From Instructions to ODRL Usage Policies: An Ontology Guided Approach

Daham M. Mustafa
Fraunhofer FIT
Sankt Augustin, Germany
daham.mohammed.mustafa@fit.fraunhofer.de

Abhishek Nadgeri
Fraunhofer FIT
Sankt Augustin, Germany

Diego Collarana
Fraunhofer FIT
Sankt Augustin, Germany
Universidad Privada Boliviana
Cochabamba, Bolivia

Benedikt T. Arnold
Fraunhofer FIT
Sankt Augustin, Germany
RWTH Aachen University
Aachen, Germany

Christoph Quix
RWTH Aachen University
Aachen, Germany

Christoph Lange
Fraunhofer FIT
Sankt Augustin, Germany
RWTH Aachen University
Aachen, Germany

Stefan Decker
Fraunhofer FIT
Sankt Augustin, Germany
RWTH Aachen University
Aachen, Germany

ABSTRACT

This study presents an approach that uses large language models such as GPT-4 to generate usage policies in the W3C Open Digital Rights Language *ODRL* automatically from natural language instructions. Our approach uses the *ODRL* ontology and its documentation as a central part of the prompt. Our research hypothesis is that a curated version of existing ontology documentation will better guide policy generation. We present various heuristics for adapting the *ODRL* ontology and its documentation to guide an end-to-end *KG* construction process. We evaluate our approach in the context of dataspaces, i.e., distributed infrastructures for trustworthy data exchange between multiple participating organizations for the cultural domain. We created a benchmark consisting of 12 use cases of varying complexity. Our evaluation shows excellent results with up to 91.95% accuracy in the resulting knowledge graph.

VLDB Workshop Reference Format:
Daham M. Mustafa, Abhishek Nadgeri, Diego Collarana, Benedikt T. Arnold, Christoph Quix, Christoph Lange, and Stefan Decker. From Instructions to ODRL Usage Policies: An Ontology Guided Approach. VLDB 2024 Workshop: LLM+KG.

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Daham-Mustaf/LLM4ODRL

Proceedings of the VLDB Endowment. ISSN 2150-8097.

1 INTRODUCTION

The data economy increasingly relies on dataspaces, which are distributed infrastructures for data exchange among multiple participating organizations based on data sovereignty and interoperability principles. To achieve interoperability in dataspaces, initiatives such as the International Data Spaces Association (IDSA) [3] and the Gaia-X European Association for Data and Cloud strongly rely on Semantic Web technologies. Semantic Web technologies are not easily accessible to non-technical users – this problem also holds in dataspaces. The specifications of both IDSA and Gaia-X rely on the W3C Open Digital Rights Language (*ODRL*) to describe the usage policies of assets. Formulating usage policies in *ODRL* requires familiarity with the RDF graph data model, its serializations, and the *ODRL* concepts. This requirement has become a significant entry barrier with the adoption of dataspaces in highly digitalized sectors such as car manufacturing, transportation, and the cultural sector. For example, to create the *ODRL* policy from the instruction, "Painting *X* from Museum *M* is authorized for display within Germany but prohibited within the USA", a domain expert is required to know all the terminology from *ODRL*, including *Asset*, *Permission*, *Constraint*, etc.

The *ODRL* version 2.2, a W3C recommendation since February 2018 [15], is an ontology for expressing rights over physical or digital goods. *ODRL* enables owners and consumers to effectively express the terms and obligations governing digital asset access and use. We show a high-level overview of the *ODRL* information model in Figure 1. The main class of the Core Vocabulary is *Policy*, which acts as a container for *Rules*. A *Set* serves as the default subclass of *Policy* for conveying a *Rule* over an *Asset* to define general terms of usage without specific *Constraint* or Duty. An *Offer*, as a subclass of *Policy* in the Core Model, describes the rules presented by the assigning parties and specifies the conditions for the recipient party of these rules. An *Agreement* is a subclass of

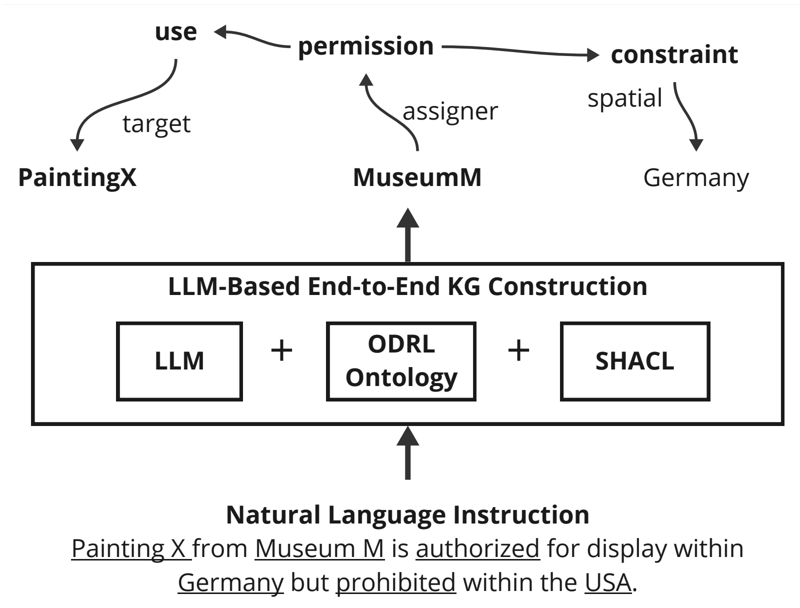

Figure 1: Our method takes a natural language description of a usage policy as input. Using an LLM, a curated description of the *ODRL* ontology and associated SHACL shapes, we generate a *KG* that corresponds to the *ODRL* representation of the described policy.

Policy in the core model, which encompasses all terms governing the usage agreement between an Assigner (the party that proposes the policy statements) and an *Assignee* (the party that receives the policy statements) regarding an *Asset*. A *Permission* specifies *Actions* over an *Asset*. Permissions can also be linked with a duty, especially when the action is obligatory. In contrast to permissions, *Prohibitions* restrict specific actions.

To alleviate the barrier of familiarity with *ODRL* and Semantic Web technologies for creating usage policies in dataspaces, we contribute with an end-to-end approach that generates usage policies in *ODRL* directly from natural language instructions given by an application domain expert. Our approach employs Large Language Models (LLMs) combined with a hand-crafted ontology description based on the ODRL specification. We equip our approach with a self-correction component that evaluates the generated output and applies rules. We evaluate our approach with GPT-3.5-turbo, GPT-4, and GPT-4o on twelve realistic use cases from the cultural domain, with increasing complexity, about the described usage permissions. We rely on SHACL (Shapes Constraint Language [10]) shapes to perform the evaluation from the syntax and semantic perspective. Our results show a promising path in ontology-guided KG generation.

2 RELATED WORK

Formally a knowledge graph *KG* is defined as a tuple given by $KG = (\mathcal{E}, \mathcal{R}, \mathcal{T}^+)$ where $\mathcal{E}$ denotes the set of all vertices in the graph representing entities, $\mathcal{R}$ is the set of edges representing relations, and $\mathcal{T}^+ \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of all *KG* triples [13]. The task of *KG* construction can be defined as, given unstructured sources $S = (s_1, s_2, \ldots, s_n)$, a model M being trained to approximate the function $F(S) \rightarrow KG$ [17]. Traditionally, *KG* construction comprises several tasks, namely Name Entity Recognition (NER) [9], Relation Extraction (RE) [16], Entity Linking (EL) [14] and Coreference Resolution (CR) [12]. Early, the construction of knowledge graphs relied on a pipeline architecture and tools to effectively create and incrementally update a knowledge graph [8]. With the advent of deep learning approaches, each of these tools was further improved [18]. However, in recent years, Large Language Models (LLMs) have caused a paradigm shift in Natural Language Processing (NLP) by creating generalist agents [2]. The remarkable properties of LLMs now enable end-to-end KG construction. Early attempts included BERT style models [4] to extract entities and associated relations [11], whereas Guo et al. [6] propose end-to-end knowledge graph construction with BERT. The work by Han et al. [7] aligns with our work, using a larger LLM (GPT-4) for KG construction. However, to correct errors in the previously generated knowledge graph, they use a small fine-tuning LLM (*T5*). While more efficient during inference, this also poses the method's main drawback due to the need for labeled data that might be unavailable.

3 METHOD

The KG construction process begins by developing an LLM Guidance Template (*LGT*)[1], a framework for guiding the LLM-based KG construction. We provide a Task Description (*TD*) in natural language with the specific requirements and constraints for the desired ODRL. The LLM then uses the *LGT* and *TD* to construct an *ODRL KG* end-to-end. After generating the *ODRL* knowledge graph, our approach incorporates a Self-Correction Model (*SCM*)

[1]The LLM guidance templates used in this study can be found in the templates directory on GitHub.

to refine the generated KG. The *SCM* utilizes predefined rules to correct inconsistencies and errors within the *KG*, resulting in a refined *ODRL* knowledge graph. Figure 2 illustrates the complete workflow of our approach, where we explain the process of generating the *ODRL* and subsequently refining it using the *SCM*. Now, we describe each module in detail.

3.1 Ontology-Guided KG Construction

From the perspective of computer science, An ontology is a formal, explicit specification of a shared conceptualization [5]. The *LGT* in this approach is based on the *ODRL* version 2.2 ontology[2] This ontology defines *ODRL* classes, core concepts, and includes various OWL axioms (e.g., domain, range, and others) that represent relationships between classes and properties. The ontology uses SKOS[3] annotations (`skos:definition`, `skos:note`, `rdfs:label`) for metadata, conceptual definitions, and labels. We pass this ontology directly to the LLM, leveraging both its formal structure and human-readable annotations. The LLM then interprets this information to generate an *ODRL* KG.

Relying solely on the *ODRL* ontology challenges LLMs in accurately predicting complex *KG*s, especially with multiple constraints (e.g., date, time, location, special conditions). LLMs may misinterpret ontology elements, leading to errors such as:

The LLM-generated ODRL policies often exhibit structural misunderstandings of the ODRL ontology. A critical issue is the misuse of class instances within the `odrl:Constraint` structure. For example, the LLM might incorrectly use `odrl:use` (an `odrl:Action` instance) as the `odrl:leftOperand` of a Constraint, where an `odrl:LeftOperand` instance is required. This error goes beyond simple property value mismatches. It represents a fundamental misinterpretation of ODRL class relationships and the specific structure of the `odrl:Constraint` class. Such mistakes can lead to logically inconsistent and structurally invalid ODRL policies.

Another significant issue is the introduction of undefined properties. LLMs, in their attempt to generate coherent policies, sometimes employ synonyms or related concepts that are not officially defined in the ODRL ontology. A prime example of this is the use of `odrl:location` in place of the standard `odrl:spatial` property. While `odrl:location` might seem intuitively correct and semantically appropriate, it is not a part of the official ODRL vocabulary. This seemingly minor deviation can lead to invalid policy expressions, as ODRL processors and enforcement mechanisms are designed to work with a specific set of predefined terms. Such undefined properties, despite their apparent semantic relevance, can render the entire policy uninterpretable within the ODRL framework, highlighting the need for strict adherence to the standardized ODRL vocabulary in policy generation.

These errors lead to inconsistencies and misalignments with the *ODRL* standard. To address these issues, we enhance the *LGT* with additional context about *ODRL* syntax and semantics. This enrichment aids LLMs in better distinguishing between various data types and object properties within the ontology, resulting in more accurate ODRL policy generation.

These errors cause inconsistencies and misalignments with the *ODRL* standard. To mitigate these issues, we enrich the *LGT* with additional context about *ODRL* syntax and semantics, aiding LLMs in distinguishing between various data types and object properties within the ontology.

3.2 Ontology, Syntax, Semantics, and Examples (*OSES*) Insights

We translate the structured *ODRL* ontology into plain text, incorporating syntactic and semantic aspects to guide the LLM. However, directly importing the text of the *ODRL* W3C recommendation[4] into the *LGT* file presents a significant challenge. The main issue lies in the clarity guidance. The recommendations often contain lengthy explanations and duplicated information, leading to confusion and hallucinations. In response to these challenges, we undertake a process of distillation [1], extracting essential information from the *ODRL* ontology and recommendations. As part of this distillation process, we have created guidelines inside the LGT file for the main classes of the *ODRL* information model and their properties, detailing how they should be utilized by the LLM. This distilled information is then organized into three key dimensions within the LGT file, considering three main aspects: Firstly, we address *ODRL*'s syntactic interpretation, the syntactic structure defines how its main entities are aligned and how they should be used. Secondly, we consider *ODRL* Semantics: *ODRL* `odrl:Duty` instances can represent obligations when linked to `odrl:Policy` via `odrl:obligation` or conditions for permission activation when linked to `odrl:Permission` via `odrl:duty`.

Finally, integrated *ODRL* examples in the LGT provide crucial practical guidance for policy implementation, significantly aiding LLMs in understanding and selecting appropriate relations for the output. These examples serve several key functions in enhancing the LLM's ability to generate accurate and compliant ODRL policies: These carefully curated examples in the LGT equip LLMs with a rich set of patterns and templates, enabling them to recognize and replicate valid ODRL structures. This significantly improves their ability to generate policies that are not only syntactically correct but also semantically meaningful and aligned with ODRL standards. The examples serve as a practical reference, bridging the gap between abstract ODRL concepts and their concrete implementation in policy formulation.

LGT guides the LLM to generate the *ODRL KG* by solving tasks. Separate LGTs are established for each *ODRL* type (*Agreement*, *Rule*, *Offer*), providing guidance for accurate results. Despite careful design, there may be inaccuracies or discrepancies, including syntactic errors and semantic inconsistencies, in the output generated by LLM. Therefore, we apply a third self-correction model.

3.3 *ODRL* Self-Correction Model

For ODRL self-correction, we establish 17[5] rules for *Agreement*, 16 for *Offer*, and *ODRL Rule*. These rules, termed correction rules, are derived from the ODRL W3C recommendation and ontology relations, ensuring alignment with the official ODRL specification and leveraging the semantic structure defined in the ontology. *ODRL*

[2]https://www.w3.org/ns/odrl/2/ODRL22.ttl
[3]https://www.w3.org/2004/02/skos/

[4]https://www.w3.org/TR/odrl-model/
[5]ODRL self-correction rules

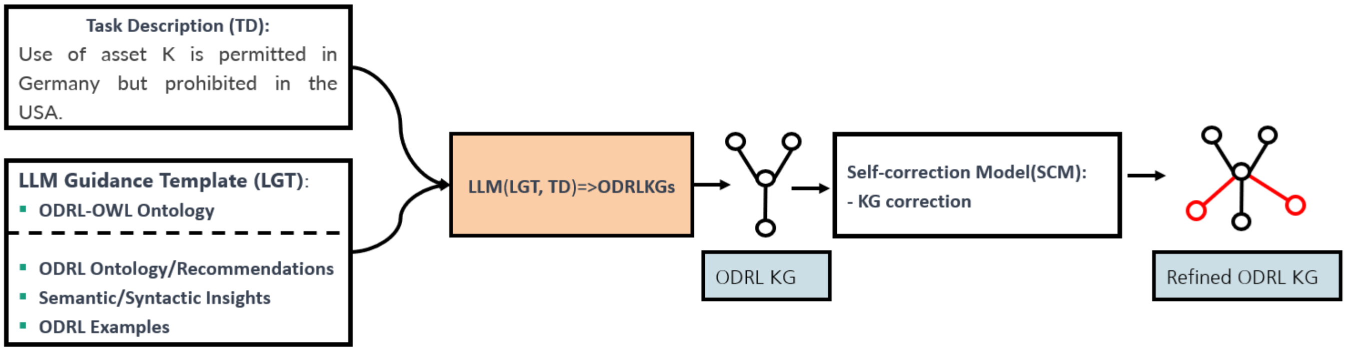

Figure 2: Approach. The LLM Guidance Template (*LGT*) comprises two modules, i.e., the *ODRL* ontology in Turtle serialization and contextual insights in PDF format. *KG* is constructed by passing Task Description (*TD*) and the *LGT* as input to the LLM. Next, the Self-correction Model (*SCM*) refines the generated *KG* by providing the TD, the first version of the *KG*, and the *ODRL* Self-Correction Rules (*SCR*) as input for the LLM to produce the refined *KG*.

Self-Correction Rules (SCR), rules are expressed in human-readable language, which allows for accurate interpretation and application by the LLM. The inputs include the *TD* along with it resulting *ODRL KG*, as has been detailed previously in section 3.1, and the SCR. These inputs are then passed to the LLM. We have two options for applying self-correction. The first option is to define each rule separately and iterate over them. The second option is to consolidate all rules into one prompt and then pass it to the LLM. Although this method is less expensive than the first option, it may slightly compromise accuracy due to the large text processing required by the LLM. The SHACL violation messages alone do not effectively guide the LLM to produce KG correction. This is due to the stateless nature of LLMs, which do not retain memory of previous interactions or learn from individual correction attempts. As such, simply passing SHACL violation messages back to the LLM does not guarantee improved KG. The LLM compares the *ODRL KG* policy with the *SCR*. If any inconsistencies are detected, the LLM makes adjustments to ensure compliance with the *SCR*. This results in a refined *ODRL KG*, as illustrated in Figure 2. Unlike SHACL shapes, which primarily detect violations without modifying the *KG*, the LLM can autonomously refine the KG based on detected rule violations.

4 EVALUATION

Evaluating outputs from LLMs is crucial due to their probabilistic nature. Unlike deterministic systems, LLMs may generate varying outputs for the same task, as LLMs may still generate probabilistic outputs even when provided with factual data. We conducted a comprehensive set of 108 experiments to evaluate the performance of three different LLM models: GPT-3.5-turbo, GPT-4, and GPT-4o. For each use case, we employed three different methodologies: Ontology-Guided, OSES Insights, and Refinement, resulting in 36 experiments per model. The experiments aimed to assess the models' capabilities in managing and enforcing digital rights.

Dataset: Inspired by the context of the project Datenraum Kultur(DRK)[7], We designed a dataset of twelve different use cases.[8] These use cases represent tasks defined in plain text, designed to prompt LLMs to generate ODRL KGs. These use cases are derived from real scenarios in the context of the DRK and encapsulate different criteria for determining the appropriate application of policies to assets where digital rights need to be managed and enforced in the cultural domain. Each use case serves as a task description for the LLM. The dataset includes 4 *Agreement*, 5 *Offer* and 3 *Set* of type *ODRL Policy*. Case 1, illustrated below, demonstrates a typical scenario for ODRL policy generation. This example policy regulates access to the ShowTimesAPI' dataAPI managed by DE_Staatstheater_Augsburg, a German cultural organization. The policy's main objective is to control access to valuable cultural assets stored in the dataAPI. Access is restricted to subscribers such as Regional Newspaper', Culture Research Institute', and Cultural Platform Bavaria'. Furthermore, access is limited to users located in Germany, and usage rights expire on May 10, 2025.

Criteria Definition and Constraint Establishment: Criteria C_1–C_9 Table 1 are defined for each use case based on the W3C *ODRL* recommendation, focusing on both semantic and syntactic representations. Each criterion sets expectations for an *ODRL* representation, ensuring that the selected use cases comply with both the structural and content constraints necessary for effective digital rights management. The next step is mapping to *SHACL* constraints for validation.

SHACL Shape Creation: Each criterion is translated into a *SHACL* shape with associated constraints generated for this study[9]. This constraint encompasses five shapes: `PolicyShape`, `PermissionShape`, `PartyShape`, `AssetShape`, and `ConstraintShape` collectively evaluate 22 ranges of properties for `Agreement`, 20 for `Offer`, and 16 for `Asset` policy types. SHACL shapes were manually crafted to reflect the specific requirements of ODRL policies. These shapes define constraints that valid ODRL policies must satisfy, including structural integrity (e.g., presence of required elements), proper use of ODRL vocabulary, correct relationships between policy components, and adherence to data type specifications. The manual creation of these shapes allowed for precise control over the validation criteria, ensuring that all nuances of ODRL policy structure

[7]https://datenraum-kultur.fit.fraunhofer.de/
[8]The use cases can be found in the YAML file on GitHub.
[9]SHACL shapes for ODRL KGs evaluation

Table 1: ***ODRL*** **Criteria and Violations**

Name	Assertion	SHACL Shape
C_1: odrl:uid	Instances of classes `Policy`, `Asset`, `Party`, and `Constraint` **MUST** be associated with an (`odrl:uid`). This ensures unique identification of ODRL entities.	PolicyShape
C_2: Data type specification	**MUST** explicitly specify the corresponding XML Schema Definition (XSD) data type. This ensures proper data validation and interpretation.	All Shapes
C_3: Meta info	**MUST** include mandatory meta-information: `dc:creator`, `dc:title`, `dc:description`, and `dc:issued`. These provide essential context for the policy.	PolicyShape
C_4: Function	`Offer` **MUST** have one `odrl:assigner` of type `Party`. `Agreement` **MUST** have one `odrl:assigner` and `odrl:assignee` of type `Party`. This defines the parties involved in the policy.	PartyShape
C_5: Relation	Policy **MUST** have one `odrl:target` property with an object of type `Asset`. This specifies the asset to which the policy applies.	AssetShape
C_6: Action	Policy **MUST** have one `odrl:action` property of type `Action`. This defines the permitted, prohibited, or obligated action on the asset.	PermissionShape
C_7: Rule	Policy **MUST** contain at least one rule specifying actions on `Assets`, which can be `Permission`, `Prohibition`, or `Obligation`. Rules are the core of ODRL policies.	PolicyShape
C_8: Constraint	`Rule` Constraint **MUST** have properties `leftOperand` (`LeftOperand`), `operator` of type (`Operator`), and `rightOperand` (Literal, IRI, or `RightOperand`). These define the conditions under which a rule applies.	ConstraintShape
C_9: *ODRL* Extension	New elements introduced by LLM **MUST** either be explicitly defined in *ODRL* or adhere to the *ODRL* Profile Mechanism.[6] This ensures compatibility with the ODRL standard.	All shapes

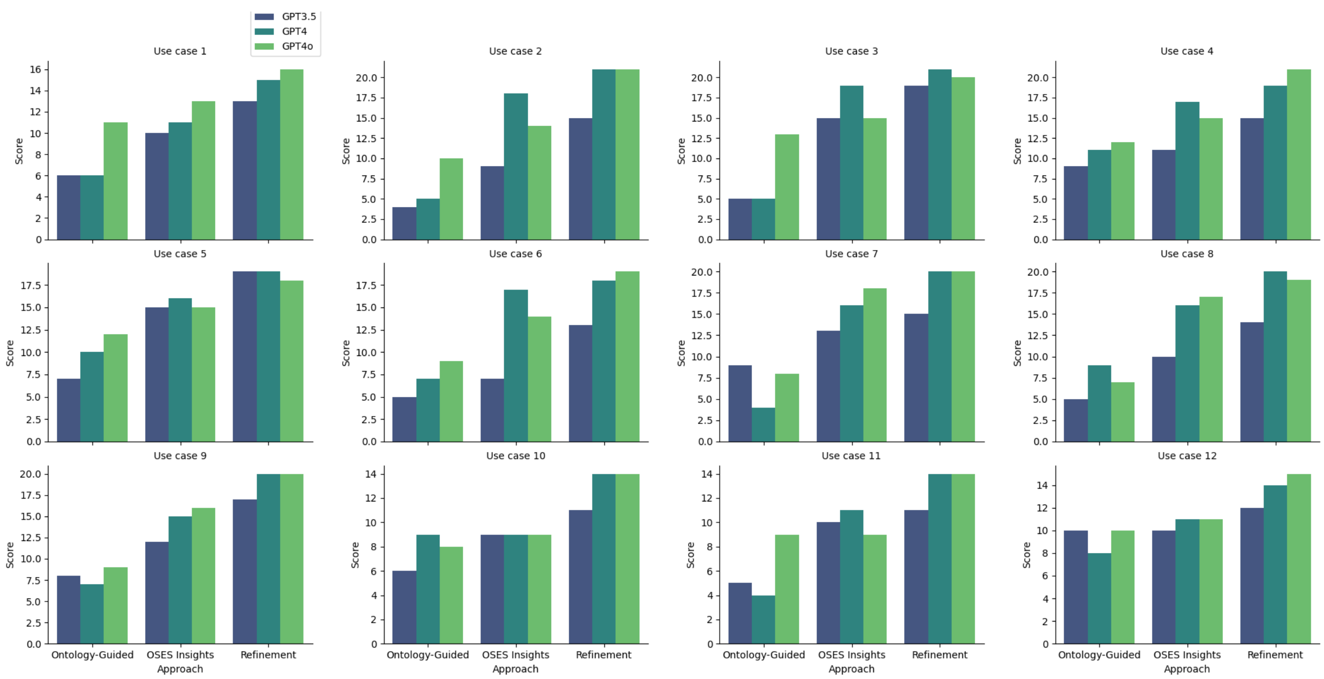

Figure 3: Figure 3 illustrates performance across 12 use cases for GPT-3.5-turbo, GPT-4, and GPT-4o. LLMs learn progressively from ontology input, with improved context and examples enhancing performance. The Refinement method, combining ontology guidance and self-correction, consistently achieves the highest scores across all models and use cases, demonstrating its effectiveness in generating accurate ODRL policies.

were captured. This hand-crafted approach enabled us to tailor the shapes to the specific requirements of our use cases and the ODRL standard.

Assignment of Scores:

The evaluation of our generated ODRL policies involves a comprehensive scoring system based on SHACL validation. Our scoring mechanism is binary: each Focus Node and property shape within the data graph is evaluated against the corresponding SHACL shape, receiving a score of 1 if it fully satisfies the SHACL shape constraints, and 0 if it fails to meet one constraints. This granular approach allows for a detailed assessment of each component of the ODRL policy. **Evaluation Tools and Process:**

For the evaluation process, we primarily utilized the *SHACL Playground*[10], an online tool that provided a user-friendly interface for testing and visualization of SHACL validation results. This tool allowed us to input our manually crafted SHACL shapes and the generated ODRL policies, providing immediate feedback on compliance with the defined constraints. The SHACL Playground facilitated a thorough examination of each policy, highlighting any violations of the SHACL shapes and allowing us to assess the structural and semantic correctness of the generated ODRL policies across different use cases and LLM models.

Aggregation and Analysis:

After evaluating individual use cases, we aggregated the results to provide a comprehensive overview. Scores were totaled for each approach (e.g., basic generation, ontology-guided, refinement), performance was compared across different LLM models (GPT-3.5-turbo, GPT-4, GPT-4o), and trends and patterns in policy generation accuracy were identified. The aggregated scores serve as key metrics for assessing the quality of generated ODRL policies. Higher scores indicate greater accuracy and better adherence to ODRL standards, reflecting the precision of policy representation for each use case. Comparative analysis of scores helps identify the most effective approaches and models for ODRL policy generation.

Result: In the Ontology-Guided approach, the LLM faced challenges in extracting information solely from *ODRL* ontology. The large size compounded these challenges. Without the assistance of real *ODRL* examples, the LLM struggled to accurately predict the *ODRL* Figure 3. However, in the *OSES* Insights approach, the *ODRL KG* is enhanced with properties and relations. By formulating the ontology in text and providing semantic, syntactic, and real examples, the guidance provided to the LLM is improved. Refined ODRL KG quality improves through self-correction rules[11] targeting crucial omitted properties. This iterative process enhances KG completeness and accuracy, with effectiveness highly dependent on the initial KG generation quality. Initially, we ran each use case once and scored the output as the first exploration. the results indicate varying degrees of performance across the LLM variants and methodologies, with GPT-4 and GPT-4o exhibiting similar performance patterns across the use cases.

This Accuracy formula as shown in Equation 1 allows us to quantify the performance of our LLM-based approach in generating ODRL policies. Each correctly generated element of the policy contributes to the total obtained score (R), while the total possible score (T) represents the number of elements required for a fully compliant ODRL policy. By applying this formula to our use cases, we can effectively measure and compare the accuracy of different approaches and models in ODRL policy generation. The formula is defined as follows:

$$\text{Accuracy} = \frac{\sum R}{\sum T} \times 100\% \tag{1}$$

where:

- R: Total obtained score
- T: Total possible score

Example calculation for Use Case 1 (with refinement):

$$\frac{217}{236} \times 100\% \approx 91.95\%$$

5 CONCLUSIONS

This work proposes a novel methodology for generating KGs from textual data using an LLM. We demonstrate that interpreting the ontology in plain text significantly boosts LLM accuracy. The LLM can compare KGs with textual predefined rules presented in human-readable text and refine the results. This capability to refine KGs underscores the advantages of employing LLM in KG construction. One limitation of our current work is that it is evaluated on using the *ODRL* ontology. However, our approach is generic and can be extended to diverse ontologies. In our current methodology, *LGT* guidelines for KG construction (section 3.1) and selfcorrection rules (section 3.3) have been derived and formulated from the W3C Recommendation and *ODRL* ontology, which involves manual interpretation and analysis. A potential area for future work could be to automate this task, e.g., by identifying and extracting the most important parts of the ontology using a statistical analysis of existing knowledge graphs that are built upon the ontology. An LLM could be used to verbalize the remaining parts.

ACKNOWLEDGMENTS

This work was supported by the German Ministry for Research and Education (BMBF) project WestAI (Grant no. 01IS22094D) and the German Federal Commissioner for Culture and the Media via Datenraum Kultur (Grant no. 2522DIG012).

REFERENCES

[1] Yuvanesh Anand, Zach Nussbaum, Brandon Duderstadt, Benjamin Schmidt, and Andriy Mulyar. 2023. Gpt4all: Training an assistant-style chatbot with large scale data distillation from gpt-3.5-turbo. *GitHub* (2023).

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[3] Tobias Dam, Andreas Krimbacher, and Sebastian Neumaier. 2023. Policy Patterns for Usage Control in Data Spaces. *arXiv preprint arXiv:2309.11289* (2023).

[4] Jacob Devlin, Ming-Wei Chang, and Kenton Lee. 2019. Google, KT, Language, AI: BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[5] Thomas R Gruber. 1993. A translation approach to portable ontology specifications. *Knowledge acquisition* 5, 2 (1993), 199–220.

[6] Qingyan Guo, Yang Sun, Guanzhong Liu, Zijun Wang, Zijing Ji, Yuxin Shen, and Xin Wang. 2021. Constructing Chinese historical literature knowledge graph based on BERT. In *Web Information Systems and Applications: 18th International Conference, WISA 2021, Kaifeng, China, September 24–26, 2021, Proceedings 18*. Springer, 323–334.

[10] SHACL Playground: https://shacl.org/playground/

[11] https://github.com/Daham-Mustaf/LLM4ODRL/blob/main/correction_report.py

[7] Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. 2023. PiVe: Prompting with Iterative Verification Improving Graph-based Generative Capability of LLMs. *arXiv preprint arXiv:2305.12392* (2023).

[8] Marvin Hofer, Daniel Obraczka, Alieh Saeedi, Hanna Köpcke, and Erhard Rahm. 2023. Construction of knowledge graphs: State and challenges. *arXiv preprint arXiv:2302.11509* (2023).

[9] Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. 2024. A survey on recent advances in named entity recognition. *CoRR* abs/2401.10825 (2024). https://doi.org/10.48550/ARXIV.2401.10825 arXiv:2401.10825

[10] H. Knublauch and D. Kontokostas. 2017. *Shapes Constraint Language (SHACL)*. Recommendation REC-shacl-20170720. W3C. https://www.w3.org/TR/2017/REC-shacl-20170720/

[11] Abhijeet Kumar, Abhishek Pandey, Rohit Gadia, and Mridul Mishra. 2020. Building knowledge graph using pre-trained language model for learning entity-aware relationships. In *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*. IEEE, 310–315.

[12] Ruicheng Liu, Rui Mao, Anh Tuan Luu, and Erik Cambria. 2023. A brief survey on recent advances in coreference resolution. *Artificial Intelligence Review* (2023), 1–43.

[13] Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6418–6425.

[14] Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. 2022. Neural entity linking: A survey of models based on deep learning. *Semantic Web* 13, 3 (2022), 527–570.

[15] S. Villata and R. Iannella. 2018. *ODRL Information Model 2.2*. Recommendation REC-odrl-model-20180215. W3C. https://www.w3.org/TR/2018/REC-odrl-model-20180215/

[16] Zhao Xiaoyan, Deng Yang, Yang Min, Wang Lingzhi, Zhang Rui, Cheng Hong, Lam Wai, Shen Ying, and Xu Ruifeng. 2023. A Comprehensive Survey on Deep Learning for Relation Extraction: Recent Advances and New Frontiers. *arXiv preprint arXiv:2306.02051* (2023).

[17] Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. 2022. Generative Knowledge Graph Construction: A Review. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates. https://doi.org/10.18653/v1/2022.emnlp-main.1

[18] Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities. *arXiv preprint arXiv:2305.13168* (2023).