An LLM-Aided Enterprise Knowledge Graph (EKG) Engineering Process

Emanuele Laurenzi, Adrian Mathys, Andreas Martin

FHNW - University of Applied Sciences and Arts Northwestern Switzerland
adrian.mathys@students.fhnw.ch, emanuele.laurenzi@fhnw.ch, andreas.martin@fhnw.ch

Abstract

Conventional knowledge engineering approaches aiming to create Enterprise Knowledge Graphs (EKG) still require a high level of manual effort and high ontology expertise, which hinder their adoption across industries. To tackle this issue, we explored the use of Large Language Models (LLMs) for the creation of EKGs through the lens of a design-science approach. Findings from the literature and from expert interviews led to the creation of the proposed artefact, which takes the form of a six-step process for EKG development. Scenarios on how to use LLMs are proposed and implemented for each of the six steps. The process is then evaluated with an anonymised data set from a large Swiss company. Results demonstrate that LLMs can support the creation of EKGs, offering themselves as a new aid for knowledge engineers.

Introduction

The amount of data being generated today is expanding quickly (Song et al. 2019), and the corporate environment is becoming more complex (Simsek et al. 2022). Enterprises are increasingly looking for methods to gain insightful information and make better decisions from their knowledge assets. Enterprise Knowledge Graphs (EKGs) have become a potent tool for this purpose (Abu-Salih 2021). They have the ability to preserve relationships and meaning (Kejriwal 2019) in data when combining them from diverse sources (Zou 2020). However, because it often calls for subject-matter experts with in-depth knowledge of the knowledge base (Kejriwal 2019), the development process of knowledge graphs from differently structured, domain-specific data poses major obstacles (Simsek et al. 2022). This paper investigates a promising new approach (Bi et al. 2023; Zhu et al. 2023) to enterprise knowledge graph creation by introducing probabilistic language models to the process. Through the use of generative artificial intelligence, this method enables the automated construction of knowledge graphs in various domains. In this work, the terms ”knowledge graph” and ”ontology” are used interchangeably, which follows a more academic stance than an industrial one. The paper is structured as follows. First, section describes the related work that motivates the worthiness of this study. Section reports an enterprise knowledge graph engineering process, which derives from the findings from the literature review and experts' reviews. Next, the suggested artifact is described in section , while a running example of how to use the process is described in section . The evaluation and related findings are reported in section . Finally, section concludes the paper.

Related Work and Research Objective

Knowledge graphs have drawn a lot of interest (Ehrlinger and Wöß 2016; Hogan et al. 2021) because of their ability to represent complex information in a structured and interconnected format (Chaudhri et al. 2022). Their inherent nature of capturing and modelling relationships among entities allows for meaningful analyses (Angles et al. 2017) and reasoning (Hogan et al. 2021). Based on the provenance of the data, two types of knowledge graphs can be distinguished. Open knowledge graphs draw their underlying data from publicly available sources. Proprietary knowledge graphs, on the other hand, obtain data from non-public, enterprise internal bases (Chaudhri et al. 2022).

Either way, creating a knowledge graph becomes especially difficult when working with large amounts (Fensel et al. 2020) of differently structured data from various sources (Tamašauskaitė and Groth 2023). Many approaches have been proposed for creating knowledge graphs, including bottom-up (Li et al. 2020), top-down (Li et al. 2020) and in-between methods (Ji et al. 2022; Tamašauskaitė and Groth 2023). The procedures generally consist of multiple steps that deal with data extraction, knowledge fusion and information representation. In detail, entities are identified and harmonised, corresponding relationships are detected and textual descriptions (labels) are added.

The reliance on human expertise constitutes a major impediment to the process. This is due to the sheer amount of data (Li et al. 2020; Tamašauskaitė and Groth 2023), which makes manual knowledge graph creation labour-intensive (Ruan et al. 2016), expensive (Ji et al. 2022) and prone to human errors (Kejriwal 2019). At the same time, there is a shortage of relevant experts in many domains (Abu-Salih 2021). The growing number of applications of knowledge graphs in different fields makes it especially important to automate the building steps of knowledge graphs (Ji et al. 2022). In particular, knowledge graph creation from large-

scale unstructured content should exhibit a high degree of automation (Kejriwal 2019).

A promising strategy for addressing the challenges involved in knowledge graph creation is generative artificial intelligence (AI) (Bi et al. 2023; Zhu et al. 2023). Generative AI methods can leverage probabilistic large language models (LLMs) – language models that have been pre-trained on substantial amounts of textual data – in such a way that machines become able to understand and generate natural language (and other modalities) based on custom input (Yang 2023). This capability can be used for various knowledge graph generation tasks (Yang 2023; Zhu et al. 2023), such as entity and relation extraction and question answering (Zhu et al. 2023).

Despite the high expectations for generative AI, there is still much to learn about how it can be used to build knowledge graphs (Zhu et al. 2023). While LLM-based generative AI techniques (such as ChatGPT) have been thoroughly studied and employed in many fields (Yang 2023), there has not been much research on their application for building knowledge graphs (Zhu et al. 2023), let alone enterprise knowledge graphs. Instead, in many instances, domain-specific knowledge graphs have been built using more traditional methodologies (Dong, Yu, and Li 2021; Ruan et al. 2016; Song et al. 2019). In addition, numerous existing studies have focused on open data sets (e.g., Wikidata) based on information generally available on the internet (Chaudhri et al. 2022; Google 2023; Li et al. 2020; schema.org 2022).

Given the relevance of the topic in both research and industry, this work investigated the potential of generative AI techniques employing probabilistic language models for the creation of Enterprise Knowledge Graphs (EGKs). To achieve this research objective, we adopted the Design Science Research (DSR) strategy (Vaishnavi, Kuechler, and Petter 2004), as it is adequate for the creation of novel artefacts that are relevant in both research and the real world.

An Enterprise Knowledge Graph Engineering Process

This section describes the findings of the first DSR phase, the Awareness of Problem, which derived from a literature review and experts' interviews. Interviews have been conducted with two senior knowledge engineers, both working in tech companies that build knowledge graphs for other companies; one company is based in Germany, and the other is based in California. Interviewee 1 and the company make extensive use of the Semantic Web Stack. Interviewee 2 and the company use Object-Role modeling (OMR) to conceptualize enterprise knowledge. Insights from the two knowledge engineers were synthesized and extended with existing literature. The result is a top-down process for building ontologies and knowledge graphs in company settings. The process is detailed below.

1. **Formulate informal competency questions.**
   Informal competency questions are expressed in natural language and place demands on an underlying ontology. An ontology must be able to answer these questions (Grüninger and Fox 1995). The Interviewees emphasize that there is no standardized set of questions that fits every scenario or domain. Good informal competency questions should be defined in a stratified manner. This means that higher-level questions require the solution of lower-level ones (Grüninger and Fox 1995).
2. **Construct the ontology schema.**
   Once the domain and scope of the knowledge engineering endeavor have been defined, the interviewees propagate a model-first approach when it comes to enterprise knowledge graph creation. This top-down approach – where a (visual) ontology schema is built as one of the first steps – is also outlined by Li et al. (2020) and Tamašauskaitė and Groth (2023). When no existing ontology can be reused, a new one needs to be constructed. This can be done by enumerating important terms and relationships in the given domain. These terms are then classified into classes, properties, and facets (Noy and McGuinness 2001).
3. **Extract data and knowledge and integrate it into the knowledge graph.**
   In the literature (Ji et al. 2022; Li et al. 2020; Tamašauskaitė and Groth 2023), knowledge extraction is described as the extraction of entities, attributes and relationships from data sources. As soon as the required data sources are identified, knowledge can be extracted from them according to the previously constructed ontology schema. This allows to enrich the initial ontology schema. In a procedure called knowledge fusion or integration, a mapping is performed between the ontology and the extracted knowledge (Tamašauskaitė and Groth 2023). The knowledge is integrated by aligning the extracted entities, attributes and relationships to the classes and properties defined in the ontology schema (Li et al. 2020). As a result of entity and relation extraction (and knowledge processing), it is possible to construct RDF triples and build a knowledge graph. The underlying data can be stored in various ways, including relational databases, triple stores and graph databases (Tamašauskaitė and Groth 2023).
4. **Validate the mapping of the data to the ontology.**
   Both interviewees and Li et al. (2020) and Tamašauskaitė and Groth (2023) agree that it is important to continuously validate the alignment of the extracted data with the ontology schema. This can be done based on manual sampling (Li et al. 2020) or in an automated way by taking advantage of the structure and constraints defined in the ontology schema (Tamašauskaitė and Groth 2023). Interviewee 1 recommends the use of SHACL for automated validation of triples in RDF.
5. **Create an instance of a knowledge graph.**
   Creating an instance refers to populating the structure defined in the ontology with actual transaction data (Noy and McGuinness 2001). To populate a knowledge graph, interviewee 1 and his company usually retrieve RDF data (instances) from structured sources. This is done by converting formats like JSON, XML, CSV and relational databases to RDF triples. Instances can be displayed visually, allowing navigation through the graph

structure and the discovery of related knowledge. However, in order to enable effective use of the knowledge graph, querying is required (Tamašauskaitė and Groth 2023). The standard query language for RDF graphs is SPARQL.

6. **Maintain and extend the knowledge graph.**
   As knowledge constantly changes and evolves, knowledge graphs are never complete and need to be continuously updated. The aspect of maintaining a graph relates to integrating new data from sources (Tamašauskaitė and Groth 2023), while knowledge completion can be performed through deductive reasoning to extend the knowledge graph. Deductive reasoning enables the establishment of new relations among given entities or to derive new entities based on existing knowledge (Ji et al. 2022; Tamašauskaitė and Groth 2023). A language that can be used to define rules for deductive reasoning is SWRL. With SWRL, one can create a chain of reasoning where the output of one rule becomes the input for the next rule (Horrocks et al. 2004).

The LLM-aided Enterprise Knowledge Graph Engineering Process

This section presents the results of the second DSR phase, the Suggestion Phase. Specifically, in this phase, we addressed the concern of how probabilistic language models can support the creation of enterprise knowledge graphs. Thus, we built from the previously presented six steps.

The proposed LLM-aided EKG engineering process is illustrated in Figure 1. For each process step, the figure depicts a suggestion of what a large language model can contribute. The process steps are detailed in the following sub-sections.

LLM to Generate Informal Competency Questions

To ensure that the LLM is aware of what is meant by informal competency questions, sources such as Noy and McGuinness (2001) can be cited. The language model should also be fed with a textual description of the area of knowledge for which the questions need to be formulated, including information about existing concepts, their properties and relationships. Once the LLM has been fed with informal competency questions and the given domain, it can be asked to generate corresponding competency questions. It is suggested to encourage diverse questions by varying the prompts. Examples of such prompts could be:

- What are the key questions someone might have about [topic, concept, property, relationship]?
- Generate competency questions that cover a range of complexity, from basic facts to specific nuances.

The LLM should then answer with a list of questions. The model can be provided with feedback on the generated questions to improve them iteratively.

LLM to Generate an Ontology Schema

LLMs can be used to generate an enterprise knowledge graph schema by extracting the important terms, their characteristics and relations within a knowledge area. In the first step, what is expected from an ontology schema should be explained to the model. The LLM should then be presented with input data from which the schema needs to be constructed. These might be textual descriptions that include mentions of the classes, properties, relationships and facets prevalent in the knowledge area. Prompt engineering should be applied to formulate varying prompts that evoke different answers from the LLM. Examples of such prompts are as follows:

- Based on the provided input data, extract an ontology schema by retrieving important terms/classes, properties and relationships. Create a table with the following columns: subject, object, predicate.
- For each class, list its properties and the properties' facets in a table with the columns class, property and facet.

After receiving responses, the model can be guided with feedback on the results. Follow-up prompts can be used to ask about the structure of the ontology. Examples might be:

- What is the relationship between [class A] and [class B]?
- Are there any constraints to [property C]?

LLM to Extract Data and Convert them to RDF

LLMs can assist in entity extraction, resolution, and format conversion. The list of entities from the previous step might include some terms that refer to the same thing. The LLM can be guided to recognize this.

- As [class D] and [class E] are the same thing, call them both [class D].

In order to obtain results in a structured format, input data can be provided to the LLM along with an explicit request to format them in RDF(S) or OWL.

- Based on the ontology schema and description provided, convert the identified classes, properties, relationships and facets to a description in OWL.

To convert data to RDF, the data source can be provided in its structured origin format and the output can be requested in Turtle syntax (Beckett et al. 2014). It shall be assessed whether all the classes and properties from the source have been classified correctly in the generated RDF code.

LLMs can be used to generate SHACL shapes to express conditions and restrictions on RDF graphs. This can be done from natural language input or based on an existing RDF(S) representation. A Turtle code can be used as a basis for the generation of SHACL constraints. These can be derived from the ontology schema (facets) but also from instances. For example, an LLM should be able to identify a date format from an instance and provide the right data type. When a generated SHACL shape is incomplete or wrong, it can be refined by means of prompts.

- Add a property [property D] with data type Integer.
- Change the SHACL shape so that [term A] is a class which may have several [class B].
- Set the minCount of the property [property A] to 2.

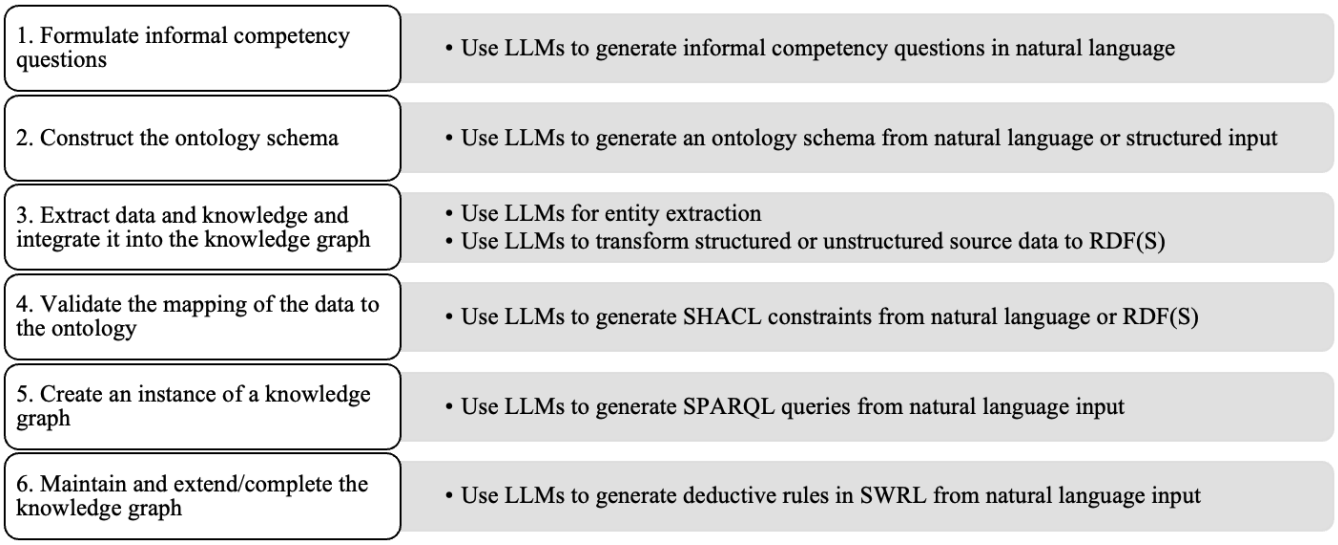

Figure 1: The LLM-aided Enterprise Knowledge Graph Engineering Process.

LLM to Generate Queries in SPARQL

For an LLM to be able to create SPARQL queries, it needs to be aware of the ontology schema and the respective instances. Then it can be asked to transform a natural language query into SPARQL.

- Write a SPARQL query that extracts the people who were born in 1990 from the above dataset.
- Write a SPARQL query that extracts all the organisationUnits without a departmentHead that are located in locationCountry Switzerland.
- Write a SPARQL query that summarises the total revenue generated in 2022.

The resulting queries might lack some conditions or there might be misinterpretations.This can be corrected by asking the LLM to change certain elements of the SPARQL query.

LLM to Generate Deductive Rules in SWRL

In order to generate deductive rules to infer new relationships or entities based on the existing knowledge graph, the LLM must be aware of the ontology and instances. Next, the design of deductive rules can be guided by asking the LLM to take on a specific role.

- Imagine that you are a manager who needs to make important decisions based on the knowledge graph. Think about additional insights that could be gained from the provided data set.

For a formal representation of the resulting rules, SWRL code should be generated. The LLM can propose its own rules or it can be asked to represent customized rules that involve instances from the input data:

- Suggest some deductive rules for the above dataset and define them in SWRL.
- Write a SWRL rule to infer a new Boolean property highPriority for instances where the dueDate is in less than five days.

One can always use follow-up prompts to refine the rules to include additional conditions or to change certain parts.

Use of the LLM-aided Enterprise Knowledge Graph Engineering Process

In this section, we describe the DSR phase implementation, where LLMs have been used in each of the six process steps. For this, five LLMs were considered: ChatGPT 3.5/4 (OpenAI), Bard (Google), Claude 2 (Anthropic) and Code LLaMA 34b (Meta). These have been selected for the following reasons: they are easy to access (through a user interface), are usable at either no cost or at a reasonable price, are the latest LLMs available in the market, which are known to best performant on applications like translation, question answering, and program synthesis.

The open data set on summer Olympic medals (Agrawal 2019) is used as input. The data set consists of a brief textual description as well as instances in a CSV file. The models are fed with prompts and data that correspond to the respective task.

LLM to Generate Informal Competency Questions

The models are asked to generate informal competency questions about the Summer Olympic Games knowledge domain. This is done by submitting the above-mentioned textual description along with a suitable request to the model:

- Provide some competency questions for the knowledge domain of the Summer Olympic Games provided. Before doing so, think about the questions one might have about the domain.

One can ask for more specific information on a specific topic with a prompt like this:

- What are questions one might have about medals?

LLM to Generate an Ontology Schema

As the previous conversation about informal competency questions continues, the models already have contextual awareness, and the description of the Olympic Games domain can be easily recalled.

- Build an ontology schema for the Summer Olympic Games knowledge domain from above. Proceed by retrieving important terms/classes, properties, relationships and constraints (facets).

Further, the models can be asked to list the results in a well-readable way:

- Create a table with the columns subject, object, predicate.

With another prompt, the properties and facets per class can be listed:

- For each class, list its properties and the properties' facets in a table with the columns class, property, facet.

Finally, specific questions about the structure of the ontology can be asked:

- What is the relationship between Event and Discipline?

The procedure can be repeated by providing a structured input with instances to the model. For instance, some rows of the Summer Olympics CSV file can be provided along with the following prompt:

- Below, I provide you with some instances of the domain. Based on this, provide an ontology schema.

LLM to Extract Data and Convert them to RDF

By continuing the conversation from before, it is possible to build on the Summer Olympic Games domain knowledge already established by the LLMs. Although entities have already been extracted before, the LLMs are again asked for a list of them:

- Extract all the entities in the Olympic Summer Games domain. Do not include examples, only the entity names.

Further prompts may be used to consolidate similar entities or to change or add certain entities. Considering the outcomes of ontology schema creation and entity resolution, a comprehensive list of classes, relationships, properties and facets should now be available. This structural ontology information can be converted to RDF(S) format in order build a knowledge graph. The following prompt is used:

- Based on what you know about the Olympic Summer Games ontology schema, convert the identified classes, properties, relationships and facets to a description in RDF(S) format using OWL.

In addition to the structural data, instances are converted to RDF by supplying the model with samples in CSV format. Precise instructions are given to the LLMs:

- In the following, you are provided with some instances of the Summer Olympic Games data set. This set corresponds to the ontology schema of the Summer Olympic Games built above. Proceed as instructed: 1. Organise and display the data as a table. 2. Transform the instances to RDF while using Turtle syntax. Consider the underlying Summer Olympic Games ontology schema.

LLM to Generate Constraints in SHACL

In this LLM application scenario, the models are used to generate SHACL shapes for validation. As input, the ontology schema and the RDF/Turtle representations of instances from before are used.

- Based on the ontology schema about the Summer Olympic Games and the instances, generate some SHACL shapes that include constraints for classes or properties.

The information that there are only three types of medals can be encoded in a SHACL shape by means of a natural language request:

- There are only three types of medals: gold, silver and bronze. Consider this in a SHACL shape.

There are many more options for restrictions in the Olympic Games domain. For example, it can be declared that a sport encompasses multiple disciplines, including several events.

- Generate SHACL shapes denoting that a sport may encompass multiple disciplines. Each discipline can include several events.

LLM to Generate Queries in SPARQL

On the basis of the generated ontology schema and some instances of the Olympic Summer Games data set, search queries are provided in natural language. The LLMs are then prompted to transform these queries to SPARQL.

- In the following, you are provided with some instances of the Summer Olympic Games data set. This set corresponds to the ontology schema of the Summer Olympic Games built above. Proceed as instructed: 1. Organise and display the data as a table. 2. Transform the instances to RDF while using Turtle syntax. Consider the underlying Summer Olympic Games ontology schema.

As a result, the LLMs should now have generated a Turtle representation of the instances. This is a good foundation for generating queries. The following instructions are given:

- Based on this Turtle representation of the Olympic Summer Games instances, generate a SPARQL query that extracts all the women who have won a gold medal.
- Generate a SPARQL query based on the above data set that retrieves all athletes who have won multiple medals.
- The athletes come from different countries. Generate a SPARQL query that lists the countries in descending order based on the number of medals won by their athletes.

LLM to Generate Deductive Rules in SWRL

The final application scenario addresses the generation of deductive rules for reasoning and inference. First, the LLMs are asked to provide ideas for such rules based on the ontology schema and available instances.

- Imagine that you are a manager who needs to make important decisions based on the Olympic Summer Games ontology. Think about additional insights that could be gained from the provided data set.

For a formal representation of these insights, the SWRL language is used.

- Represent some of the suggested insights as deductive rules in SWRL. For each generated SWRL rule, add a description of what it does. Remember to consider the provided dataset of the Olympic Summer Games.

SWRL is also able to build so-called chains of reasoning. These are constructs consisting of multiple rules, where the output of one rule serves as input for the next. Such a chain of reasoning can be evoked with a prompt like this:

- Write a SWRL rule to identify all athletes that have participated in multiple events.

Evaluation of the LLM-aided Knowledge Graph Engineering Process

To assess the usefulness of the LLM-aided knowledge graph engineering process in practice, the evaluation was conducted on a real data set from an anonymous company (Company X). The data set consists of a textual description and a corresponding CSV file with multiple instances. The latter were anonymized.

ChatGPT 3.5 and ChatGPT 4 achieved the best implementation results. As they both stem from the same model family, only ChatGPT 4 is considered in the evaluation. In addition, Claude 2 is taken into account. The two models are applied to the same six steps while utilizing the domain-specific input data. The correctness of the generated results is validated in four ways:

1. Manual validation for natural language text.
2. PlantUML (Ezoic 2023) for visualising ontology schemas.
3. RDF grapher (web tool) for validating RDF code.
4. AllegroGraph (Franz 2023) for implementing and querying RDF graphs.

The results of the evaluation are listed below.

- **LLM to generate informal competency questions**. The competency questions generated by the LLMs were well tailored to the business area of Company X. An example from GPT 4 is provided in Figure 2. *Lesson learned:* LLMs are well-suited for the formulation of informal competency questions. The generated competency questions exhibited a high degree of relevance with respect to the given domain. Understandably, in cases where not much information was available from the given knowledge area, the competency questions became less relevant.
- **LLM to generate an ontology schema**. Both LLMs were capable of extracting important terms from the ontology and visualizing them in PlantUML. Figure 3 provides Claude 2's result. *Lesson learned:* The LLMs demonstrated good performance with some shortcomings. It was observed that the output improved with additional knowledge provided. While the initial extraction was done from natural language text, the concepts of the ontology were refined by both LLMs after providing

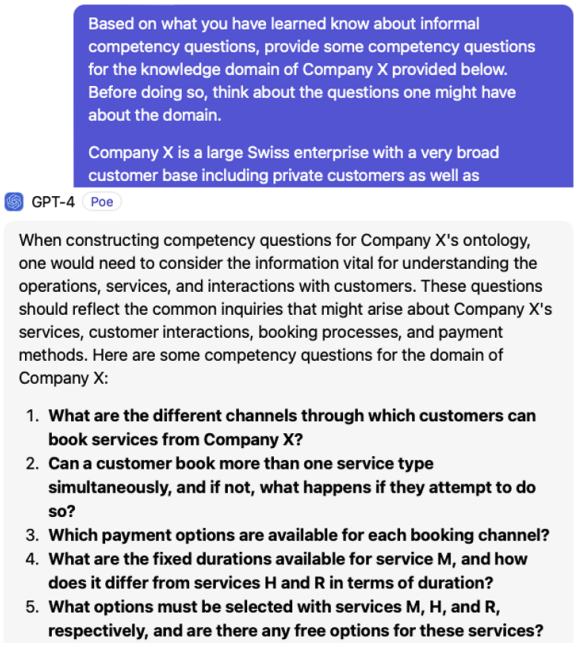

Figure 2: Competency questions generated by GPT 4.

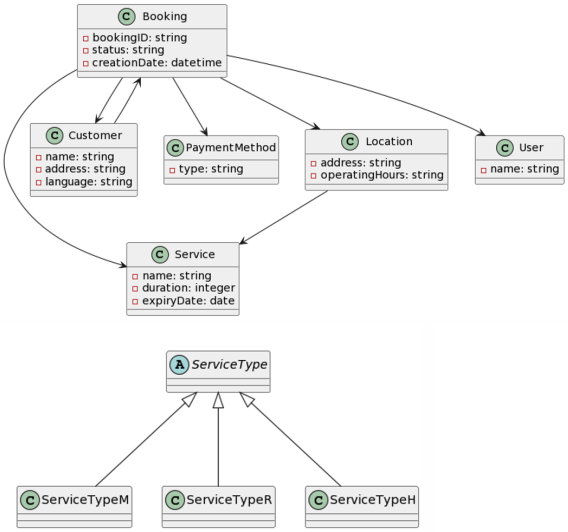

Figure 3: PlantUML ontology schema generated by Claude 2.

RDF data. However, none of the language models managed to extract all the properties from the provided data (textual description and instances). From the real data set, GPT 4 missed two properties and Claude 2 even more. This finding underlines the need to thoroughly inspect the generated ontology schema manually.

- **LLM to extract data and convert them to RDF**. Claude 2 and GPT 4 accurately extracted structural ontology data and instances and transform them to RDF/Turtle while maintaining ontology alignment. The RDF code was successfully validated by RDF grapher (see excerpt in Fig-

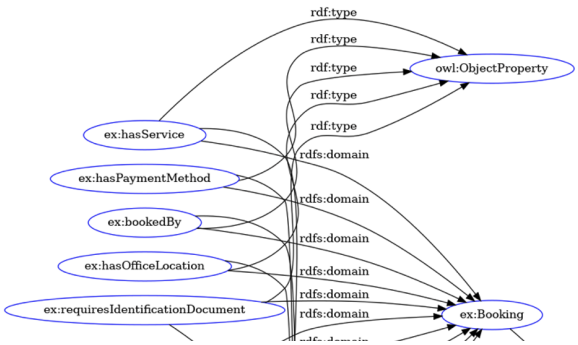

Figure 4: OWL ontology schema generated by GPT 4 and visualised in RDF grapher.

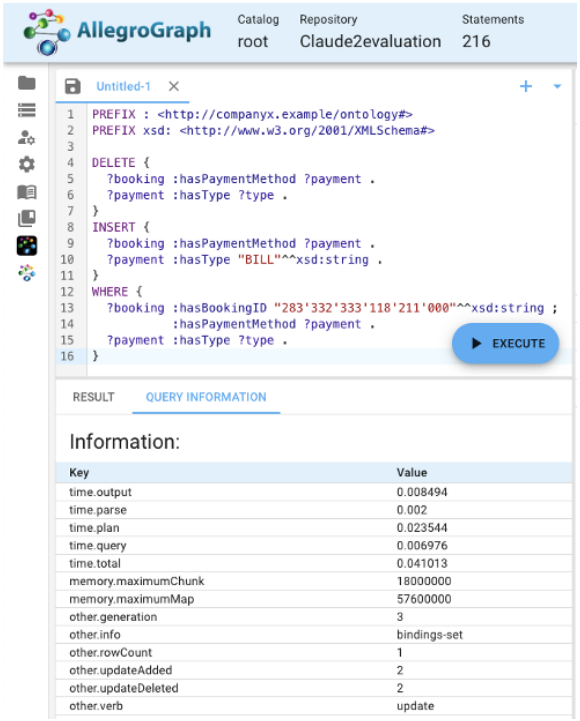

Figure 5: SPARQL update query generated by Claude 2 and executed in AllegroGraph.

ure 4) and imported into the knowledge graph platform AllegroGraph. *Lesson learned:* Promising results were achieved in the experiments that applied LLMs for generating RDF code. While the same properties as before were missing for both models, GPT 4 did not initially define any relationships among the classes. They were only added after being instructed to do so, which required an additional prompt. Afterward, the ontology was successfully validated by RDF grapher. With Claude 2, the relations among the classes were well-defined from the beginning. However, the prefix lines required some manual syntax adjustments. After these changes, the schema was also validated correctly. Both ontology schemas could now be imported into AllegroGraph without further issues. The integration of a small number of instances into the knowledge graph progressed relatively well. Claude 2 and GPT 4 were both able to transform three individual CSV instances of the real data set to turtle syntax while maintaining alignment with the underlying ontology schema. While no adaptations were necessary for GPT 4, the code generated by Claude 2 required some small changes. Finally, the code from both LLMs was successfully validated and integrated into AllegroGraph. The learning that can be drawn from this approach is that it does not work for large numbers of instances. Both LLMs refused to process prompts or provide complete results when confronted with more than three instances of the evaluation set. This makes it almost impossible to integrate a larger data sample into the knowledge graph using GPT 4 or Claude 2.

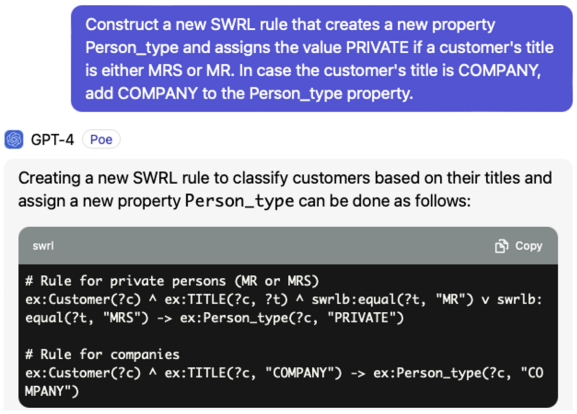

Figure 6: SWRL rule generated by GPT 4.

- **LLM to generate constraints in SHACL**. The two LLMs generated SHACL shapes that were well-suited for the given ontology. The shapes were validated in RDF grapher and integrated into AllegroGraph to demonstrate their correctness. *Lesson learned:* The LLMs proved to be helpful in generating SHACL shapes. GPT 4 returned an extensive set of constraints tailored to the provided instances from the real-world data. The shapes included restrictions to specific datatypes or classes, often accompanied by minimum conditions. The SHACL shapes suggested by Claude 2 were less numerous, but similarly well defined. Based on the instance data, in many cases, the correct data types were selected, such as a date or an enumeration. Nevertheless, some constraints were initially wrong or incomplete. However, these issues were quickly resolved with corresponding prompts and the SHACL shapes were successfully imported into AllegroGraph.
- **LLM to generate queries in SPARQL**. Although some of the generated SPARQL queries required minor corrections, they could be successfully executed against the instances in AllegroGraph. The results were observed to be correct. This also applies to update queries, as Figure 5 demonstrates. *Lesson learned:* Great potential was identified in the LLM-supported generation of SPARQL

queries. A prerequisite for this to work is the awareness of an LLM with the underlying ontology schema. In the experiments, the generated queries were executed against the knowledge graph implemented in AllegroGraph. This allowed verifying their validity based on the query results, which included actual instances. Apart from some erroneous namespaces or prefixes and repeated requests to the LLMs to recall the ontology structure, the generated queries were processed without any problems and their outputs were correct. Notably, these queries included information needs across multiple classes and their properties. GPT 4 and Claude 2 also showcased their ability to generate SPARQL update queries. Specifically, they succeeded in writing a query that changed a property value of an existing instance. Despite these positive results, it is important to emphasise that the person writing the prompts must be well acquainted with the knowledge domain. While, it helped to provide the actual names of classes and properties, there was no need to cumbersomely pre-build the query structure in natural language. It was sufficient to simply name the terms used in the ontology. Case sensitivity was not required. For example, the following prompt was provided:

  – Extract all bookings that were paid for in cash.

  This prompt resulted in a valid SPARQL query, incorporating the two classes Booking and PaymentMethod and the hasType property for the latter.

- **LLM to generate deductive rules in SWRL**. The LLMs proved useful in suggesting and generating SWRL rules concerning the real-world data set. An example is illustrated in Figure 6. *Lesson learned:* Both GPT 4 and Claude 2 showed to be valid in suggesting and formulating deductive rules. With respect to the knowledge domain, both language models gave relevant insights that were gained from the existing instances. The rules were represented in SWRL and were aligned with the ontology. As AllegroGraph does not support the execution of SWRL rules, the LLMs were asked to transform the set of rules into SPARQL queries. Next, the resulting SPARQL queries were successfully executed in AllegroGraph. The queries included rather extensive clauses such as aliases, counting operations, filters and grouping.

Discussion

The capability of LLMs to generate informal competency questions and enterprise knowledge graphs can be leveraged by practitioners to ensure that they align with the business area and cover all relevant topics. By employing LLMs for extracting and converting (even unstructured) data to RDF, the manual effort is expected to be reduced significantly. The need for specialist expertise is also assumed to decrease. However, the experiments underlined the importance of human supervision of the output generated. In addition, limitations were observed when processing larger numbers of instances.

LLMs can also contribute to enforcing data quality and consistency within a knowledge graph by generating SHACL shapes. In practice, with a large number of properties, this is a rather tedious task that could be easily automated based on the nature of the instances provided.

Another important aspect in terms of practical usefulness is the LLM-supported query generation. This approach enables business users without knowledge of the SPARQL query language to fulfill their information needs. While familiarity with the ontology schema is still required, LLMs can help to bridge the knowledge gap. The same applies to the formulation of deductive rules. While LLMs can provide inspiration to decision-makers to infer new knowledge, they can also represent these insights in an executable, structured format such as SWRL. As a result, business-relevant decisions can be made more quickly.

Conclusion and Future Work

The study presents an LLM-aided enterprise knowledge graph engineering process. The latter integrates the use of LLMs into phases of enterprise knowledge engineering. The knowledge engineering process was derived by combining findings from primary and secondary data, where the former were collected by interviewing senior knowledge engineers of tech companies. The process has been implemented using current large language models available in the market and accessible via user interfaces. These were used for evaluation purposes along with a real dataset provided by a large Swiss company. The two language models GPT 4 and Claude 2 achieve remarkable results in generating code in W3C recommendations such as RDF(S), SHACL, OWL, SPARQL and SWRL. However, their performance varies depending on the quality and amount of input provided. In addition, the generated output requires manual validation, which is left to the knowledge engineer. Nevertheless, if these obstacles can be overcome, LLMs are an aid to democratise the creation of knowledge graphs, which might become a practice for a domain or business experts.

A valid follow-up work regards the evaluation of the proposed process in comparison to a human-driven enterprise knowledge graph engineering process. An additional future work worth mentioning concerns fine-tuning experiments on LLMs to add expertise about enterprise knowledge graph engineering projects, which could result in better performance than generic LLMs.

Acknowledgements

We are thankful to Marton Bur from RelationalAI and Wolfgang Schell from metaphacts for their kind availability during the data collection phase of this work.

References

Abu-Salih, B. 2021. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185: 103076.

Agrawal, D. 2019. Summer Olympics Medals (1976-2008). https://www.kaggle.com/datasets/divyansh22/summer-olympics-medals/data. Accessed: 2023-09-03.

Angles, R.; Arenas, M.; Barcelo, P.; Hogan, A.; Reutter, J.; and Vrgoc, D. 2017. Foundations of Modern Query

Languages for Graph Databases. *ACM Computing Surveys (CSUR)*, 50: 1–40.

Beckett, D.; Berners-Lee, T.; Prud'hommeaux, E.; and Carothers, G. 2014. RDF 1.1 Turtle. https://www.w3.org/TR/turtle/. Accessed: 2023-05-03.

Bi, Z.; Chen, J.; Jiang, Y.; Xiong, F.; Guo, W.; Chen, H.; and Zhang, N. 2023. CodeKGC: Code Language Model for Generative Knowledge Graph Construction. *arXiv*, 2304.09048v1.

Chaudhri, V. K.; Baru, C.; Chittar, N.; Dong, X. L.; Genesereth, M.; Hendler, J.; Kalyanpur, A.; Lenat, D. B.; Sequeda, J.; Vrandečić, D.; and Wang, K. 2022. Knowledge graphs: Introduction, history, and perspectives. *AI Magazine*, 43: 17–29.

Dong, B.; Yu, H.; and Li, H. 2021. A knowledge graph construction approach for legal domain. *Tehnicki Vjesnik*, 28: 357–362.

Ehrlinger, L.; and Wöß, W. 2016. Towards a Definition of Knowledge Graphs. *SEMANTICS: Posters and Demos Track*, 48.

Ezoic. 2023. PlantUML at a Glance. https://plantuml.com. Accessed: 2023-09-10.

Fensel, D.; Şimşek, U.; Angele, K.; Huaman, E.; Kärle, E.; Panasiuk, O.; Toma, I.; Umbrich, J.; and Wahler, A. 2020. *Knowledge Graphs: Methodology, Tools and Selected Use Cases*. Springer.

Franz. 2023. AllegroGraph Cloud. https://allegrograph.com/products/cloud/. Accessed: 2023-10-23.

Google. 2023. Welcome to the Gemini era. https://deepmind.google/technologies/gemini/#introduction. Accessed: 2023-10-23.

Grüninger, M.; and Fox, M. S. 1995. Methodology for the Design and Evaluation of Ontologies.

Hogan, A.; Blomqvist, E.; Cochez, M.; D'amato, C.; Melo, G. D.; Gutierrez, C.; Kirrane, S.; Gayo, J. E. L.; Navigli, R.; Neumaier, S.; Ngomo, A.-C. N.; Polleres, A.; Rashid, S. M.; Rula, A.; Schmelzeisen, L.; Sequeda, J.; Staab, S.; and Zimmermann, A. 2021. Knowledge Graphs. *ACM Computing Surveys*, 54: 1–37.

Horrocks, I.; Patel-Schneider, P. F.; Boley, H.; Tabet, S.; Grosof, B.; and Dean, M. 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. https://www.w3.org/submissions/SWRL/. Accessed: 2023-04-15.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Yu, P. S. 2022. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33: 494–514.

Kejriwal, M. 2019. *Domain-Specific Knowledge Graph Construction*. Springer International Publishing. ISBN 978-3-030-12374-1.

Li, F.; Xie, W.; Wang, X.; and Fan, Z. 2020. Research on Optimization of Knowledge Graph Construction Flow Chart. 1386–1390. IEEE. ISBN 978-1-7281-5244-8.

Noy, N. F.; and McGuinness, D. L. 2001. Ontology Development 101: A Guide to Creating Your First Ontology.

Ruan, T.; Xue, L.; Wang, H.; Hu, F.; Zhao, L.; and Ding, J. 2016. Building and Exploring an Enterprise Knowledge Graph for Investment Analysis. In Groth, P.; Simperl, E.; Gray, A.; Sabou, M.; Krötzsch, M.; Lecue, F.; Flöck, F.; and Gil, Y., eds., *The Semantic Web – ISWC 2016*, 418–436. Cham: Springer. ISBN 978-3-319-46547-0.

schema.org. 2022. Schema.org. https://schema.org/. Accessed: 2023-08-10.

Simsek, U.; Kärle, E.; Angele, K.; Huaman, E.; Opdenplatz, J.; Sommer, D.; Umbrich, J.; and Fensel, D. 2022. A Knowledge Graph Perspective on Knowledge Engineering. *SN Computer Science*, 4: 16.

Song, D.; Schilder, F.; Hertz, S.; Saltini, G.; Smiley, C.; Nivarthi, P.; Hazai, O.; Landau, D.; Zaharkin, M.; Zielund, T.; Molina-Salgado, H.; Brew, C.; and Bennett, D. 2019. Building and Querying an Enterprise Knowledge Graph. *IEEE Transactions on Services Computing*, 12: 356–369.

Tamašauskaitė, G.; and Groth, P. 2023. Defining a Knowledge Graph Development Process Through a Systematic Review. *ACM Transactions on Software Engineering and Methodology*, 32: 1–40.

Vaishnavi, V.; Kuechler, W.; and Petter, S. 2004. Design Science Research in Information Systems.

Yang, Z. 2023. Chinese tech giant Baidu just released its answer to ChatGPT. https://www.technologyreview.com/2023/03/16/1069919/baidu-ernie-bot-chatgpt-launch/. Accessed: 2023-10-12.

Zhu, Y.; Wang, X.; Chen, J.; Qiao, S.; Ou, Y.; Yao, Y.; Deng, S.; Chen, H.; and Zhang, N. 2023. LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities. *arXiv*, 2305.13168v1.

Zou, X. 2020. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series*, 1487.