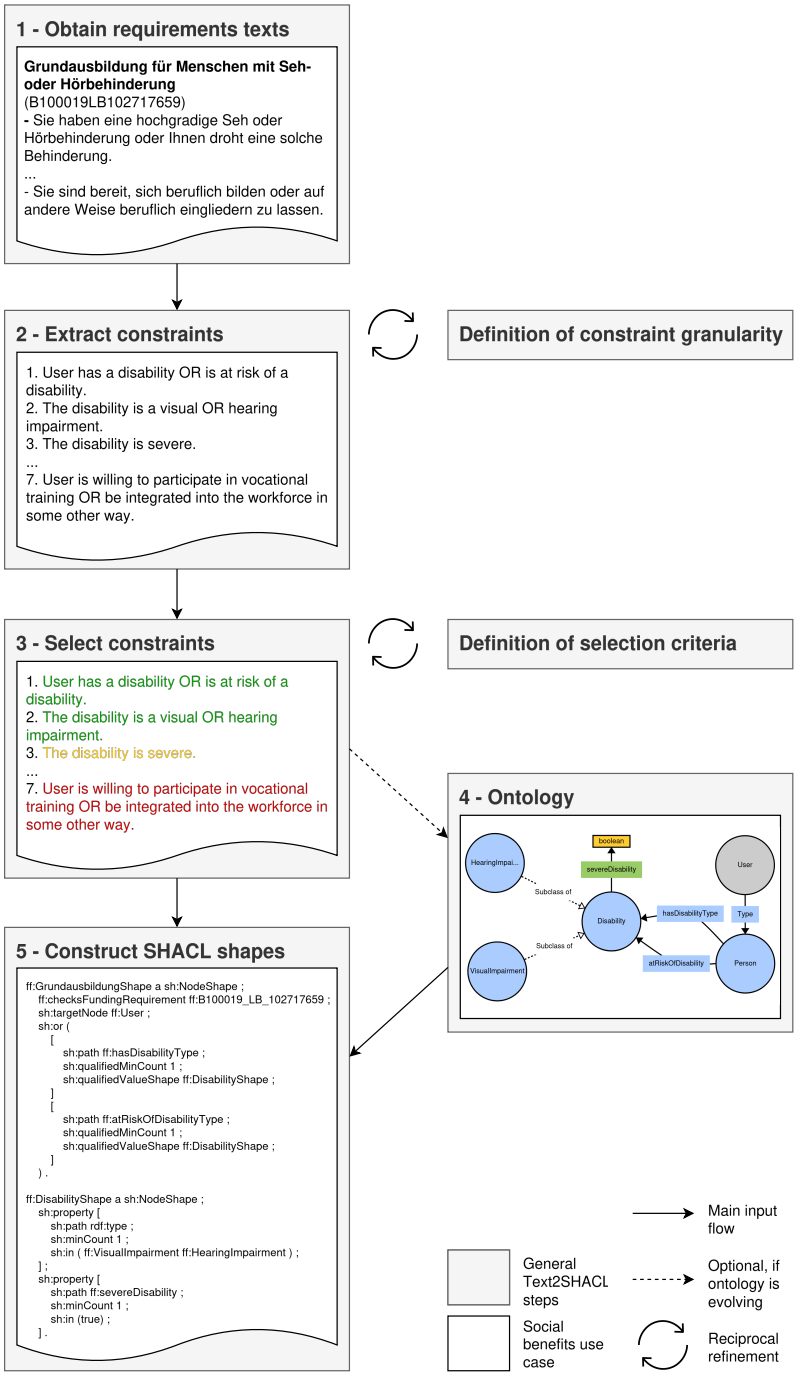


Figure 8: Process for converting NL requirements into SHACL shapes. Illustrated with excerpts from a real-world social benefit description for "Basic training for people with visual or hearing impairments". Ontology visualized with WebVOWL (https://service.tib.eu/webvowl/).

may require the presence (e.g., receiving another benefit) or absence (e.g., not receiving another benefit) of a fact; (2) such facts may be physical (e.g., a disability) or legal (e.g., reaching retirement age); (3) they must serve to distinguish between eligible and ineligible individuals; and (4) be verifiable prior to application, aligning with our goal of informing users proactively (e.g., having to request a notice to assert that one is legally an employee is excluded). Ensuring distinctiveness (3) may involve aggregating requirements when details are irrelevant for eligibility (e.g., whether unemployment was registered electronically or in person), or decomposing them when atomic elements - such as the presence, type, and severity of a disability in the *basic training* case - each independently influence the status. The latter also enhances scalability, as such fine-grained constraints are more likely to be reused across benefits, making distinctiveness relevant not only locally but corpus-wide. Finally, in our case, this step also involves translating the original German texts into English, the primary modeling language for improved accessibility and interoperability.

Select constraints. Another point to consider is the practical feasibility of operationalizing the extracted constraints within the downstream validation task. This may depend on (1) the expressivity of the chosen formalism (SHACL Core) and (2) the accessibility of the information required to assess conformance. In our approach, that information must be obtainable from end-users without legal expertise. In principle, this requires decomposing legal concepts into concrete, physical facts during constraint extraction. For example, it is more intuitive to request a person's date of birth than to ask whether they meet the legal retirement age as defined in 7a SGB II. In practice, however, many requirements texts, including this example extracted from *citizen benefit*, do not explicate such legal terms directly. Despite this limitation, we deliberately constrain our work to a single, self-contained text corpus to maintain a clear focus on the Text2SHACL task, rather than diverting attention to an analysis of external legal sources. Thus, we retain "legal facts" in our constraint definition and propose a three-tier flagging scheme for selecting tractable constraints, as illustrated in Figure 8:

- **Green (G):** The user can directly provide the required information without guidance, e.g. confirming a hearing or visual impairment.
- **Yellow (Y):** The constraint involves domain-specific terms that are not immediately understandable from the text alone but could be clarified using external sources.
- **Red (R):** The nature or complexity of the required information renders for-

malization with SHACL Core infeasible; for example, a user's "willingness" to participate in "vocational training" or "other measures" is inherently indeterminate, as it requires them to express a disposition without knowing the measures or funding opportunities.

G and Y constraints are both retained, but Y constraints require further processing before use in a downstream task - either by encoding the meaning of the disputed terms in a domain ontology and inserting an inferential reasoning step to semantically enrich the data graphs before validation, or by providing appropriate explanations during user interaction. In contrast, R constraints were excluded. This triage aims to balance faithfulness to the source with practicality within the envisioned validation workflow.

Ontology. Ontologies are a key tool for bridging the gap between NL and formalization in Text2SHACL. For instance, Donkers and Petrova [28] draw on several existing vocabularies related to fire safety regulations, while Nuyts et al. [80] re-use concepts from the Building Element Ontology[19]. However, to the best of our knowledge, no existing domain ontology fully accommodates the specific task of modeling user profiles relevant to social benefit eligibility in Germany.

Ontology development is a complex task that typically follows a rigorous methodology including iterative testing and refinement, deep domaine expertise, and a comprehensive review of related vocabularies ([83, 79], *i.a.*). As such, a full-fledged ontology development process was beyond the scope of this work. Instead, we adopted a pragmatic approach in which the ontology evolved alongside the annotation: When existing terms proved insufficient to express a selected constraint, we extended the ontology with the necessary classes, properties, or individuals using the custom `ff` namespace. We emphasized strict class hierarchies, well-defined property domains and ranges, and enriching terms with NL annotations for which we selectively incorporated foundational vocabularies such as `rdf`, `rdfs`, and `xsd`. The ontology was constructed in the open-source editor Protégé Desktop[20].

Whether an evolving or fixed ontology is employed will depend on the specific focus of a given project and the maturity of Semantic Web technologies in the target domain. This contingency is reflected in Figure 8 by the optional linkage representing how the identified constraints may impact the ontology. Irrespective of the chosen approach,

[19] https://pi.pauwel.be/voc/buildingelement/index-en.html
[20] https://protege.stanford.edu/

the solid linkage from the ontology to the SHACL shapes graph underscores the indispensable role of a shared ontology in enabling interoperability between the data and shapes graphs.

Construct SHACL shapes. The final step is to map the selected constraints to suitable constraint components from the SHACL Core vocabulary. The sample shapes graph in Figure 8 illustrates key characteristics of our human-authored shapes graphs. First, each benefit is represented by a main node shape, named after the benefit (e.g., `ff:GrundausbildungShape`) and referencing its IDLB to disambiguate the associated administrative service. Second, this main shape targets the node `ff:User`, reflecting that we adopt the vantage point of the individual using the downstream application and purposefully forego a more generic class target, such as `ff:Person`, to be able to distinguish other involved individuals (e.g., a dependent child). Third, while the verbalized constraints suggest linearity, the tightly coupled network of property and node shapes in their SHACL representation reveals the structural complexity of the underlying semantics. For the construction of SHACL shapes, this means that the selected constraints cannot be considered in isolation but must account for interactions between constraints across multiple shape components.

5.1.3 Evaluation

To assess the quality of the generated SHACL shapes graphs, we employed a two-step evaluation procedure combining synthetic RDF data and expert annotation.

First, we manually constructed RDF user profiles based on the selected constraints and validated them against the generated shapes graphs using the pySHACL library[21]. To ensure syntactic correctness, we employed pySHACL's built-in `meta-shacl` mode, which validates the shapes graph against the meta shapes graph (§ 2.1.3). Subsequently, to assess the semantic adequacy of the shapes graphs - that is, their alignment with our interpretation of the requirements texts - the profiles were purposefully designed to either satisfy or violate specific constraints. If validation results deviated from the expected behavior, or indicated a violation of the meta shapes graph, the tested shapes graphs were revised accordingly until

We systematically built a dedicated set of synthetic RDF data for each social benefit, aimed at covering a variety of distinctive scenarios. Starting with a seed data graph conforming to all constraints, we derived additional test cases by systematically

[21] https://pypi.org/project/pyshacl/0.9.5/

varying constraint satisfaction. To ensure comprehensive yet manageable coverage, we applied tailored heuristics based on the type of each constraint:

- **Numeric range constraints** (e.g., "user must be at least 15 years old"): These restrict a value to lie within a continuous or discrete interval on the number line and are defined with respect to the interval boundaries. For each such constraint, we aimed to construct:
  - one conforming edge-case (e.g., age set to 15),
  - one non-conforming edge-case (e.g., age set to 14).
- **Categorical constraints** (e.g., "user must live in Germany"): These require membership in an exhaustively articulated set of permissible values. For each such constraint, we aimed to construct:
  - up to two conforming cases from the set (e.g., residence in Germany),
  - one non-conforming case (e.g., residence in Tuvalu).

For both types, we also accounted for missing information (e.g., age or residence are not defined). Consistent with our constraint definition, which requires both presence and absence conditions to be explicitly specified, this case was expected to result in non-conformance.

Second, two domain experts - who have worked extensively with SHACL within the German administrative context as key contributors to the FörderFunke project[22] - reviewed each shapes graph. This review aimed to mitigate potential circularity, since both data and shapes graphs stemmed from the author's interpretation of the requirements, and to ensure comprehensive coverage beyond the heuristically derived user scenarios.

The expert annotators were provided with detailed guidelines, a structured annotation sheet, and the annotation targets - namely, the original requirements texts, the selected constraints, and the shapes graphs. All materials are digitally attached to this thesis (Appendix F). The annotation sheet allowed annotators to accept or reject the proposed formalization of each social benefit, where acceptance indicated agreement with all three components of the target. In case of rejection, annotators

[22] https://foerderfunke.org/

were asked to specify which aspect(s) (benefit selection, requirements interpretation, or SHACL representation) were responsible and provide a free-text explanation.

If both experts independently accepted a target, it was directly included in the ground truth. If at least one of expert rejected a target, we approached disagreement resolution following Oortwijn et al. [82] who emphasize that the appropriate resolution strategy depends on the source of disagreement. Accordingly, we conducted a joint review session with both annotators to determine which conflicts required revisiting task definitions or annotation guidelines, aligning conceptual understanding among annotators, correcting inconsistencies or errors, or addressing interpretive differences through discussion [82]. Revised shapes graphs were then re-annotated in a second round using the same procedure. To avoid unresolved annotations from inhibiting the datasets' suitability as ground truth, a researcher from our team was prepared to act as a final arbitrator in case disagreements persisted after the second round.

To quantify the extent of inter-annotator agreement, we report Cohen's Kappa (κ) [19] after each annotation round, based on binary "accept"/"reject" labels. Kappa scores range from -1 to $+1$, where 0 indicates agreement expected by chance and 1 perfect agreement. It is formally defined as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

where p_o is the observed agreement between annotators, and p_e is the expected agreement by chance.

5.2 Results

Based on the tailored methodology proposed in the previous section, we constructed a Text2SHACL dataset for social benefit eligibility assessment, publicly available in this project's GitHub repository[23], that consists of four parts:

Requirements Texts. A curated set of 21 NL requirements texts, each outlining the name, IDLB, and eligibility criteria of a social benefit as issued by the PVOG. Since the PVOG does not guarantee the completeness of English translations, we include the original German texts. The documents vary substantially in length (min: 24, max: 405, mean: 122 words) and degree of structure. While they consistently employ itemization, items range from fragmented phrases to multi-sentence paragraphs, and

[23] https://github.com/semantic-systems/text-to-SHACL.git

are sometimes combined with continuous text. From these texts, 178 constraints were extracted, of which 59% were accepted (GREEN), 25% conditionally accepted (YELLOW), and 16% rejected (RED) during selection. Owing to the manual review of textual quality during benefit selection, all texts contain at least one accepted constraint. Full statistics are reported in Table 3.

Table 3: Distribution of extracted and selected constraints. Colors distinguish accepted (GREEN), conditionally accepted (YELLOW), and rejected (RED) constraints. Relative frequency is calculated over 178 constraints, extracted from 21 requirements texts.

Constraint type	**Absolute frequency**	**Relative frequency**	**Min per file**	**Max per file**	**Mean per file**
GREEN	105	0.590	1	18	5.00
YELLOW	44	0.247	0	8	2.10
RED	29	0.163	0	5	1.38
TOTAL	178	1.000	3	25	8.48

Ontology[24]**.** A human-authored, task-specific ontology that formally encodes the concepts needed to assess eligibility for the 21 selected social benefits. It defines 40 classes, 53 object properties, 52 data properties, and 33 individuals. As shown in Figure 9, it has a star-like topology centered on the *Person* class, reflecting that most requirements apply to personal attributes of the user and their dependents. Another cluster forms around the *Event* class, which has the most subclasses, mirroring the versatility of the concept as well as the fact that several benefits are triggered by formalized occurrences such as an insolvency or insurance event. Only three other classes have subclasses, each with a maximum of three, and limited to a single hierarchical level. 73% of data properties are Boolean-valued, alongside a smaller set of numeric and temporal types. The largest number of individuals is associated with the class *Social benefit*, which highlights the interdependent nature of benefit entitlements, where determining eligibility for one benefit may require reasoning over past and present receipt of, or entitlements to, others.

SHACL Gold. A set of 21 human-generated SHACL shapes graphs, each representing the selected constraints form one of the requirements texts using the proposed ontology. The number of triples per graph ranges from 11 to 164 (mean: 62.57), reflecting variation in constraint complexity and density in the underlying requirements texts. In total, the shapes graphs comprise 59 node shapes and 119 property

[24] https://seikpold.github.io/social-benefits-ontology/v1-2025-06-05.ttl

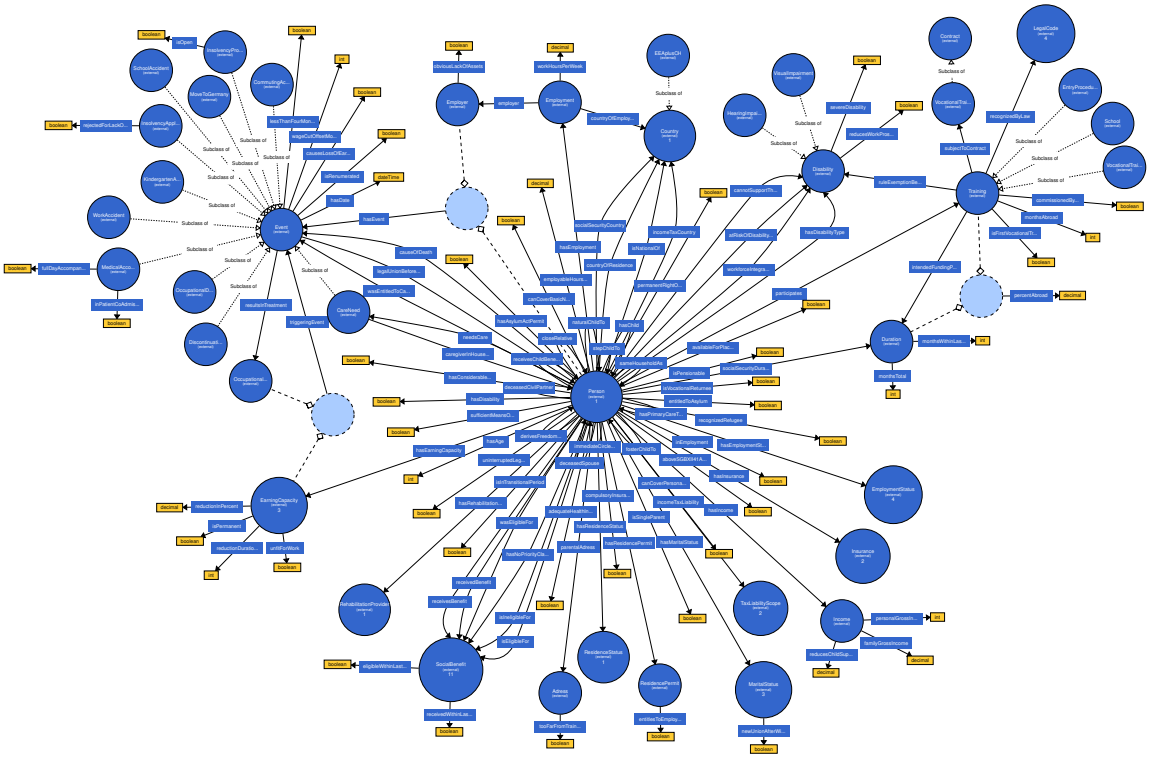


Figure 9: Ontology visualization. Dark blue circles are classes (with class names and individual counts, if any), edges are properties, and yellow rectangles are data types. All terms are from the `ff` namespace. Visualized with WebVOWL (https://service.tib.eu/webvowl/).

shapes. The only explicit target declaration is the node target *User* consistently used across all shapes graphs. We observe 508 individual SHACL constraint parameters - nearly three times the number of extracted constraints, underscoring the finer granularity of SHACL's constraint components relative to the intermediate human formalization step. Figure 10 shows that the parameters span a wide range of the constraint component types as defined in the SHACL specification [64]. The most frequent are cardinality constraints, notably `sh:minCount`, which reflects the need to ensure the presence of all relevant data for conclusive eligibility assessment. This is followed by shape-based constraints, notably `sh:property`, which enables validating values nodes against a property shape, thereby extending the validation scope deeper into the graph.

All finalized SHACL Gold graphs are unanimously accepted by the expert annotators and successfully pass the validation tests - that is, they accurately differentiate

between compliant and non-compliant data graphs in their dedicated set of synthetic user profiles and conform with the syntax rules in the meta shapes graph. High inter-annotator agreement after the first round ($\kappa = 0.91$), which increased to perfect agreement ($\kappa = 1$) following disagreement resolution, demonstrates the reliability of the annotation procedure and the robustness of the resulting gold standard. Arbitration was not required.

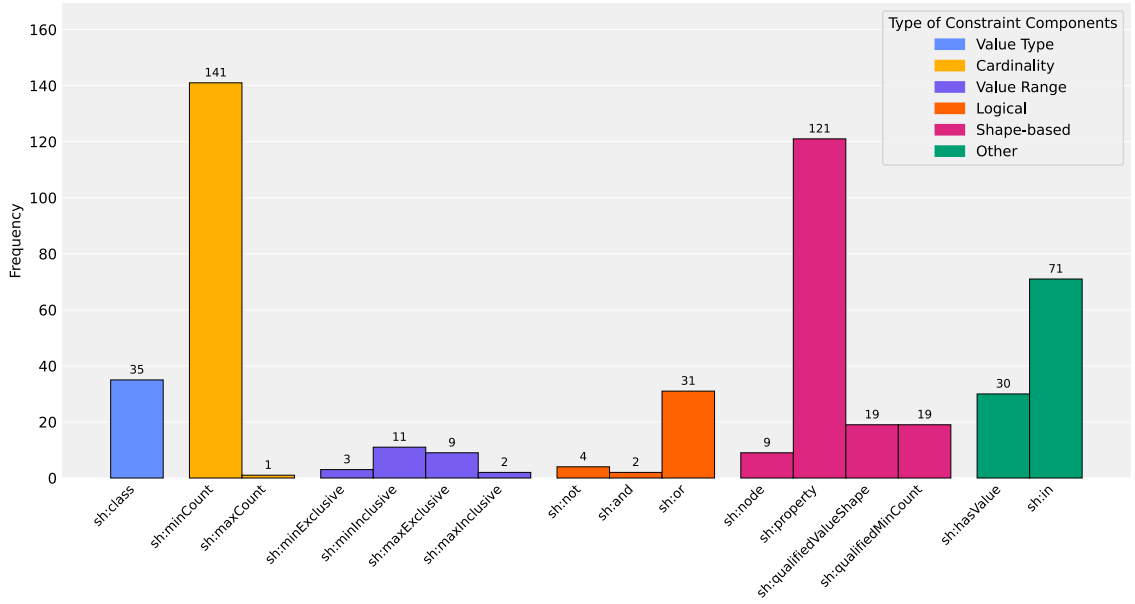


Figure 10: Distribution of constraint parameters in SHACL Gold. *X-axis:* Constraint parameters grouped by their type as defined in the SHACL specification [64]. *Y-axis:* Number of occurrences across all 21 human-generated shapes graphs.

RDF User Profiles. A set of 314 synthetic RDF user profiles composed of 21 subsets, designed to systematically probe one of the generated SHACL shapes graphs. The number of profiles per graph ranges from 4 to 29 (mean: 15), mirroring the variation in the size of the shapes graphs. On average, 37% of the profiles per shapes graph were purposefully designed to be eligible for the corresponding benefit. The systematic inclusion of non-conforming data graphs due to missing values in addition to inaccurate values accounts for the moderate bias toward ineligible cases.

5.3 Discussion

5.3.1 Text2SHACL Process

In response to our first research question - *how can eligibility requirements for social benefits be systematically translated from natural language text into SHACL shapes graphs?* - the results demonstrate that the methodology presented in Section 5.1 supports a robust manual conversion of requirements texts to shapes graphs. Beyond its practical utility, the schema yields two more general insights relevant to the broader Text2SHACL task.

First, the mapping process is fundamentally shaped by assumptions about the completeness of the ontology. Under an OWA, the vocabulary remains extensible, allowing new concepts to be introduced dynamically as texts are processed. In this case, the scope of the resulting shapes graph is primarily determined by the input text, with the ontology ensuring terminological consistency. Conversely, the CWA treats the ontology as a complete conceptualization of the target vocabulary. Consequently, the ontology's expressivity limits what aspects of the input can be formalized. While our manual annotations adopt the OWA to account for the absence of a predefined ontology, the prompting experiments in Section 6 deliberately assume a fixed ontology to guide constraint selection by the LLMs.

Second, although the proposed steps capture recurring patterns in existing Text2SHACL literature and is generally domain-independent, their concrete implementation is closely tied to the intended use of the resulting shapes graphs. In our validation scenario, data graphs are derived from user queries, requiring the constraint selection to consider which information users can realistically supply. The constraint definition similarly accounts for the user, addressing their application status. By contrast, if the goal were to validate completed applications, the same requirements texts could be used, but the interpretive lens might shift toward alignment with predefined fields in application forms.

A central motivation for the proposed framework is to enhance transparency and reproducibility by explicitly articulating the interpretive assumptions and design choices that underlie the process of translating NL text to SHACL. Nonetheless, the manual execution of the task also brought to light several persistent modeling challenges that will be discussed in-depth in the next section.

5.3.2 Text2SHACL Challenges

In answer to our second research question - *what challenges hinder the faithful representation of eligibility requirements for social benefits with SHACL Core?* -, this subsection discusses systemic obstacles encountered while manually performing Text2SHACL on social benefit eligibility requirements. They fall into two broad categories: (1) Limitations of SHACL Core and (2) Domain-specific formalization challenges.

Limitations of SHACL Core. We identify three recurring types of eligibility requirements from the dataset that are, in principle, formalizable, but push the expressive boundaries of SHACL Core. We define them as follows:

- **Arithmetic requirements** involve the application of numeric operations like summation, which SHACL Core does not support. For example, eligibility for *unemployment benefit* requires that a person "has been subject to compulsory insurance for at least 12 months..." and specifies various periods that can be counted toward this total. Thus, rigorously evaluating this rule requires summing over multiple durations first.

- **Temporal requirements** specify that a given condition must be satisfied either at a particular point in time or within a defined temporal interval. For instance, the previously cited requirement for *unemployment benefit* continues: "...within the last 30 months before registering as unemployed". While some SHACL Core parameters allow for basic operations on date-types (e.g., `sh:lessThan`), it can neither compute time intervals with respect to a specific date nor condition constraints on a temporal reference.

- **Defeasible requirements** can be overridden by other requirements [93]. For instance, *transitional allowance for people with disabilities* requires 12 months of social insurance within the last 3 years. However, the 3-year-limit for this (and other) requirements is waived for vocational returnees with disabilities. While SHACL Core allows neutralizing a shape with `sh:deactivated`, it lacks mechanisms for *conditional* deactivation to directly model such exceptions.

To handle arithmetic and temporal constraints, we adopted a pragmatic workaround by encoding the outcome of complex computations in boolean-typed data properties such as `ff:compulsoryInsuranceFor12In30` that indicate whether the relevant conditions holds (e.g., whether a person has been insured for at least 12 months

within the last 30 months). Although this approach enables SHACL to validate these higher-level abstractions, it merely defers the necessary reasoning steps: Either to the user, who must manually sum up the relevant insurance periods in the specified time window, or to an external rule system that pre-computes these values from lower level user input before validation. This reduces the overall expressiveness of the KB, as key information remains embedded in unstructured NL definitions of abstract terms rather than being formally defined. It also undermines re-usability, since the resulting properties are tailored to highly specific use cases.

For defeasible constraints, we used SHACL's `sh:or` parameter to model alternative eligibility paths. For example, a user must either comply with a node shape that represents conditions for vocational returnees, or to another one that encodes the default case. Although functional, this approach introduces repetition: nearly identical constraints must be copied across multiple shapes, with only minor variations. Furthermore, this technique necessitates significant logical restructuring of the original requirements - from a "default-plus-exception" format to a "scenario 1.1, scenario 1.2,..." branching structure. This transformation increases the risk of interpretive inconsistencies and amplifies the structural complexity of the resulting shapes graphs.

In sum, given its standardized syntax and broad implementation support, SHACL Core is a practical choice for many validation tasks, especially when considering automatic generation of SHACL shapes. Nonetheless, recognizing the described limitations in the context of social benefit requirements is essential. Workarounds may come at the cost of additional complexity, reduced modularity, or limited machine interpretability, and may necessitate the construction of more precise SHACL SPARQL constraints depending on the application requirements.

Domain-Specific Formalization Challenges. The second group of challenges comprises types of requirements that pose fundamental obstacles to formalization. These challenges arise not from technical limitations, but from the nature of the requirements themselves, which resist reduction to deterministic constraints, including:

- **Self-referential requirements** relate to mutable personal characteristics, where a person's willingness or ability to change them may depend on their eligibility for funding. For example, one requirement for *training allowance* is that a person "will participate in one of the following measures..." and continues to list qualified educational measures - but whether a person is willing and

able to participate in one of them may depend on the receipt of *training allowance*. Thus, it circularly conditions eligibility on a criterion that may itself depend on the outcome of eligibility assessment.

- **Discretionary requirements** grant administrative authorities a legally sanctioned range of freedom in decision-making when determinative criteria are absent [6]. For instance, *funding for activation and professional integration measures* requires that "the need for the benefit [is] determined in a discussion with [the applicant's] integration specialist" without detailing the rules guiding that determination, which suggests that the specialist is granted discretionary power. It should be noted that we use the term discretion in a descriptive sense with respect to the given text corpus and do not account for possible specifications in external legal sources.

We address both cases by excluding them from formalization with an *R*-flag. On the one hand, this decision weakens the bindingness of the validation results in the downstream application as some requirements are not being assessed. On the other hand, they are grounded in pragmatic and ethical considerations.

We exclude self-referential requirements to avoid prematurely deeming users ineligible and thereby withholding information about potentially relevant benefits. This approach is intuitive in the case cited above where the requirement concerns the decision to engage in a funded program. However, identifying requirements as self-referential entails subjective judgments about a user's capacity or motivation to change their circumstances. For example, that someone working two hours fewer than required would be willing and able to adjust their schedule if they knew it unlocked a benefit is theoretically possible, but considerably less clear in practice. This reflects a trade-off between overwhelming users with conditional entitlements and failing to inform them of easily attainable benefits. One way to navigate this tension is by capturing users' preferences and flexibility before eligibility validation, as demonstrated in the current interaction flow in the FörderFunke prototype[25].

One evident reason to exclude discretionary requirements is that it is practically infeasible to formalize an outcome that is not conditioned on concrete criteria. However, even if approximating a formalization were possible by analyzing precedents or external sources, its desirability is questionable. Discretion is essential in administrative law, preventing cases not foreseen by the legislature from paralyzing the

[25] https://foerderfunke.org/

administration [6]. Furthermore, full automation may challenge Article 22 of the General Data Protection Regulation (GDPR), which protects individuals from decisions with legal or similarly significant effects made solely by automated systems [33]. As the German Ethics Council notes, even systems intended to merely support decision-making risk undermining human agency and accountability because humans are susceptible to automation bias, i.e. the tendency to defer unreservedly to machine-generated recommendations [25]. Given that automated eligibility assessment can also be used to support decisions about real applicants, we exclude discretionary requirements not only for practical reasons but also out of a normative commitment to safeguarding human judgment in individual case assessments.

6 Prompting Experiments

The second phase of our research targets the potential of LLMs to automate the Text2SHACL task. The goal is to build on the insights from the first part and state-of-the-art (SOTA) language modeling to examine the feasibility of a scalable, end-to-end system to perform Text2SHACL in the social benefits context.

6.1 Methodology

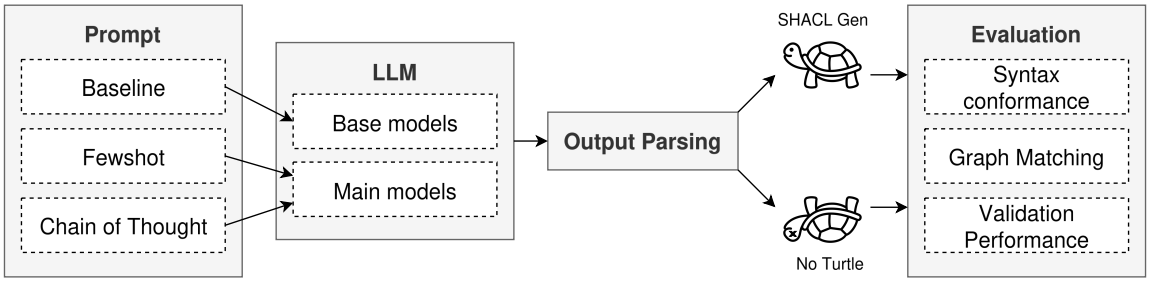


Figure 11: Overview of experimental setup.

We conduct three prompting experiments - Baseline (Base), Few-Shot (FS), and Chain-of-Thought (CoT). For each experiment, we generate a prompt by automatically populating a manually constructed template tailored to the respective prompting technique (Section 6.1.1). These prompts are used to invoke several LLMs, with FS and CoT experiments using a subset (main models) from the initial model selection (base models) (Section 6.1.2).

For each successful model call, we attempt to extract valid Turtle output by first locating candidate RDF content from fenced code blocks marked with ```` ```turtle ````, or from RDF indicators such as `@prefix`, `@base`, `<http://`, and `PREFIX`, and then parsing the identified substring with `rdflib`[26] to confirm syntactic validity. If valid Turtle is produced, it is saved (SHACL Gen).

Finally, during evaluation we consider the logged metadata of each iteration and the parsed output, if any, to assess performance on three groups of metrics: syntax conformance, graph matching, and validation performance (Section 6.1.3). An overview

[26] https://rdflib.readthedocs.io/en/stable/

of this experimental setup is presented in Figure 11, with further implementation details discussed in the following subsections.

Throughout this chapter, we refer to an *experiment configuration* as a specific combination of prompting technique and LLM (e.g., FS + LLaMA 8B), a *run* as the processing of the full dataset (i.e., 21 requirements texts) under a given configuration, and an *iteration* as the processing of a single requirements text.

6.1.1 Prompt Construction

To construct the LLM prompts, we adapt the ZS, FS, and CoT prompting techniques introduced in Section 2.2.2 to our use case. Figures 12 and 13 illustrate the prompt templates employed in each experimental condition, with blue font highlighting variable components that are dynamically populated at runtime.

Base. In the base experiment, we apply ZS prompting, providing only the minimal information required to perform the Text2SHACL task. The template (Figure 12) comprises three variable components:

1. The NL **directive** specifies the task objective and strict requirements for the output format such as compliance with SHACL syntax, Turtle serialization, and exclusive use of SHACL Core and the provided ontology.

2. The **ontology**, developed as detailed in Section 5.1.2, defines the vocabulary over which the generated shapes graphs are constructed and applied. It is serialized in Manchester Syntax [51] that closely resembles NL, to cater to the NLP capabilities of LLMs.

3. The **input** consists of an original requirements text, enriched with the benefit name and IDLB to enable the LLM to annotate shapes graphs consistent with the human-authored ground truth. It is the only prompt component that changes at each iteration.

Beginning with a human-authored task description, we iteratively refine the directive and prompt template, alternating between visual inspection of outputs and LLM-assisted revisions, inspired by recent work that leverages LLMs for prompt optimization [122, 89]. These preliminary explorations focus on three test benefits and are conducted using mistral-large-instruct, one of the base models.

Fewshot. In the FS experiment, we modify the base template in two ways (Figure 13). First, we add a section for examples, each comprising a requirements text and its corresponding ground truth shapes graph (Figure 13b). Second, we revise the directive to reference the examples, introducing only minimal changes to the base phrasing in order to isolate the effect of incorporating the FS example. The effectiveness of the FS approach is highly dependent on the choice of examples [70, 69], requiring careful consideration of two key questions: (1) how to define the pool of candidate examples, and (2) how to select the most suitable examples from this pool. We address these challenges using K-Fold Cross-Validation (KFCV) to dynamically construct the candidate set and embedding-based retrieval for selection.

```
{base-directive}

--- BEGINNING OF ONTOLOGY ---

{ontology}

--- END OF ONTOLOGY ---

Text:

{input}

SHACL:
```

Figure 12: Base template. The prompt includes a directive, the ontology as additional information, and a requirements text as input. Blue font indicates variable components.

KFCV is widely used for estimating the out-of-sample performance of machine-learning models [56]. The idea is to split the dataset into k approximately equal subsets (folds), successively holding out one fold $\mathcal{D}^{\text{test}}$ for evaluation and using the remaining *k–1* folds $\mathcal{D}^{\text{train}}$ to train the model, and finally averaging the evaluation results across folds [3]. Algorithm 1 outlines our adaptation of this procedure for the FS setting. Unlike traditional machine learning, it does not involve retraining the LLM. Instead, $\mathcal{D}^{\text{train}}$ serves as the pool of candidate examples, while $\mathcal{D}^{\text{test}}$ comprises the prompt inputs [97]. This strategy makes full use of the limited dataset by evaluating all 21 instances and reduces sampling variance compared to a single static split. In our implementation, we use $k = 3$.

Embedding-based retrieval draws on semantic similarity in an embedding space to identify the most relevant examples for a given input, which has proven highly effective in prior work [36, 69]. Following this strategy, we compute input embeddings using the `all-mpnet-base-v2` model [101], a transformer-based encoder that maps sentences to 768-dimensional dense vectors. For each test input x_i, we use LangChain's `SemanticSimilarityExampleSelector` class[27] to retrieve the exam-

[27] https://python.langchain.com/api_reference/core/example_selectors/langchain_

ples from $\mathcal{D}^{\text{train}}$ whose embeddings have the highest cosine similarity to that of x_{i}.

Notation. Let $x_i \in \mathcal{X}$ be a requirements text, $y_i \in \mathcal{Y}$ the corresponding ground truth shapes graph, k the number of folds, M an LLM that generates an output $\hat{y}$ given an input x and a set of examples $\mathcal{E}$, and $S(\hat{y}, y)$ a scoring function - representing our evaluation framework (Section 6.1.3) - that compares $\hat{y}$ against ground truth y. Then:

Algorithm 1 k-fold cross-validation for fewshot prompting.

Require: $D \leftarrow \{(x_1, y_1), \ldots, (x_N, y_N)\}$
Require: $k \in \mathbb{Z}^+$ such that $k < N$
Require: $M : \mathcal{X} \times \mathcal{E} \rightarrow \hat{\mathcal{Y}}$
Randomly partition D into k disjoint subsets: $D_1, \ldots, D_k$
for $i = 1$ to k **do**
    $\mathcal{D}^{\text{test}} \leftarrow D_i$
    $\mathcal{D}^{\text{train}} \leftarrow D \setminus D_i$
    for all $x \in \mathcal{D}^{\text{test}}$ **do**
        Embedding-based retrieval of $\mathcal{E}$ from $\mathcal{D}^{\text{train}}$
        Generate output $\hat{y} \leftarrow M(x, \mathcal{E})$
    end for
end for
return Mean score over all predictions

$$\frac{1}{|\hat{Y}|} \sum_{j=1}^{|\hat{Y}|} S(\hat{y}_j, y_j)$$

Chain-of-Thought. In the CoT experiment, we employ the same example retrieval strategy as for FS but modify the example template to accommodate the demonstration of a step-by-step solution process (Figure 13c). CoT [118] and related prompting techniques [123, 115] are based on the idea to emulate human cognition. Following this approach, we re-use the methodology for manually performing Text2SHACL presented in Section 5.1.2. Specifically, we decompose the thought process into three steps: (1) translation of the original German text to English, with identification of verbatim elements (benefit name and IDLB) intended for direct incorporation into the output; (2) extraction and selection of individual constraints based on our three-tier flagging system; and (3) conversion of the selected G and Y constraints to

`core.example_selectors.semantic_similarity.SemanticSimilarityExampleSelector.html#main-content`

SHACL. We populate the CoT example template with the correct solution, given an example input, for each step.

```
{fs-directive | cot-directive}

--- BEGINNING OF ONTOLOGY ---

{ontology}

--- END OF ONTOLOGY ---

--- BEGINNING OF EXAMPLES ---

{fs-examples | cot-examples}

--- END OF EXAMPLES ---

TEXT:
{input}

SHACL:
```

(a) FS and CoT prompt template.

```
TEXT:
{input}

SHACL:
```turtle
{shacl}
```
```

(b) FS example template.

```
TEXT:
{input}

SOLUTION:
Step 1 - Understand the input text. Translate it from German to
   English and structure the relevant information:
    - Benefit name (German and English)
    - IDLB identifier
    - Eligibility requirements (in English)

{structured-text}

Step 2 - Break down the requirements text into individual
   eligibility requirements. Decide for each requirement whether
   to include it in the SHACL graph by assigning it one of the
   following labels:
    - GREEN: A layperson can easily assess if they meet the
   condition -> Include
    - YELLOW: A layperson could likely assess if they meet the
   condition with some explanation -> Include
    - RED: The requirement is ambiguous or is at the discretion
   of third parties and cannot be clearly assessed -> Exclude

{decomposition}

Step 3 - Generate the SHACL graph in Turtle syntax. Model only
   the GREEN and YELLOW requirements using ontology terms and
   SHACL Core components. The final answer is:

```turtle
{shacl}
```
```

(c) CoT example template.

Figure 13: FS and CoT templates. The FS and CoT prompts (a) extend the base templates with tailored directives and one or more examples. A single FS example (b) provides an input–output pair for SHACL generation, while a single CoT example (c) demonstrates step-by-step reasoning from input text to SHACL output. Blue font indicates variable components.

For both FS and CoT, we restrict prompts to a single example to isolate the effect of prompt type - direct input-output pairs versus reasoning-based demonstrations - in our initial exploration of the task. Nonetheless, the system is designed to accommodate an arbitrary number of examples, presenting a potential direction for future investigation.

All materials for prompt construction are available in the GitHub repository[28], in-

[28] https://github.com/semantic-systems/text-to-SHACL.git

cluding the ontology, experiment-specific directives and the complete prompt string for each iteration.

6.1.2 Model Selection

The model selection is conducted in two steps, combining theoretical and empirical considerations. First, we identify a set of candidate models for the base experiment based on pre-defined selection criteria. Second, the three models demonstrating the most promising performance in the base experiment are selected for further analysis.

Four principal considerations guide model selection in the first step. First, all deployed models are required to be open-source in acknowledgment of the sensitive personal data to be processed and the high level of accountability mandated in the public sector. Second, we focus on instruction-tuned models to meet the demand for strict adherence to the distinctive task description and output format. Third, given the voluminous prompts including, among other components, a comprehensive ontology for the entire dataset, large context window sizes were prioritized. Specifically, all selected models support a context length of at least 128K tokens, with qwen-qwq-32b (QwQ) extending this capability to 131K tokens [90]. Fourth, we purposefully sample models with different optimization priorities, allowing us to empirically assess their effect on task-specific performance in step two of the selection process.

To implement this strategy in practice, we source models from a range of LLMs offered through the Chat AI platform. Chat AI is an independent service offered by the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) that provides access to SOTA LLMs hosted on local high-performance computing (HPC) systems. The service guarantees that data is neither stored nor used for secondary purposes without explicit consent, making it a well-suited alternative to commercial providers in terms of both computational and ethical requirements [29]. More technical details are available in the Chat AI documentation[29].

As a result of the considerations outlined above, we select the following seven models for the base experiment:

- **llama-3.1-8b-instruct (Llama 8B)**: Released as part of Meta's Llama 3 collection in July 2024 [30], this model represents the most computationally efficient option in the selection and is well-suited for lower-resource environments. Available at: https://huggingface.co/meta-llama/Llama-3.

[29] https://docs.hpc.gwdg.de/services/chat-ai/

1-8B-Instruct

- **llama-3.3-70b-instruct (Llama 70B)** and **qwen-2.5-72b-instruct (Qwen)**: Llama 70B [30] and Qwen [121] are selected as medium-scale models with robust overall performance from two different providers. In particular, Llama 70B allows isolating size-related effects relative to Llama 8B, whereas Qwen stands out for having the most recent knowledge cutoff in this collection, together with QwQ, in September 2024 [40], possibly a relevant factor given the relative novelty of the SHACL standard [64]. Available at: https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct (Llama 70B) and https://huggingface.co/Qwen/Qwen2.5-72B-Instruct (Qwen)

- **llama-3.1-sauerkrautlm-70b-instruct (Sauerkraut)**: VAGO Solution released this model based on Meta's Llama 3.1 70B Instruct in August 2024 [110]. Selectively fine-tuned on German-English data using the Spectrum approach, Sauerkraut was chosen for its German-language capabilities that may offer an edge when processing German administrative texts. Available at: https://huggingface.co/VAGOsolutions/Llama-3.1-SauerkrautLM-70b-Instruct

- **mistral-large-instruct (Mistral)**: Scaling train-time compute, including model size, has dominated advancements in LLMs for years [48, 61]. In combination with SOTA performance on code and reasoning benchmarks, this motivated the choice of Mistral, one of the largest available models via Chat AI at the time of writing with 123B parameters [40]. Introduced by theMistralAI Team in July 2024 [77], it is available at: https://huggingface.co/mistralai/Mistral-Large-Instruct-2407.

- **qwen-qwq-32b (QwQ)** and **deepseek-r1-distill-llama-70b (DeepSeek)**: The January 2025 release of DeepSeek R1 highlighted the potential of reinforcement learning to enhance LLM performance on complex reasoning tasks [22]. To address slow inference and timeout issues with the 671B-parameter flagship, we select a smaller and more efficient version from the suite of Distill models fine-tuned on the reasoning-oriented dataset originally developed for DeepSeek R1. Based on Llama 70B that is also included in this selection, it outperforms other Distill variants on most benchmarks and isolates the effects of reasoning fine-tuning. Additionally, we evaluate QwQ [90], which is based on Qwen 2.5 and demonstrates the effectiveness of reinforcement learning and tool-based rewards even for a considerably smaller, 32B-parameter model. This facilitates controlled assessment of the imminent

impact of reinforcement learning without the intermediary support of reasoning fine-tuning. Available at: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-70B (DeepSeek) and https://huggingface.co/Qwen/QwQ-32B (QwQ)

After the base experiments, we retained Sauerkraut and Mistral, which demonstrate the strongest syntactic abilities and consistently rank among the top-three models across all metrics, as well as QwQ, notable for its robust validation performance, as main models in the FS and CoT experiments. The selection and all results are detailed in Section 6.2.

6.1.3 Evaluation

From the task description (Section 4), we derive two core criteria for evaluating a successful Text2SHACL system: *syntax conformance* and *semantic consistency*. Syntax conformance requires that the system's output adheres to the SHACL syntax rules. To distinguish general RDF serialization capabilities from SHACL-specific syntactic correctness, we separately evaluate *Turtle Syntax Conformance (TSC)* and *SHACL Syntax Conformance (SSC)*. Semantic consistency evaluates whether the generated shapes graph faithfully encodes all constraints from the input text that can be expressed with the provided ontology. This reflects a conceptual shift in this thesis: whereas manual generation permitted ontology evolution, automated generation relies on a fixed ontology to provide additional guidance to the LLM regarding which constraints to include.

To evaluate semantic consistency, we employ two complementary approaches. First, we measure the similarity between the generated shapes graphs and the gold graphs using established graph matching techniques, including *Graph Edit Distance (GED)*, *G-BERTScore (G-BS)*, and the triple-level metrics *Triple-F1 (T-F1)*, *Triple-Precision (T-P)*, *Triple-Recall (T-R)* [41, 46, 94]. Second, we propose assessing validation performance by comparing how a generated and a ground truth shapes graph validate a common set of relevant data graphs, which yields *Validation-F1 (V-F1)*, *Validation-Precision (V-P)*, *Validation-Recall (V-R)*, and *Validation-Accuracy (V-Acc)*.

Following, we formally define the metrics used to quantify syntax conformance and semantic consistency, and describe their practical implementation. **Turtle Syntax**

Conformance (TSC). TSC measures the proportion of outputs that produce well-formed Turtle documents within a given experiment run. Let n denote the total number of outputs, and let $t_i \in \{0, 1\}$ indicate whether the i-th output is valid Turtle. Then:

$$\text{TSC} = \frac{1}{n} \sum_{i=1}^{n} t_i$$

To determine t_i, we assess parsability with `rdflib`[30] as described in Section 6.1.

SHACL Syntax Conformance (SSC). SSC measures the proportion of outputs that are well-formed SHACL documents within a given experiment run. Let $s_i \in \{0, 1\}$ indicate whether the i-th output is valid under SHACL syntax. Then:

$$\text{SSC} = \frac{1}{n} \sum_{i=1}^{n} s_i$$

If $t_i = 0$, then s_i is automatically set to 0. Otherwise, we re-use our approach from the manual shapes graph generation based on pySHACL's[31] `meta-shacl` feature (Section 5.1.3) to determine adherence to SHACL Core syntax rules.

Graph Edit Distance (GED). GED quantifies the structural similarity between a generated shapes graph G^{gen} and a reference G^{gold} by computing the minimum cost of transforming G^{gen} into a graph isomorphic to G^{gold} [1]. It is widely used in the evaluation of LLM-generated knowledge graphs [94, 46, 41].

Let $\gamma(G^{\text{gen}}, G^{\text{gold}})$ denote the set of all possible edit sequences to transform G^{gen} into G^{gold}, and let $c(e_i)$ be the cost of edit operation e_i, i.e., the insertion, deletion, or substitution of a node or edge. Then, we adopt the following formalization from Abu-Aisheh et al. [1]:

$$\text{GED}(G^{\text{gen}}, G^{\text{gold}}) = \min_{e_1,\ldots,e_k \in \gamma(G^{\text{gen}}, G^{\text{gold}})} \sum_{i=1}^{k} c(e_i)$$

[30] https://rdflib.readthedocs.io/en/stable/
[31] https://pypi.org/project/pyshacl/0.9.5/

To ensure comparability across graphs, we normalize GED to the interval $[0, 1]$ by dividing a graph-specific upper bound, defined as the number of edits required to delete all nodes and edges from G^{gen} and insert all nodes and edges from G^{gold} [94, 46, 41]. GED = 0 indicates that the two graphs are isomorphic, while GED = 1 indicates maximal dissimilarity.

In practice, we use the `graph_edit_distance` function from the NetworkX package[32], which implements the exact algorithm by Abu-Aisheh et al. [1]. As in previous work [94, 46, 41], we set $e_i = 1$ for all edit operations. However, unlike these studies we introduce a timeout of 120 seconds to ensure computational feasibility given our large gold graphs with up to 165 triples, which compares to a maximum of 7 triples in the WebNLG+2020 [37] dataset used by Han et al. [46] and Ghanem and Cruz [41] and a reported average of 4 edges per graph in the custom dataset by Saha et al. [94]. Exploratory experiments indicate that extending the timeout to up to 10 minutes does not result in noticeable improvements (Appendix A).

Triple-Match. To account for the preservation of specific statements in addition to structural similarity, we compute Triple-Match metrics based on the exact match between RDF triples in G^{gen} and G^{gold}. Following previous work, we treat the problem as a multi-label classification task, where each unique triple across both graphs represents a label, and each graph a label set [41, 46]. Then, we compute precision (T-P), recall (T-R), and F1 score (T-F1) using the standard definitions, assuming:

- **True Positives (TP)**: A label is present in both, G^{gen} and G^{gold}
- **False Positives (FP)**: A label is present in G^{gen}, while absent in G^{gold}
- **False Negatives (FN)**: A label is absent in G^{gen}, while present in G^{gold}

In practice, we normalize all triples by lower-casing, trimming whitespace, and standardizing blank nodes to the form `_:blank`. The resulting generated and gold graphs are encoded as indicator vectors over the set of all unique normalized triples using scikit-learn's[33] `MultiLabelBinarizer` class, and precision, recall, and F1 score are computed via the corresponding scikit-learn functions.

G-BERTScore (G-BS). BERTScore measures the semantic similarity between two

[32] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.similarity.graph_edit_distance.html

[33] https://scikit-learn.org/stable/index.html

text sequences using contextual embeddings derived from a transformer-based model, and is commonly used ([108, 99, 78], *i.a.*) because it overcomes limitations of string-based metrics by robustly handling paraphrasing and long-range dependencies [27]. Given a reference sequence and a candidate, it computes the maximum pairwise cosine similarity between token embeddings, produced with the transformer-based encoder BERT [27], to estimate recall and precision, with the F1 score defined as their harmonic mean [125]. To adapt this token-level metric to graphs, Saha et al. [94] propose treating each RDF triple as a sentence, and computing G-BS in three steps:

1. Compute BERTScore F1 for all gold–generated triple pairs.
2. Solve a linear assignment problem to find the mapping between gold and generated triples that maximizes the total BERTScore.
3. Compute precision G-BS$_P$ and recall G-BS$_R$ based on the optimal assignment and report F1 as the final G-BS.

Formally, let $S \in [0,1]^{m \times n}$ be the BERTScore matrix between the gold triples $m = |G^{\text{gold}}|$ and generated triples $n = |G^{\text{gen}}|$. Let $A \subseteq \{(i,j)\}$ denote the optimal assignment that maximizes the total BERTScore.

$$A = \arg\max_{A'} \sum_{(i,j) \in A'} S_{ij}$$

Then, G-BS is defined as:

$$\text{G-BS}_P = \frac{1}{n} \sum_{(i,j) \in A} S_{ij}, \quad \text{G-BS}_R = \frac{1}{m} \sum_{(i,j) \in A} S_{ij}, \quad \text{G-BS} = \frac{2 \cdot \text{G-BS}_P \cdot \text{G-BS}_R}{\text{G-BS}_P + \text{G-BS}_R}$$

In line with prior work [46, 41, 94], we compute BERTScores using the official Python package[34] with the default RoBERTa-large model [71]. To solve the linear sum assignment problem, we use SciPy's implementation[35], which supports rectangular cost matrices, which we are likely to incur given that we do not fix the size of the generated graphs.

Validation Performance. To evaluate the semantic adequacy of the generated shapes graph G^{gen} from a functional perspective, we propose a set of metrics that

[34] https://pypi.org/project/bert-score/
[35] https://docs.scipy.org/doc/scipy/index.html

compares validation output of G^{gen} and G^{gold}. Given a set of RDF data graphs $\mathcal{D} = \{D_1, \ldots, D_N\}$, each graph $D_i \in \mathcal{D}$ is validated against the generated and the reference shapes graph. Based on the binary outcome, we define:

- **TP**: D_i conforms to both G^{gen} and G^{gold}
- **True Negatives (TN)**: D_i conforms to neither G^{gen} nor G^{gold}
- **FP**: D_i conforms to G^{gen}, but not to G^{gold}
- **FN**: D_i conforms to G^{gold}, but not to G^{gen}

Based on these labels, we compute precision (V-P), recall (V-R), accuracy (V-Acc), and F1 score (V-F1) using standard definitions implemented in `scikit-learn`[36].

Note that the validity of this evaluation depends critically on the representativeness of the data graphs $\mathcal{D}$, which should ideally include diverse and challenging positive and negative validation cases. To this end, we leverage the synthetic user profiles constructed for evaluating manually created shapes graphs (Section 5.1.3), and compute the validation metrics over the full set of profiles.

To obtain the semantic consistency metrics for an entire experiment run, we average over the results of all outputs. Unless indicated otherwise, we set all semantic consistency metrics to their worst possible value if an output is no valid Turtle; if it is valid Turtle, but no valid SHACL, we set the validation performance metrics to their worst value and compute the graph matching metrics.

To produce more reliable results in light of the non-deterministic nature of LLMs, we run each experiment configuration three times, and report the mean and standard deviation (SD) across all runs. The only exception is Qwen, where one run is omitted because it resulted in consistent timeouts, and the exact model version was no longer accessible via the GWDG API during re-runs.

[36] https://scikit-learn.org/stable/index.html

6.2 Results

6.2.1 Base

Syntax Conformance. Figure 14 summarizes syntax conformance in the base experiment. Mistral and Sauerkraut clearly outperform all other models on both TSC and SSC. For Turtle, performance declines significantly from Mistral to Sauerkraut to Llama 70B, after which TSC scores fall below 0.50. Across all models, a consistent drop from TSC to SSC highlights the increased syntactic complexity of SHACL compared to general RDF; even Mistral reaches only 0.64 on SSC, maintaining a lead of 0.13 over Sauerkraut. The performance gap to the mid-field models, Llama 70B and Qwen, is similarly large but not statistically significant due to their high SD - particularly in the case of Qwen, where results are averaged over only two runs. QwQ follows at an intermediate position, while Llama 8B and DeepSeek trail notably with scores around 0.14.

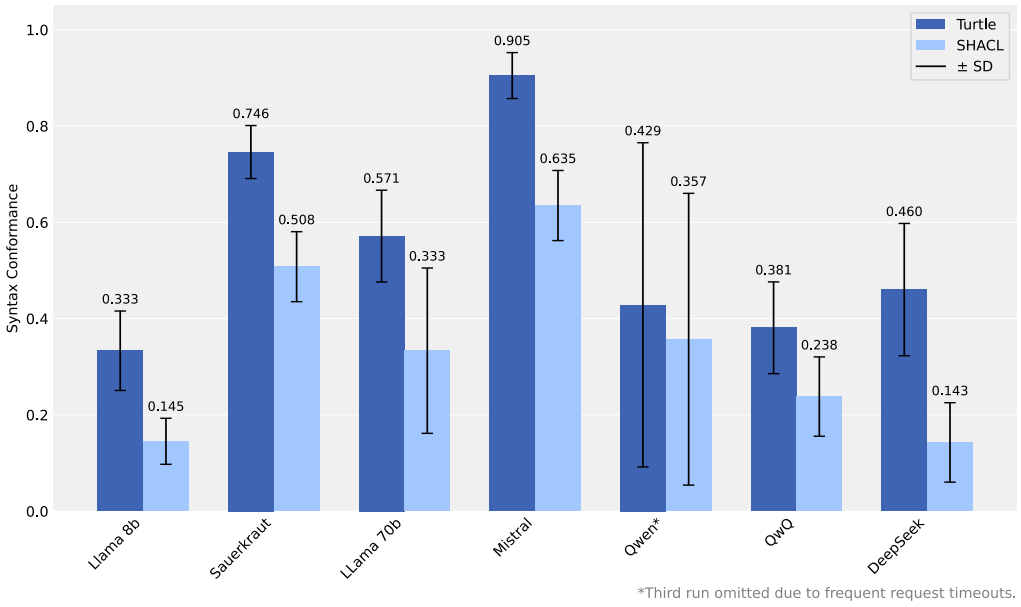


Figure 14: Baseline syntax conformance by model. *Y-axis*: Turtle Syntax Conformance and SHACL Syntax Conformance. *X-axis*: Model.

Those results indicate that LLMs frequently generate ill-formed SHACL shapes. Since these models are known to be susceptible to hallucination [52], we hypothesize that invented SHACL terms that superficially conform to RDF syntax but lack

grounding in the official SHACL vocabulary[37], might contribute to the observed syntactic failures. Although the meta shapes graph does not explicitly flag neologisms within the SHACL namespace, it may penalize the use of unknown terms indirectly, because the structural constraints naturally draw on the existing vocabulary. An analysis of the proportion of generated `sh` terms not belonging to SHACL Core (Appendix C) reveals that DeepSeek and Llama 8B are most inventive, with $\approx 14\%$, consistent with their poor syntax performance. However, for all other models the rate is negligible ($< 4\%$). Thus, while hallucination may be a contributing factor in the weakest models, it does not provide a comprehensive explanation for the observed syntactic failures.

To analyze the syntax failures in greater detail, Figure 15 presents an error analysis based on the distribution of source shapes in the validation reports. A source shape indicates the shape in the meta shapes graph that a given focus node in the generated shapes graph failed to satisfy [64]. Intuitively, each source shape corresponds to a type of syntax error. To reduce complexity, we focus on top-level errors, excluding nested violations that arise from shapes referencing other shapes through parameters like `sh:node` or `sh:or`. 17 distinct top-level source shapes are observed, with Figure 15 illustrating the average proportion of each error across all types, ordered from most (E01) to least (E17) frequent. We now examine the two most common error types across models in detail; definitions and illustrative examples of the ten most common ones are provided in Appendix B.

E01 occurs when a property shape does not have the property `sh:path` exactly once or its value is not a well-formed `sh:PathShape` or `sh:IRI`. Extracting the NL messages associated with this error type from the validation report reveals three distinct ways in which E01 is triggerd: (1) the absence of a `sh:path` property, (2) the presence of multiple `sh:path` properties, or (3) the node referenced by `sh:path` is not well-formed. Among those, (1) is the most common. An example is the shapes graph in Listing 3, generated by DeepSeek for the benefit *Funding for activation and professional integration measures*. Here, another term from the SHACL vocabulary, `sh:predicate`, is used instead of `sh:path`.

E02 indicates a violation of the disjointness between well-formed node shapes and property shapes, as defined in the SHACL specification [64]. For example, Listing 4, generated by DeepSeek, demonstrates a shape that is neither a valid node shape, due to the presence of `sh:path`, nor a valid property shape, due to multiple

[37] https://www.w3.org/ns/shacl.ttl

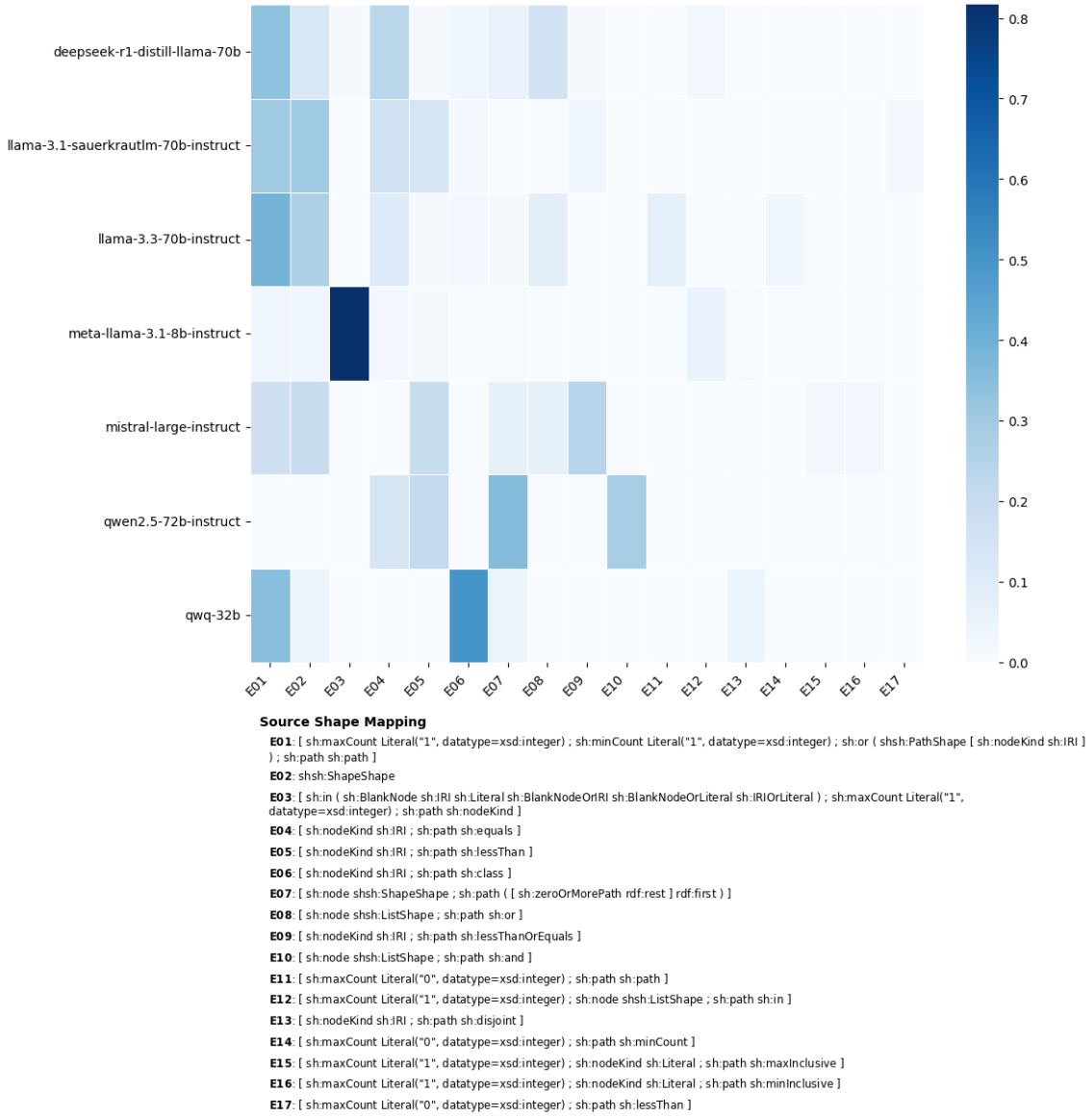


Figure 15: Heatmap of relative source shape frequency in the base experiment. *Y-axis*: Model. *X-axis*: Top-level source shapes. Values indicate the proportion of validation reports generated by a given source shape from the meta shapes graph, averaged over all base runs for each model.

values for `sh:path` (lines 12-15). Using object properties `ff:deceasedSpouse` and `ff:deceasedCivilPartner` to represent *Statutory accident insurance pension compensation* in this example appears plausible, suggesting that the LLM successfully picks up semantic cues from the text and ontology, but struggles with the SHACL format.

```
1 @prefix ff: <https://foerderfunke.org/default#> .
2 @prefix sh: <http://www.w3.org/ns/shacl#> .
3
4 ff:User
5   a sh:Shape ;
6   sh:property [
7       sh:predicate ff:receivesBenefit ;
8       sh:hasValue ff:B100019_LB_102716305 ;
9     ] ;
10  sh:property [
11      sh:predicate ff:needsCare ;
12      sh:class ff:CareNeed ;
13    ] .
```

Listing 3: Generated shapes graph with SHACL syntax error E01.

```
1 @prefix ff: <https://foerderfunke.org/default#> .
2 @prefix sh: <http://www.w3.org/ns/shacl#> .
3
4 ff:UserShape a sh:NodeShape ;
5     sh:targetClass ff:User ;
6     sh:property [
7         sh:path ff:isEligibleFor ;
8         sh:hasValue ff:B100019_LB_102799525 ;
9     ] , [
10        sh:path ff:hasMaritalStatus ;
11        sh:hasValue ff:Married , ff:CivilUnion ;
12    ] , [
13        sh:path ff:deceasedSpouse , ff:deceasedCivilPartner ;
14        sh:hasValue [] ;
15    ] .
```

Listing 4: Generated shapes graph with SHACL syntax error E02.

Notably, Llama 8B exhibits a distinct error profile, with an overwhelming focus on **E03**, which is rarely to never produced by other models. It characterizes a viola-

tion of the rule that `sh:nodeKind` may have at most one value, which must be one of the listed SHACL node kinds. The error messages reveal that E03 is predominantly caused by the use of non-existing SHACL terms, such as `sh:NodeTest` or `sh:Resource`, supporting our earlier conclusion that for this specific model, hallucination of SHACL terms may be a relevant factor in syntax failures. While E03 is a peculiarity of Llama 8B, almost half of all identified error types involve node kind restrictions to `sh:IRI`, indicating a tendency to incorrectly assign blank nodes or literals across models.

Semantic Consistency. Table 4 summarizes semantic consistency results for all models in the base experiment, juxtaposing an assessment over all iterations (Table 4a) with an assessment over well-formed SHACL output only (Table 4b), thereby isolating semantic consistency from syntactic failures. Sauerkraut and Mistral, which excel in syntax conformance, predictably maintain leading positions across most metrics under the full evaluation. However, when focusing on Table 4b, Qwen and QwQ demonstrate superior performance over the other models on most metrics, indicating that despite frequent syntactic failures, their valid outputs more closely match the ground truth. Overall, performance differences narrow in the syntax-filtered evaluation - except for validation performance, discussed below - and all models except Llama 70B place second at least once, resulting in a less definitive performance hierarchy than in the previous evaluations.

Examining Table 4a in greater detail, the consistently high GED scores - never below 0.57 - and low Triple Match scores - never above 0.30 - demonstrate that the generated shapes graphs differ considerably from the topology and composition of the gold graphs across models. Nonetheless, the comparatively high G-BS scores suggest partial alignment in contextual embeddings, particularly for Mistral and Sauerkraut. This may be attributable to the use of contextually relevant terminology from the provided ontology, consistent with our earlier observation that even ill-formed generated SHACL shapes employ vocabulary appropriate to the given benefit (Listing 4).

Validation performance scores in both tables indicate that the generated shapes graphs not only diverge substantially from the human-generated gold standards, but also perform poorly in functional terms, with V-F1 consistently remaining below 0.1. This result is primarily due to low V-P scores, whereas V-R is higher across nearly all models, reaching as much as 1.00 in the case of Qwen (Table 4b).

The sole exception to this trend is QwQ, which exhibits a more balanced relationship

between V-P and V-R. Moreover, it outperforms all other models on V-F1, V-P, and V-Acc, which is remarkable for two reasons. First, QwQ's superior performance even in the full evaluation setting (Table 4a) means that its validation abilities are sufficiently strong to compensate for its poor syntax performance. Without the effect of syntax failures (Table 4b), QwQ further extends its lead, especially on V-Acc where it surpasses its nearest competitor, Sauerkraut, by nearly 0.80. Second, QwQ's high GED score indicates that it generates the shapes graphs with the greatest structural divergence from the ground truth. Its elevated validation performance despite this divergence demonstrates SHACL's tolerance for syntactic variation and reinforces the necessity of employing functional metrics alongside morphological metrics when evaluating Text2SHACL systems.

The predominant pattern - high V-R, and low V-P - indicates that the SHACL validation produces few FN, leading to high recall, and many FP, resulting in low precision. Based on a manual inspection of a random subset of outputs, we hypothesize that imprecise target declarations may cause those results. Specifically, we observe that the generated shapes graphs repeatedly fail to declare the main target node `ff:User`, instead including no target definition (e.g., Listing 3) or declaring a target class (e.g., Listing 4). Consequently, these shapes may generate no focus nodes within the data graph, resulting in trivial conformance and explaining the low number of FN. Additionally, we anticipate a substantial bias towards non-conformance in the ground truth dataset, as validation metrics are computed across all data graphs, thereby diminishing the proportion of profiles specifically designed to conform to a given shapes graph. This negative bias in the data would explain the high number of FP for a system that predicts mostly positive instances.

To investigate this hypothesis, Figure 16 presents the proportion of generated shapes graphs that correctly declare the target node `ff:User`, alongside V-R and V-P scores. For clarity, we focus on a subset of three models, including the outlier model (QwQ) as well as the model exhibiting the clearest instance of the general pattern (Mistral). Two clear patterns emerge from the base experiment: On the one hand, QwQ, which achieves the most balanced V-P and V-R, produces the `ff:User` target node most reliably, with a significant lead of more than 0.75 over the other models. On the other hand, the models with high V-R and low V-P rarely (Sauerkraut) or never (Mistral) declare the correct target node. Appendix D confirms that the remaining models follow the same trend.

These observations support the hypothesis that imprecise target declarations contribute to the significant discrepancy between V-R and V-P, as well as the gener-

Table 4: Baseline performance by model. Metrics include Graph Edit Distance (GED), G-BERTScore (G-BS), Triple Match Precision, Recall, and F1 (T-P, T-R, T-F1), and Validation Precision, Recall, F1, and Accuracy (V-P, V-R, V-F1, V-Acc). **Bold** indicates the best score per model, underlined second best.

(a) Evaluation on all outputs, invalid syntax is penalized with worst scores.

Model	GED ↓	G-BS ↑	T-F1 ↑	T-P ↑	T-R ↑	V-F1 ↑	V-P ↑	V-R ↑	V-Acc ↑
Llama 8b	0.874 ±0.029	0.215 ±0.053	0.035 ±0.008	0.045 ±0.016	0.032 ±0.009	0.006 ±0.003	0.003 ±0.002	0.122 ±0.060	0.010 ±0.013
Sauerkraut	0.674 ±0.015	0.565 ±0.051	0.164 ±0.022	0.193 ±0.014	0.150 ±0.027	0.014 ±0.004	0.007 ±0.002	0.413 ±0.099	0.071 ±0.024
Llama 70b	0.761 ±0.045	0.372 ±0.052	0.115 ±0.004	0.165 ±0.006	0.095 ±0.004	0.010 ±0.006	0.005 ±0.003	0.302 ±0.192	0.005 ±0.003
Mistral	**0.577** ±0.025	**0.649** ±0.024	**0.243** ±0.003	**0.298** ±0.006	**0.215** ±0.008	0.020 ±0.002	0.010 ±0.001	**0.603** ±0.028	0.011 ±0.002
Qwen*	0.794 ±0.162	0.348 ±0.269	0.121 ±0.086	0.140 ±0.098	0.112 ±0.081	0.012 ±0.010	0.006 ±0.005	0.357 ±0.303	0.009 ±0.009
QwQ	0.798 ±0.055	0.290 ±0.082	0.120 ±0.050	0.151 ±0.055	0.106 ±0.048	**0.024** ±0.021	**0.020** ±0.018	0.044 ±0.048	**0.220** ±0.054
Deepseek	0.796 ±0.075	0.308 ±0.084	0.094 ±0.023	0.136 ±0.034	0.073 ±0.018	0.004 ±0.003	0.002 ±0.002	0.095 ±0.082	0.002 ±0.002

(b) Evaluation on outputs with well-formed SHACL only.

Model	GED	G-BS	T-F1	T-P	T-R	V-F1	V-P	V-R	V-Acc
Llama 8b	0.577 ±0.163	0.718 ±0.165	0.099 ±0.041	0.147 ±0.074	0.078 ±0.028	0.039 ±0.014	0.020 ±0.007	0.852 ±0.257	0.068 ±0.089
Sauerkraut	0.555 ±0.021	0.745 ±0.037	0.220 ±0.016	0.258 ±0.015	0.203 ±0.021	0.027 ±0.004	0.014 ±0.002	0.807 ±0.098	0.147 ±0.074
Llama 70b	0.604 ±0.021	0.611 ±0.029	0.232 ±0.054	0.342 ±0.078	0.190 ±0.042	0.030 ±0.002	0.016 ±0.001	0.847 ±0.168	0.016 ±0.001
Mistral	0.525 ±0.024	0.703 ±0.016	0.273 ±0.025	0.337 ±0.040	0.244 ±0.015	0.032 ±0.004	0.016 ±0.002	0.956 ±0.077	0.018 ±0.001
Qwen*	0.520 ±0.013	**0.818** ±0.016	**0.286** ±0.021	0.335 ±0.025	**0.260** ±0.016	0.034 ±0.001	0.018 ±0.001	**1.000** ±0.000	0.022 ±0.006
QwQ	**0.481** ±0.040	0.690 ±0.039	0.281 ±0.073	**0.382** ±0.076	0.237 ±0.077	**0.095** ±0.086	**0.080** ±0.073	0.162 ±0.147	**0.942** ±0.082
Deepseek	0.608 ±0.036	0.699 ±0.210	0.238 ±0.062	0.363 ±0.056	0.180 ±0.057	0.018 ±0.017	0.009 ±0.009	0.500 ±0.433	0.009 ±0.009

*Third run omitted due to frequent request timeouts.

ally poor validation performance observed across most models. Furthermore, they demonstrate that QwQ generates the most active shapes graphs with respect to the dataset, generating conditions that effectively lead to non-conformance. Nonetheless, its results remain sobering in absolute terms - with a low V-F1, suggesting that the generated constraints often still diverge from the ground truth.

Promisingly, Figure 16 suggests that the FS and CoT improve this situation. They exhibit near-perfect to perfect target definitions across models, and narrow the gap between V-P and V-R. The subsequent sections will further explore the performance over different prompts, after deriving our selection of three main models from the base results.

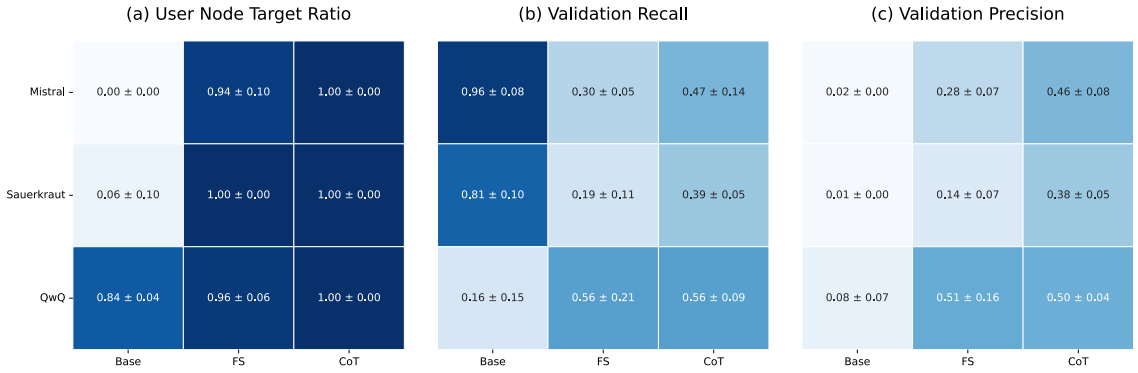


Figure 16: Heatmap of target definitions for different experiment configurations. Juxtaposes average proportion of generated shapes graphs declaring the target node `ff:User` (a), validation recall (b), and validation precision (c). *Y-axis*: Model. *X-axis*: Experiment, including baseline (base), fewshot (FS), and Chain-of-Thought (CoT). Results are averaged over well-formed shapes graphs only.

Main Models. Based on both syntax conformance and semantic consistency results, we retain three LLMs for further analysis. First and second, Mistral and Sauerkraut are selected for consistently outperforming the other models on the syntax metrics as well as most semantic metrics. While their performance advantage on semantic consistency diminishes when considering only syntactically valid outputs, they remain the most promising candidates given the Text2SHACL task definition, which requires the generation of well-formed SHACL (Section 4). Third, we retain QwQ due to its exceptional validation performance. We deem this metric particularly important because it operationalizes the core task requirement that the generated shapes graphs must correctly distinguish between conforming and non-conforming data graphs with respect to the constraints expressed in the input text.

6.2.2 Fewshot and Chain-of-Thought

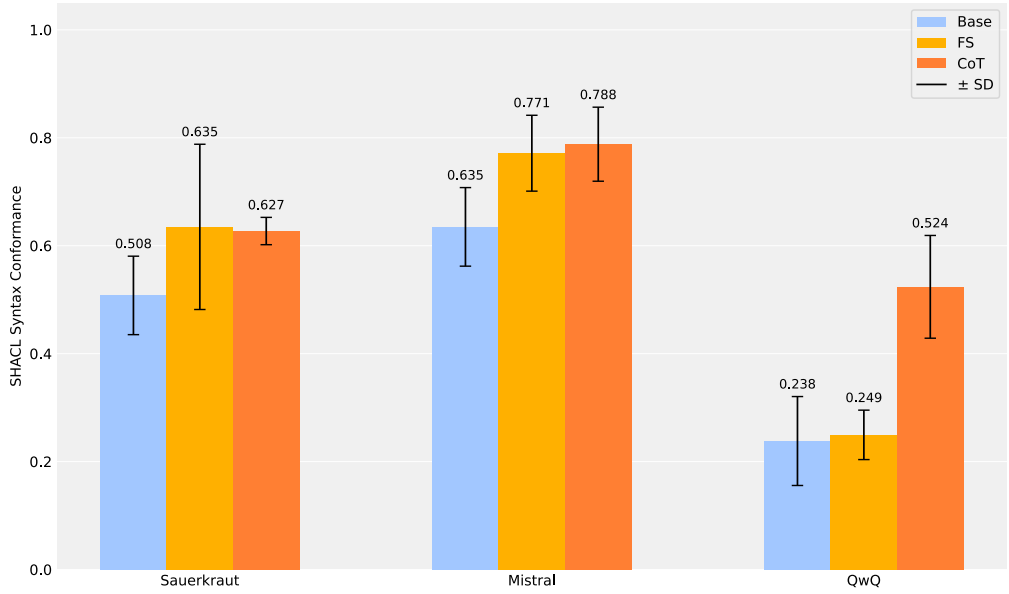


Figure 17: SHACL Syntax conformance in different experiments. *Y-axis*: Proportion of outputs with well-formed SHACL. *X-axis*: Model.

Syntax Conformance. Figure 17 presents SSC scores across experiments for each main model. The performance ranking remains consistent, with Mistral achieving the highest scores, followed by Sauerkraut and QwQ under all prompts. While the base prompt yields the lowest proportion of well-formed SHACL output across models, the impact of FS and CoT prompting is not consistent. Mistral and Sauerkraut show their largest gains with the FS prompt, while CoT has minimal impact. In the case of Sauerkraut, it even induces a slight decline of SSC, though the reduced SD also indicates more stable results. By contrast, QwQ demonstrates a distinct trend: FS has negligible effect, but CoT yields the largest improvement observed across all conditions, with SSC increasing from 0.25 to 0.52. This suggests that QwQ, with its reasoning-oriented training objective, utilizes the reasoning trajectory provided in the CoT prompt most efficiently.

Semantic Consistency. To assess the impact of prompting techniques on semantic consistency, Table 5 reports performance relative to the base prompt. Across nearly all metrics, all models benefit from both FS and CoT prompts. The only exception

is V-R, which declines under both FS and, to a lesser extent, CoT for Sauerkraut and Mistral. Despite this, other validation metrics improve markedly, particularly V-Acc with gains of up to 0.53 for Sauerkraut (FS) and even 0.74 for Mistral (CoT).

These results indicate convergence between V-P and V-R for both models, accompanied by overall increases in V-F1 and V-Acc, reflecting a shift toward the performance pattern exhibited by QwQ in the base experiment. As shown in Figure 15, this development under FS and CoT coincides with a substantial increase in correctly declared target nodes. Given the concurrent drop in V-R, this suggests that while FS and CoT examples guide models toward shape graphs that generate focus nodes more effectively, shapes continue to diverge semantically from the ground truth and produce not only more TN but also FN on the negatively skewed dataset.

Regarding graph matching metrics, the positive trend relative to the base experiment is consistent across models, albeit with model-specific variation that mirrors how FS and CoT impacts SSC. Specifically, Sauerkraut achieves the largest gains with the FS prompt rather than CoT across most metrics, except for GED, where improvements remain modest. In contrast, Mistral benefits most from CoT, though the margin over FS is limited. Finally, in the case of QwQ, CoT consistently induces substantially larger performance gains than FS across all metrics. It is the only model to attain a significant gain on GED and enhance consistently across all metrics, including V-R, highlighting its efficient use of CoT examples to approximate the ground truth graphs more closely, both structurally and semantically.

Nonetheless, QwQ's consistent gains must be interpreted in light of its comparatively low baseline performance. For instance, its base V-R score was below 0.05, putting a gain of 0.25 into perspective. To account for this, we also identify the best-performing prompt configuration in absolute terms for each model, as reported in Table 6.

Before turning to these results, we derive our choice of the "best" prompt per model. For Mistral and QwQ the outcome we select CoT, since both models consistently attain the highest gains relative to base on syntax conformance and nearly all semantic consistency metrics. The sole exception is V-R for Mistral, but this does not offset the other gains in validation performance.For Sauerkraut, the choice between FS and CoT is less clear-cut, as both perform comparably across many metrics. Nevertheless, we consider CoT the more favorable setting due to its stronger improvement in V-F1, which we view as particularly indicative of downstream utility, given the functional demands of the Text2SHACL task.

Table 5: Fewshot (FS) and chain-of-thought (CoT) performance relative to baseline (base). Each sub-table correspond to one model. Metrics include Graph Edit Distance (GED), G-BERTScore (G-BS), Triple Match Precision, Recall, and F1 (T-P, T-R, T-F1), and Validation Precision, Recall, F1, and Accuracy (V-P, V-R, V-F1, V-Acc). **Bold** indicates the best score per model.

(a) Sauerkraut

Prompt	GED ↓	G-BS ↑	T-F1 ↑	T-P ↑	T-R ↑	V-F1 ↑	V-P ↑	V-R ↑	V-Acc ↑
Base	0.674	0.565	0.164	0.193	0.150	0.014	0.007	**0.413**	0.071
± 1SD	±0.015	±0.051	±0.022	±0.014	±0.027	±0.004	±0.002	±0.099	±0.024
△ **FS**	–0.062	+**0.105**	+**0.285**	+**0.270**	+**0.307**	+0.067	+0.073	–0.305	+**0.527**
△ SD	+0.032	+0.034	+0.029	+0.039	+0.024	+0.025	+0.026	-0.047	+0.148
△ **CoT**	–**0.076**	+0.084	+0.271	+0.261	+0.281	+**0.214**	+**0.220**	–0.182	+0.516
△ SD	+0.002	-0.031	-0.011	-0.002	-0.016	+0.021	+0.022	-0.072	-0.001

(b) Mistral

Prompt	GED	G-BS	T-F1	T-P	T-R	V-F1	V-P	V-R	V-Acc
Base	0.577	0.649	0.243	0.298	0.215	0.020	0.010	**0.603**	0.011
± 1SD	±0.025	±0.024	±0.003	±0.006	±0.008	±0.002	±0.001	±0.028	±0.002
△ **FS**	–0.024	+0.057	+0.251	+0.236	+0.262	+0.176	+0.197	–0.379	+0.695
△ SD	+0.002	+0.027	+0.046	+0.045	+0.034	+0.044	+0.052	-0.004	+0.091
△ **CoT**	–**0.049**	+**0.072**	+**0.286**	+**0.306**	+**0.272**	+**0.312**	+**0.336**	–0.251	+**0.740**
△ SD	+0.010	+0.022	+0.015	+0.003	+0.012	+0.047	+0.038	+0.055	+0.048

(c) QwQ

Prompt	GED	G-BS	T-F1	T-P	T-R	V-F1	V-P	V-R	V-Acc
Base	0.798	0.290	0.120	0.151	0.106	0.024	0.020	0.044	0.220
± 1SD	±0.055	±0.082	±0.050	±0.055	±0.048	±0.021	±0.018	±0.048	±0.054
△ **FS**	–0.046	+0.133	+0.196	+0.186	+0.202	+0.097	+0.098	+0.083	+0.016
△ SD	-0.022	-0.041	-0.003	-0.010	+0.006	+0.003	+0.006	-0.020	-0.006
△ **CoT**	–**0.131**	+**0.232**	+**0.280**	+**0.287**	+**0.271**	+**0.243**	+**0.238**	+**0.246**	+**0.299**
△ SD	-0.032	-0.056	-0.014	-0.022	-0.012	+0.009	+0.012	-0.008	+0.041

Based on these criteria, all models perform best with the CoT prompt. Table 6 demonstrates that Mistral consistently achieves the strongest results across metrics in this setting, reaffirming its leading position throughout all experiments evident from Table 9 in Appendix E. This indicates that it not only contains particularly rich knowledge about the SHACL standard based on its training, but also effectively leverages demonstrations at inference-time to further approximate the intended output.

Nonetheless, its peak V-F1 of 0.33 still suggests substantial room for improvement in producing effective shapes with LLMs. While Sauerkraut ranks second in graph matching metrics, QwQ surpasses it in terms of V-F1. This divergence underscores that structural alignment with a reference graph is not the only way to succeed at the Text2SHACL task, because the set of appropriate SHACL shapes graphs may be diverse.

Table 6: Best prompt per model. Metrics include Graph Edit Distance (GED), G-BERTScore (G-BS), Triple Match Precision, Recall, and F1 (T-P, T-R, T-F1), and Validation Precision, Recall, F1, and Accuracy (V-P, V-R, V-F1, V-Acc). **Bold** highlights best model, underlined is second best.

Model	**Prompt**	GED ↓	G-BS ↑	T-F1 ↑	T-P ↑	T-R ↑	V-F1 ↑	V-P ↑	V-R ↑	V-Acc ↑
Sauer-kraut	CoT	0.598 ±0.018	0.649 ±0.020	0.435 ±0.011	0.454 ±0.012	0.431 ±0.011	0.228 ±0.025	0.227 ±0.024	0.230 ±0.027	0.588 ±0.023
Mistral	CoT	**0.528** ±0.035	**0.721** ±0.046	**0.529** ±0.018	**0.605** ±0.010	**0.487** ±0.020	**0.333** ±0.048	**0.346** ±0.039	**0.352** ±0.083	**0.751** ±0.050
QwQ	CoT	0.668 ±0.023	0.522 ±0.026	0.400 ±0.036	0.438 ±0.034	0.377 ±0.036	0.267 ±0.030	0.258 ±0.029	0.290 ±0.040	0.519 ±0.095

6.3 Discussion

We structure the discussion of our third research question - *how do different prompting techniques affect the syntactic and semantic quality of SHACL shapes graphs automatically generated by LLMs from eligibility requirement texts?* - in two parts: first syntax, then semantics. For each, we begin by establishing the base performance with minimal guidance and then discuss the main observed effects of FS and CoT prompting.

Syntactic Conformance. In the base experiment, we observe substantial performance differences between individual models on both Turtle and SHACL syntax, with Mistral emerging as the most effective across the board. As the largest model under evaluation, its parameters appear to capture more extensive knowledge of RDF and SHACL and be better equipped to follow formal output constraints even in the absence of guiding examples. Nonetheless, a notable discrepancy among models sharing the Llama 70B backbone, including Sauerkraut, DeepSeek, and Llama 70B, highlights the influence of different fine-tuning strategies in addition to size. In particular, the reasoning-focused objective used in DeepSeek appears to come at the cost of syntactic reliability.

Overall, SHACL proves substantially more challenging than plain RDF. This may be due to its inherently more restrictive and complex structure, as well as the greater prevalence of RDF in pre-training data, given its earlier standardization in 1999 [119] and its foundational role in various Semantic Web standards (Section 2.1).

The results further highlight that the most frequent reasons for syntactic failures relate to assigning invalid property values - particularly when `sh:IRI` is required - and constructing ill-formed property shapes. For example, we observe the confusion of `sh:path` with `sh:predicate`. On the one hand, this demonstrates familiarity with SHACL vocabulary, which includes `sh:predicate` as well as RDF graphs, where "path" and "predicate" are closely related concepts. On the other hand, these terms are not used as defined by SHACL, highlighting the weakness of systems like LLMs that lack explicit ontological grounding and cannot reliably interpret terms according to their formal semantics.

The effect of FS and CoT on syntactic quality is overall positive for all main models and preserves the ranking from Mistral to Sauerkraut to QwQ. However, effective example design - balancing prompt complexity with performance gains - must consider model-specific predispositions: CoT yields substantial benefits only for the reasoning model QwQ, offering no significant improvement over FS for the others. Furthermore, these prompting techniques are insufficient to bridge the performance gap between TSC and SSC, indicating that reliable automatic SHACL generation from requirements texts with LLMs necessitates either further prompt engineering or employing models with greater SHACL-specific knowledge embedded in their pre-trained and fine-tuned parameters.

Semantic Consistency. The observed patterns in the base experiment suggest that Mistral and Sauerkraut are best equipped to generate shapes graphs that align semantically with the reference graphs, a benefit partially attributable to their syntactic strengths. Despite these relative strengths, the generated shapes graphs generally diverge substantially from the ground truth in both structure and function, evidenced by consistently weak graph matching and validation scores.

Moreover, the elevated G-BS scores relative to other metrics suggest that LLMs are more effective at extracting relevant cues from the input text and ontology to generate semantically related terminology, than at reproducing the structural form of the ground truth. While this may in part reflect the metric's gradual, similarity-based design, the previous interpretation also aligns with prior research indicating that LLMs, primarily trained on NL, often have difficulty with generating structured

output [39, 59].

The results reveal a distinctive pattern in validation performance with QwQ emerging as the top performer. This appears to result from it being the only model to consistently declare the correct primary target node `ff:User` thereby enabling meaningful constraint validation. In contrast, other models frequently omit the required target declaration, leading to trivially conforming shapes. This discrepancy is exacerbated by the negatively biased dataset, which accentuates the distinction between models that generate focus nodes - a precondition of detecting negative instances - and those that do not.

With FS and CoT prompting, the discrepancy in validation performance between models narrows, converging toward the pattern initially observed in QwQ. Both Mistral and Sauerkraut begin to consistently include the correct primary target node declaration, suggesting that demonstrating this requirement through examples guides models to process this output constraint more effectively than merely describing it in the directive, as done in the ZS prompt. As a result, FS and CoT notably enhance validation performance across models, including further improvements for QwQ, accompanied by corresponding gains in graph matching scores.

However, these two sets of metrics reflect different objectives. Graph matching measures structural alignment with the reference graph, whereas validation performance captures whether the generated shapes function as intended. While both tend to improve in parallel under FS and CoT prompting, the base results illustrate that there is no necessary connection between generating the shapes graphs with the closest structural alignment with the ground truth (Mistral) and those that perform best on the validation task (QwQ).

This leads to two important insights for future evaluation. First, graph matching alone is not a reliable proxy for success on the Text2SHACL task; with its functional lens, validation performance captures a distinct and important aspect of semantic quality. Second, it draws attention to alternative optimization strategies that target functional correctness more directly. Specifically, we note that the inclusion of ground-truth shapes graphs in FS and CoT prompts encourages models to emulate human-authored examples, thereby prioritizing structural similarity. However, a more outcome-oriented alternative may involve conditioning the model on user profiles along with their expected validation outcomes. While this approach targets validation performance more directly, future work investigating its effects will need to account for adequate evaluation through realistic and representative user data.

7 Conclusion

This thesis investigated the Text2SHACL task for social benefit eligibility assessment through human annotation and prompt engineering.

The human annotation yielded a procedural schema outlining the main steps of Text2SHACL as well as a novel dataset of SHACL-annotated social benefit requirements. These contributions provide concrete resources for practitioners and researchers and offer a more nuanced understanding of the task. Specifically, it highlighted the versatile role of the underlying - fixed or evolving - ontology. Moreover, we conclude that complete representation of eligibility requirements with SHACL Core is neither feasible, due to the intricacy of arithmetic, temporal, and defeasible requirements, nor necessarily desirable, as demonstrated by the pragmatic and ethical concerns regarding self-referentiality and discretion. These findings imply that effective eligibility assessment requires integrating more expressive formalization tools and challenge the prospect of fully automated, legally binding eligibility assessments.

The interoperability and generalizability of the proposed ontology, though effective within this study's scope, is limited due to its reliance on a custom namespace and focus on a narrow subset of social benefits. For a more efficient representation of the domain, future research should pursue a systematic ontology development process that incorporates established domain vocabularies and can accommodate a broader range of benefits. We recommend the well-established, tool-supported methodology outlined by Noy et al. [79] as a foundation.

Additionally, future work should explore integrating SHACL-SPARQL constraints into the ontology, as exemplified by Anim et al. [2]. This approach enables inference of complex legal facts from simple user inputs, addressing SHACL Core's limited expressivity - not by increasing the complexity of shapes graphs at the expense of automation, but by validating semantically enriched data.

The prompt engineering experiments leveraged the developed resources to investigate the effect of prompting techniques on the quality of LLM-generated shapes graphs. For this, we first established a baseline using ZS prompting. We found that the best-performing models demonstrate a general capacity to generate functional shapes graphs and employ the prescribed vocabulary. However, performance varies substantially across model size and training paradigms, and even the top models exhibit frequent syntactic errors and limited semantic alignment with the ground

truth.

The FS and CoT experiments boil down to four key findings. First, providing demonstrative examples generally enhances syntactic and semantic quality. Second, while CoT emerges as the most effective strategy overall, its impact differs by model type, with the reasoning model QwQ benefiting the most. Third, Mistral consistently outperforms the others across nearly all metrics, indicating that prompting alone cannot fully compensate for lack of SHACL-knowledge contained in the model parameters. Fourth, strong validation scores do not presuppose high structural similarity with the ground truth.

Despite improvements, given low overall validation performance, the practical utility of generated shapes graphs remains limited. A promising direction to further enhance performance in future research is incorporating exemplary data graphs instead of reference shapes graphs, along with their expected validation results into prompts. This approach targets validation performance more directly than through structural similarity with ground truth. We recommend constructing a more balanced and comprehensive set of data graphs to strengthen the robustness of validation metrics for this optimization objective. For example, future work could examine the viability of the open-source tool proposed by Vecovska and Jovanovik [112] for automatically generating RDF data conforming to SHACL shapes to counterweigh the negative bias in the dataset.

References

[1] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An exact graph edit distance algorithm for solving pattern recognition problems. In *ICPRAM 2015 - Proceedings of the International Conference on Pattern Recognition Applications and Methods, Volume 1, Lisbon, Portugal, 10-12 January, 2015*, pages 271–278. SciTePress.

[2] Joseph Anim, Livio Robaldo, and Adam Z. Wyner. 2024. A shacl-based approach for enhancing automated compliance checking with rdf data. *Information*, 15(12):759. 10.3390/info15120759.

[3] Sylvain Arlot and Alain Celisse. 2010. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79. 10.1214/09-SS054.

[4] Vahan Arsenyan, Spartak Bughdaryan, Fadi Shaya, Kent Wilson Small, and Davit Shahnazaryan. 2024. Large language models for biomedical knowledge graph construction: Information extraction from EMR notes. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing, BioNLP@ACL 2024, Bangkok, Thailand, August 16, 2024*, pages 295–317. Association for Computational Linguistics.

[5] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2nd edition. Cambridge University Press, Cambridge, UK. Online publication July 2010.

[6] Chris Backes and Mariolina Eliantonio. 2014. Administrative law. In Jaap Hage and Bram Akkermans, editors, *Introduction to Law*, pages 189–210. Springer, Cham.

[7] Benjamin Baisch, Dagmar Müller, Corinna Zollner, Laura Castiglioni, and Christina Boll. 2023. Barrieren der Inanspruchnahme monetärer Leistungen für Familien.

[8] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. 2014. RDF 1.1 Turtle: Terse RDF Triple Language. The latest edition is available at `http://www.w3.org/TR/turtle/`.

[9] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web. *Scientific American*, 284(5):34–43.

[10] Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, Tomasz Sternal, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Łukasz Flis, Hannes Eberhard, Hubert Niewiadomski, and Torsten Hoefler. 2025. Reasoning models: A blueprint. *CoRR*.

[11] Dan Brickley and R.V. Guha. 2014. RDF Schema 1.1. The latest edition is available at `http://www.w3.org/TR/rdf11-schema/`.

[12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[13] Bundesrepublik Deutschland. 1975. Sozialgesetzbuch Erstes Buch – Allgemeiner Teil: SGB I. Promulgated on 11 December 1975 (BGBl. I S. 3015); latest revision available at `https://www.gesetze-im-internet.de/sgb_1/`.

[14] Gavin Carothers and Andy Seabourne. 2014. RDF 1.1 N-Triples. The latest edition is available at `http://www.w3.org/TR/n-triples/`.

[15] Lei Chen, Haifei Zhang, Ying Chen, and Wenping Guo. 2012. Blank nodes in RDF. *J. Softw.*, 7(9):1993–1999. 10.4304/JSW.7.9.1993-1999.

[16] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Er-

ica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

[17] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*.

[18] Andrea Cimmino, Alba Fernández-Izquierdo, and Raúl García-Castro. 2020. Astrea: Automatic generation of SHACL shapes from ontologies. In *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 497–513. Springer.

[19] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46. 10.1177/001316446002000104.

[20] Richard Cyganiak, David Wood, and Markus Lanthaler. 2014. RDF 1.1 Concepts and abstract syntax. The latest edition is available at `http://www.w3.org/TR/rdf11-concepts/`.

[21] Dataport AöR. 2025. PVOG und Landesredaktionssystem. Accessed: 2025-05-31.

[22] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J.

Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. 2025. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948. 10.48550/ARXIV.2501.12948.

[23] Thomas Delva, Birte De Smedt, Sitt Min Oo, Dylan Van Assche, Sven Lieber, and Anastasia Dimou. 2021. RML2SHACL: RDF generation taking shape. In *K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021*, pages 153–160. ACM.

[24] Deutscher Bundestag. 2013. Gesetz zur förderung der elektronischen verwaltung: Egovg. Promulgated on 25 July 2013 (BGBl. I S. 2749); latest revision available at `https://www.gesetze-im-internet.de/egovg/index.html#BJNR274910013BJNE002900130`.

[25] Deutscher Ethikrat. 2023. *Mensch und Maschine - Herausforderungen durch Künstliche Intelligenz*. Deutscher Ethikrat, Berlin.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

[28] Alex Donkers and Ekaterina Petrova. 2024. Converting fire safety regulations to SHACL shapes using natural language processing. In *Proceedings of the 3rd International Workshop on Natural Language Processing for Knowledge Graph Creation co-located with 20th International Conference on Semantic Systems (SEMANTiCS 2024), Amsterdam, The Netherlands, September 17, 2024*, volume 3874 of *CEUR Workshop Proceedings*, pages 97–111. CEUR-WS.org.

[29] Ali Doosthosseini, Jonathan Decker, Hendrik Nolte, and Julian M. Kunkel. 2024. Chat AI: A seamless slurm-native solution for hpc-based services. *CoRR*, abs/2407.00110. 10.48550/ARXIV.2407.00110.

[30] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 herd of models. *CoRR*, abs/2407.21783. 10.48550/ARXIV.2407.21783.

[31] Hans Dubois and Anna Ludwinek. 2015. Access to social benefits: Reducing non-take-up.

[32] Martin J. Dürst and Michel Suignard. 2005. Internationalized Resource Identifiers (IRIs). *RFC*, 3987:1–46.

[33] European Parliament and Council of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the Eu-

ropean Union, L 119, 4 May 2016, pp. 1–88. Consolidated version available at https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[34] Code for Müenster. 2022. Müensterhack 2022. https://github.com/codeformuenster/muensterhack/blob/master/2022.md. Accessed: 2025-05-24.

[35] Fabien Gandon and Guus Schreiber. 2014. RDF 1.1 XML Syntax. The latest edition is available at http://www.w3.org/TR/rdf-syntax-grammar/.

[36] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3816–3830. Association for Computational Linguistics.

[37] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

[38] José Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. 2017. *Validating RDF Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

[39] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10932–10952. Association for Computational Linguistics.

[40] Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen Datenschutzbeauftragter. 2025. Available models. Accessed: 2025-04-10.

[41] Hussam Ghanem and Christophe Cruz. 2024. Fine-tuning vs. prompting: Evaluating the knowledge graph construction with llms. In *Proceedings of the 3rd International Workshop on Knowledge Graph Generation from Text (Text2KG), co-located with the Extended Semantic Web Conference (ESWC 2024)*, volume 3747, page 18, Hersonissos, Greece. CEUR-WS.org.

[42] Danielle G. E. Gomes, Pierre Pottier, Robert Crystal-Ornelas, Emma J. Hudgins, Vivian Foroughirad, Lía L. Sánchez-Reyes, and Kaitlyn M. Gaynor. 2022. Why don't we share data and code? Perceived barriers and benefits to public archiving practices. *Proceedings of the Royal Society B: Biological Sciences*, 289(1987). Original work published November 30, 2022, 10.1098/rspb.2022.1113.

[43] Thomas R. Gruber. 1993. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220. 10.1006/KNAC.1993.1008.

[44] Philipp Hagedorn and Markus König. 2021. Rule-based semantic validation for standardized linked building models. In *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering*, volume 98, pages 772–787. Springer, Cham.

[45] Philipp Hagedorn, Pieter Pauwels, and Markus König. 2023. Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language. *Automation in Construction*, 156. 10.1016/j.autcon.2023.105106.

[46] Jiuzhou Han, Nigel Collier, Wray L. Buntine, and Ehsan Shareghi. 2024. Pive: Prompting with iterative verification improving graph-based generative capability of llms. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6702–6718. Association for Computational Linguistics.

[47] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780. 10.1162/NECO.1997.9.8.1735.

[48] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[49] Thomas Hofweber. 2023. Logic and ontology. *The Stanford Encyclopedia of Philosophy*.

[50] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers.

[51] Matthew Horridge and Peter F. Patel-Schneider. 2012. OWL 2 web ontology language: Manchester syntax (second edition). The latest edition is available at `http://www.w3.org/TR/owl2-manchester-syntax/`.

[52] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2):42:1–42:55. 10.1145/3703155.

[53] Tobias Hummel, Leon Martin, and Andreas Henrich. 2024. Assessing the fairness of software repositories using RDF and SHACL. In *Knowledge Graphs in the Age of Language Models and Neuro-Symbolic AI - Proceedings of the 20th International Conference on Semantic Systems, 17-19 September 2024, Amsterdam, The Netherlands*, volume 60 of *Studies on the Semantic Web*, pages 160–175. IOS Press.

[54] Ana Iglesias-Molina and Daniel Garijo. 2023. Towards assessing FAIR research software best practices in an organization using rdf-star. In *Proceedings of the Posters and Demo Track of the 19th International Conference on Semantic Systems co-located with 19th International Conference on Semantic Systems (SEMANTiCS 2023), Leipzing, Germany, September 20 to 22, 2023*, volume 3526 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[55] IT-Planungsrat. 2025. Produkte des IT-Planungsrats. Accessed: 2025-06-01.

[56] Garud Iyengar, Henry Lam, and Tianyu Wang. 2024. Is cross-validation the gold standard to estimate out-of-sample model performance? In *Advances in Neural Information Processing Systems*, volume 37, pages 94736–94775. Curran Associates, Inc.

[57] Choi Ji-Woong. 2020. Automatic construction of SHACL schemas for RDF knowledge graphs generated by R2RML mappings. *Journal of the Korea Society of Computer and Information*, 25(8):9–21. 10.9708/jksci.2020.25.08.009.

[58] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438. 10.1162/TACL_A_00324.

[59] Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting asymmetry for synthetic training data generation: Synthie and the case of information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1555–1574. Association for Computational Linguistics.

[60] Dan Jurafsky and James H. Martin. 2009. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International.

[61] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

[62] Gregg Kellogg, Dave Longley, and Pierre-Antoine Champin. 2020. JSON-LD 1.0. The latest edition is available at `https://www.w3.org/TR/json-ld11/`.

[63] Holger Knublauch. 2017. SHACL and OWL compared. Accessed: 2025-05-30.

[64] Holger Knublauch and Dimitris Kontokostas. 2017. Shapes Constraint Language (SHACL). The latest edition is available at `https://www.w3.org/TR/shacl/`.

[65] Wonsik Ko and Robert A. Moffitt. 2024. Take-up of social benefits. In Klaus F. Zimmermann, editor, *Handbook of Labor, Human Resources and Population Economics*. Springer, Cham.

[66] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

[67] Ora Lassila and Deborah L. McGuinness. 2001. The role of frame-based representation on the semantic web. *Linköping Electronic Articles in Computer and Information Science*, 6(5).

[68] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998.

Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324. 10.1109/5.726791.

[69] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, pages 100–114. Association for Computational Linguistics.

[70] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35. 10.1145/3560815.

[71] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

[72] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8086–8098. Association for Computational Linguistics.

[73] Nandana Mihindukulasooriya, Mohammad Rifat Ahmmad Rashid, Giuseppe Rizzo, Raúl García-Castro, Óscar Corcho, and Marco Torchiano. 2018. RDF shape induction using knowledge base profiling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018*, pages 1952–1959. ACM.

[74] Nandana Mihindukulasooriya, Sanju Tiwari, Carlos F. Enguix, and Kusum Lata. 2023. Text2KGBench: A benchmark for ontology-driven knowledge graph generation from text. In *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part II*, volume 14266 of *Lecture Notes in Computer Science*, pages 247–265. Springer.

[75] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev

Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech 2010*, pages 1045–1048.

[76] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11048–11064. Association for Computational Linguistics.

[77] Mistral AI team. 2024. Large enough. Accessed: 2025-04-10.

[78] Iftitahu Ni'mah, Meng Fang, Vlado Menkovski, and Mykola Pechenizkiy. 2023. NLG evaluation metrics beyond correlation analysis: An empirical metric preference checklist. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1240–1266. Association for Computational Linguistics.

[79] Natalya F. Noy, Deborah L. McGuinness, et al. 2001. Ontology development 101: A guide to creating your first ontology.

[80] Emma Nuyts, Jeroen Werbrouck, Ruben Verstraeten, and Louise Deprez. 2023. Validation of building models against legislation using SHACL. In *Proceedings of the 11th Linked Data in Architecture and Construction Workshop, Matera, Italy, June 15-16, 2023*, volume 3633 of *CEUR Workshop Proceedings*, pages 164–175. CEUR-WS.org.

[81] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio José Rodríguez Méndez, and Armin Haller. 2023. Learning SHACL shapes from knowledge graphs. *Semantic Web*, 14(1):101–121. 10.3233/SW-223063.

[82] Yvette Oortwijn, Thijs Ossenkoppele, and Arianna Betti. 2021. Interrater disagreement resolution: A systematic procedure to reach consensus in annotation tasks. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 131–141.

[83] Monica Palmirani, Michele Martoni, Arianna Rossi, Cesare Bartolini, and Livio Robaldo. 2018. Pronto: Privacy ontology for legal reasoning. In *Electronic Government and the Information Systems Perspective - 7th International Conference, EGOVIS 2018, Regensburg, Germany, September 3-5, 2018, Proceedings*, volume 11032 of *Lecture Notes in Computer Science*, pages 139–152. Springer.

[84] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Trans. Knowl. Data Eng.*, 36(7):3580–3599. 10.1109/TKDE.2024.3352100.

[85] Harshvardhan J. Pandit, Declan O'Sullivan, and Dave Lewis. 2018. Using ontology design patterns to define SHACL shapes. In *Proceedings of the 9th Workshop on Ontology Design and Patterns (WOP 2018) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 9th, 2018*, volume 2195 of *CEUR Workshop Proceedings*, pages 67–71. CEUR-WS.org.

[86] Paolo Pareti and George Konstantinidis. 2021. A review of SHACL: From data validation to schema reasoning for RDF graphs. In *Reasoning Web. Declarative Artificial Intelligence - 17th International Summer School 2021, Leuven, Belgium, September 8-15, 2021, Tutorial Lectures*, volume 13100 of *Lecture Notes in Computer Science*, pages 115–144. Springer.

[87] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

[88] Eric Prud'hommeaux, José Emilio Labra Gayo, and Harold R. Solbrig. 2014. Shape expressions: An RDF validation and transformation language. In *Proceedings of the 10th International Conference on Semantic Systems, SEMANTiCS 2014, Leipzig, Germany, September 4-5, 2014*, pages 32–40. ACM.

[89] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

[90] Qwen Team. 2025. QwQ-32B: Embracing the power of reinforcement learning. Accessed: 2025-04-10.

[91] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and

Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical Report.

[92] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

[93] Livio Robaldo, Francesco Pacenza, Jessica Zangari, Roberta Calegari, Francesco Calimeri, and Giovanni Siragusa. 2023. Efficient compliance checking of RDF data. *J. Log. Comput.*, 33(8):1753–1776. 10.1093/LOGCOM/EXAD034.

[94] Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7716–7740. Association for Computational Linguistics.

[95] Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2339–2352. Association for Computational Linguistics.

[96] Guus Schreiber and Yves Raimond. 2014. RDF 1.1 Primer. The latest edition is available at `http://www.w3.org/TR/rdf11-primer/`.

[97] Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Hoyle, and Philip Resnik. 2024. The prompt report: A systematic survey of prompt engineering techniques. *CoRR*.

[98] David Senkýr. 2019. SHACL shapes generation from textual documents. In *Enterprise and Organizational Modeling and Simulation - 15th International Workshop, EOMAS 2019, Held at CAiSE 2019, Rome, Italy, June 3-4, 2019,*

Selected Papers, volume 366 of *Lecture Notes in Business Information Processing*, pages 121–130. Springer.

[99] Joel Shor, Ruyue Agnes Bi, Subhashini Venugopalan, Steven Ibara, Roman Goldenberg, and Ehud Rivlin. 2023. Clinical bertscore: An improved measure of automatic speech recognition performance in clinical settings. In *Proceedings of the 5th Clinical Natural Language Processing Workshop, ClinicalNLP@ACL 2023, Toronto, Canada, July 14, 2023*, pages 1–7. Association for Computational Linguistics.

[100] Mareike Sielaff and Felix Wilke. 2024. Die Nichtinanspruchnahme von Grundsicherung als Bewältigungsstrategie. In Michael Opielka and Felix Wilke, editors, *Der weite Weg zum Bürgergeld, Perspektiven der Sozialpolitik*, pages 107–131. Springer, Wiesbaden.

[101] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33*, volume 33, pages 16857–16867. Curran Associates, Inc.

[102] Sozialplattform. 2025. Schnittstellen und Anbindung. https://sozialplattform.de/schnittstellen-anbindung. Accessed: 2025-05-24.

[103] Statistisches Bundesamt. 2025. Regionalschlüssel (RS). Accessed: 2025-05-31.

[104] Simon Steyskal and Karen Coyle. 2017. SHACL Use Cases and Requirements. The latest edition is available at https://www.w3.org/TR/shacl-ucr/.

[105] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

[106] Mohammad Mustafa Taye. 2010. Understanding semantic web and ontologies: Theory and applications. *Journal of Computing*, 2(6):182–192.

[107] David Thaler, Tony Hansen, and Ted Hardie. 2015. Guidelines and registration procedures for URI schemes. *RFC*, 7595:1–19. 10.17487/RFC7595.

[108] Inigo Jauregi Unanue, Jacob Parnell, and Massimo Piccardi. 2021. Berttune: Fine-tuning neural machine translation with bertscore. In *Proceedings of the*

59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021, pages 915–924. Association for Computational Linguistics.

[109] United Nations Human Rights Council (UNHRC). 2022. Report of the special rapporteur on extreme poverty and human rights Olivier de Schutter. UN Doc A/HRC/50/38.

[110] VAGO Solutions. 2024. SauerkrautLM bekommt Zuwachs – Das neue Llama-3.1-70b-Instruct. Accessed: 2025-04-10.

[111] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

[112] Marija Vecovska and Milos Jovanovik. 2024. RDFGraphGen: A synthetic RDF graph generator based on SHACL constraints. *CoRR*, abs/2407.17941. 10.48550/ARXIV.2407.17941.

[113] Julie Vinck, Jo Lebeer, and Wim Van Lancker. 2018. Non-take-up of the supplemental child benefit for children with a disability in belgium: A mixed-method approach. *Social Policy & Administration*, 53:357–384. 10.1111/spol.12457.

[114] W3C OWL Working Group. 2012. OWL 2 Web Ontology Language. The latest edition is available at `http://www.w3.org/TR/owl-overview`.

[115] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[116] Zhi Wang, Zixu Chu, Tuan V. Doan, et al. 2024. History, development, and principles of large language models: An introductory survey. *AI Ethics*. 10.1007/s43681-024-00583-7.

[117] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and

William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

[118] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[119] World Wide Web Consortium. 1999. W3c issues recommendation for resource description framework (rdf). W3C Press Release.

[120] World Wide Web Consortium (W3C). 2025. About us. `https://www.w3.org/about/`. Accessed: 2025-04-29.

[121] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115. 10.48550/ARXIV.2412.15115.

[122] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[123] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

[124] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[125] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav

Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[126] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5017–5033. Association for Computational Linguistics.

[127] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Graph Edit Distance Timeout

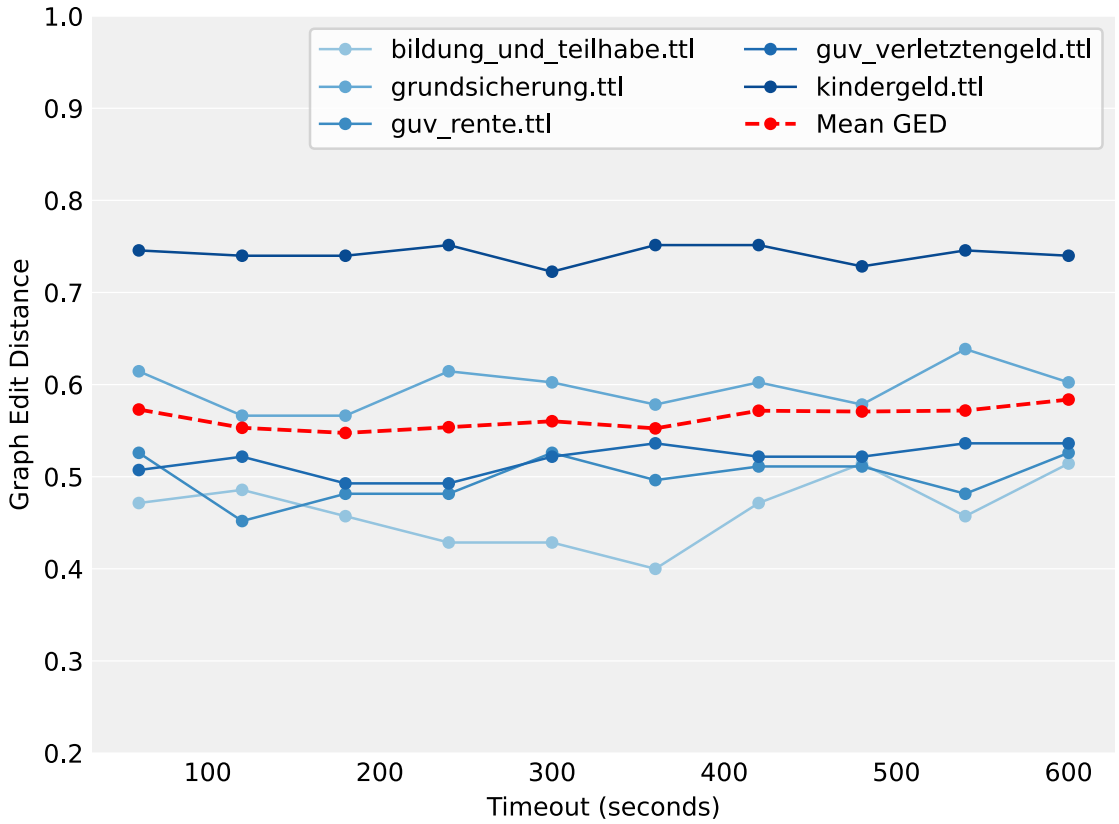


Figure 18: Graph Edit Distance vs. Timeout. GED was computed using NetworkX's exact algorithm with timeouts ranging from 60 to 600 seconds in 1-minute steps, across six machine-generated SHACL shape graphs randomly selected from run baseline_0ex0fcv_1745697193 with model Mistral Large Instruct. The red line shows the mean GED across all benefits at each timeout.

B Definition of Source Shapes

Table 7: Definition of source shapes. Presents the 10 most frequently triggered meta shapes graph source shapes from the base experiment, illustrated with simplified examples drawn from actual generated data. IDs (E01,...,E10) match the legend in 15.

Error Type		**Example**	
Description	**Source Shape**	**Description**	**Invalid Shape**
E01. A SHACL Property Shape must have the property sh:path exactly once and its value must be a well-formed shsh:pathShape, or an IRI.	`[ sh:maxCount "1" ^^xsd:integer ; sh:minCount "1"^^xsd:integer ; sh:or ( shsh:PathShape [ sh:nodeKind sh:IRI ] ) ; sh:path sh:path ]`	An object of sh:property does not have a property sh:path.	`[ sh:property [ sh:predicate ff:needsCare ; sh:class ff:CareNeed ] ]`
E02. A shape must be either a well-formed node shape or property shape.	`shsh:ShapeShape`	A shape is not a valid node shape because it has the property sh:path and not a valid property shape because it has multiple values for sh:path.	`[ sh:property [ sh:path ff:deceasedSpouse , ff:deceasedCivilPartner ; sh:hasValue [ ] ] ]`
E03. The property sh:nodeKind can have at most one value, which must be one out of the listed types.	`[ sh:in ( sh:BlankNode sh:IRI sh:Literal sh:BlankNodeOrIRI sh:BlankNodeOrLiteral sh:IRIOrLiteral ) ; sh:maxCount "1"^^xsd:integer ; sh:path sh:nodeKind ]`	Using an invalid node kind, like sh:Node.	`[ sh:property [ sh:path ff:hasChild ; sh:nodeKind sh:Node ] ]`
E04. The value of sh:equals must be an IRI.	`[ sh:nodeKind sh:IRI ; sh:path sh:equals ]`	Using a literal like “false” as value for sh:equals.	`[ sh:property [ sh:path ff:canCoverBasicNeeds ; sh:equals "false"^^xsd:boolean ] ]`

E05. The value of sh:lessThan must be an IRI.	`[ sh:nodeKind sh:IRI ; sh:path sh:lessThan ]`	Using a literal like 1500 as value for sh:lessThan.	`[ sh:property [ sh:path ff:personalGrossIncome ; sh:lessThan 1500 ] ]`
E06. The value of sh:class must be an IRI.	`[ sh:nodeKind sh:IRI ; sh:path sh:class ]`	Using a blank node as value for sh:class.	`sh:class [ sh:or ( ff:WorkAccident ff:CommutingAccident ff:OccupationalDisease ) ]`
E07. Each member of a SHACL list must be a well-formed shape.	`[ sh:node shsh:ShapeShape ; sh:path ( [ sh:zeroOrMorePath rdf:rest ] rdf:first ) ]`	An item in sh:or (a SHACL list) is not a well-formed shape because it contains a path sh:equals that points to a Literal instead of an IRI.	`sh:or ( [ sh:path ff:hasDisability ; sh:datatype xsd:boolean ; sh:equals true ] )`
E08. The value of sh:or must be a valid RDF list.	`[ sh:node shsh:ListShape ; sh:path sh:or ]`	Assigning sh:or a blank node instead of an explicit RDF list or Turtle's short-hand parenthesis notation.	`sh:or [ sh:path ff:receivesBenefit ; sh:hasValue ff:L100040_LB_-8665924 ]`
E09. The value for sh:lessThanOrEquals must be an IRI.	`[ sh:nodeKind sh:IRI ; sh:path sh:lessThanOrEquals ]`	Using a literal like "now() - P1Y" as a value for sh:lessThanOrEquals.	`sh:property [ sh:path ff:hasDate ; sh:lessThanOrEquals [ sh:path ff:hasDate ; sh:value "now() - P1Y"^^xsd:dateTime ] ]`
E10. The value of sh:and must be a well-formed RDF list.	`[ sh:node shsh:ListShape ; sh:path sh:and ]`	Assigning sh:and a blank node instead of an explicit RDF list or Turtle's short-hand parenthesis notation.	`sh:and [ sh:path ff:isSingleParent ; sh:hasValue true ]`

C Conformance with SHACL Vocabulary

Table 8: Ratio of unknown SHACL terms by Prompt and Model. Average ratio of SHACL terms that are not part of the full SHACL vocabulary (Unknown SHACL) or its more limited subset SHACL Core (Unknown SHACL Core) per generated shapes graph. Lower values indicate fewer invented terms, i.e. better vocabulary conformance.

Prompt	Model	Unknown SHACL Ratio	Unknown SHACL Core Ratio
Base	DeepSeek-R1-Distill-LLaMA-70B	0.108	0.140
	SauerKrautLM-70B-Instruct	0.022	0.024
	LLaMA-3.3-70B-Instruct	0.008	0.039
	Meta-LLaMA-3.1-8B-Instruct	0.130	0.137
	Mistral-Large-Instruct	0.002	0.002
	Qwen2.5-72B-Instruct	0.000	0.022
	QwQ-32B	0.012	0.012
FS	SauerKrautLM-70B-Instruct	0.004	0.004
	Mistral-Large-Instruct	0.000	0.000
	QwQ-32B	0.015	0.015
CoT	SauerKrautLM-70B-Instruct	0.002	0.002
	Mistral-Large-Instruct	0.003	0.003
	QwQ-32B	0.003	0.003

Lower Unknown SHACL vs. SHACL Core ratios in Base indicate use of advanced SHACL features against the prompt directive - an effect absent in FS and CoT. DeepSeek and LLaMA 80B are most inventive (approximately 14% for SHACL Core), aligning with their poor syntax performance. Besides, invention rates are low and have limited explanatory power for syntactic errors.

D Heatmap of Target Definitions Over All Models

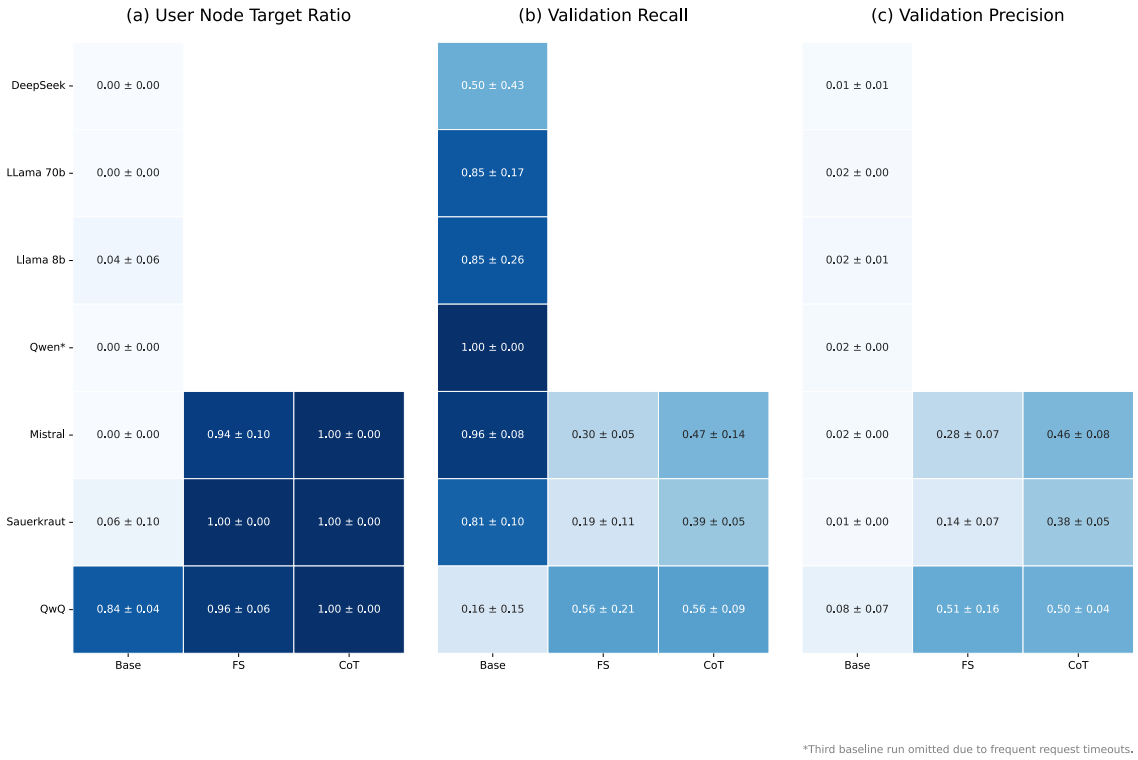


Figure 19: Heatmap of target declarations (all models). Juxtaposes average proportion of generated shapes graphs declaring the target node `ff:User` (a), validation recall (b), and validation precision (c). *Y-axis*: Model. *X-axis*: Experiment, including baseline (base), fewshot (FS), and Chain-of-Thought (CoT). Results are averaged over well-formed shapes graphs only.

E Performance of Main Models Over all Prompts

Table 9: Performance of main models on all prompts. *Rows:* Each row corresponds to a combination of prompt (Base = baseline, FS = fewshot, CoT = chain-of-thought) and model. *Columns:* Each column corresponds to one of the main evaluation metrics, including Graph Edit Distance (GED), G-BERTScore (G-BS), Triple Match Precision, Recall, and F1 (T-P, T-R, T-F1), and Validation Precision, Recall, F1, and Accuracy (V-P, V-R, V-F1, V-Acc). **Bold** indicates the best score per prompt, the overall best results per metric are additionally marked with an asterix.

Prompt + Model		GED ↓	G-BS ↑	T-F1 ↑	T-P ↑	T-R ↑	V-F1 ↑	V-P ↑	V-R ↑	V-Acc ↑
Base	Sauerkraut	0.674 ±0.015	0.565 ±0.051	0.164 ±0.022	0.193 ±0.014	0.150 ±0.027	0.014 ±0.004	0.007 ±0.002	0.413 ±0.099	0.071 ±0.024
	Mistral	**0.577** ±0.025	**0.649** ±0.024	**0.243** ±0.003	**0.298** ±0.006	**0.215** ±0.008	0.020 ±0.002	0.010 ±0.001	**0.603** ±0.028	0.011 ±0.002
	QwQ	0.798 ±0.055	0.290 ±0.082	0.120 ±0.050	0.151 ±0.055	0.106 ±0.048	**0.024** ±0.021	**0.020** ±0.018	0.044 ±0.048	**0.220** ±0.054
FS	Sauerkraut	0.612 ±0.047	0.669 ±0.084	0.449 ±0.051	0.463 ±0.053	0.456 ±0.052	0.080 ±0.028	0.080 ±0.028	0.108 ±0.052	0.598 ±0.172
	Mistral	**0.553** ±0.026	**0.706** ±0.051	**0.494** ±0.048	**0.534** ±0.051	**0.478** ±0.042	**0.196** ±0.045	**0.207** ±0.053	**0.224** ±0.024	**0.707** ±0.091
	QwQ	0.752 ±0.033	0.423 ±0.042	0.316 ±0.047	0.337 ±0.046	0.307 ±0.054	0.121 ±0.024	0.118 ±0.024	0.127 ±0.028	0.236 ±0.048
CoT	Sauerkraut	0.598 ±0.018	0.649 ±0.020	0.435 ±0.011	0.454 ±0.012	0.431 ±0.011	0.228 ±0.025	0.227 ±0.024	0.230 ±0.024	0.588 ±0.023
	Mistral	**0.528*** ±0.035	**0.721*** ±0.046	**0.529*** ±0.018	**0.605*** ±0.010	**0.487*** ±0.020	**0.333*** ±0.048	**0.346*** ±0.039	**0.352*** ±0.083	**0.751*** ±0.050
	QwQ	0.668 ±0.023	0.522 ±0.026	0.400 ±0.036	0.438 ±0.034	0.377 ±0.036	0.267 ±0.030	0.258 ±0.029	0.290 ±0.040	0.519 ±0.095

F Digital Appendix: Annotation Materials

The digital appendix provides read-only access to the original materials used for the expert annotation to evaluate the human-generated SHACL shapes graphs. For each annotator, a separate Google Drive folder contains the following:

- Guidelines: The instructions provided to the annotators outlining the annotation procedure and criteria.
- Annotation Sheet: A spreadsheet capturing the annotator's evaluations and comments.
- Files Round 1 and Files Round 2: Reviewed input texts and corresponding SHACL shapes graphs for each annotation round.

Access the materials here:

- Annotator 1
- Annotator 2

Note on AI Usage

This thesis was generated with the help of the following generative artifical intelligence tools:

- DeepL for translation because the author is a non-native English speaker
- Grammarly to correct grammatical mistakes.
- ChatGPT for discussion of ideas, improvement of wording, debugging.
- GitHub Copilot for code completion.