Appendices

A Full Queries to the LLM

QueryQ234_all: 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 }}

QueryQ5: We are creating the ontology in the drinking water distribution network domain. Step 5 involves relationship mapping for the input entities:

Table 5: (Semi) automatic methods evaluation

Methods	(Semi) auto	C1	C2	C3	C4	C5	C6	C7	C8
[5]	Auto	5	Y	Very Clear	NLP	Not Applicable	Y	biomedical literatures	Y
[16]	Auto	4	Y	Clear	X20WL tool	Yes	Y	XML file associated with graph schema	Y
[3]	Auto	5	Y	Clear	Automatic Extraction Dataset System (AEDS) tool, NLP, Protégé 5.0.0 software / Protégé IDE	No, unable to find; Not applicable; Yes	N	webpages	Y
[21]	Auto	5	N	Clear	word2vec	Not Applicable	N	External data base	Y
[28]	Auto	4	Y	Clear	LDA	Not Applicable	N	Textual document	Y
[14]	Auto	4	N	Clear	KG generator, Dbpedia, Association Rule Mining	Yes; Yes; Not Applicable	N	unstructured text corpus	Y
[7]	Auto	4	Y	Clear	Text2Onto, SimMetrics	Yes, Yes	Y	BiblioDem	Y
[25]	Auto	5	N	Very Clear	RelExOnt Althorithm	No, unable to find	Y	texts on agriculture domain	Y
[35]	Auto	4	N	Clear	NLP	Not Applicable	N	Web dictionaries	Y
[13]	Auto	4	N	Clear	Name Entity Recognizer (NER), HTML Parser	Not Applicable	N	Website	Y
[26]	Auto	3	N	Unclear	Apache Jena	Yes	N	Data from job portals with the key, value based extraction technique	Y
[43]	Semi-auto	5	Y	Very Clear	Fact++ and celfie	Yes, Yes	Y	Previous data base in the domain	Y
[11]	Semi-auto	4	Y	Clear	Ont + CABASC, Ont + LCR-Rot-hop	Not Applicable, Not Applicable	N	6 free books	Y
[23]	Semi-auto	4	N	Clear	Sebanca	No, unable to find	N	Textual document	Y
[24]	Semi-auto	5	Y	Very Clear	part-of-speech (POS), Cmap Tools, Protégé	Not Applicable; Yes, Yes	N	Textual document	Y
[36]	Semi-auto	5	Y	Very Clear	RCA, FCA	Not Applicable	Y	Arabic text	Y
[43]	Semi-auto	3	N	Clear	NLP and Crowdsourcing	Not Applicable	Y	academic papers	Y
[37]	Semi-auto								
[12]	Semi-auto	3	Y	Unclear	Machine learning algorithms	Not Applicable	N	financial data	Y

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.

QueryQ6: 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.

QueryQ7: 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.

B Examples of Results

Domain glossary, taxonomy and predication

gpt-3.5-turbo:
Pipes:
Synonyms: ["Tubes", "Conduits"]
Taxonomy: "Transportation": ["Pipes"]
Predication: ["Material", "Diameter", "Length"]

7b-beta
Pipe:
Synonyms: ["conduit", "water transport system"]
Taxonomy: "Drinking water distribution network": ["Water transport system", "Pipe"]
Predication:
- "Material" (AP): ["PVC", "Steel", "Cast iron", "Asbestos cement"]
- "Diameter" (AP): ["mm"]
- "Length" (AP): ["m"]
- "Flow rate" (AP): ["L/s"]

gpt-4
Pipes:
Synonyms: ["Conduits", "Aqueducts"]

Taxonomy: "Water Transport Infrastructure": ["Pipes"]
Predication:
- "AP": ["Material", "Diameter", "Length"]
- "CP": ["Location (composed of Street, City, Postal Code)"]
- "RP": ["Connected to Reservoirs", "Connected to Pump Stations"]

---

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"]

Parthood

---

gpt-3.5-turbo
- Pipes: partOf: Pump Stations
- Pipes: partOf: Reservoirs

7b-beta
- Pipe: partOf: Reservoir (when connected to a reservoir)
- Pipe: partOf: Pump (when connected to a pump)

---

gpt-4
- Pipes: partOf: Drinking Water Distribution Network
- Valves: partOf: Drinking Water Distribution Network

---

gpt-4-turbo
1. Water Distribution Network: hasPart: Pipe
2. Water Distribution Network: hasPart: Pump

Turtle Syntax of Initial Ontology

---

gpt-3.5-turbo
```
dwdn:Pipes rdf:type owl:Class;  dwdn:Pipes rdfs:subClassOf
dwdn:DrinkingWaterDistributionNetwork.
```

7b-Beta
```
dwdn:Pipe rdf:type owl:Class; dwdn:Pipe rdfs:subClassOf dwdn:Entity;
dwdn:Pipe dwdn:partOf dwdn:Reservoir;
dwdn:Pipe dwdn:partOf dwdn:Pump; dwdn:Pipe dwdn:partOf dwdn:Valve;
dwdn:Pipe dwdn:partOf dwdn:Junction; dwdn:Pipe dwdn:partOf dwdn:Manhole.
```

---

gpt-4
```
dw:PumpStations rdf:type owl:Class ; dw:PumpStations rdfs:subClassOf
```

```
dw:DrinkingWaterDistributionNetwork.
```

---

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  ].
```

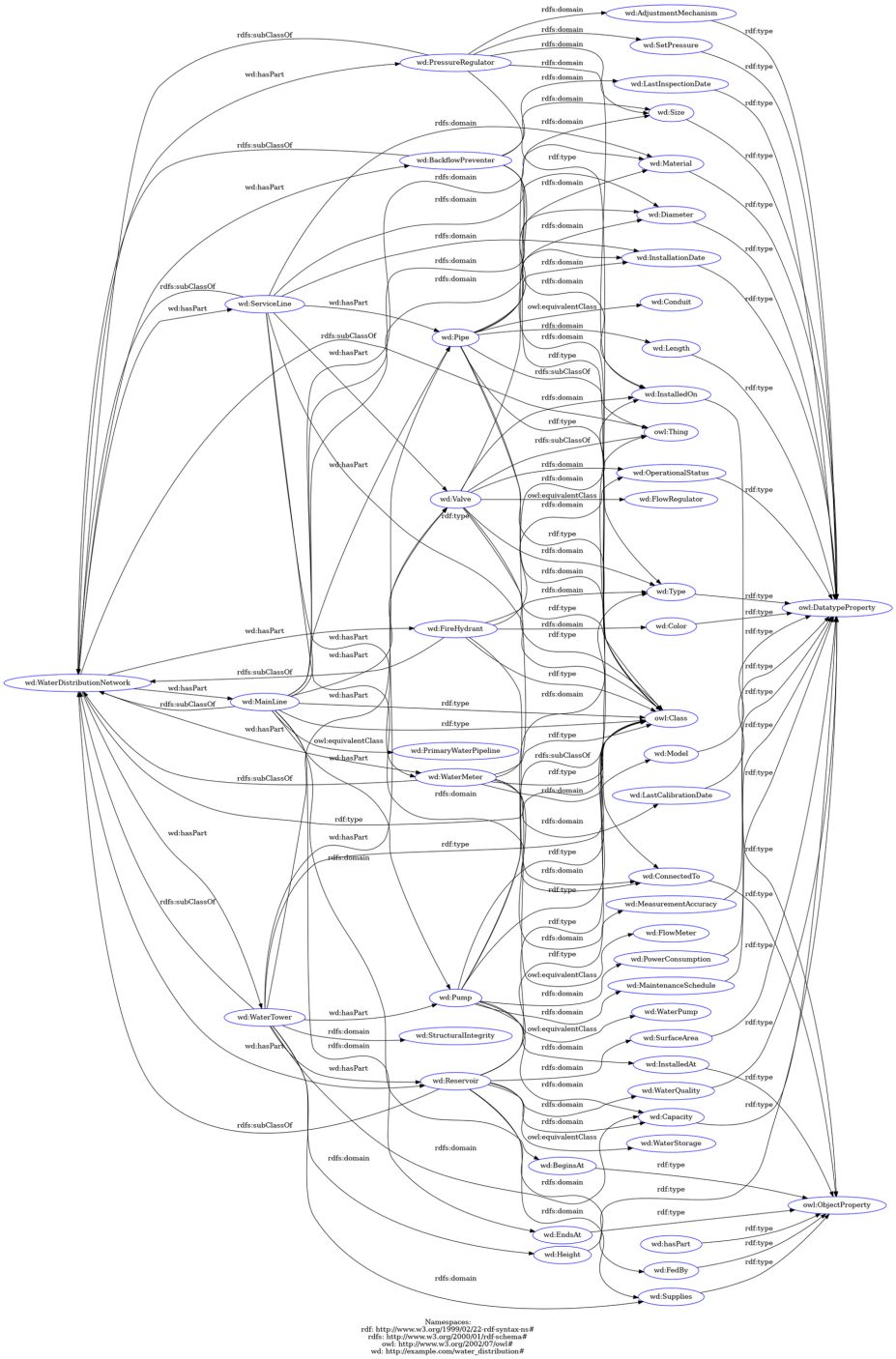

Fig. 4: Final ontology generated by gpt-4-turbo

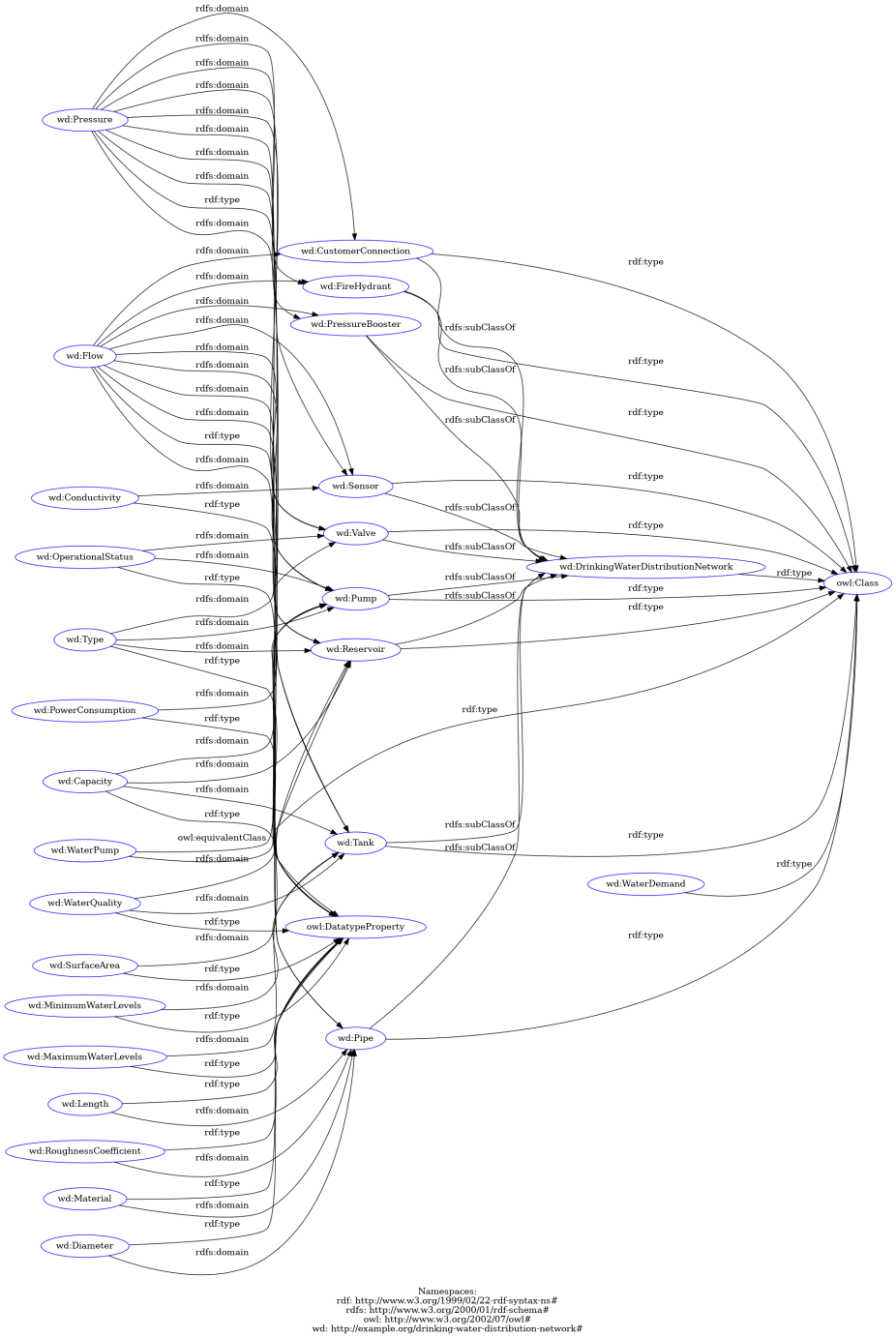

Fig. 5: Finetuned Ontology