Validating Semantic Artifacts With Large Language Models

Nilay Tufek[1][0000−0002−0446−1299], Aparna Saissre Thuluva[1][0000−0002−7564−4279], Valentin Philipp Just[2][0009−0005−9449−1027], Fajar J. Ekaputra[3,4][0000−0003−4569−2496], Tathagata Bandyopadhyay[1][0000−0001−5271−883X], Marta Sabou[3][0000−0002−7149−5843], and Allan Hanbury[4][0000−0002−7149−5843]

[1] Technology Department, Siemens AG, Munich Germany
[2] Inst. of Computer Engineering, TU Wien, Vienna, Austria
[3] Inst. Data, Process and Knowledge Management, WU Vienna, Vienna, Austria
[4] Inst. of Informatics, TU Wien, Vienna, Austria

Abstract. As part of knowledge engineering workflows, semantic artifacts, such as ontologies, knowledge graphs or semantic descriptions based on industrial standards, are often validated in terms of their compliance with requirements expressed in natural language (e.g., ontology competency questions, standard specifications). Key to this process is the translation of the requirements in machine-actionable queries (e.g., SPARQL) that can automate the validation process. This manual translation process is time-consuming, error-prone and challenging, especially in areas where domain experts might lack knowledge of semantic technologies. In this paper, we propose a Large Language Models (LLMs) based approach to translate requirements texts into SPARQL queries and test it in validation use cases related to SAREF and OPC UA Robotics. F1 scores of 88-100% indicate the feasibility of the approach and its potential impact on ensuring high quality semantic artifacts and further uptake of the semantic technologies (industrial) domains.

Keywords: Semantic Artifacts, Validation, LLM, OPC UA

1 Introduction

Knowledge Engineering (KE) aims to elicit, capture, conceptualise and formalise knowledge in the form of *semantic artifacts* that support information systems, in particular in the area of Artificial Intelligence. A semantic artifact is a "machine-actionable formalisation of a conceptualisation, enabling sharing and reuse by humans and machines" [1]. Examples of such artifacts range from taxonomies and thesauri, to ontologies and knowledge graphs.

Ensuring the quality of a semantic artifact is often achieved by validating that it complies with certain requirements. For example, in ontology engineering, ontologies are validated against competency questions or requirements elicited

when building (or reusing already existing) ontologies [2]. We represent this process graphically in Fig. 1a: given a set of competency questions or requirements expressed in natural language, a semantic artifact is built (or selected for reuse). The validation of the semantic resource happens by translating the requirements from natural language to a machine-actionable language such as SPARQL (or SHACL) and applying these to the artifacts. Validation fails when the queries return no answer, as the information expressed by the requirements could not be found in the semantic artifact.

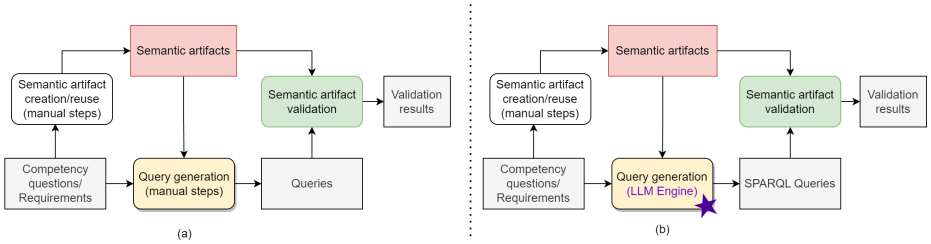

Fig. 1. Validation of semantic artifacts: (a) traditional approach; (b) with LLMs.

Interestingly, this pattern of validating semantic artifacts through the formalisation of requirements can also be found more broadly in communities that adopt semantic technologies [3]. This is the case with industrial standards such as ISA, IEC, OPC UA, etc., which provide standardized semantics for the industry automation field [4]. These standards provide the specification of their semantics in textual documents (e.g., in PDF format). Their adopters create information artifacts (e.g., descriptions of devices e.g., in XML format) in compliance with the standard's rules to ensure interoperability and integration in their domain. To benefit from formal and machine-actionable semantics in terms of automated query and reasoning capabilities, several of these standards are aligned to the Semantic Web (SW) technology stack [5, 6, 7, 8]. As a result, standard adopters create *semantic artifacts* (expressed in RDF), which need to comply with the textual requirements expressed in the standard specification (aka compliance rules). Compliance is checked by validating these industrial semantic artifacts through formal queries derived from the standard specification.

The key *challenge* in all these validation settings is that translating textual requirements into formal queries manually is time consuming, tedious and error-prone [9]. In industrial settings, this is exacerbated by *scale* (e.g., OPC UA alone has about 140 specification documents each containing 30-100 textual compliance rules) and reduced expertise with semantic technologies [10]. LLMs such as GPT-3, have shown a versatile applicability for performing language processing tasks [11]. While their use to address KE tasks is being investigated [12, 13, 14], there is, so far limited focus on semantic artifact validation.

To fill this gap, we investigate how to automatically generate SPARQL queries from textual requirements using LLMs (Fig.1b) as a key enabler for semantic artifact validation. We describe our methodology for devising suitable prompts and experimentally investigate the feasibility of this approach in two

uses cases: (i) the open source Smart Applications REFerence (SAREF) ontology [15] and (ii) an OPC UA Robotics ontology derived from OPC UA Robotics nodeset file[5] using the formal mapping to RDF defined in [7]. The experimental data and the source code are publicly available [16]. Results of F1 score ranging between 88-100% demonstrate the feasibility of the approach.

An overview of related work (Sect. 2), is followed by the description of our methodology (Sect. 3), its adaptation to the two use cases and the experiments conducted (Sect. 4). We close with lessons learned and conclusions (Sect. 5).

2 Related Work

The support of LLMs for KE has attracted much research interest in recent years. Allen et al. [14] highlights two possible uses of LLMs for KE: (i) LLM as a component in a hybrid neuro-symbolic system to support the KE process, and (ii) LLM as a stand-alone approach to KE. Further, they emphasize the capability of LLM as a general-purpose technology to transform natural language into formal language. Meyer et al. [17] elaborate their experiments on potential application areas where LLMs can assist with Knowledge Graph (KG) engineering.

However, much of these recent works are focused on KG construction and completion, marked with the availability of benchmarks (e.g., Text2KGBench [18]) and challenges (e.g., LLM-KBC[6]). In contrast, support for other KE tasks, including the validation of semantic artefacts, is limited or only reported as part of a larger process. The identification of incorrect KG triples has been briefly discussed in [19] as part of a KG generation process. Meyer et al. conducted experiments with SPARQL query generation where their initial results show that the standard prompt engineering is not sufficient [17]. Therefore, there is a need for a methodology to support validation of semantic artefacts, including generating SPARQL queries from textual requirements.

3 Proposed Methodology

LLMs are pre-trained models on billions or trillions of parameters [20]. Therefore, in general, they can generate SPARQL queries from natural language sentences and questions, in other words, Natural Language Query (NLQ). The generated queries are syntactically correct. However, in most cases they may not be semantically correct as they still lack the context and instructions to generate the queries desired by the user [17]. In this work, we developed a methodology to improve the syntactic and semantic correctness of SPARQL queries generated from LLMs. We mainly use prompt engineering, which means giving knowledge required by an LLM (e.g., in the form of additional context such as text

[5] `https://github.com/OPCFoundation/UA-Nodeset/blob/latest/Robotics/Opc.Ua.Robotics.NodeSet2.xml`

[6] `https://lm-kbc.github.io/challenge2023/`

documents, excel sheets, knowledge graphs, etc.) and giving it instructions or guidelines in the form of natural language text to perform the task a user wants.

There are several ways to provide instructions to LLMs [21]. Our methodology relies on prompt engineering with zero-shot learning and has three steps:

- *Step 1: Schema preparation*: provides context/schema to the LLM such as the semantic artifact, based on which the queries should be generated;
- *Step 2: Instruction template adaptation*: a list of instructions for the LLM is created by adaptation of an instruction template, and
- *Step 3: LLM configuration* towards optimal and consistent results.

The development of this methodology was an iterative process that involved performing multiple experiments on LLMs for SPARQL query generation. At every iteration, we extended the context and fine-tuned the instructions to the LLM to improve the correctness and accuracy of the generated SPARQL query. As part of our methodology, we developed a prompt template for the SPARQL query generation (cf. Listing 1.1). The prompt template contains three placeholders: `SCHEMA` of the semantic artefacts (for Step 1), `Domain-Specific-Instructions` (for Step 2), and `NLQ` to be translated into SPARQL.

Listing 1.1. Prompt template for SPARQL query generation using LLMs

```
By using this ontology:
{SCHEMA}
Instructions:
1. First, generate the paths and relations.
2. Then generate SPARQL query using those paths and relations.
3. Define all the necessary prefixes fully and use only those prefixes in
     the query.
4. Use precise prefixes while generating a SPARQL query; do not give
    multiple options for prefixes in the query.
5. Do not use '/' or '\textbackslash' inside the WHERE clause; define the
     full path in prefix.
6. Do not include any explanations or apologies in your responses.
7. Do not respond to any questions that ask for anything else than for
    you to construct a SPARQL query.
8. Do not include any text except the SPARQL query generated.
{Domain-Specific-Instructions}
{NLQ}
```

3.1 Step 1: Schema Preparation

The schema part of the prompt is the semantic artifact. Should the artifact contain ontologies with complex (OWL) axioms, this leads to challenges for the LLM. Therefore, a pre-processing step is required to simplify the OWL axioms and present them as simple triples prior to being used by the LLM. In this case, there is a token limit for the schema. In our case, we used OpenAI's GPT-4 [11] LLM engine where the token size limit is 8192 tokens. In contrast to this approach, the schema can be given to an LLM by uploading documents. In this

way, the token limit can be further extended. However, when the size of the schema is larger, it should be divided into chunks and it should be kept in a vector database. Moreover, appropriate chunks should be given to the LLM for query generation, however, this is a part of our future work.

3.2 Step 2: Instruction Template Adaptation

Listing 1.1 shows a prompt template containing the list of instructions provided to an LLM for query generation. The instruction part is divided into two sections.

General instructions (instructions 1-8): are domain agnostic and can be used to generate SPARQL queries in any domain. They were enriched in an iterative way through experiments. Therefore, this list is not complete, but rather a baseline to be further extended. The instructions have these roles:

- *Instructions 1-2* define the main goal of the prompt, i.e., query generation. In the case of semantic artifacts with complex query paths between entities with multiple levels, it is challenging for an LLM to generate the paths and relations between entities correctly. Therefore, these first two instructions ensure that the LLM first summarizes the paths and relations between entities before using them in SPARQL queries. This leads to better results.
- *Instruction 3-5* instruct on how the prefixes should be defined and used in the generated query to provide accurate results.
- *Instructions 6-8* request that only a SPARQL query is returned so that it can be executed on a triple store without further processing.

Listing 1.2. An example of domain-specific instructions for OPC UA domain

```
9. Use 'hasProperty' relation while generating the SPARQL query to get to
     a variable node.
10. Use 'hasComponent' relation while generating the SPARQL query to get
    to an object node.
11. Subjects and objects are entities in a generated SPARQL query, so
    they should always start with a capital letter.
```

Domain specific instructions: the LLM requires domain specific instructions to generate accurate queries for domain specific use cases. In our experiments, we mainly focused on generating the SPARQL queries OPC UA where the structure of information models is complex. Therefore, we also might need to provide domain specific instructions such as in Listing 1.2. In OPC UA there are mainly two types of relations between entities: "hasComponent" (refers from an Object node to another Object node) and "hasProperty" (refers from an Object node to a variable node). We observed that the LLM could not use these relationships correctly to generate SPARQL queries before being explicitly instructed on how to make use of these domain specific information (instructions 9-11). A general lesson learned from our experiments is that, when complex domain knowledge is present in the use cases, additional instructions need to be provided that ensure a correct interpretation of this knowledge.

3.3 Step 3: LLM Configuration

Besides making use of a prompt template, it is crucial to configure the LLM appropriately for each use case. For query generation purposes, the LLM needs to be accurate rather than creative, i.e., deterministic in its answers. Therefore, the temperature of the LLM for this use case should be set to zero in order to enable producing consistent results across several executions. In this project, we used Azure OpenAI Service with GPT 4.0 Engine.

4 Experimental Evaluation

The aim of these experiments is to evaluate the quality of SPARQL queries produced using LLM. To this end, we evaluated our method in the following two use cases according to the evaluation strategy described in Section 4.1. The result of the evaluation is reported in Section 4.2.

Use Case 1 (UC1): Validation with Compliance Rules. In our prior work, we described the process of extracting compliance rules from OPC UA companion specifications [9]. These compliance rules fulfill a similar role to competency questions as they describe the requirements to comply with an OPC UA companion specification. The information model of an OPC UA companion specification can be converted from XML into an OWL ontology and subsequently can be queried with SPARQL queries [7]. To convert these textual compliance rules to SPARQL queries the LLM-based semantic artifact validation method can be used.

In UC1, we chose the OPC UA Robotics companion specification and its information model for our evaluation. Robotics information model describes the components of a robot and how they should interact and communicate with each other. The compliance rules, to comply with the information model are explicitly and implicitly stated in the text and tables in the specification document. We extracted 25 compliance rules in textual form. The related template file and compliance rules can be found in [16].

Use Case 2 (UC2): Validation with Competency Questions. Competency Questions represent the requirements of ontologies [22] and can be used to ensure the capability of an ontology in answering them. The competency questions are normally formulated in natural language and have to be converted into SPARQL queries to validate the ontology. Therefore this use case can also be addressed with the methodology presented in this paper.

For our evaluation purpose, we chose the SAREF ontology for UC2 since it covers similar domains as OPC UA with its companion specifications but is less complex than OPC UA. To this end, we instantiated data and developed a set of 15 competency questions related to the TemperatureSensor concept in SAREF Core. The related template file, competency questions list, and end-to-end implementation are located in [16].

4.1 Evaluation Strategy

First, for both use cases, the NLQs and their expected, corresponding SPARQL query outputs were established as ground truth. Then the NLQ and the semantic artifacts were provided to the LLM-based semantic artifact validation system. The received `<SPARQL;result>` pairs, were then compared with the ground truth. Differing results were marked as *False* and matching results were evaluated with SPARQL and ontology experts. Following this, results matching the expectation were labelled as *True*, and finally, all results were evaluated in terms of precision, recall, and F1 score.

Furthermore, there are two different ways of using GPT 4.0 in our evaluation context: GPT Web UI and APIs. The first method involved running each sentence through the Azure-based GPT 4.0 Web UI, with prepared prompts using a consistent template populated with the ontology and the question. The second method was to execute the same NLQs through GPT 4.0 API Calls. We tested each use case with both methods and evaluated the results.

4.2 Experimental Results

We evaluated the LLM-based semantic artifact validation methodology on both SAREF and OPC UA robotics following the evaluation design explained above.

The OPC UA Robotics KG was evaluated with 25 compliance rules pertaining to validation. Out of the compliance rules, 13 correspond to a valid implementation on the knowledge graph, while there is no implementation for 12 of them on the relevant knowledge graph.

Table 1. Semantic Artifact Validation Results over Compliance Rules for OPC UA Robotics

Source & Sending Method	Number of NLQ	Type of NLQ	Precision	Recall	F1 score
Robotics (over Azure Web UI)	25 (13/12)	Compl. Rules	.88	.88	.88
Robotics (over Azure API)	25 (13/12)	Compl. Rules	.88	.88	.88
Robotics* (whole experiment)	50 (26/24)	Compl. Rules	.88	.88	.88

Similarly, 15 competency questions for validating the SAREF KG were used. Among these, 10 were identified as valid questions, while 5 were not as shown in Table 2. Although everything remained consistent in terms of sentences, template, ontology, the LLM engine, and configurations, minor discrepancies are noticed between the Web UI results and API Call results.

4.3 Discussion

The main lessons we learned during our work include:

– *Verified prompt templates play a crucial role.* The main challenge of our work was optimizing the prompt template, as even small changes to the template

Table 2. Semantic Artifact Validation Results over Competency Questions for SAREF

Source & Sending Method	Number of NLQ	Type of NLQ	Precision	Recall	F1 score
SAREF (over ChatGPT UI)	15 (10/5)	Competency Q.	1.00	1.00	1.00
SAREF (over ChatGPT API)	15 (10/5)	Competency Q.	0.92	0.95	0.93
SAREF* (whole experiment)	30 (20/10)	Competency Q.	0.95	0.97	0.96

showed deviation in the results. Templates that worked for some sentences in the dataset, may not work for other sentences. Therefore, we optimized the template iteratively to derive the template in Listing 1.1, which was successfully applied for all the sentences in our datasets. Consistently using this template lead to improved quality and accuracy of the generated queries, and is one of the key reusable results of our work.

- *In-depth domain and modeling semantics are challenging for LLMs.* The modeling semantics of OWL axioms and the domain knowledge necided in the OPC UA relations could only be processed in our workflow when these semantic structures were rewritten and presented in a simplified manner.
- *Execution modality impacts on results.* The query generation experiments were conducted using API and web-page based execution modalities (Sect. 4.1). In general, the results were better on the Web UI. Identifying the reason for this still requires deeper investigation.
- *The complexity of the semantic artifact negatively impacts the results' quality.* The macro average F1 score results for UC1 are 88% across all experiments (Table 1). Moreover, 100% F1 score result was achieved through Web UI, but this fell to 93% when using the API call (Table2). The overall average result for UC2 is 96%. OPC UA Robotics had a more complex NLQ and semantic artifacts and achieved a lower score.

5 Conclusion and Future Work

We proposed a methodology to support semantic artifact validation task with LLMs. We evaluated our approach with two different semantic artifacts from the industry domain and open source artifacts. We investigated how we could use textual requirements, such as competency questions and compliance rules, for the task. Consequently, we generated SPARQL queries through LLM and did this using two different methods including ChatGPT Web UI and API Calls. The results ranged between 88% and 100% in terms of F1 score, which can be considered promising for the future. Furthermore, throughout this study, we had the opportunity to conduct numerous experiments and observations. We expressed how important prompt engineering is in the new era of LLM usage for KE, especially on semantic artifact validation, and its challenges.

In the future, we plan to develop and evaluate our approach for scalability and robustness within the industry domain and beyond. Furthermore, we plan to continue our research on LLMs for other KE tasks in the industry domain.

References

[1] Oscar Corcho et al. *A maturity model for catalogues of semantic artefacts*. English. WorkingPaper. June 2023. DOI: 10.48550/arXiv.2305.06746.

[2] María Poveda-Villalón et al. "LOT: An industrial oriented ontology engineering framework". In: *Engineering Applications of Artificial Intelligence* 111 (2022), p. 104755.

[3] Elwin Huaman, Elias Kärle, and Dieter Fensel. "Knowledge graph validation". In: *arXiv preprint arXiv:2005.01389* (2020).

[4] Thomas Burns, John Cosgrove, and Frank Doyle. "A Review of Interoperability Standards for Industry 4.0." In: *Procedia Manufacturing* 38 (2019), pp. 646–653.

[5] Franz Georg Listl et al. "Utilizing ISA-95 in an Industrial Knowledge Graph for Material Flow Simulation-Semantic Model Extensions and Efficient Data Integration". In: *Procedia CIRP* 120 (2023), pp. 1558–1563.

[6] Ahmadi Seyedamir, Borja Ramis Ferrer, and Jose L Martinez Lastra. "An ISA-95 based ontology for manufacturing systems knowledge description extended with semantic rules". In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. IEEE. 2018, pp. 374–380.

[7] Rainer Schiekofer et al. "A formal mapping between OPC UA and the Semantic Web". In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Vol. 1. IEEE. 2019, pp. 33–40.

[8] Jean-Charles Leclerc et al. "Use of Ontologies to Structure and Manage Digital Technical Data of Industrial Assets: First Steps Towards an Ecology of Knowledge in Multi-Energies Industry". In: *Proceedings http://ceur-ws.org ISSN* 1613 (2022), p. 0073.

[9] Nilay Tufek et al. "Towards Extraction of Validation Rules from OPC UA Companion Specifications". In: *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE. 2023, pp. 1–8.

[10] Yashoda Saisree Bareedu et al. "Deriving semantic validation rules from industrial standards: An OPC UA study". In: *Semantic Web* Preprint (2022), pp. 1–38.

[11] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[12] Jeff Z. Pan et al. "Large Language Models and Knowledge Graphs: Opportunities and Challenges". In: *Transactions on Graph Data and Knowledge* 1.1 (2023), 2:1–2:38. DOI: 10.4230/TGDK.1.1.2. URL: https://drops.dagstuhl.de/entities/document/10.4230/TGDK.1.1.2.

[13] Shirui Pan et al. "Unifying Large Language Models and Knowledge Graphs: A Roadmap". In: *IEEE Transactions on Knowledge and Data Engineering* (2024), pp. 1–20. DOI: 10.1109/TKDE.2024.3352100.

[14] Bradley P. Allen, Lise Stork, and Paul Groth. "Knowledge Engineering Using Large Language Models". In: *Transactions on Graph Data and Knowledge* 1.1 (2023), 3:1–3:19. DOI: 10.4230/TGDK.1.1.3. URL: https://drops.dagstuhl.de/entities/document/10.4230/TGDK.1.1.3.

[15] ETSI 2021. *Official ETSI portal for SAREF*. 2021. URL: https://saref.etsi.org/index.html.

[16] Nilay Tufek. *Siemens-OKE/llm-query-pipeline*. https://github.com/Siemens-OKE/llm-query-pipeline. Year.

[17] Lars-Peter Meyer et al. "Llm-assisted knowledge graph engineering: Experiments with chatgpt". In: *arXiv preprint arXiv:2307.06917* (2023).

[18] Nandana Mihindukulasooriya et al. "Text2KGBench: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text". In: *The Semantic Web – ISWC 2023*. Ed. by Terry R. Payne et al. Cham: Springer Nature Switzerland, 2023, pp. 247–265. ISBN: 978-3-031-47243-5.

[19] Hanieh Khorashadizadeh et al. "Exploring In-Context Learning Capabilities of Foundation Models for Generating Knowledge Graphs from Text". In: *arXiv preprint arXiv:2305.08804* (2023).

[20] Rafal Jozefowicz et al. "Exploring the limits of language modeling". In: *arXiv preprint arXiv:1602.02410* (2016).

[21] Fina Polat, Ilaria Tiddi, and Paul Groth. "Testing Prompt Engineering Methods for Knowledge Extraction from Text". In: ().

[22] Camila Bezerra, Fred Freitas, and Filipe Santana. "Evaluating Ontologies with Competency Questions". In: *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Vol. 3. 2013, pp. 284–285. DOI: 10.1109/WI-IAT.2013.199.