LARGE LANGUAGE MODEL FOR ONTOLOGY LEARNING IN DRINKING WATER DISTRIBUTION NETWORK DOMAIN

Yiwen Huang[1], Erkan Karabulut[1] and Victoria Degeler[1]

[1]*University of Amsterdam, Amsterdam, Netherlands*

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 construction method as a basis to facilitate the LLMs for ontology learning. The proposed approach is based on Retrieval Augmented Generation (RAG), and passed queries to LLMs are based upon the manual ontology method – UPON Lite ontology. Two different variants of LLMs have been experimented with, and they all demonstrate the capability of ontology learning to varying degrees. This approach shows promising initial results in the direction of (semi-) automated ontology learning using LLMs and makes the ontology construction process easier for people without prior domain expertise.The final ontology was evaluated by the domain expert and ranked according to the defined criteria. Based on the evaluation results, the final ontology could be used as a base version, but it requires further fine-tuning by 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 them one of the most popular tools in the AI field [1]. A 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 queries and summarizing texts.

An ontology, defined as "a formal, explicit specification of a shared conceptualization" [2], 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 [3]. Manual construction of ontology offers the advantage of precision and structure, but takes a lot of time. In recent years, LLM-based (semi) automatic ontology construction has gained popularity [3]. It refers to a (part of) or the complete ontology construction 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 paper 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 construction. This hybrid method could increase the efficiency of ontology construction while maintaining the structure during the ontology construction process. The drinking water distribution network (DWDN) was chosen as a case study to test the

EKAW 2024 Workshops, Tutorials, Posters and Demos, 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2024), November 26-28, 2024, Amsterdam, The Netherlands.

y1kelinblue@gmail.com (Y. Huang); e.karabulut@uva.nl (E. Karabulut); v.o.degeler@uva.nl (V. Degeler)
0000-0003-2710-7951 (E. Karabulut); 0000-0001-7054-3770 (V. Degeler)

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 construction method?***

The remainder of this paper is organized as follows: First, a literature review on ontology construction and existing ontology construction 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 construction 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 construction methods [4, 5]. In order to find the best method that could be used as a framework for (semi) automatic ontology construction, eight evaluation criteria, either derived from the original paper [4] (domain analysis, level of detail, evaluation, documentation, maintenance, reusability, sample application) or tailored to this paper (domain specificity), were selected to assess the quality of the manual methods.

Two methodologies stand out after the evaluation: the UPON Lite Ontology [6] and the NeOn Methodology [7]. 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 Lite Ontology provides direct and concise instructions for ontology construction. Given the context of this study, the UPON Lite 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 [8] and the product development [9].

2.2. (Semi) Automatic ontology construction

Zulkipli et al. [10] 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[1].

Most methodologies contain detailed descriptions and clearly outline the automated ontology construction process. They all used various tools to facilitate automation, such as NLP [11, 12] and the RelExOnt algorithm [13]. There are also a variety of input data formats, but most are plain text, such as databases and websites, and several methods use NLP for automation.

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" [14]. It is often associated with ontology engineering and machine learning.

[1] https://github.com/DiTEC-project/Large-Language-Model-for-Ontology-Learning-In-Drinking-Water-Distribution-Network-Domain

The LLMs are already used in various tasks that are relevant to different aspects of ontology construction such as ontology matching, and competency question generation [15, 16], they could work as an assistant in this process [17, 18]. However, none of them addresses the problem of ontology construction directly with LLMs.

ChatGPT performs well in the task of term typing, recognizing a type taxonomy, and discovering non-taxonomic relations between types [17]. Besides ChatGPT, the open source models such as 'Flan-T5' also demonstrates ability to perform these tasks [17]. Furthermore, LLM demonstrates exceptional capability in translating natural language sentences into description logic [19].

2.3.1. 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 [20]. 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 [21], which has the potential to mitigate this risk by augmenting the generative AI with facts from 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].

3. Use case: Drinking water distribution network

A drinking water distribution network (DWDN) [22] 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[2]. The DWDN serves several purposes such as providing water for households, supplying water for firefighting and supporting industrial processes [22].

There is currently no existing DWDN ontology, as the currently published ontologies in the water networks domain focus on the drinking water quality [23, 24] or water resources management [25] rather than the DWDN itself. EPANET is a widely used software for modeling and analysis of DWDNs[3]. It includes various data that 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 as an external source for performing the RAG method.

4. Methodology

This section outlines the complete ontology construction process, including the RAG construction, the strategy of query engineering and the pipeline. The final goal is to construct an ontology and represent it in OWL. The flowchart is available in Table 1.

4.1. Framework of the RAG

The RAG enables the LLMs to generate contextual responses, and it is the core framework of ontology construction in this paper. This paper implements the Basic RAG, which could be segmented into three parts. The LangChain library from Hugging Face is employed[4].

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

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

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

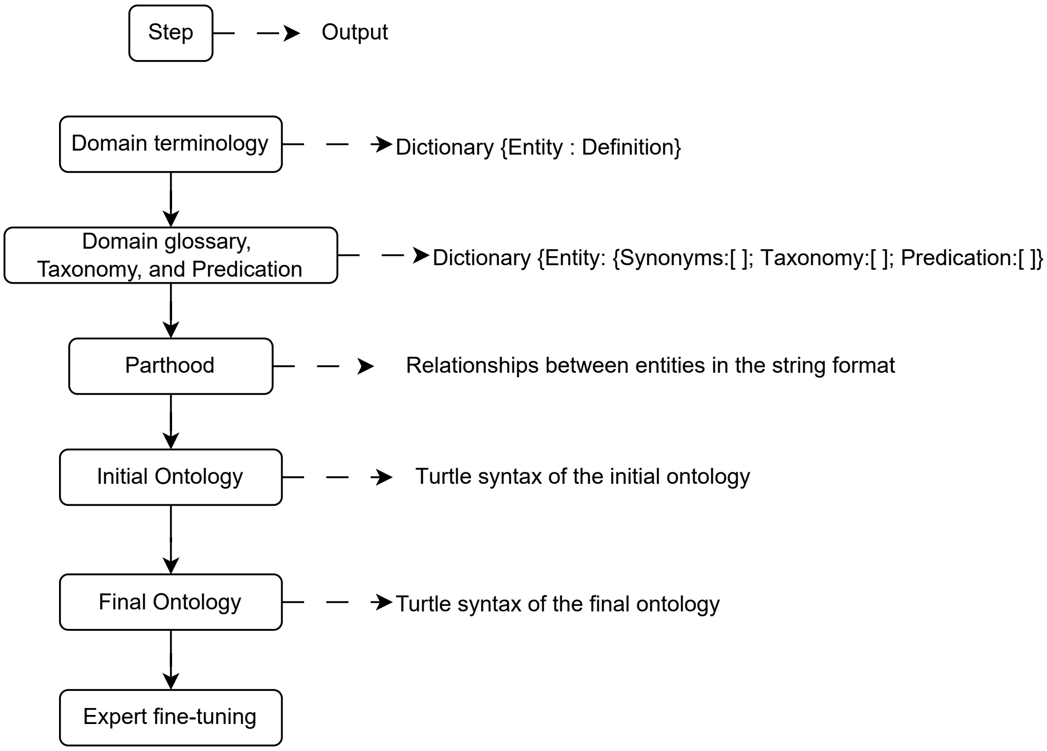
Step
Output
Domain terminology
Dictionary {Entity : Definition}
Domain glossary, Taxonomy, and Predication
Dictionary {Entity: {Synonyms:[ ]; Taxonomy:[ ]; Predication:[ ]}
Parthood
Relationships between entities in the string format
Initial Ontology
Turtle syntax of the initial ontology
Final Ontology
Turtle syntax of the final ontology
Expert fine-tuning

Figure 1: The proposed approach is based on the RAG, and queries are generated from the UPON Lite ontology.

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 OpenAI embedding model – "OpenAIEmbeddings". In our use case, the resource is coming from EPANET. This resource comprehensively covers both physical and non-physical components in the water networks, as well as simulation models for water flow.

Retrieval Information retrieval is the process of extracting relevant information from a given input document based on a user query [26]. 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 [27]. The search arguments define K to set the retriever to return the top K chunks based on the MMR score. It's a trade-off between the diversity and relevance of the result. Typically, the value of K falls within the range of 5 to 10, depending on the size of the divided chunks. Based on some small experiments, the result is better when k is 10, therefore, k was set to 10.

The LLMs sometimes produce results with errors. The algorithm is configured to attempt retrieval up to three times per query. If the first attempt fails, the algorithm will retry the same query in the same block. If the second attempt also fails, it will make a third and final attempt.

Table 1 summarizes for all the mentioned hyperparameters to ensure reproducibility.

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 paper: 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 also 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,

Table 1
Hyper-parameters

Retriever Parameter Settings	**Values**
Search_type	"mmr"
Search_kwargs	10
Max_retries	3
LLM Parameter Settings	**Values**
7b- beta: Max_length	16384
7b- beta : Max_new_tokens	8096
All LLMs: Temperature	0.5
Chunks Settings	**Values**
Chunk_size	256
Chunk_overlap	20

and a user query specified for each step. 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 paper are based on the UPON Lite 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 Lite 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 create the query in a specific way so that the LLM returns an answer in the desired format.

Prompt engineering, or querying is about creating the "recipe" to guide the LLMs to perform the desired task [28]. There are several papers that have introduced query patterns to enhance prompt engineering [29, 30], but these identified patterns are all tailored for OpenAI models[6]. After experimenting with a promising template pattern [29]: *"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.

The original query templates were created, tested, and adjusted based on the outputs. Placeholders are usually used in the query 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. During this stage, the existing queries are fine-tuned based on the output from the different

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

models, by adjusting the content details and changing the position of the content within the query. 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. Although the answers in the last phase already possess some initial structure and quality, it's crucial to refine them to further optimize the results.

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, the final ontology step was introduced to give LLMs more room and time to generate the final Turtle syntax, thereby enriching the initial ontology that only contains entities and their relationships.

4.3. Query pipeline

This subsection provides an overview of the query, including its structure, purpose, and components. The queries are derived from the UPON Lite ontology.

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.

Domain glossary, taxonomy, and predication The second step is to determine the domain glossary (synonyms), taxonomy, and predication (property) 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 different entities. 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 have difficulties to generate correct ontology Turtle syntax without explicit instruction, which is not provided by the UPON Lite Ontology. Therefore, this query is tailored by us without reference to the UPON Lite 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 for each of the six steps. It is a generic fine-tuning step and is not specific to, e.g., error correction.

4.4. Query Code

This subsection details the queries sent to the LLMs at each step.

4.4.1. Domain terminology

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.

4.4.2. Domain glossary, taxonomy, and predication

We are creating the ontology in the drinking water distribution network domain. Steps 2, 3, and 4 involve creating Synonyms, Taxonomy, and Predication for the input entity: {ResultQ1} based on the step descriptions. There should not be any repeated answers between steps 2, 3, and 4. A term can be either a synonym, a taxonomical term, or a predicate.
Step 2: Provide 0-2 domain glossary (synonyms) for the input entity. The terms of the lexicon are associated with a textual description, indicating also possible synonyms; Having produced a first lexicon, you could, in this step, enrich it by associating a textual description with each entry. You can enrich the lexicon by associating a textual description with each entry.
Step 3: Provide taxonomy for the input entity. Domain terms are organized in a generalization/specialization (ISA) hierarchy; The first is a taxonomy based on the specialization relation, or the ISA relationship connecting a more specific concept to a more general one (such as invoice ISA business document). You must not only identify ISA relations between existing terms but also introduce more

abstract terms or generic concepts seldom used in everyday life that are extremely useful in organizing knowledge. During this step, you thus provide feedback to the two previous knowledge levels—lexicon and glossary—since taxonomy building is also an opportunity to validate the two previous levels and extend them with new terms. You must find a good balance between the breadth of the taxonomy, or average number of children of intermediate nodes, and its depth, or levels of specialization and the granularity of taxonomy leaves.
Step 4: Provide predication (CP, AP, RP) for the input entity. Terms representing properties from the glossary are identified and connected to the entities they characterize; This step is similar to a database design activity, as it concentrates on the properties that, in the domain at hand, characterize the relevant entities. You generally identify atomic properties (AP) and complex properties (CP). The former can be seen as printable data fields (such as unit price), and the latter exhibit an internal structure and have components (such as address composed of, say, street, city, postal code, and state). Finally, if a property refers to other entities (such as a customer referred to in an invoice) it is called a reference property (RP). In a relational database, an RP is represented by a foreign key. The resulting predicate hierarchy is organized with the entity at the top, and then a property hierarchy below it, where nodes are tagged with CP, AP, and RP.
Here's how you can structure it:
{"Entity1":{ "Synonyms": ["synonym1"],
"Taxonomy":
"term1": ["subterm1"],
"Predication": ["property1"]
# Add more or delete properties as needed }}

4.4.3. Parthood

We are creating the ontology in the drinking water distribution network domain. Step 5 involves relationship mapping for the input entities: ResultQ1. Step 5: Parthood (meronymy). Complex entity names connected to their components, with all names needing to be present in the glossary; This step concentrates on the 'architectural' structure of business entities, or parts of composite entities, whether objects, processes, or actors, by eliciting their decomposition hierarchy (or part-whole hierarchy). To this end, you would analyze the structure and components an entity exhibits, creating the hierarchy based on the partOf (inverse hasPart) relationship. Parthood can also be applied to immaterial entities (such as a regulation subdivided into sections and articles or a process subdivided into sub-processes and activities). Please identify and map the clear relationships between entities and ensuring that no conflicting relationships exist. For example, avoid situations where entity A is considered a part of entity B while simultaneously entity B is also considered a part of entity A. You don't need to provide the explanation. You can structure it like this: Entity: Relationship: Entity.

4.4.4. Initial ontology

We are creating the ontology in the drinking water distribution network domain. Step 6 involves creating the ontology schema based on the input: {ResultQ5}.
Step 6: Please produce the formally encoded ontology by using the Web Ontology Language, or OWL, based on this input.
When constructing an ontology schema, follow these steps:
1) Define prefixes for readability.
2) Create classes to represent entities.
3) Organize classes hierarchically using subclass relationships.
Please return the turtle syntax encompassing all classes and their relationships, excluding any explanatory text. Here is an example of ontology schema in another domain:
# Define prefixes
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns> .

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema> .
@prefix owl: <http://www.w3.org/2002/07/owl> .
@prefix ex: <http://example.org/ontology> .
# Create classes and relationships
ex:Animal rdf:type owl:Class .
ex:Mammal rdf:type owl:Class;
rdfs:subClassOf ex:Animal.
```

4.4.5. Final ontology

We are creating the ontology in the drinking water distribution network domain. Step 7 is ontology finalization, which integrates the knowledge gathered in previous steps: step 2 (Synonyms), step 3 (Taxonomy), and step 4 (Predication). The results from these steps are stored in the dictionary {ResultQ234}. This is in the ontology schema: {ResultQ6}.
Your task is formally encoding the previous result and combining it with the provided ontology schema. When generating the answers, you need to keep everything from the ontology schema, but you don't need to provide any explanation. You should provide a complete ontology by repeating these steps:
1. Identify the key of the input, which represents the entity in the ontology, and use it as the class name in the turtle syntax.
2. Define equivalent classes (e.g. equivalentClass) for each entity based on synonyms. Two classes may be stated to be equivalent.
3. Incorporate the taxonomy of each entity as relationships (e.g. rdfs: subClassOf). Class hierarchies may be created by making one or more statements that a class is a subclass of another class.
4. Define properties for each entity based on predication. Properties can be used to state relationships between individuals or from individuals to data values. If there are repetitive properties between entities, you can simply add information on top of existing properties rather than creating duplicates. Here is an example of encoded information in turtle syntax:

```
ex:Person rdf:type owl:Class.
# Equivalent Classes
ex:Individual rdf:type owl:Class ;
owl:equivalentClass ex:Person.
# Taxonomy Relationships
ex:Employee rdf:type owl:Class ;
rdfs:subClassOf ex:Person.
# Properties
ex:hasChild rdf:type owl:ObjectProperty ;
rdfs:domain ex:Person.
```

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

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

included among these frequent terms, except in the response from 7b-beta. Below is the example answer returned by 7b-beta of this step:

{'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 as printable data fields - atomic properties (AP) , internal structure and components - complex properties (CP) and the property refer to other attributes - reference properties (RP) [6]. 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 Table 2.

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 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 successful in generating Turtle syntax, they showed inconsistent stability and errors such as repetitive class creation or incomplete input information. The example result is provided in Table 2.

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. Examples from this model are illustrated in Figure 2 which presents the relationship between the main line (Large-diameter pipe) and water distribution network. Figure 3 shows the properties of the entity fire hydrant.

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 a property and hydrant as an entity, while essential entities such as water demand were missing. Meanwhile, not all entities have a definition and one domain expert mentioned that this is a challenge for them to understand them. 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 3.

In this paper, the overall performance of the models is qualitatively ranked along three dimensions: time (the duration required to generate the answer), scalability (the length of returned final ontology), cost (whether the use of the LLM is free or incurs a cost). The gpt-4-turbo has best performance in terms of cost-effectiveness and scalability. gpt-4-turbo is followed by gpt-4. The 7b-beta model is characterized

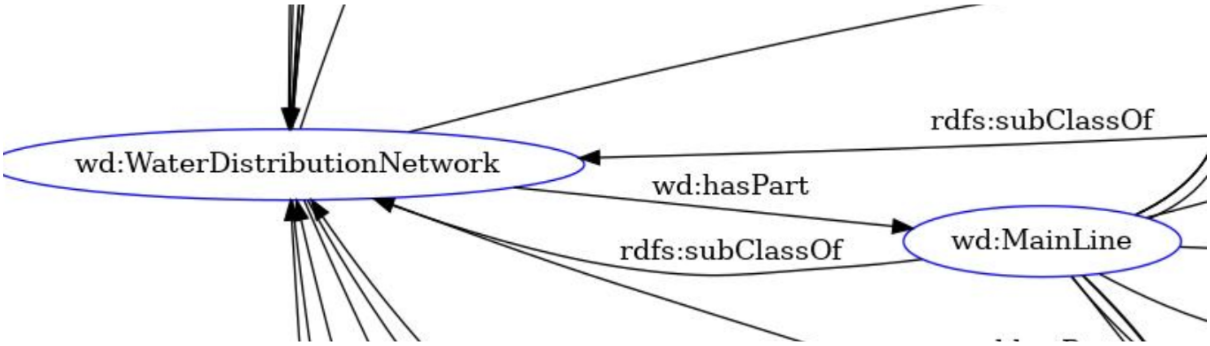

Figure 2: First zoom-in example of the final ontology

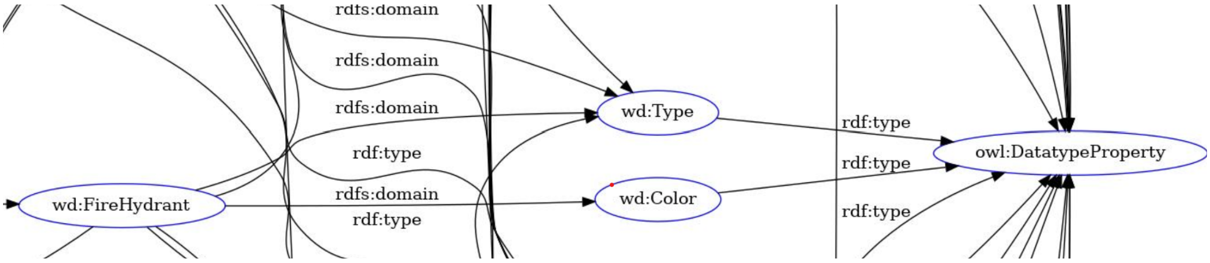

Figure 3: Second zoom-in example of the final ontology

Table 2
Gpt-4-turbo example answers of Domain Glossary, Taxonomy, Predication; Parthood; and Initial ontology

Domain Glossary, Taxonomy, and Predication	
Pipe	
Synonyms	Conduit, Tubing
Taxonomy	Water Transport Infrastructure: Pipe
Predication - AP - CP - RP	 Material, Diameter, Length Installation Information Connected to Pump, Connected to Valve
Parthood	
Water Distribution Network	
hasPart	Pipe
hasPart	Pump
Initial Ontology	
	`ex:WaterDistributionNetwork rdf:type owl:Class.`
	`ex:Pipe rdf:type owl:Class.`
	`ex:WaterDistributionNetwork rdfs:subClassOf [`
	`  owl:hasPart ex:Pipe, ex:Pump, ex:Valve, ex:Reservoir,`
	`  ex:WaterTower, ex:Meter, ex:Hydrant, ex:PressureRegulator,`
	`  ex:BackflowPreventer, ex:ServiceLine ].`

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 Figure 4. 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

	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
Similarity	0.39	0.48	0.54	0.5

Table 3
Comparison of Model Performance

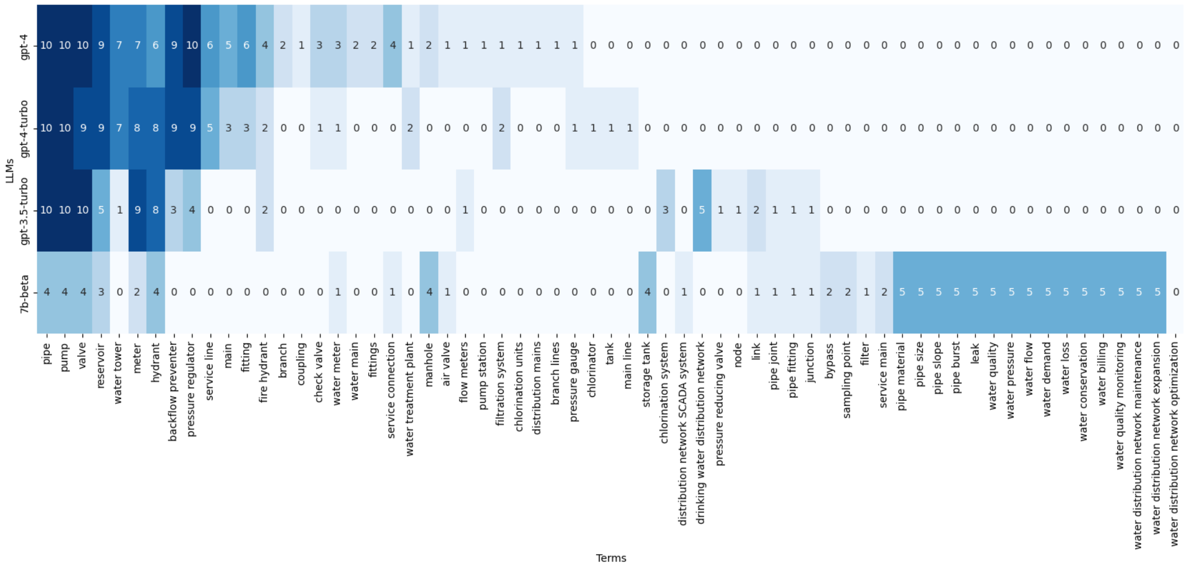

Figure 4: Frequency of terms of each LLM

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 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 from the water network company Vitens, as a senior data scientist 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.

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 the best answer by comparison was selected 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 feedback point 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 paper 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 paper 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.

Despite these constraints, future work could focus on evaluations. In selecting methods, multiple criteria are being developed to choose the best ontology construction methods. These criteria are also work in progress, standard rules about criteria selection could be introduced in the future. Additionally, the current result evaluation process lacks objectivity. For example, the time is not precisely measured by the timer. In the future, multiple domain experts should be incorporated in the evaluation process, and evaluation criteria should be quantified.

7. Conclusion

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

This paper 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 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.

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 increases the likelihood of achieving a better result. In this paper, 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.

Acknowledgements. This work was supported by The Dutch Research Council (NWO), in the scope of the Digital Twin for Evolutionary Changes in water networks (DiTEC) project, number 19454.

References

[1] C. Filippo, G. Vito, S. Irene, B. Simone, F. Gualtiero, Future applications of generative large language models: A data-driven case study on chatgpt, Technovation 133 (2024) 103002.

[2] T. Gruber, What is an ontology, 1993.

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

[4] A. Sattar, E. S. M. Surin, M. N. Ahmad, M. Ahmad, A. K. Mahmood, Comparative analysis of methodologies for domain ontology development: A systematic review, International Journal of Advanced Computer Science and Applications 11 (2020).

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

[6] A. De Nicola, M. Missikoff, A lightweight methodology for rapid ontology engineering, Communications of the ACM 59 (2016) 79–86.

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

[8] K. Farghaly, R. K. Soman, W. Collinge, P. Manu, M. H. Mosleh, C. Cheung, Construction safety ontology development and alignment with industry foundation classes (ifc), Electronic Journal of Information Technology in Construction 27 (2022) 94–108.

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

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

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

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

[13] N. Kaushik, N. Chatterjee, Automatic relationship extraction from agricultural text for ontology construction, Information processing in agriculture 5 (2018) 60–73.

[14] A. Maedche, S. Staab, Ontology learning, in: Handbook on ontologies, Springer, 2004, pp. 173–190.

[15] I. Horrocks, A language model based framework for new concept placement in ontologies (2024).

[16] Y. Rebboud, L. Tailhardat, P. Lisena, R. Troncy, Can llms generate competency questions?, in: ESWC 2024, Extended Semantic Web Conference, 2024.

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

[18] V. K. Kommineni, B. König-Ries, S. Samuel, From human experts to machines: An llm supported approach to ontology and knowledge graph construction, arXiv preprint arXiv:2403.08345 (2024).

[19] P. Mateiu, A. Groza, Ontology engineering with large language models, arXiv preprint arXiv:2307.16699 (2023).

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

[21] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[22] N. R. Council, D. on Earth, L. Studies, W. Science, T. Board, C. on Public Water Supply Distribution Systems, Assessing, R. Risks, Drinking water distribution systems: Assessing and reducing risks, National Academies Press, 2007.

[23] S. Shahsavani, A. Mohammadpour, M. R. Shooshtarian, H. Soleimani, M. R. Ghalhari, A. Badeenezhad, Z. Baboli, R. Morovati, P. Javanmardi, 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 (2023) 35.

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

[25] P. Escobar, M. d. M. Roldán-García, J. Peral, G. Candela, J. Garcia-Nieto, An ontology-based framework for publishing and exploiting linked open data: A use case on water resources management, Applied Sciences 10 (2020) 779.

[26] N. T. W. Khin, N. N. Yee, Query classification based information retrieval system, in: 2018 International conference on intelligent informatics and biomedical sciences (ICIIBMS), volume 3, IEEE, 2018, pp. 151–156.

[27] J. Carbonell, J. Goldstein, 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, 1998, pp. 335–336.

[28] V. Liu, L. B. Chilton, Design guidelines for prompt engineering text-to-image generative models, in: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, 2022, pp. 1–23.

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

[30] L. Giray, Prompt engineering with chatgpt: a guide for academic writers, Annals of biomedical engineering 51 (2023) 2629–2633.