Large Language Model for Ontology Learning In Drinking Water Distribution Network Domain

Rhea Huang[1], Erkan Karabulut[1][0000−0003−2710−7951], and Victoria Degeler[1][0000−0001−7054−3770]

Informatics Institute, University of Amsterdam, Netherlands
y1kelinblue@gmail.com; {e.karabulut, v.o.degeler}@uva.nl

Abstract. Currently, most ontologies are created manually, which is time consuming and labour intensive. Meanwhile, the advanced capabilities of Large Language Models (LLMs) have proven beneficial in various domains, significantly improving the efficiency of text processing and text generation. Therefore, this paper focuses on the use of LLMs for ontology learning. It uses a manual ontology creation method as a basis to facilitate the use of LLMs for ontology learning. Two different variants of LLMs have been experimented with, and they all demonstrate the capability of ontology learning to varying degrees. This approach proves that it is possible to automate the entire process of using LLMs to generate the ontology and make it accessible to people without prior domain expertise. It is also very efficient, not only saving significant time but also improving usability. The final ontology returned by the LLMs could be used as a base version according to the evaluation results and needs to be validated and refined by the domain experts to ensure its accuracy and completeness.

Keywords: LLM, Ontology Learning, Drinking water distribution network

1 Introduction

The widespread success of Large Language Models (LLMs) such as ChatGPT has made LLMs one of the most popular topics in the AI field [12]. The LLM is trained on massive amounts of data, and based on the observed patterns in the training data. It is capable of performing Natural Language Processing (NLP) tasks in real-time, such as generating answers to prompts and summarizing texts. Retrieval Augmented Generation (RAG), as a popular approach to building RAG, but without the need for training [11]. In addition, it could significantly increase the accuracy of results [19].

An ontology, defined as "a formal, explicit specification of a shared conceptualization" [16], usually serves as a formal model that defines the common vocabulary and relationships in a given domain.

The construction of an ontology can be categorized into two approaches: the manual approach or the (semi) automatic approach. The manual construction of the ontology is usually done by domain experts, and this process involves considerable effort [2]. The ontology engineer has to manually analyse textual data, identify relevant terms and concepts, and map them to ontology representation languages. Manual construction of ontology offers the advantage of precision and structure, but takes a lot of time. In recent years, (semi) automatic ontology creation has gained popularity [2]. It refers to a (part of) or the complete ontology creation process being handled by automated tools or algorithms. The (semi) automatic methods are efficient in terms of production time, but they lack accuracy as they lack domain expertise in the creation phase and are less structured.

Therefore, this project explores a new way of creating an ontology - combining manual and (semi) automatic methods. It uses LLMs to facilitate the process of ontology construction and the framework is derived from manual methods. The aim is to automate the process of generating an ontology while following the established steps of manual ontology creation. This hybrid method could increase the efficiency of ontology construction while maintaining the structure during the ontology creation process. The drinking water distribution network (DWDN) was chosen as a case study to test the feasibility of the approach. The research question is proposed as follows : ***How can LLMs be used to facilitate the construction of an ontology in the drinking water distribution network domain based on a manual ontology creation method?***

The remainder of this paper is organized as follows: First, a literature review on ontology learning and existing ontology creation methods is conducted in Section 2. Section 3 describes the DWDN use case. Next, Section 4 explains the approach of this research to create the ontology. This is followed by the results in Section 5. Finally, the paper discusses the limitations and future work in Section 6, and provides conclusions in Section 7.

2 Literature Review

This section analyzes existing ontology construction methods to find the best one among manual and (semi) automatic construction methods; and looks at the usage of LLMs for ontology learning and at the retrieval-augmented generation.

2.1 Manual ontology construction

Two systematic literature reviews (SLR) have investigated the ontology construction, listing in total 21 manual ontology creation methods [23; 3]. In order to find the best method that could be used as a framework for (semi) automatic ontology creation, eight evaluation criteria, either derived from the original paper [23] (domain analysis, level of detail, evaluation, documentation, maintenance, reusability, sample application) or tailored to this project (domain specificity), were selected to assess the quality of the manual methods. Two methodologies stand out after the evaluation: the UPON Life Ontology [8] and the NeOn Methodology [25]. The difference between these two methodologies lies in their level of details score. The NeOn Methodology offers high flexibility with a bit more detailed results, allowing users to select proposed scenarios based on the specific situation, whereas the UPON Life Ontology provides direct and concise instructions for ontology construction. Given the context of this study, the UPON Life Ontology is a preferred approach, as it provides comprehensive instructions and the entire construction cycle by default.

A few relevant papers from the recent years were domain-specific and could not be generalized to other domains, such as the construction industry [10] and the product development [26].

2.2 (Semi) Automatic ontology construction

Zulkipli et al. [30] discuss 19 (semi-)automatic methods for ontology construction. 8 of these methods are semi-automatic and 11 are fully automatic based on the categorisation of the paper. They did not define any criteria for evaluating these methodologies, so, in this paper, we have devised several new criteria (Description of automation (C3), Main tools or algorithm (C4), Availability of tools(C5), Input data for automation(C7), Accessibility(C8)) and augmented them with some criteria for manual methods (Evaluation(C2), Domain specificity(C6), Level of detail(C1)) to find the best (semi-)automatic methodologies. The results of the evaluation can be found in the Github supplementary material.

Most methodologies contain detailed descriptions and clearly outline the automated ontology construction process. They all used various tools to facilitate automation, such as NLP [4; 28] and the RelExOnt algorithm [15]. For methodologies using logic instead of specific tools for automation, the availability of tools is not measurable. There are also a variety of input data formats, but most are plain text, such as databases and websites, and several methods use natural language processing (NLP) for automation. Although the stage of using NLP was restricted, such as input data preprocessing or term extraction, rather than ontology learning.

2.3 Ontology learning

Ontology learning refers to the "integration of knowledge from diverse fields, utilizing technology to automatically or semi-automatically construct ontology based on various input data" [21]. It is often associated with ontology engineering and machine learning.

There has been lack of study that uses LLMs for fully automated ontology construction that without human intervension at all, but they could work as an assistant in this process [5; 18]. ChatGPT performs well in the task of term typing, recognizing a type taxonomy, and discovering non-taxonomic relations between types [5]. Meanwhile, the 'Flan-T5 LMM' fine-tuning method is effective and could ease the knowledge acquisition bottleneck. Furthermore, LLM demonstrates exceptional capability in translating natural language sentences into description logic [22].

Retrieval-augmented Generation Although LLMs have demonstrated remarkable generation capabilities, they have a significant limitation – hallucination, they may produce incorrect or missing information when queried beyond the scope of their training data [29]. This limitation affects the reliability and validity of the responses provided by LLMs. Nevertheless, this limitation is not insurmountable. In 2020, a technique known as Retrieval-Augmented Generation (RAG) was introduced [19], which has the potential to mitigate this risk by using external documents. RAG combines pre-trained parametric and non-parametric memory for language generation. It not only exploits the strong generation abilities of LLMs but also incorporates external data as relevant information input. Therefore, RAG serves as a powerful tool for customizing factual answers within specific domains. The study has been carried out to demonstrate the feasibility of developing competency questions and then used LLMs to support the automation process of ontology and knowledge graph construction via the RAG technique [18].

In general, RAG can be divided into three types: Basic RAG, Advanced RAG, and Modular RAG [13]. Different types of RAGs vary in the specifics of their operational processes. The framework of RAG usually consists of the input, retrieval memory, and generation model.

3 Use case: Drinking water distribution network

A drinking water distribution network (DWDN) is a large and complex network, it connects water treatment plants or water sources (in the absence of treatment) to customers via a network of pipes, storage facilities, valves, and pumps [1]. The DWDN serve several purposes such as providing water for households, supplying water for firefighting and supporting industrial processes [7].

There is currently no existing DWDN ontology, as the currently published ontologies in the water networks domain focus on the drinking water quality [24; 1] or water resources management [9] rather than the DWDN itself.

EPANET is a widely used software for modeling and analysis of DWDNs [2]. It includes various data could be used for modeling such as colour-coded network maps, data tables, energy consumption, response, calibration, and time series graphs. EPANET documentation contains extensive information about the DWDN domain and this information is utilized as part of our methodology while performing the RAG method.

4 Methodology

The approach of this project combines both manual and automated methods. It uses the instructions outlined in the UPON Life Ontology as queries to LLMs

[1] `https://www.epa.gov/dwreginfo/drinking-water-distribution-system-tools-and-resources`

[2] `https://www.epa.gov/water-research/epanet`

for the ontology generation via the RAG framework. This section outlines the complete ontology creation process, including the structure of the RAG, the overarching strategy of prompt engineering and the query pipeline.

4.1 Framework of the RAG

The RAG enables the LLMs to generate contextual responses, and it is the core framework of ontology creation in this project. This project implements the Basic RAG, which could be segmented into three parts. The LangChain library from Hugging Face is employed [3].

Input In this step, users provide various types of data to serve as external information sources. These documents are segmented into smaller units and transformed into vector scores using embedding models. In our use case, the resource is coming from EPANET. It is designed primarily as a research tool, and the third chapter of the EPANET website contains resources about water network modeling [4]. This resource comprehensively covers both physical and non-physical components in the water networks, as well as simulation models for water flow. Although the website is initially in HTML format, it has been converted to PDF format in order to be fed in as an input document.

Retrieval Information retrieval is the process of extracting relevant information from a given input document based on a user query [17]. The information retriever identifies the chunks that best match the query based on their vector similarity. The retriever requires two parameters as input - search type and search arguments. The search type parameter was set to Maximum Marginal Relevance (MMR). MMR offers the advantage of diversified search results. Its scoring mechanism combines the relevance of the document chunks to the user query and the novelty of the chunk compared to others [6]. The search arguments define K to set the retriever to return the top K chunks based on the MMR score. Typically, the value of K falls within the range of 5 to 10, depending on the size of the divided chunks. For this project, K was set to 10.

Generation model In the generation phase, the LLM produces a contextually relevant response that is coherent with the query and the retrieved units. Four variants of LLMs were used in this project: gpt-4-0125-preview (gpt-4), gpt-3.5-turbo-0125 (gpt-3.5-turbo), gpt-4-turbo-2024-04-09 (gpt-4-turbo) and huggingfaceh4/zephyr-7b-beta (7b-beta).

The construction of the RAG requires several parameters. The *template,* which provides a structured format for the RAG. It consists of two parts: a system template that applies to all the queries, and a user query specified for each

[3] `https://python.langchain.com/v0.1/docs/integrations/platforms/huggingface/`

[4] `https://epanet22.readthedocs.io/en/latest/3_network_model.html`

prompt. The *query,* which is defined by the user and passed to LLM via a predefined system template. The *Parser,* which converts the generated response into a human-readable string to make it more suitable for display and interpretation.

4.2 Query generation

Once the RAG environment has been configured, the next step is to create queries. The user query is used to instruct the LLM to generate the answer in the desired format and direction. The queries of this project are based on the UPON Life Ontology, which includes the creation of a domain terminology; domain glossary; taxonomy; predication; parthood, and ontology in the Turtle syntax format (a way of expressing data in the RDF data model [5]).

The creation of queries can be divided into three phases. The first phase involves experimentation, focusing on the query instructions. The primary aim is to gain a basic understanding of the answers generated by the LLMs. During this stage, the step descriptions are copied directly from the UPON Life Ontology and then tested multiple times with all LLMs. Based on the outputs, the query is slightly modified, such as adding more descriptions or removing redundant content. For example, if the output indicates a lack of understanding of a certain concept in the query, an explanation of that concept is added. At the end of this step, the core set of queries is defined, which is with step descriptions.

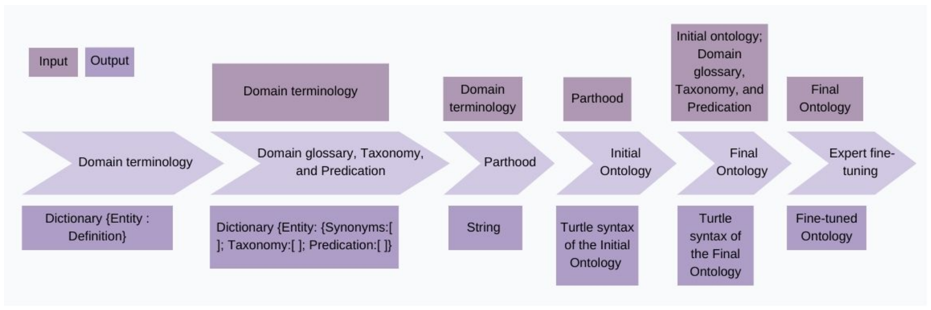
Having gained a basic understanding of the responses provided by the LLMs, the second phase shifts the focus to data structure. The primary objective of this step is to query the prompt in a specific way so that the LLM returns an answer in the desired format.

Prompt engineering is about creating the "recipe" to guide the LLMs to perform the desired task [20]. There are several papers that have introduced prompt patterns to enhance prompt engineering [27; 14], but these identified patterns are all tailored for OpenAI models [6]. After experimenting with a promising template pattern [27]: *"I am going to provide a template for your output. X is my placeholder for content. Try to fit the output into one or more of the placeholders that I list. Please preserve the formatting and overall template that I provide. This is the template: PATTERN with PLACEHOLDERS"*, only the gpt-4 model demonstrates its capability to process this template pattern effectively.

Therefore, the original prompt templates were created, tested, and adjusted based on the outputs. Placeholders are usually used in the prompt to structure the response format. For example: *"Please provide the complete answer formatted as a Python Dictionary {Entity: Definition}"*. By incorporating customized format templates into the query, the queries became longer and the answers became more structured. As a result, by the end of this phase, the queries consisted of two parts: format instructions and step descriptions.

The third stage, the fine-tuning of queries, focuses on improving both the quality and the structure of the results. Although the answers in the last phase

[5] `https://en.wikipedia.org/wiki/Turtle_(syntax)`
[6] `https://platform.openai.com/docs/models/overview`



Input
Output
Domain terminology
Domain terminology
Parthood
Initial ontology; Domain glossary, Taxonomy, and Predication
Final Ontology
Domain terminology
Domain glossary, Taxonomy, and Predication
Parthood
Initial Ontology
Final Ontology
Expert fine-tuning
Dictionary {Entity : Definition}
Dictionary {Entity: {Synonyms:[ ]; Taxonomy:[ ]; Predication:[ ]}
String
Turtle syntax of the Initial Ontology
Turtle syntax of the Final Ontology
Fine-tuned Ontology

Fig. 1: Query pipeline

already possess some initial structure and quality, it's crucial to refine them to further optimize the results. During this stage, the existing prompts are fine-tuned based on the output from the different models, by adjusting the content details and changing the position of the content within the query. For example, whether the format template or the step descriptions should come first. In the step of the ontology generation, it was observed that LLMs struggled to process large amounts of information to create the ontology Turtle syntax all at once, resulting in numerous mistakes and loss of information. As a solution, another step – final ontology was introduced to give LLMs more room and time to generate the final Turtle syntax, thereby enriching the initial ontology. At the end of this stage, all the queries can result in an overall satisfactory quality, comprising coherent answers structured in accordance with the specified query format.

4.3 Query pipeline

This subsection provides an overview of the query, including its structure, purpose, and components. The sub-subsections describe each query in detail. The query pipeline is shown in the Fig. 1. The query example for domain terminology is given below and full queries are available in the Github supplementary material.

The system template is created to provide a general structure for the input query and output. Within the system template, the "query" and "Assistant" are the indicators for the starting position of the query and answer respectively. Within the "query," its components are introduced.

1. **System template:**
   – This part provides a general template for generating answers and handling input. "template: | You are an AI assistant that follows instruction extremely well. Please give direct and complete answers. Please be truthful, if you don't know the answer, just say that you don't know, don't try to make up an answer or return the wrong answer. {query}
   Assistant"
2. **Query**

- **Introduction:** This part introduces the query and explains its context.
- **Input:** Specifies the input for the query, such as the result from previous steps.
- **Step Descriptions:** Describes the instructions for each step.
- **Answer Format:** Outlines the format for presenting the query results.
- **Example:** Provides an illustrative example of the query in action.

Domain terminology The first step creates the domain terminology by identifying the entities or concepts within the domain e.g. in the DWDN domain, the entities are "junction," "pipe," etc. The structure of the first step is important because it forms the basis of the ontology. The response should adopt a dictionary format. This format allows for easy looping of the entity within the dictionary in subsequent steps if needed.

QueryQ1: We are creating an ontology in the water distribution network domain. The first step involves creating a domain-specific terminology, or list of terms characterizing the domain at hand. This is a preliminary step to start identifying domain knowledge and drawing the boundaries of the observed domain. The drinking water distribution network could be modeled as a collection of links connected to nodes. The outcome of this step should be a domain lexicon, or information structure used to answer the question "What are the physical components typically used while building drinking water distribution networks?" Provide the output in a Python dictionary format suitable for direct iteration within a Python for loop, structured as {"Entity": "Short definition of the entity"}, with no additional explanations outside of the dictionary.

Domain glossary, taxonomy, and predication The second step is to determine the domain glossary (synonyms), taxonomy, and predication for each entity that was generated in the previous step. For example, in the DWDN domain, we may establish that 'control device' is a synonym for 'valve'; create a hierarchical taxonomy where 'butterfly valve' is a category of 'valve'; and specify that 'valve' has the property of 'material'. This step is designed to take the entire dictionary result of the domain terminology as input and return the domain glossary, taxonomy, and predication in a dictionary format.

Parthood This step involves establishing the relationships between entities that are generated in the first step. Parthood defines part-whole relationships within the ontology, such as the relationship between "pipe" and "valve" in the context of the DWDN. This step uses the entire domain terminology dictionary to replace the input placeholder in the query, provide step descriptions, and specify the requirements such as avoiding the generation of conflicting relationships.

Initial ontology Until the last step, all stages of information gathering have been completed. This step involves encoding the parthood into Turtle syntax that is machine-readable. It creates the initial ontology based on the output from the last step. Through experimentation, it was observed that the LLMs lack the ability to generate correct ontology Turtle syntax without explicit instruction, which is not provided by the UPON Life Ontology. Therefore, this query is tailored by us without reference to the UPON Life Ontology. The expected output of this step is a correct and complete initial Turtle syntax.

Final ontology This step also involves encoding information into the Turtle syntax. Ideally, the input to this step includes the initial ontology from the previous step along with domain glossary, taxonomy, and predication for all the entities. However, due to the limited processing capacity of the free model such as 7b-beta, it cannot handle very large texts at once. Therefore, the alternative approach is to use another predefined query that is slightly different in the input part which feed each entity from the dictionary individually, to gradually generate the final ontology Turtle syntax. As in the previous step, specific step descriptions and examples are developed. The expected outcome of this step is a correct and complete Turtle syntax containing all the information generated before.

Expert fine-tuning After the previous step, the LLM has generated the final ontology, and the quality of the ontology should be checked to ensure the accuracy and completeness. Therefore, domain experts evaluate it and provide suggestions for improvement.

5 Result

This result section first describes the answers generated at each step and then evaluates the performance of LLMs: gpt-4-0125-preview (gpt-4), gpt-3.5-turbo-0125 (gpt-3.5-turbo), gpt-4-turbo-2024-04-09 (gpt-4-turbo), huggingfaceh4/zephyr-7b-beta (7b-beta).

Domain terminology. The DWDN is modeled as nodes and links in the input document, the main physical components consist of pipes, pumps, valves, junctions, tanks, and reservoirs. Across all the responses generated by different LLMs, terms like pipe(s), valve(s), pump(s), reservoir(s), and hydrant(s) are the most common. Surprisingly, the term "hydrant" does not appear in the input document. However, it is a part of the drinking water distribution network, it refers to "a discharge pipe with a valve and spout at which water may be drawn from a water main" [7]. The term "junction" is not consistently included among these frequent terms, except in the response from 7b-beta. Below is the example answer returned by 7b-beta of this step:

{'pipe': 'A conduit for transporting water through a drinking water distribution network.', 'reservoir': 'A large storage tank used to store water in a drinking water distribution network.', 'pump': 'A mechanical device used to move water

[7] https://www.merriam-webster.com/dictionary/hydrant

Table 1: Example answers of Domain glossary, taxonomy, and predication

gpt-4-turbo		
Pipe		
Synonyms	["Conduit", "Tubing"]	
Taxonomy	"Water Transport Infrastructure": ["Pipe"]	
Predication	- "AP"	["Material", "Diameter", "Length"]
	- "CP"	["Installation Information"]
	- "RP"	["Connected to Pump", "Connected to Valve"]

Table 2: Example answers of Parthood

gpt-4-turbo	
Water Distribution Network	
hasPart	Pipe
hasPart	Pump

through a drinking water distribution network.', 'valve': 'A device used to control the flow of water in a drinking water distribution network.', 'hydrant': 'A device used for emergency water access in a drinking water distribution network.', 'tank': 'A container used to store water in a drinking water distribution network.', 'junction': 'A point where multiple pipes join in a drinking water distribution network.', 'manhole': 'A structure used to provide access to the drinking water distribution network for maintenance and repair.'}

Domain glossary, taxonomy, and predication. The expected result of this step is a structured dictionary with the content of domain glossary, taxonomy, and predication. All the LLMs return a structured dictionary, however, not all models are able to create the structured answer within the dictionary, as illustrated in the query. LLM gpt-3.5-turbo can only return the predication without further specification such asatomic properties (AP) and complex properties (CP). Moreover, both gpt-4 and gpt-4-turbo models generate very similar answers in this step. The example answer of all models is shown in the Table 1.

Parthood. This step generates the relationships between entities. In general, gpt-4 and gpt-4-turbo return similar results. Mostly the entities like "pump" are connected by the big concept "drinking water distribution network". Gpt-3.5-turbo and 7b-beta create relationships between the entities but put less emphasis on creating relationships under the one big system "drinking water distribution network". Also, 7b-beta sometimes returns conflicting answers, even when the query explicitly instructs not to do so. An example answer of this step is shown in the Table 2.

Initial ontology. The expected result of this step is a formal initial ontology Turtle syntax that encodes the answer from parthood. However, not all models were able to produce the correct Turtle syntax, with 7b-beta performing particularly poorly and gpt-3.5-turbo gives the most straightforward answer. The 7b-beta model either failed to generate the answer due to server errors or produced Turtle syntax with incorrect content. While the gpt models were also

Table 3: Example of Initial ontology

gpt-4-turbo
`ex:WaterDistributionNetwork rdf:type owl:Class.  ex:Pipe rdf:type owl:Class.  ex:WaterDistributionNetwork :subClassOf [  owl:hasPart ex:Pipe, ex:Pump, ex:Valve, ex:Reservoir,  ex:WaterTower, ex:Meter, ex:Hydrant, ex:PressureRegulator,  ex:BackflowPreventer, ex:ServiceLine].`

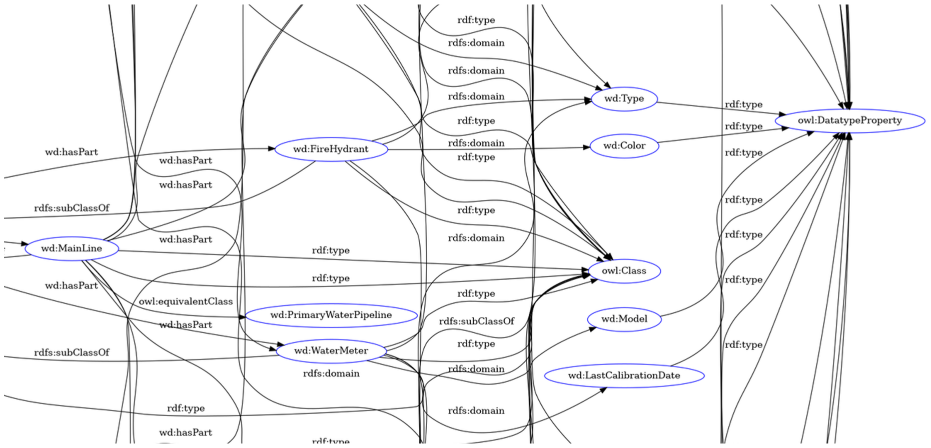


Fig. 2: Example of the final ontology

successful in generating Turtle syntax, they showed inconsistent stability and errors such as repetitive class creation or missing information. The example result is provided in the Table 3.

Final ontology. Ideally, the final ontology should contain all the information generated in the initial ontology, domain glossary, taxonomy and predication. However, all the models struggle somewhat in this aspect. Although they all follow the query instructions' structure, the result is disorganized, containing errors or miss significant amounts of information. Nevertheless, gpt-4-turbo outperforms the other models to some extent, as it can produce the Turtle file with fewer missing details and syntax errors. An example response from this model is illustrated in Figure 2.

Expert fine-tuning. Two domain experts acknowledged the quality of the ontology produced by gpt-4-turbo and suggested that it could serve as a valuable starting point. They also highlighted its significant utility for individuals lacking domain knowledge but interested in ontology construction. The main criticism centered around the overly complex responses, which included many less important terms such as color as an property and hydrant as an entity, while essential entities such as water demand were missed. Meanwhile, not all entities have a definition and one domain expert mentioned that it is a challenge for the them to understand it. All suggestions have been integrated into the final ontology.

The visualized full fine-tuned ontology is available in the Github supplementary material.

5.1 Model evaluation

The evaluation consists of two parts, the results evaluated by the domain expert and the ranking of the models based on the experience gained during the project. The full evaluation results is available in Table 4.

In this project, the overall performance of the models is qualitatively ranked along four dimensions: time (the duration required to generate the answer), scalability (the ability to handle large ontologies), cost (whether the use of the LLM is free or incurs a cost), and responsiveness (the ability to follow instructions accurately). The gpt-4-turbo has best performance in terms of cost-effectiveness, responsiveness, and scalability. gpt-4-turbo is followed by gpt-4. The 7b-beta model is characterized by being free of cost but it has a longer response time.

In addition, to better understand the stability of each LLM, the domain terminology query was run ten times and the frequency of terms graph is plotted in Fig. 3. The 7b-beta returns the same answer even on different runs, and the gpt LLMs are able to return answers that are very similar but differ in details such as the order of terms and the definitions of terms.

Meanwhile, a similarity check is performed to compare the answers generated by the same model and all queries over three runs. This check involves vectorizing the responses and then using cosine similarity to measure the difference. The similarity check score is calculated for each model with all queries, each run three different times, resulting in three answers. When compared, it is clear that there is not a high degree of similarity between the answers generated by the models, indicating the answers might be unstable. Although gpt-4 achieved the highest similarity score, suggesting that its outputs are somewhat closer to its previous output compared to the other models. The overall similarity between the answers remains are not very high for all models.

When analyzing the results of each step, the models show similar performance in the intermediate steps. The main difference between the models lies in the initial and final phases. While 7b-beta excels in the first step of domain terminology, its performance drops in the last two steps of initial ontology generation and final ontology generation. Conversely, gpt-4-turbo performs well in the final step and gpt-3.5-turbo performs well in the initial ontology construction.

The domain expert evaluated the models anonymously without prior knowledge of the model identity. The expert evaluated the result from three perspectives on the scale of 1 to 10: accuracy, ensuring factual correctness; relevance, alignment with the DWDN domain; and completeness, addressing the query comprehensively. The scores of each model are determined based on both the initial and the final ontologies. There is no reference ontology used for scoring; instead, the models are scored at once after a thorough review of all the models. All models achieve a score of 7 or above, meaning that the results are good for the first version, but not perfect. Of all the models tested, gpt-4-turbo achieved the highest score.

Model / Evaluation	7b-beta	gpt-3.5-turbo	gpt-4	gpt-4-turbo
Accuracy	8	8	8	8
Relevance	6	8	6	8
Completeness	6	6	8	9
Time (Shortest to Longest)	4th	3rd	2nd	1st
Scalability (Largest to Smallest)	4th	3rd	2nd	1st
Cost (Highest to Lowest)	4th	3rd	1st	1st
Responsiveness (Most to Least)	4th	3rd	2nd	1st
Similarity	0.39	0.48	0.54	0.5

Table 4: Comparison of Model Performance

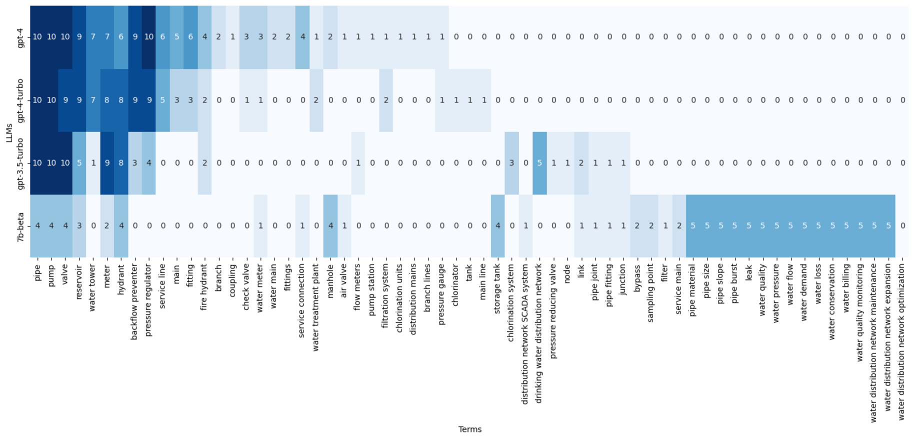


Fig. 3: Frequency of terms of each LLM

6 Discussion

This section discusses the limitations of this approach.

Instability of LLMs With each run, even when presented with the same query and model, the similarity between the answers is not very high. Despite setting the temperature (randomness) of the LLMs to a low value of 0.5, there is still significant instability in the responses. Multiple query runs were conducted and selected the best answer by comparison to reduce the randomness of the outputs. However, there remains a possibility that the model could produce better or worse results with the same query.

Parameter settings. In this research, parameter values were chosen based on commonly used values. For example, at the document splitting stage, the chunk size was set to 256 and the chunk overlap was set to 20 but they could be any value; the maximum length and the maximum new tokens were set to 16384 and 8096 respectively. These parameter configurations have the potential to affect the retriever's quality or LLM's ability to generate the answers, which in turn affects the final result. Therefore, it is possible that the project did not achieve optimal results due to these settings.

Document input. One noteworthy point of feedback from the domain expert is that the final ontology lacks certain concepts such as water demand. However, we subsequently found that the original input document doesn't extensively cover the topic of water demand. This suggests that the input document used in this project may lack sufficient information. The chosen documents must fully cover the topics that are expected to be included in the ontology.

Full automation of the process. While the methodology used in this project facilitates the generation of a basic version of the ontology within minutes, the final ontology still needs to undergo fine-tuning by the domain expert to ensure accuracy and relevance before publication and official use. The first version of the ontology generation process is fully automated, but human intervention is still required afterward.

Utilizing multiple LLM architectures. Based on the results, the model's performance may vary at different steps of the process. No single model consistently excels or fails in all steps. Therefore, using models that perform better at certain steps and combining them based on their respective strengths at each step could improve the overall accuracy of the final result.

7 Conclusion

In conclusion, this project focuses on the use of LLMs for ontology learning, combining manual and automated methods into a new approach to ontology creation. The UPON Life Ontology was chosen as the foundation for the query input to the LLMs. This hybrid ontology creation approach increases the generation speed, which significantly reduces the production time, and the accessibility of ontology creation to non-domain experts as it only requires the involvement of domain experts in the fine-tuning step.

This project tested multiple LLMs to provide insights into the capabilities of different LLM variants in ontology learning within the drinking water distribution network domain. Results from four LLMs are evaluated both qualitatively and quantitatively to provide a more comprehensive understanding of the overall quality of answers.

Currently, the final ontology requires refinement by domain experts before usage. One suggestion fro the domain expert is that the refinement could possibly start with adjusting the query so that the LLMs only return the most important concepts with detailed explanation.

Currently, the final ontology requires refinement by domain experts before it can be used. One suggestion for the domain expert is that the refinement could possibly start with adapting the query so that the LLMs only return the most important concepts with detailed explanations.

Furthermore, the choice of the LLM is crucial. While no LLM can produce answers identical to those of a domain expert, using a more advanced LLM that trained on superior data may increase the likelihood of achieving a better result.

In this project, the latest OPENAI model, gpt-4-turbo, has superior overall performance compared to other models. In the case of budget constraints, 7b-beta remains a viable option.

Last but not least, the user query plays an important role in the answer generation, as it directly influences the quality and format of the answer. During the query generation process, it's important to continually experiment to find the right balance of the query complexity. If the query is too complex, LLMs may struggle to process it correctly. Conversely, if the query is too simple, LLMs may miss the underlying concept. The development of the query template and a query generation strategy can greatly improve this process, facilitating the creation of queries that are both informative and comprehensible for LLMs to interpret accurately.

Data and code availability. The code of our approach is available at `https://github.com/DiTEC-project/Large-Language-Model-for-Ontology-Learning-In-Drinking-Water-Distribution-Network-Domain`.

Acknowledgement. This work is funded by the project DiTEC: Digital Twin for Evolutionary Changes in Water Networks (NWO 19454). We would like to thank Yvonne Hassink-Mulder for expert validation of the generated ontology.

Bibliography

[1] Ahmedi, L., Jajaga, E., Ahmedi, F.: An ontology framework for water quality management. SSN@ ISWC **1063**, 35–50 (2013)

[2] Al-Aswadi, F.N., Chan, H.Y., Gan, K.H.: Automatic ontology construction from text: a review from shallow to deep learning trend. Artificial Intelligence Review **53**(6), 3901–3928 (2020)

[3] Alfaifi, Y.: Ontology development methodology: a systematic review and case study. In: 2022 2nd International Conference on Computing and Information Technology (ICCIT). pp. 446–450. IEEE (2022)

[4] Alobaidi, M., Malik, K.M., Sabra, S.: Linked open data-based framework for automatic biomedical ontology generation. BMC bioinformatics **19**, 1–13 (2018)

[5] Babaei Giglou, H., D'Souza, J., Auer, S.: Llms4ol: Large language models for ontology learning. In: International Semantic Web Conference. pp. 408–427. Springer (2023)

[6] Carbonell, J., Goldstein, J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 335–336 (1998)

[7] Council, N.R., on Earth, D., Studies, L., Science, W., Board, T., on Public Water Supply Distribution Systems, C., Assessing, Risks, R.: Drinking wa-

ter distribution systems: Assessing and reducing risks. National Academies Press (2007)
[8] De Nicola, A., Missikoff, M.: A lightweight methodology for rapid ontology engineering. Communications of the ACM **59**(3), 79–86 (2016)
[9] Escobar, P., Roldán-García, M.d.M., Peral, J., Candela, G., Garcia-Nieto, J.: An ontology-based framework for publishing and exploiting linked open data: A use case on water resources management. Applied Sciences **10**(3), 779 (2020)
[10] Farghaly, K., Soman, R.K., Collinge, W., Manu, P., Mosleh, M.H., Cheung, C.: Construction safety ontology development and alignment with industry foundation classes (ifc). Electronic Journal of Information Technology in Construction **27**, 94–108 (2022)
[11] Fatehkia, M., Lucas, J.K., Chawla, S.: T-rag: lessons from the llm trenches. arXiv preprint arXiv:2402.07483 (2024)
[12] Filippo, C., Vito, G., Irene, S., Simone, B., Gualtiero, F.: Future applications of generative large language models: A data-driven case study on chatgpt. Technovation **133**, 103002 (2024)
[13] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2023)
[14] Giray, L.: Prompt engineering with chatgpt: a guide for academic writers. Annals of biomedical engineering **51**(12), 2629–2633 (2023)
[15] Kaushik, N., Chatterjee, N.: Automatic relationship extraction from agricultural text for ontology construction. Information processing in agriculture **5**(1), 60–73 (2018)
[16] Khadir, A.C., Aliane, H., Guessoum, A.: Ontology learning: Grand tour and challenges. Computer Science Review **39**, 100339 (2021)
[17] Khin, N.T.W., Yee, N.N.: Query classification based information retrieval system. In: 2018 International conference on intelligent informatics and biomedical sciences (ICIIBMS). vol. 3, pp. 151–156. IEEE (2018)
[18] Kommineni, V.K., König-Ries, B., Samuel, S.: From human experts to machines: An llm supported approach to ontology and knowledge graph construction. arXiv preprint arXiv:2403.08345 (2024)
[19] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
[20] Liu, V., Chilton, L.B.: Design guidelines for prompt engineering text-to-image generative models. In: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems. pp. 1–23 (2022)
[21] Maedche, A., Staab, S.: Ontology learning. In: Handbook on ontologies, pp. 173–190. Springer (2004)
[22] Mateiu, P., Groza, A.: Ontology engineering with large language models. arXiv preprint arXiv:2307.16699 (2023)
[23] Sattar, A., Surin, E.S.M., Ahmad, M.N., Ahmad, M., Mahmood, A.K.: Comparative analysis of methodologies for domain ontology development:

A systematic review. International Journal of Advanced Computer Science and Applications **11**(5) (2020)

[24] Shahsavani, S., Mohammadpour, A., Shooshtarian, M.R., Soleimani, H., Ghalhari, M.R., Badeenezhad, A., Baboli, Z., Morovati, R., Javanmardi, P.: An ontology-based study on water quality: probabilistic risk assessment of exposure to fluoride and nitrate in shiraz drinking water, iran using fuzzy multi-criteria group decision-making models. Environmental Monitoring and Assessment **195**(1), 35 (2023)

[25] Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernandez-Lopez, M.: The neon methodology framework: A scenario-based methodology for ontology development. Applied ontology **10**(2), 107–145 (2015)

[26] Wagner, A., Sprenger, W., Maurer, C., Kuhn, T.E., Rüppel, U.: Building product ontology: core ontology for linked building product data. Automation in Construction **133**, 103927 (2022)

[27] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382 (2023)

[28] Zhang, Y., Saberi, M., Chang, E.: Semantic-based lightweight ontology learning framework: a case study of intrusion detection ontology. In: Proceedings of the international conference on web intelligence. pp. 1171–1177 (2017)

[29] Zhao, J., Haffar, G., Shareghi, E.: Generating synthetic speech from spokenvocab for speech translation. arXiv preprint arXiv:2210.08174 (2022)

[30] Zulkipli, Z.Z., Maskat, R., Teo, N.H.I.: A systematic literature review of automatic ontology construction. Indones. J. Electr. Eng. Comput. Sci **28**, 878 (2022)