Schema Generation for Large Knowledge Graphs Using Large Language Models

Bohui Zhang[1] **Yuan He**[2] **Lydia Pintscher**[3] **Albert Meroño Peñuela**[1] **Elena Simperl**[1,4]

[1]King's College London [2]University of Oxford [3]Wikimedia Deutschland
[4]Technical University of Munich

{bohui.zhang, elena.simperl}@kcl.ac.uk

Abstract

Schemas are vital for ensuring data quality in the Semantic Web and natural language processing. Traditionally, their creation demands substantial involvement from knowledge engineers and domain experts. Leveraging the impressive capabilities of large language models (LLMs) in related tasks like ontology engineering, we explore automatic schema generation using LLMs. To bridge the resource gap, we introduce two datasets: YAGO Schema and Wikidata EntitySchema, along with evaluation metrics. The LLM-based pipelines effectively utilize local and global information from knowledge graphs (KGs) to generate validating schemas in Shape Expressions (ShEx). Experiments demonstrate LLMs' strong potential in producing high-quality ShEx schemas, paving the way for scalable, automated schema generation for large KGs. Furthermore, our benchmark introduces a new challenge for structured generation, pushing the limits of LLMs on syntactically rich formalisms.

1 Introduction

Graphs have emerged as a vital area of research in artificial intelligence and its foundational disciplines, significantly advancing progress across various domains, including knowledge representation and natural language processing (Sakr et al., 2021; Scherp et al., 2024; Hogan et al., 2025). This is especially evident with the rise of large language models (LLMs) (Brown et al., 2020), where graph-based methods enhance their reasoning capabilities for structured knowledge integration, and graphs serve as rich sources of structured and factual information (Zhang, 2023; Sun et al., 2023; Edge et al., 2024). Large knowledge graphs (KGs), such as Wikidata (Vrandečić and Krötzsch, 2014) and DBpedia (Auer et al., 2007), are compiled from heterogeneous sources, leading to significant quality issues like redundancy, noise, and ambiguity (Shenoy et al., 2022). Beyond data noise, KGs frequently suffer from modeling issues. A survey by Wikimedia Deutschland on Wikidata's ontology issues revealed conceptual ambiguity and inconsistent modeling in Wikidata, stemming from diverse contributor perspectives and inadequate guidelines—challenges common to large KGs (Ammalainen, 2023). This can manifest, for example, as entities for a company, its service and application being conflated, with predicates incorrectly shared between them. These quality deficits impede effective KG querying, sharing, and reuse. Critically, as KGs underpin tasks like pre-training (Chen et al., 2020; Pan et al., 2022; Yasunaga et al., 2022), retrieval-augmented generation (Xu et al., 2024; He et al., 2024a; Fang et al., 2024; Hu et al., 2025), and post-training (Agarwal et al., 2021; Li et al., 2023; Tang et al., 2024) for LLMs, ensuring KG quality is essential for maintaining the factual accuracy and reliability of these downstream AI systems (Pan et al., 2023).

Validating KGs against predefined schemas is a crucial technique for quality assessment, effective in identifying structural and semantic inconsistencies to maintain data integrity (Gayo et al., 2018; Scherp et al., 2024). These validating schemas define the expected rules and patterns for the data, focusing on specific sets of nodes (Ahmetaj et al., 2025). To represent these schemas, distinct languages including W3C standards like Shapes Constraint Language (SHACL) (Knublauch and Kontokostas, 2017), Shape Expressions (ShEx) (Prud'hommeaux et al., 2014), and PG-Schema (Angles et al., 2023) have emerged. Developing high-quality validation schemas aligned with user needs is a considerable challenge (Rabbani et al., 2022). Current automatic schema generation via pattern aggregation often inherits noise and errors from the source KGs (Rabbani et al., 2022). As a result, the primary approach involves manually writing schemas or refining those produced by basic pattern mining—a process that is both time-consuming and expensive, particularly for KGs with potentially millions of classes (Rabbani et al., 2022). The demonstrated success of LLMs in areas like ontology engineering (Lo et al., 2024; Zhang et al., 2025) and graph-based reasoning (Wang et al., 2023) highlights their proficiency with structured data. Specifically, LLMs show aptitude for processing structured information, reasoning across graph and text modalities, and performing structured generation (He et al., 2024b; Jin et al., 2024). This aptitude strongly suggests their potential for automatically generating KG validation schemas.

Building on this, this paper investigates how LLMs can undertake this automatic schema generation for large KGs, especially where such schemas are missing or incomplete. Specifically, we address the research question: *How can LLMs generate high-quality ShEx shapes for a given KG?* We identify key challenges inherent to this task. First, due to the sheer scale of large KGs, pinpointing the essential information required for accurate ShEx generation by LLMs is difficult. Second, even when relevant information is identified, investigating how LLMs can effectively leverage this structured data for schema generation is required. Finally, the relative novelty and ongoing development of the ShEx standard mean that publicly available ShEx corpora are limited, potentially hindering LLM familiarity with the required syntax and conventions. To address these challenges, this paper makes the following main contributions:

- We formulate the novel task of using LLMs to generate ShEx schemas from KGs, relevant for both the Semantic Web and LLM research.
- We introduce a benchmark comprising two datasets derived from large KGs—YAGO and Wikidata—along with associated metrics designed for comprehensive evaluation from diverse perspectives.
- We propose and evaluate pipelines designed to efficiently generate high-quality ShEx schemas by leveraging the structured generation capabilities of LLMs[1].

2 Related Work

Schema Generation KG schemas are broadly classified into semantic, validating, and emergent schemas (Hogan et al., 2021). We focus on **validating schemas**, typically defined using shapes. A *shape* identifies a set of focus nodes within a data graph and specifies constraints they must adhere to. ShEx is a prominent language for expressing such schemas, especially in Wikidata.

Existing automatic ShEx schema extraction approaches operate on KGs provided as RDF data or via query endpoints, employing a two-stage process: collecting essential information from KGs, and then extracting and refining shapes based on this information. Mihindukulasooriya et al. (2018) approached shape generation by framing cardinality and range constraint prediction as machine learning (ML) classification problems on data profiling features. The sheXer system (Fernandez-Álvarez et al., 2022) extracts shapes by iteratively exploring triples associated with target nodes and mining KG structures. A key finding is that shapes derived from representative instance examples often converge with those from larger sets, suggesting sampling can be effective for accurate shape extraction. Influenced by graph pattern mining, QSE (Rabbani et al., 2023) extracts shapes from large KGs by computing support (number of entities satisfying a constraint) and confidence (proportion of entities satisfying a constraint among applicable entities). While defined differently, these metrics are conceptually similar to sheXer's internal voting and trustworthiness scores.

Despite these advancements, studies (Rabbani et al., 2022; Fernandez-Álvarez et al., 2022; Rabbani et al., 2023) indicate that current methods often produce incomplete shapes, frequently missing essential elements like cardinality constraints, and lack standardized benchmarks for schema generation. Current evaluation practices primarily focus on generation efficiency, such as running time and memory usage. These limitations underscore the need for novel approaches to generate more comprehensive and accurate validating schemas for semantic quality and completeness.

Structured Generation Structured generation, or constrained decoding, enables LLMs to produce outputs adhering to precise formats, vital for applications like code generation (Ugare et al., 2024) and tool calling (Zhang et al., 2024). This involves generating token sequences that satisfy specified constraints, often defined by regular expressions (regex) or context-free grammars (CFGs). A key challenge is efficiently applying these constraints

[1]We will release our code upon acceptance.

over the LLM's large vocabulary without degrading speed or performance (Dong et al., 2024; Koo et al., 2024). Fine-tune-free methods primarily fall into two categories: token-level constrained decoding and prompt-level output structuring. Token-level methods directly influence token selection during decoding. They often compile regex and CFGs into automata (finite state machines or pushdown automata) that operate in the token space (Koo et al., 2024). This generation efficiency is further enhanced through techniques like states tracking and indexing (Willard and Louf, 2023), and token categorization and caching (Dong et al., 2024). Prompt-level methods leverage LLMs' instruction-following or function-calling capabilities to guide the overall output structure, which is subsequently parsed and validated. Instructor (Liu and Contributors, 2024) exemplifies this by converting Pydantic models into LLM-understandable schemas (e.g., JSON Schema) and incorporating a retry mechanism with feedback.

3 Problem Formulation

The Resource Description Framework (RDF) is a standard model for representing data and is widely used in building KGs (Cyganiak et al., 2014). We define a KG as follows:

Definition 1 (Knowledge Graph) *A KG in RDF is a directed, labeled graph $\mathcal{G}$, defined as a set of triples. Each triple $(u, p, o) \in \mathcal{G}$ consists of a subject u, a predicate p, and an object o. Given distinct sets of IRIs $\mathcal{I}$, blank nodes $\mathcal{B}$, and literals $\mathcal{L}$, the subject $u \in (\mathcal{I} \cup \mathcal{B})$ must be an IRI or a blank node, the predicate $p \in \mathcal{I}$ must be an IRI, and the object $o \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ may be an IRI, a literal, or a blank node.*

In particular, we distinguish non-blank subject nodes based on their role in the taxonomy: classes and instances. Classes are primarily connected to their superclasses using predicates such as `subClassOf`. Instances are linked to their corresponding classes using predicates like `instanceOf`. To validate instances within a class, we can define a validating schema in ShEx as follows:

Definition 2 (Shape) *A shape schema in ShEx consists of a set of shapes $\mathcal{S}$. Each shape is associated with a class c in a KG. A shape $s = (\alpha, \Psi) \in \mathcal{S}$ comprises a shape label α and a set of constraints $\psi \in \Psi$ in the form $\psi = (p, \tau, \kappa)$, where p is a predicate, τ is a node constraint, and κ specifies cardinality.*

Dataset		YAGOS	WES
Classes		36	50
Constraints	Sum	678	1,874
	Mean	18.83	37.48
	Median	18	35
Instances	Sum	1,227,509	2,127,696
	Mean	34,097.47	42,553.92
	Median	1,104	1,564

Table 1: Dataset statistics, including the number of classes, total number of constraints, average schema length, median constraint length, total number of instances across all classes, and the mean and median number of instances per class.

Predicates used in shape constraints are drawn from the KG. Node constraints τ may belong to several categories, including (1) node kind constraints, (2) datatype constraints, (3) values constraints, (4) XML Schema string facet constraints and (5) XML Schema numeric facet constraints. In this work, we focus on the first three categories. Cardinality $\kappa = (n, m)$ specify the allowable number of occurrences of a predicate-object pair, where $n \in \mathbb{N}$ and $m \in \mathbb{N} \cup \{*\}$, with '$*$' indicating an unbounded upper limit. ShEx examples are provided in Appendix B. The task of schema generation can now be defined as:

Definition 3 (Schema Generation) *Given a KG $\mathcal{G}$, and a class $c \in \mathcal{G}$ representing a set of nodes, the objective is to generate a shape schema $\mathcal{S}$ describing the triples involving nodes in the KG.*

Schema generation is a challenging task for both traditional rule-based methods (Rabbani et al., 2023) and ML approaches like LLMs, primarily due to several factors. Large KGs often contain quality issues, and pattern extraction may inadvertently encode these issues into generated shapes. Moreover, there is a lack of benchmarks and high-quality ShEx ground truths, making training and evaluation for ML-based approaches particularly difficult.

4 Schema Benchmark

4.1 Dataset

To address the need for benchmarks, we introduce two new dataset, YAGO Schema (YAGOS) and Wikidata EntitySchema (WES). Together, comprising 86 ShEx schema with a total of 2,552 constraints. Detailed statistics are presented in Table 1, and the dataset construction process is outlined in Appendix A. Each schema in our datasets targets a specific class within the respective KG. While ShEx offers a rich syntax, the schemas in our datasets employ a community-prevalent subset to enhance simplicity and readability while remaining functional (detailed specification in Appendix B). In our setting, to simplify the syntax representation and ease evaluation and comparison, a shape schema $\mathcal{S}$ includes a *start shape* that targets the focus class. The remaining shapes in the schema serve as references, each describing one or more classes that specify the range of objects allowed for certain predicates in the start shape. The schemas predominantly use four fundamental node constraint types: node kind constraints (e.g., IRI), datatype constraints (e.g., `xsd:decimal`, `rdf:langString`), value set constraints (often specifying object values, e.g., `[schema:Organization]`), and shape references (e.g., `@<Person>`, which links to another defined shape like `<Person> { rdf:type [ schema:Person ] }`).

4.2 Evaluation Metrics

Novel metrics are essential for evaluating the quality of constructed schema. Simple text matching is unreliable, since the order of constraints and naming of shapes in ShEx affect textual similarity but not semantic correctness. In this work, we evaluate using the ShExJ (JSON format), converted from ShExC, which enables structured analysis at shape and constraint levels. Our evaluation uses two metric types: similarity, with graph edit distance at the shape level, and classification, with F1-score as the main metrics on the constraint level.

4.2.1 Similarity Metrics

Given that automatically generated schemas often require manual refinement (Rabbani et al., 2022), quantifying the structural similarity between a generated schema and its ground truth counterpart is crucial. To facilitate this comparison, we model each shape schema as a rooted, labeled tree graph. The shape label (or focus node) serves as the root, predicates form the first level of child nodes, node constraints linked to predicates occupy the next level, and cardinalities appear as leaf nodes, as shown in Figure 2b. Based on this graph representation, we employ the Graph Edit Distance (GED) (Abu-Aisheh et al., 2015), specifically the Tree Edit Distance (Zhang and Shasha, 1989), to measure the dissimilarity between a generated schema $\mathcal{S}'$ and the ground truth schema $\mathcal{S}$. GED represents the minimum cost required to transform $\mathcal{S}'$ into $\mathcal{S}$ using a sequence of edit operations:

$$\mathcal{D}(\mathcal{S}', \mathcal{S}) = \min_{e_1,\cdots,e_{\mathscr{L}} \in \gamma(\mathcal{S}',\mathcal{S})} \sum_{i=1}^{\mathscr{L}} c(e_i) \quad (1)$$

where $\gamma(\mathcal{S}', \mathcal{S})$ is the set of all valid edit paths transforming $\mathcal{S}'$ to $\mathcal{S}$, and $c(e_i)$ is the cost of an individual edit operation e_i. Simply averaging raw GED scores across a dataset can be misleading. Smaller schemas naturally yield lower scores, possibly hiding significant relative errors, whereas larger schemas might skew the average. To mitigate this size bias, we normalize the GED by the maximum potential edit cost related to the ground truth schema's size, defined as $3 \cdot |\mathcal{S}|$.

$$\tilde{\mathcal{D}}(\mathcal{S}', \mathcal{S}) = \frac{1}{3 \cdot |\mathcal{S}|} \mathcal{D}(\mathcal{S}', \mathcal{S}) \quad (2)$$

The normalized GED (NGED) ranges from 0 (identical schemas) to potentially above 1 (if the transformation cost exceeds deleting the ground truth, e.g., due to many additions in $\mathcal{S}'$). We report the average GED and average NGED over the dataset of N schemas:

$$\begin{aligned} \text{GED} &= \frac{1}{N} \sum_{i}^{N} \mathcal{D}(\mathcal{S}_i', \mathcal{S}_i) \\ \text{NGED} &= \frac{1}{N} \sum_{i}^{N} \tilde{\mathcal{D}}(\mathcal{S}_i', \mathcal{S}_i) \end{aligned} \quad (3)$$

The metrics offer an interpretable measure of the structural accuracy of generated schemas against ground truth, considering both the absolute number of edits and the relative error normalized by size.

4.2.2 Classification Metrics

Since each ShEx constraint comprises three elements (predicate, node constraint, and cardinality), we propose several levels of matching criteria, inspired by Fernandez-Álvarez et al. (2022), where constraints of different specificity can get positive votes. These criteria reflect practical usage scenarios and accommodate variations in large KGs modeling, ensuring meaningful evaluation without sacrificing flexibility.

Exact Matching We define an exact match between two constraints ψ and ψ' as $\mathcal{E}_{\text{exact}}(\psi, \psi')$, where all elements must match:

$$\mathcal{E}_{\text{exact}}(\psi, \psi') = \mathbb{I}[(p \equiv p') \wedge (\tau \equiv \tau') \wedge (\kappa \equiv \kappa')] \quad (4)$$

Approximate Class Matching Node constraints involving value shapes (e.g., class constraints) can be matched approximately due to the following reasons. First, upper ontologies in large KGs are often noisy and redundant. Substituting a class with a similar one may preserve semantics. Second, classes used in ground truth often represent broad coverage, not exhaustive correctness. Third, ShEx allows flexibility using the `EXTRA` keyword, which tolerates constraints beyond negative definitions. Thus, a generated constraint can be considered approximately equivalent to the ground truth if both involve value shapes and meet either of the following conditions: (1) the class specified in the ground truth constraint is a subclass of the one(s) in the generated constraint, or (2) in the WES dataset, the class used in the generated constraint corresponds to the value-type constraint defined for the predicate. We formalize the criteria as follows:

$$\mathcal{E}_{\text{subclass}}(\tau, \tau') = \begin{cases} 1, & \exists c \in \mathcal{C}(\tau),\ \exists c' \in \mathcal{C}(\tau'),\ c \sqsubseteq c' \\ 1, & \exists c \in \mathcal{C}(p),\ \exists c' \in \mathcal{C}(\tau'),\ c \sqsubseteq c' \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\mathcal{E}_{\text{subclass}}(\psi, \psi') = \mathbb{I}[(p \equiv p') \wedge \mathcal{E}_{\text{subclass}}(\tau, \tau') \wedge (\kappa \equiv \kappa')] \tag{6}$$

Here $\mathcal{C}(\tau)$ represents the set of classes defined in the node constraint τ, and $\mathcal{C}(p)$ refers to the list of value-type constraints for the predicate p from Wikidata.

Datatype Matching For node constraints, we further define a relaxed criterion based on datatype compatibility. This criteria defines if the datatype of the node constraint matches, while requiring exact matches for the predicate and cardinality:

$$\mathcal{E}_{\text{datatype}}(\psi, \psi') = \mathbb{I}[(p \equiv p') \wedge (d(\tau) \equiv d(\tau')) \wedge (\kappa \equiv \kappa')] \tag{7}$$

where $d(\tau)$ extracts the datatype from the node constraint. Datatypes within the dataset's scope are converted into four general categories: `xsd:dateTime`, `xsd:decimal`, `xsd:string`, and IRI. The list of datatypes is extensible and can be redefined depending on the characteristics of the input KGs.

Loosened Cardinality Besides node constraints, we relax the evaluation by allowing broader matches on cardinality. This is particularly useful when the exact cardinality range is less critical, and a looser bound—such as simply requiring optional presence—is sufficient for validation:

$$\mathcal{E}(\kappa, \kappa') = \begin{cases} 1, & 0 \le n' \le n \le m \le m' \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$\mathcal{E}_{\text{cardinality}}(\psi, \psi') = \mathbb{I}[(p \equiv p') \wedge (\tau \equiv \tau') \wedge \mathcal{E}(\kappa, \kappa')] \tag{9}$$

Combined Relaxations The relaxation strategies described above can also be combined to accommodate a wider range of scenarios:

$$\mathcal{E}_{\text{subclass+cardinality}}(\psi, \psi') = \mathbb{I}[(p \equiv p') \wedge \mathcal{E}_{\text{subclass}}(\tau, \tau') \wedge \mathcal{E}(\kappa, \kappa')] \tag{10}$$

$$\mathcal{E}_{\text{datatype+cardinality}}(\psi, \psi') = \mathbb{I}[(p \equiv p') \wedge (d(\tau) \equiv d(\tau')) \wedge \mathcal{E}(\kappa, \kappa')] \tag{11}$$

Give a generated schema and a ground truth schema, we report macro-averaged precision, recall, and F1-score, defined by:

$$\begin{aligned} \text{Precision} &= \frac{|\{\psi \mid \mathcal{E}(\psi, \psi') = 1, \psi \in \Psi\}|}{|\Psi|} \\ \text{Recall} &= \frac{|\{\psi \mid \mathcal{E}(\psi, \psi') = 1, \psi \in \Psi\}|}{|\Psi'|} \\ F_1 &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{12}$$

5 Experimental Setup

5.1 Models

To evaluate the capabilities of LLMs in shape generation, we compare against several baseline models. Our primary non-ML baseline is sheXer (Fernandez-Álvarez et al., 2022), a well-established system for KG shape extraction. For LLM-based comparisons, we selected `gpt-4o-mini`[2] and DeepSeek-V3 (DeepSeek-AI, 2024). Input data, including triples and global KG information, is retrieved using SPARQL queries against locally deployed SPARQL endpoints.

5.2 Prompt Engineering

To investigate how LLMs can effectively generate ShEx schemas and understand the impact of information type and generation mode, we designed experiments incorporating different types of information extracted from a KG. Based on the nature of this information, we developed and evaluated three distinct few-shot prompt engineering settings, summarized in Table 2.

[2] `https://platform.openai.com/docs/models/gpt-4o-mini`

Information Type	Local	Global	Triples
Input			
Class URI & label	✓	✓	✓
Class description		✓	
Predicate URI & label		✓	
Predicate description*		✓	
Instance triples	✓		
Predicate triples		✓	✓
Predicate frequencies		✓	
Datatype of objects		✓	
Cardinality distributions		✓	
Wikidata constraints*		✓	
Output			
Full ShEx schema script	✓		✓
Formatted constraints		✓	

Table 2: Prompt engineering settings. Information types marked with '$*$' are primarily available for WES dataset.

Local Setting This setting provides LLMs with a number of representative instance examples of the target class, along with their associated triples (typically 1-hop neighbors) from KGs. Instances are selected based on specific criteria. Wikidata entities are sorted by ID length and numeric value of their ID. The popularity and importance of entities is loosely correlated with their IDs, as important or fundamental concepts were added early in Wikidata's history. YAGO entities are sorted by the number of distinct predicates associated with them, prioritizing those with richer connections. The rationale is to allow LLMs to infer patterns directly from how entities of that class are described in the KG. In this setting, LLMs are asked to directly generate the complete ShEx schema, guided by few-shot examples of schemas.

Global Setting This setting focuses on aggregated, schema-level information about the target class and its predicates. As shown in Table 2, input comprises: (a) basic metadata (URIs, labels, and descriptions for the class and relevant predicates); (b) predicate usage statistics (predicate frequency and cardinality distributions); (c) datatype information (datatypes for predicate objects, and class distributions for objects that are URIs, which aids in identifying potential referenced shapes); (d) representative triple examples associated with the predicates; and (e) any available KG-specific constraints (for Wikidata, this includes predefined predicate constraints such as value-type[3] (range) and subject-type[4] (domain)). This setting aims to provide LLMs with a comprehensive, high-level summary of the class's structure and the characteristics of its predicates. LLMs are then tasked with generating these constraints in a structured JSON format, which is subsequently converted programmatically into ShEx. The structured generation process is detailed in Section 5.3.

Triples Setting Inspired by findings that representative samples can suffice for shape extraction (Fernandez-Álvarez et al., 2022), this setting provides the LLM with a set of triples focused on the usage of specific predicates relevant to the entities. Different from the local setting, this setting focuses on a set of predicates and provides example triples where those predicates appear, potentially sampled across many different instances. The goal is to highlight common patterns for individual predicates. Similar to the local setting, the LLM generates the ShEx schema with few-shot examples.

5.3 Structured Generation

Even though LLMs excel at structured generation with well-defined JSON schemas, directly applying the full ShEx syntax to the LLM's constraint decoding process is still challenging due to its complexity. To address this, our structured generation pipeline first simplifies the ShEx syntax within the context of the benchmark by converting it into a more manageable, JSON schema-like representation using ShExJ syntax. This process generates Pydantic models, each corresponding to a segment of ShEx syntax, to guide LLM constraint generation. We leverage the Instructor (Liu and Contributors, 2024) to enhance the LLM's structured generation capabilities.

We adopt a decomposed, two-step structured generation workflow specifically for the global setting (see Table 2), where the LLM processes global information from KGs. The first step is cardinality prediction. Given global information for a specific predicate relevant to the target class, the LLM is prompted to predict its cardinality, comprising the minimum and maximum occurrence values. The second step is node constraint prediction, applying on the predicates accepted by the previous step. Based on the global information, the LLM predicts the node constraint for the predicate's objects. If a datatype is evident from the input information,

[3]https://www.wikidata.org/wiki/Q21510865

[4]https://www.wikidata.org/wiki/Q21503250

Models	Settings	YAGO Schema					Wikidata EntitySchema				
		P	R	F1	GED	NGED	P	R	F1	GED	NGED
sheXer	/	0.111	0.081	0.092	34.08	0.581	0.106	0.099	0.096	90.30	0.833
`gpt-4o-mini`	local	0.575	0.550	0.559	**16.61**	0.308	**0.358**	0.224	0.264	82.14	0.697
	global	0.421	0.362	0.388	21.03	0.366	0.306	0.328	0.312	**54.44**	**0.484**
	triples	**0.631**	**0.564**	**0.591**	18.36	0.344	0.328	0.196	0.237	72.44	0.577
DeepSeek-V3	local	0.536	0.524	0.526	18.56	0.348	0.322	0.240	0.263	66.16	0.585
	global	0.478	0.484	0.479	16.83	**0.295**	0.304	**0.343**	**0.318**	57.36	0.494
	triples	0.535	0.505	0.510	21.89	0.468	0.269	0.277	0.269	55.80	0.488

Table 3: Results of LLMs comparing with baseline models across different settings. Note that for YAGO Schema, entities of triple examples are sorted by predicate count, and for WES, entities are sorted by their IDs. Five entities and their related triples are retrieved and feed into LLMs. The highest scores are set in **bold**.

Settings	Matching Criteria		YAGO Schema			Wikidata EntitySchema		
	Node Constraint	Cardinality	P	R	F1	P	R	F1
local	Exact	Exact	0.536	0.524	0.526	0.322	0.240	0.263
	Subclass	Exact	0.551	0.540	0.542	0.413	0.303	0.335
	Datatype	Exact	0.634	0.625	0.626	0.537	0.396	0.437
	Exact	Loosened	0.589	0.582	0.580	0.366	0.280	0.303
	Subclass	Loosened	0.604	0.598	0.596	0.466	0.352	0.384
	Datatype	Loosened	0.753	0.751	0.746	0.613	0.464	0.507
global	Exact	Exact	0.434	0.439	0.435	0.304	0.343	0.318
	Subclass	Exact	0.521	0.528	0.523	0.482	0.550	0.507
	Datatype	Exact	0.698	0.709	0.701	0.577	0.655	0.606
	Exact	Loosened	0.441	0.446	0.442	0.394	0.451	0.415
	Subclass	Loosened	0.532	0.541	0.535	0.638	0.738	0.676
	Datatype	Loosened	0.751	**0.764**	**0.755**	**0.793**	**0.910**	**0.839**
triples	Exact	Exact	0.535	0.505	0.510	0.269	0.277	0.269
	Subclass	Exact	0.554	0.523	0.528	0.371	0.377	0.367
	Datatype	Exact	0.630	0.591	0.600	0.536	0.549	0.534
	Exact	Loosened	0.581	0.550	0.554	0.320	0.334	0.322
	Subclass	Loosened	0.604	0.570	0.575	0.449	0.461	0.448
	Datatype	Loosened	**0.759**	0.720	0.725	0.675	0.699	0.677

Table 4: Results of DeepSeek-V3 under different matching criteria.

the LLM outputs this datatype. If the predicate's objects are instances of another specific classes, the LLM outputs the URIs of these referenced classes, forming the basis for a ShEx shape reference. If the predicate's objects are restricted to a fixed list of literal values, the LLM generates this complete list. Finally, if none of these conditions are strongly indicated, the LLM defaults to a general node kind constraint, which for our current datasets is typically fixed as IRI.

6 Results

Table 3 presents a comparative performance analysis of the baseline models on YAGOS and WES datasets. On YAGOS, `gpt-4o-mini` achieved the highest F1 (0.591) in its triples setting, while DeepSeek-V3's global setting showed the best structural similarity (lowest NGED of 0.295). YAGOS's generally stronger results are attributed to its ontology being largely derived and refined from schema.org, along with its smaller scale—having only half the average number of constraints and 80% of the average number of instances per class compared to WES. For the more challenging WES dataset with a complex predicate vocabulary and twice the constraints per class, DeepSeek-V3 in the global setting yielded the highest F1 (0.318). Both `gpt-4o-mini` and DeepSeek-V3 performed well on the NGED score in their global setting. Compared to the non-ML baseline sheXer, these results underscore the potent schema generation capabilities of LLM-based approaches.

We further analyzed the performance of models under different matching criteria, using DeepSeek-V3 as an example (Table 4). Full results for other models are in Appendix C. In general, loosening

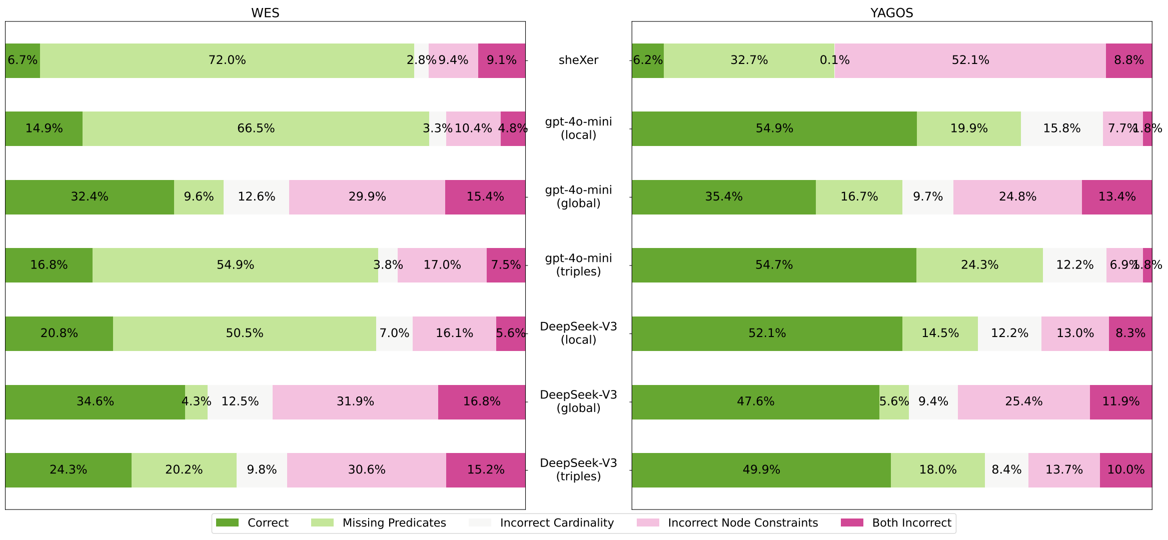


Figure 1: Error distribution across models and settings on WES (Left) and YAGOS (Right) datasets. The figure shows five categories: four error types and correctly generated constraints.

the matching criteria leads to improved evaluation scores, as expected. Notably, when combining datatype abstraction with loosened cardinality, DeepSeek-V3 in the global setting achieved impressive F1 scores (0.755 on YAGO, 0.839 on WES). These high scores suggest that LLMs can generate schemas for certain practical validation scenarios, particularly where the primary goal is to ensure predicate completeness and general object constraints.

As for individual criteria, allowing datatype abstraction provided the most significant performance gain, especially in the global setting. This indicates LLMs can effectively process given datatype information, finding it less challenging than inferring cardinality. Allowing approximate subclass matching improved classification scores more for WES compared with YAGOS, suggesting predicting the precise class list for referenced shapes is harder for WES, while generating related object classes is not. These findings indicate that although LLM-generated schemas may benefit from refinement, the required adjustments are generally less intensive or domain-specific than those for existing automated approaches, while still supporting effective validation.

Figure 1 shows the distribution of results across five categories (correct and four error types) for models and settings on WES and YAGOS. Error types are: (1) missing predicates, (2) incorrect cardinality, (3) incorrect node constraint, and (4) both cardinality and node constraint are incorrect. Global settings effectively reduce missing predicates for models on both datasets. Missing predicates are more frequent on WES than YAGOS, likely due to WES's larger set of candidate predicates. Global settings also tend to reduce instances of only incorrect node constraints, particularly on YAGOS. However, global settings often show a higher rate of errors where both cardinality and node constraint are incorrect, especially on the WES dataset.

7 Conclusion

The persistent challenge of ensuring data quality in large KGs necessitates effective and automated methods for generating validation schemas. This paper explored the application of LLMs to this task, introducing the first benchmark for ShEx schema generation from KGs. This benchmark, comprising two datasets and custom metrics, enabled a thorough assessment revealing LLMs's ability to produce high-quality shape schemas. Beyond its direct contributions to Semantic Web practices, this work provides a new benchmark for evaluating the nuanced graph understanding and structured generation capabilities of LLMs. Future research could aim to broaden the benchmark's scope by incorporating more ShEx features—such as additional constraint types (e.g., string and numeric facet constraints) and structural elements (e.g., imported schemas and annotations)—as well as by including diverse schema languages like PG-Schema. Moreover, advancing the models, particularly by enhancing their structured generation ability for complex schemas will be crucial.

Limitations

The size of the dataset is constrained by limited resources available for its construction. Creating high-quality shape schemas is a time-consuming process that requires multiple rounds of refinement by volunteer knowledge engineers. As a result, the current size of the dataset makes it difficult to convincingly evaluate the effectiveness or efficiency of traditional machine learning or deep learning approaches—such as embedding-based models—on this task.

Additionally, our evaluation of LLMs is limited to those accessible via APIs. Models that can be run locally, such as those with 8B or 14B parameters, are not included. Preliminary experiments indicate that these smaller, locally runnable LLMs struggle to generate valid ShEx schemas and lack sufficient structured generation capabilities to reliably produce correct constraint syntax. Nevertheless, we believe that with further development, such models could become strong candidates for this task.

Ethics Statement

The YAGOS and WES datasets are developed with a strong commitment to ethical AI principles. They contain no personal, sensitive, or identifiable information and are free from harmful, offensive, or misleading content. Both datasets strictly comply with responsible AI guidelines.

Acknowledgments

This work was supported by the NMES Enterprise & Engagement Partnerships fund. Co-funded by the SIEMENS AG and the Technical University of Munich, Institute for Advanced Study, Germany.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods - Volume 1*, ICPRAM 2015, page 271–278, Setubal, PRT. SCITEPRESS - Science and Technology Publications, Lda.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Shqiponja Ahmetaj, Iovka Boneva, Jan Hidders, Katja Hose, Maxime Jakubowski, Jose Emilio Labra Gayo, Wim Martens, Fabio Mogavero, Filip Murlak, Cem Okulmus, Axel Polleres, Ognjen Savković, Mantas Šimkus, and Dominik Tomaszuk. 2025. Common Foundations for SHACL, ShEx, and PG-Schema. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 8–21, New York, NY, USA. Association for Computing Machinery.

Daria Ammalainen. 2023. Wikidata Ontology Issues — Suggestions for prioritisation 2023. Accessed: March 12, 2025.

Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. 2023. PG-Schema: Schemas for Property Graphs. *Proc. ACM Manag. Data*, 1(2).

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: a nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg. Springer-Verlag.

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. *CoRR*, abs/2005.14165.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8635–8648, Online. Association for Computational Linguistics.

Richard Cyganiak, David Wood, and Markus Lanthaler. 2014. RDF 1.1 Concepts and Abstract Syntax. W3c recommendation, W3C.

DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. *Preprint*, arXiv:2412.19437.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2024.

XGrammar: Flexible and Efficient Structured Generation Engine for Large Language Models. *CoRR*, abs/2411.15100.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization.

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. REANO: Optimising Retrieval-Augmented Reader Models through Knowledge Graph Generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2094–2112, Bangkok, Thailand. Association for Computational Linguistics.

Daniel Fernandez-Álvarez, Jose Emilio Labra-Gayo, and Daniel Gayo-Avello. 2022. Automatic extraction of shapes using sheXer. *Knowledge-Based Systems*, 238:107975.

Jose Emilio Labra Gayo. 2022. WShEx: A language to describe and validate Wikibase entities. *Preprint*, arXiv:2208.02697.

Jose Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. 2018. *Validating RDF Data*. Springer International Publishing.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024a. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yuan He, Zhangdie Yuan, Jiaoyan Chen, and Ian Horrocks. 2024b. Language Models as Hierarchy Encoders. In *Advances in Neural Information Processing Systems*, volume 37, pages 14690–14711. Curran Associates, Inc.

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. *ACM Comput. Surv.*, 54(4).

Aidan Hogan, Xin Luna Dong, Denny Vrandečić, and Gerhard Weikum. 2025. Large Language Models, Knowledge Graphs and Search Engines: A Crossroads for Answering Users' Questions. *Preprint*, arXiv:2501.06699.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph Retrieval-Augmented Generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4145–4157, Albuquerque, New Mexico. Association for Computational Linguistics.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large Language Models on Graphs: A Comprehensive Survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8622–8642.

Holger Knublauch and Dimitris Kontokostas. 2017. Shapes Constraint Language (SHACL). W3c recommendation.

Terry Koo, Frederick Liu, and Luheng He. 2024. Automata-based constraints for language model decoding. In *First Conference on Language Modeling*.

Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. 2023. GraphAdapter: Tuning Vision-Language Models With Dual Knowledge Graph. In *Advances in Neural Information Processing Systems*, volume 36, pages 13448–13466. Curran Associates, Inc.

Jason Liu and Contributors. 2024. Instructor: A library for structured outputs from large language models.

Andy Lo, Albert Q. Jiang, Wenda Li, and Mateja Jamnik. 2024. End-to-End Ontology Learning with Large Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Nandana Mihindukulasooriya, Mohammad Rifat Ahmad Rashid, Giuseppe Rizzo, Raúl García-Castro, Oscar Corcho, and Marco Torchiano. 2018. RDF shape induction using knowledge base profiling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1952–1959, New York, NY, USA. Association for Computing Machinery.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large Language Models and Knowledge Graphs: Opportunities and Challenges. *Transactions on Graph Data and Knowledge*, 1(1):2:1–2:38.

Xuran Pan, Tianzhu Ye, Dongchen Han, Shiji Song, and Gao Huang. 2022. Contrastive Language-Image Pre-Training with Knowledge Graphs. In *Advances in Neural Information Processing Systems*, volume 35, pages 22895–22910. Curran Associates, Inc.

Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. 2014. Shape expressions: an RDF validation and transformation language. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, page 32–40, New York, NY, USA. Association for Computing Machinery.

Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2022. SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 260–263, New York, NY, USA. Association for Computing Machinery.

Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2023. Extraction of Validating Shapes from Very Large Knowledge Graphs. *Proc. VLDB Endow.*, 16(5):1023–1032.

Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei R. Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM*, 64(9):62–71.

Ansgar Scherp, Gerd Groener, Petr Škoda, Katja Hose, and Maria-Esther Vidal. 2024. Semantic Web: Past, Present, and Future. *Transactions on Graph Data and Knowledge*, 2(1):3:1–3:37.

Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro Szekely. 2022. A study of the quality of Wikidata. *Journal of Web Semantics*, 72:100679.

Fabian M. Suchanek, Mehwish Alam, Thomas Bonald, Lihu Chen, Pierre-Henri Paris, and Jules Soria. 2024. YAGO 4.5: A Large and Clean Knowledge Base with a Rich Taxonomy. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 131–140, New York, NY, USA. Association for Computing Machinery.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model with Knowledge Graph. *Preprint*, arXiv:2307.07697.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. GraphGPT: Graph Instruction Tuning for Large Language Models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 491–500, New York, NY, USA. Association for Computing Machinery.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. 2024. Improving LLM Code Generation with Grammar Augmentation. *CoRR*, abs/2403.01632.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledge base. *Commun. ACM*, 57(10):78–85.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. 2024. Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2905–2909, New York, NY, USA. Association for Computing Machinery.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep Bidirectional Language-Knowledge Graph Pretraining. In *Advances in Neural Information Processing Systems*, volume 35, pages 37309–37323. Curran Associates, Inc.

Bohui Zhang, Valentina Anita Carriero, Katrin Schreiberhuber, Stefani Tsaneva, Lucía Sánchez González, Jongmo Kim, and Jacopo de Berardinis. 2025. OntoChat: A Framework for Conversational Ontology Engineering Using Language Models. In *The Semantic Web: ESWC 2024 Satellite Events: Hersonissos, Crete, Greece, May 26–30, 2024, Proceedings, Part I*, page 102–121, Berlin, Heidelberg. Springer-Verlag.

Jiawei Zhang. 2023. Graph-ToolFormer: To Empower LLMs with Graph Reasoning Ability via Prompt Augmented by ChatGPT. *ArXiv*, abs/2304.11116.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. 2024. Don't Fine-Tune, Decode: Syntax Error-Free Tool Use via Constrained Decoding. *Preprint*, arXiv:2310.07075.

A Dataset Construction

The YAGOS dataset was constructed using YAGO 4.5 (Suchanek et al., 2024) as the input KG. Ground truth ShEx schema construction was informed by the existing SHACL schema and the official YAGO 4.5 design document[5]. While the provided documentation offered a starting point, it was not exhaustive, with only a subset of key properties and constraints defined. Therefore, the final ground truth ShEx scripts were manually crafted by retaining constraints explicitly mentioned in the design document and adding missing predicates relevant to each class. This was achieved by querying the KG to understand common property usage for entities belonging to the target classes and aligning these findings with the schema's intended scope.

The WES dataset was developed through a semi-automatic process based on the Wikidata (Vrandečić and Krötzsch, 2014) truthy statements RDF dump (from qEndpoint's Wikidata release 1.16.1[6]), involving knowledge engineers assisted by tools for pattern extraction and data querying. Target classes for the dataset were selected from three main sources:

1. Community-curated Wikidata EntitySchema directory: We drew upon schemas available in the Wikidata EntitySchema directory[7]. Due to the variability in quality and completeness of these community contributions, we manually selected a subset of those deemed relatively high-quality to serve as initial drafts. These selected drafts were then collaboratively reviewed, refined, and standardized by our knowledge engineers to meet consistent quality criteria.

2. Classes mapped from YAGOS: A set of classes was chosen by mapping them from the YAGOS dataset. Since YAGO facts are partially derived from Wikidata, creating schemas for these corresponding classes in WES allows for a comparison of modeling scope between the two KGs.

3. Classes mapped from Wikipedia Categories: Additional classes identified through mappings from relevant Wikipedia Categories[8], focusing on well-defined concepts suitable for schema modeling.

[5]https://yago-knowledge.org/data/yago4.5/design-document.pdf

[6]https://github.com/the-qa-company/qEndpoint/releases/tag/v1.16.1

[7]https://www.wikidata.org/wiki/Wikidata:Database_reports/EntitySchema_directory

Property Type	Count	Proportion (%)	Rank
WikibaseItem	1,607	14.30%	2
Quantity	646	5.75%	3
String	322	2.86%	4
Url	99	0.88%	5
Time	63	0.56%	7
Monolingualtext	60	0.53%	8

Table 5: Distribution of primary property datatypes in Wikidata.

For each selected class, the following iterative process was undertaken by knowledge engineers:

1. Predicate Identification and Prioritization: A comprehensive list of predicates associated with entities of the target class was compiled, along with their occurrence frequencies. Based on usage frequency and modeling importance (as detailed in Table 5, which outlines key Wikidata property types considered), a working set of candidate properties was selected.

2. Predicate Inclusion and Refinement: Candidate predicates were evaluated for inclusion based on their semantic appropriateness (i.e., factual suitability for the class) and through property denoising and deduplication. The latter step involved identifying and consolidating functionally similar or overlapping properties common in Wikidata by selecting the most representative and predominantly used one. For instance, to model an item's inception, one would choose the most suitable property from options like P580 (start time), P571 (inception), etc.

3. Cardinality Determination: The cardinality for each included predicate was established. Predicates were generally assigned an optional cardinality (e.g., minimum 0 and maximum '$*$' for zero or more, or minimum 0 and maximum 1 for at most one). This default was overridden if high frequency and semantic necessity strongly suggested a mandatory presence (a minimum of 1) or a more specific range.

[8]https://en.wikipedia.org/wiki/Wikipedia:Contents/Categories

4. Node Constraint Specification: The constraints on the object values of each predicate were defined.
   - Datatype Constraints: If a predicate consistently uses a specific datatype, a direct datatype constraint was applied.
   - Value Set: If objects were consistently drawn from a small, fixed list of literal values or specific URIs, a value set constraint was used.
   - Shape Reference: If objects were typically instances of or subclass of a few related classes, a shape reference was created. Knowledge engineers selected the most relevant object class(es) based on observed distributions and schema readability.
   - Node Kind: If neither a specific datatype, value set, nor a clear referenced shape was appropriate, a general node kind constraint (typically 'IRI') was applied.

To ensure the quality and consistency of datasets, three volunteers, all active researchers and engineers in ontology engineering with prior schema generation experience, were recruited from Wikidata community events. This process was supported by semi-automatic tools that incorporated a series of SPARQL queries to gather relevant statistics and patterns from Wikidata (representative examples of these queries are provided in Listings 1, 2, 3, and 4). Schema annotation involved three experts independently annotating each class's schema. Their annotations were collected, and a constraint was included in the final ground truth ShEx schema if at least two experts independently proposed an equivalent formulation. Discrepancies or cases with less than two-thirds agreement were resolved through discussion among the annotators or, if consensus could not be reached, the contentious constraint was omitted to maintain high confidence in the final dataset.

```
SELECT DISTINCT ?predicate (COUNT(
    DISTINCT ?subject) AS ?count)
WHERE {
  ?subject wdt:P31 wd:Q1248784 ;
           ?predicate ?object .
}
GROUP BY ?predicate
ORDER BY DESC(?count)
```

Listing 1: SPARQL query to retrieve predicates used with instances of Airport (Q1248784) and their usage frequency.

```
SELECT DISTINCT ?predicate ?datatype
WHERE {
  ?subject wdt:P31 wd:Q1248784 ;
           ?predicate ?object .
  BIND (datatype(?object) AS ?datatype)
}
```

Listing 2: SPARQL query to identify datatypes of objects for predicates associated with Airport (Q1248784).

```
SELECT (COUNT(DISTINCT ?subject) AS ?
    count)
WHERE {
  ?subject rdf:type schema:Book .
  FILTER NOT EXISTS {
    ?subject schema:illustrator ?object
  }
}
```

Listing 3: SPARQL query to count instances of `schema:Book` lacking a `schema:illustrator` predicate.

```
SELECT ?cardinality (COUNT(DISTINCT ?
    subject) AS ?count)
{
  SELECT DISTINCT ?subject (COUNT(?
      object) AS ?cardinality)
  WHERE {
    ?subject rdf:type schema:Book ;
             schema:illustrator ?object
                 .
  }
  GROUP BY ?subject
}
GROUP BY ?cardinality
ORDER BY DESC(?count)
```

Listing 4: SPARQL query to determine the distribution of `schema:illustrator` predicate occurrences per `schema:Book` instance (i.e. cardinality distribution).

YAGO 4.5 is licensed under a CC BY-SA license, and Wikidata is licensed under the CC0 license[9]. Our datasets are under the same licenses as the KGs from which they were derived, respectively.

B ShEx Specification

The ShEx was initially proposed in 2014, with its current specification published in 2019. The language continues to evolve to incorporate new functionalities addressing the diverse requirements of KG validation, as evidenced by extensions like WShEx for Wikidata EntitySchemas (Gayo, 2022). ShEx is often preferred for KG validation over traditional ontologies for several reasons. First, while ontologies can perform some validation tasks, their primary design focus is typically on entailment and

[9] https://www.wikidata.org/wiki/Wikidata:Licensing

reasoning, which can lead to less expressive or less direct validation capabilities (Mihindukulasooriya et al., 2018). Furthermore, ontologies are not inherently designed to validate specific subsets of focus nodes extracted from a KG in the same granular way ShEx allows. Second, ShEx benefits from coexisting textual (ShExC) and JSON (ShExJ) syntaxes, with readily available tools like shex.js[10] and PyShEx[11] for conversion between them. This ShExC-ShExJ interoperability is advantageous in the context of LLMs, which can effectively process and generate JSON-structured data. This makes ShEx a promising candidate for LLM-driven structured data generation and knowledge validation tasks, broadening its adoption and impact.

In our datasets, the ground truth ShEx schemas utilize a simplified yet functional subset of the ShEx language. This approach prioritizes core validation requirements while ensuring the schemas remain human-understandable. An illustrative example, the schema for ‘Museum (Q33506)’ from the WES dataset, is shown in Figure 2.

Node Constraints A key feature adopted in our dataset, and emphasized in the Wikidata EntitySchema initiative, is the use of shape references to create modular and interconnected schemas, often facilitated by `IMPORT` declarations in more complex scenarios. As shown in the example, the constraint on the predicate ‘country (P17)’ specifies that its object values must belong to the class representing countries. This is achieved using a shape reference, `@<Country>`, which points to the definition of the ‘Country’ shape (Figure 2a, lines 17-20), ensuring that objects of the ‘country (P17)’ predicate are instances of ‘country (Q6256)’.

Cardinality Cardinalities in ShExC are represented by the strings ‘?’, ‘+’, ‘$*$’ (following notation similar to the XML specification) and ranges such as $\{m,\}$ to indicate that at least m elements are required. In ShExJ, they are represented by ‘min’ and ‘max’ values indicating their lower and upper bounds.

C Experimental Details

LLM-based experiments were conducted by accessing the LLM APIs. The similarity metrics leverage the Zhang-Shasha algorithm, implemented using an open-source package[12].

Listing 5 presents a sample input from the global setting experiments, derived from the few-shot examples, for the class ‘film award (Q4220917)’ and predicate ‘organizer (P664)’.

Table 6 details the performance of the `gpt-4o-mini` model across six different compositions of matching criteria, evaluating its precision, recall, and F1-score on both datasets. A clear trend observable for `gpt-4o-mini` is the consistent improvement in F1-scores as the matching criteria are relaxed. For instance, in the triples setting on the YAGO dataset, the F1-score climbs from 0.591 under ’Exact’ node constraint and ’Exact’ cardinality matching to 0.695 when ’Datatype’ abstraction is allowed for node constraints and cardinality is ’Loosened’. Similarly, on the more challenging Wikidata EntitySchema dataset, the global setting sees its F1-score rise from 0.312 (Exact/Exact) to 0.651 (Datatype/Loosened). This demonstrates that while `gpt-4o-mini` produces a solid number of perfectly accurate constraints, its output contains an even larger proportion of constraints that are valid under more flexible, practical interpretations, particularly when considering datatype abstractions.

Compared to DeepSeek-V3, `gpt-4o-mini` performs better under strict exact-matching conditions. However, as the matching criteria are relaxed, DeepSeek-V3 surpasses `gpt-4o-mini`. This indicates that while `gpt-4o-mini` excels at generating precisely accurate constraints, DeepSeek-V3 is more capable of producing constraints that fulfill the core functional requirements of ShEx, making it more practical for real-world applications.

[10] https://github.com/shexjs/shex.js

[11] https://github.com/hsolbrig/PyShEx

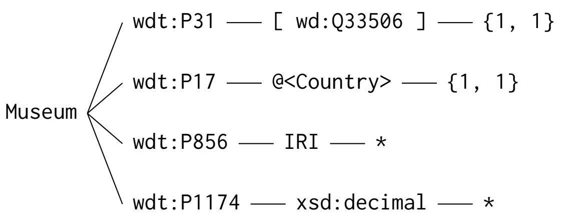
[12] https://pypi.org/project/zss/

```
<Museum> EXTRA wdt:P31 {
  # WikibaseItem property
  # instance of
  wdt:P31   [ wd:Q33506 ] ;
  # country
  wdt:P17   @<Country> ;
  ...

  # URL, String, Quantity, Time property
  # official website
  wdt:P856  IRI * ;
  # visitors per year
  wdt:P1174 xsd:decimal * ;
  ...
}

<Country> EXTRA wdt:P31 {
  # country
  wdt:P31   [ wd:Q6256 ] ;
}
```

(a)


Museum
wdt:P31 —— [wd:Q33506] —— {1, 1}
wdt:P17 —— @<Country> —— {1, 1}
wdt:P856 —— IRI —— *
wdt:P1174 —— xsd:decimal —— *

(b)

Figure 2: Example ShEx schema fragment for 'Museum (Q33506)' from the WES dataset: (a) the ShExC textual representation, where comments above each constraint indicate the label of its predicate, and (b) its corresponding tree structure representation used for similarity metrics.

Settings	**Matching Criteria**		**YAGO Schema**			**Wikidata EntitySchema**		
	Node Constraint	Cardinality	P	R	F1	P	R	F1
local	Exact	Exact	0.575	0.550	0.559	0.370	0.175	0.222
	Subclass	Exact	0.598	0.572	0.581	0.407	0.190	0.242
	Datatype	Exact	0.640	0.611	0.621	0.613	0.290	0.368
	Exact	Loosened	0.614	0.590	0.597	0.408	0.198	0.249
	Subclass	Loosened	0.640	0.616	0.623	0.456	0.217	0.275
	Datatype	Loosened	0.685	0.658	0.666	0.698	0.338	0.427
global	Exact	Exact	0.421	0.362	0.388	0.306	0.328	0.312
	Subclass	Exact	0.421	0.362	0.388	0.326	0.349	0.333
	Datatype	Exact	0.681	0.587	0.628	0.570	0.618	0.587
	Exact	Loosened	0.428	0.367	0.394	0.326	0.350	0.334
	Subclass	Loosened	0.428	0.367	0.394	0.346	0.372	0.354
	Datatype	Loosened	0.739	0.636	0.681	0.635	**0.685**	**0.651**
triples	Exact	Exact	0.631	0.564	0.591	0.335	0.200	0.242
	Subclass	Exact	0.650	0.581	0.608	0.391	0.232	0.281
	Datatype	Exact	0.698	0.621	0.652	0.643	0.395	0.472
	Exact	Loosened	0.678	0.607	0.634	0.377	0.231	0.276
	Subclass	Loosened	0.697	0.625	0.652	0.440	0.267	0.320
	Datatype	Loosened	**0.744**	**0.664**	**0.695**	**0.754**	0.477	0.563

Table 6: Results of `gpt-4o-mini` under different matching criteria.

Matching Criteria		**YAGO Schema**			**Wikidata EntitySchema**		
Node Constraint	Cardinality	P	R	F1	P	R	F1
Exact	Exact	0.111	0.081	0.092	0.106	0.099	0.096
Subclass	Exact	0.111	0.081	0.092	0.106	0.099	0.096
Datatype	Exact	0.536	0.393	0.445	0.211	0.189	0.188
Exact	Loosened	0.111	0.081	0.092	0.149	0.129	0.129
Subclass	Loosened	0.111	0.081	0.092	0.151	0.131	0.131
Datatype	Loosened	**0.630**	**0.465**	**0.525**	**0.356**	**0.297**	**0.304**

Table 7: Results of sheXer under different matching criteria.

```
{
  "class_uri": "http://www.wikidata.org/entity/Q4220917",
  "class_label": "film award",
  "class_description": "recognition for cinematic achievements",
  "predicate_uri": "http://www.wikidata.org/prop/direct/P664",
  "predicate_label": "organizer",
  "predicate_description": "person or institution organizing an event",
  "triple_examples": [
    "{'s': 'wd:Q3910523', 'p': 'wdt:P664 (organizer)', 'o': ['wd:Q2288813 (Italian
       National Syndicate of Film Journalists)']}",
    "{'s': 'wd:Q11624249 (Fujimoto Prize)', 'p': 'wdt:P664 (organizer)', 'o': ['wd:
       Q114256803']}",
    "{'s': 'wd:Q18640780 (Florida Film Critics Circle Awards)', 'p': 'wdt:P664 (
       organizer)', 'o': ['wd:Q3074282 (Florida Film Critics Circle)']}",
    "{'s': 'wd:Q106867277 (NBR Freedom of Expression)', 'p': 'wdt:P664 (organizer)',
        'o': ['wd:Q1133614 (National Board of Review of Motion Pictures)']}",
    "{'s': 'wd:Q109259295 (Gotham Independent Film Award for Breakthrough Nonfiction
        Series)', 'p': 'wdt:P664 (organizer)', 'o': ['wd:Q892112 (Independent
       Feature Project)']}"
  ],
  "frequency": "1.73% instance(s) in the class use the predicate",
  "cardinality_distribution": "1.73% instances in the class have 1 object(s) when
     using the predicate",
  "datatype_of_objects": "IRI",
  "object_class_distribution": "13.33% of subjects have objects in class wd:Q43229 (
     organization), 6.67% of subjects have objects in class wd:Q101007233 (film
     critics association), 6.67% of subjects have objects in class wd:Q10689397 (
     television production company)",
  "subject_type_constraint": "Based on the subject type constraint of Wikidata, the
     item described by such predicates should be a subclass or instance of ['wd:
     Q170584 (project)', 'wd:Q288514 (fair)', 'wd:Q464980 (exhibition)', 'wd:
     Q1190554 (occurrence)', 'wd:Q14136353 (fictional occurrence)', 'wd:Q15275719 (
     recurring event)', 'wd:Q15900616 (event sequence)', 'wd:Q107736918 (series of
     concerts)'].",
  "value_type_constraint": "Based on the value type constraint of Wikidata, the
     value item should be a subclass or instance of ['wd:Q5 (human)', 'wd:Q43229 (
     organization)', 'wd:Q49773 (social movement)', 'wd:Q4164871 (position)', 'wd:
     Q14623646 (fictional organization)', 'wd:Q15275719 (recurring event)', 'wd:
     Q16334295 (group of humans)', 'wd:Q30017383 (fictional organism)']."
}
```

Listing 5: An example of global information input.