Domain Knowledge Distillation from Large Language Model: An Empirical Study in the Autonomous Driving Domain

Yun Tang
yun.tang@warwick.ac.uk

Antonio A. Bruto da Costa
antonio.bruto-da-costa@warwick.ac.uk

Xizhe Zhang
Jason.Zhang@warwick.ac.uk

Irvine Patrick
patrick.irvine@warwick.ac.uk

Siddartha Khastgir
S.Khastgir.1@warwick.ac.uk

Paul Jennings
Paul.Jennings@warwick.ac.uk

***Abstract*—Engineering knowledge-based (or expert) systems require extensive manual effort and domain knowledge. As Large Language Models (LLMs) are trained using an enormous amount of cross-domain knowledge, it becomes possible to automate such engineering processes. This paper presents an empirical automation and semi-automation framework for domain knowledge distillation using prompt engineering and the LLM ChatGPT. We assess the framework empirically in the autonomous driving domain and present our key observations. In our implementation, we construct the domain knowledge ontology by "chatting" with ChatGPT. The key finding is that while fully automated domain ontology construction is possible, human supervision and early intervention typically improve efficiency and output quality as they lessen the effects of response randomness and the butterfly effect. We, therefore, also develop a web-based distillation assistant enabling supervision and flexible intervention at runtime. We hope our findings and tools could inspire future research toward revolutionizing the engineering of knowledge-based systems across application domains.**

***Index Terms*—large language model, domain ontology distillation, autonomous driving**

I. Introduction

Large language models (LLMs), such as GPT-3 [1], Codex [2], and ChatGPT [3] have made remarkable progress. Trained using an enormous amount of indiscriminate data from the entire internet, these LLMs embed knowledge from different domains, which are thus capable of answering questions, writing codes, drawing pictures, or translating languages across application areas [4]–[6]. In this paper, we aim to investigate if and how the knowledge of a specific application domain, e,g., scenario-based testing of autonomous vehicles, can be extracted to facilitate subsequent tasks, e.g. automatic testing scenario generation.

Safety verification and validation (V&V) of autonomous vehicles (AVs) are challenging due to the complexity of the AVs and their operating environment. Scenario-based testing of AVs [7], [8] has been a new V&V paradigm compared to distance-based approaches, where the performance of AVs is evaluated against the types of scenarios they pass instead of the countless miles they travel.

All authors are with WMG, University of Warwick, Coventry, United Kingdom.

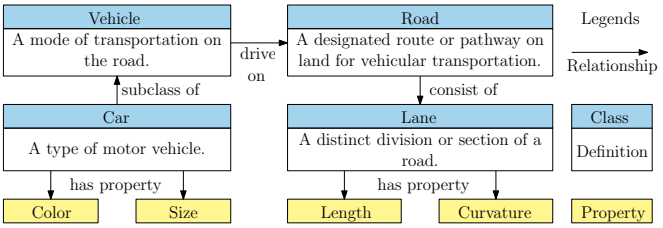

Fig. 1. Visualization of an ontology example adapted from the OpenXOntology [17], manually designed for road traffic domain.

Many scenario-generation methods have been proposed, e.g., [9]–[14]. However, those methods are mostly "parameter samplers" instead of "scenario explorers", meaning they are proposed to sample critical parameter values given a fixed list of scenario parameters toward their generation directions. Still, they cannot systematically explore different functional scenarios [15], e.g. different road networks, traffic actors, and their manoeuvres. Our recent work [16] applies *Systems Theoretic Process Analysis* (STPA) to explore different scenario types at the functional scenario level; however, such a method requires "domain knowledge" and "manual effort" extensively and hence does not scale.

Recently, to eliminate the "manual effort", combinatorial sampling-based methods are proposed to systematically generate different scenarios given a form of domain knowledge, e.g., either *Operational Design Domain* (ODD) [18] or *Ontology* [19]. The scenario ontology (Figure 1) is a form of domain knowledge aiming to encapsulate all the relevant physical entities, their relationships, as well as their associated events and activities, which thus has the potential to generate any scenario. However, no approaches have been proposed to automatically "distil" such "domain knowledge" in any form from scratch for subsequent automation tasks, such as scenario generation, until it becomes feasible with the recent progress in Artificial General Intelligence (AGI)

Manuscript 1418 submitted to 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). Received May 28, 2023.

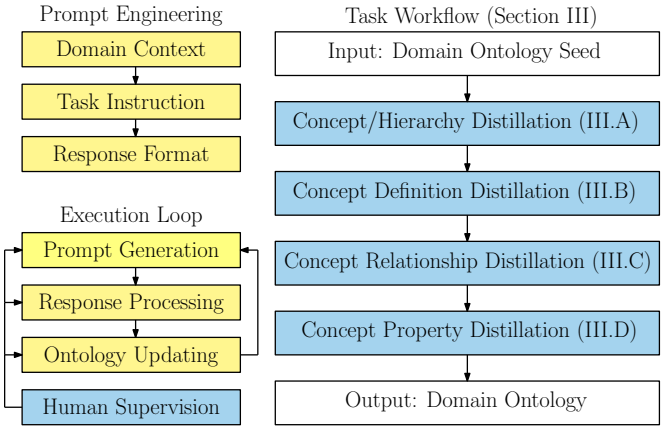

Fig. 2. Our domain ontology distillation framework with three main components, i.e., Prompt Engineering, Task Workflow and Execution Loop.

such as ChatGPT [3]. In this paper, we conduct an empirical study by "chatting" with ChatGPT and discuss our findings in constructing a driving scenario domain ontology. Our contributions are as follows:

- We are the first, to the best of our knowledge, to propose an empirical automation and semi-automation framework for domain knowledge distillation with LLMs.
- We discuss our key observations and recommendations covering the entire distillation lifecycle in depth.
- We present our web-based domain ontology distillation assistant to facilitate runtime human supervision, addressing the key challenges faced in the automatic ontology distillation experiment.

This paper is organized as follows: Section II presents an overview of our empirical distillation framework, Section III demonstrates the application of the framework in the autonomous driving domain and discusses our key observations based on the distillation results, Section IV presents our web-based distillation assistant and Section V concludes the paper.

II. Distillation Framework Overview

A. Ontology 101

The term *Ontology* is defined as the description of domain **concepts** (often referred as **classes**, e.g., *Car*, *Lane*, *Road*), the **properties** of the concepts (e.g., *Color*, *Size*, *Length*), and the **relationships** (e.g., *subclass_of*, *consists_of*, *drive_on*) between the concepts. [20]. The manual ontology construction process usually consists of the following steps (adapted from [20], [21]): 1) Define the application domain; 2) Define all the relevant classes; 3) Organize the classes in a superclass-subclass hierarchy; 4) Define the properties associated with each class; and 5) Define the relationships between each pair of classes.

In the next section, we present our empirical ontology distillation framework designed based on the required steps.

B. Framework Overview

The LLMs such as ChatGPT [3] work in a question-answer (or instruction-response) mechanism, which enables us to extract and format the knowledge via "prompt engineering". To make this empirical study beneficial to most of the public, we limit the model to the browser version of ChatGPT [3], with default generation settings (e.g., temperature [22], and numbers of tokens [23]) but provides unlimited free-tier usage. Figure 2 presents our empirical distillation framework, consisting of three main components, i.e., Task Workflow, Prompt Engineering, and Execution Loop.

Prompt Engineering When "chatting" with ChatGPT, the prompt template consists of three parts, i.e., domain context (why), task instruction (what), and response format (how). The domain context part introduces the background context for the subsequent requests, e.g., "*I have a road driving scenario ontology as shown below ...*". The task instruction part instructs the LLM on what information is expected, e.g., "*Add 10 new relevant concepts, terms or entities to the ontology ...*". Lastly, to facilitate the automated processing of the responses, the response format part specifies the machine-readable format, e.g., "*Output the new ontology in DOT format.*". Note that DOT [24] (a graph description language, examples can be found in Figure 3) is used in this study to describe the ontology class hierarchy as it is widely supported by major programming languages.

Task Workflow We start with a seed ontology of the application domain and go through a list of distillation tasks (i.e., concept/hierarchy distillation, concept definition distillation, concept relationship distillation, and concept property distillation), wherein each task we repeatedly request new knowledge from ChatGPT to augment and improve the ontology. The reasons for such a workflow design are as follows:

1) During our preliminary concept-distillation experiments, ChatGPT returns a wide range of concepts, including highly relevant, irrelevant, and sometimes duplicated ones, during the looped execution. As discussed in [5], [6], with more specific context information and good examples come improved semantic accuracy and more focused responses. Thus, we need to provide illustrative examples in the prompt to distil those highly relevant concepts while eliminating the rest. This is essential, especially for the first request, as subsequent responses highly depend on the previous results, i.e., the butterfly effect applies. As a result, we introduce the seed ontology in the first request consisting of only highly abstract concepts but still sufficient to focus the scope and demonstrate the basic ontology structure, e.g., superclass-subclass relationship, in the DOT format.

2) The basis of an ontology is formed by the concepts and the classification hierarchy (organized by the pairwise superclass-subclass relationships between concepts). As we keep updating the ontology hierarchy, the location of individual concepts in the hierarchy also changes, and so do their definitions, non-hierarchical relationships (e.g., the *drive on* relationship in *vehicles drive on roads*) and properties. As a result, the distillation tasks for the hierarchy-dependent knowledge are performed after the ontology hierarchy has been constructed and fixed.

Execution Loop In each task, there is an execution loop

Manuscript 1418 submitted to 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). Received May 28, 2023.

consisting of prompt generation, response processing and ontology updating, which continues until any stopping criterion is met, e.g., ChatGPT stops presenting new information or the ontology graph has reached a pre-defined breadth or depth. If ChatGPT returns irrelevant or erroneous results, the execution loop can be paused, repeated, reverted or resumed manually at any step to ensure satisfactory distillation results. In each step, we start a new conversation with ChatGPT instead of using the existing conversation sessions. Such a looped execution mechanism is proposed for the following reasons:

1) It is impractical to extract all the information with one request due to the limit on the maximum number of tokens [23] and occasional browser connection timeout exceptions (ChatGPT slowly generating a large body of text may encounter timeout error) per request.

2) As ChatGPT memorizes its previous requests and responses in the same conversation session, it may return similar undesirable responses as in the previous responses. Hence, we design the prompt schema to be self-sufficient and start a new conversation for each request to avoid such scenarios.

3) The looped execution mechanism improves the distillation quality and lessens the bufferfly effect by enabling manual supervision and early optimization. For example, in each step of the concept/hierarchy distillation task, instead of asking ChatGPT only to append new concepts while preserving the existing hierarchy, we request it to re-design the hierarchy from scratch considering all the concepts, explicitly requesting it to remove irrelevant concepts and merge duplicated ones. Such a step-wise re-design allows ChatGPT to optimize the hierarchy globally.

In the next section, we will cover the details of each element in the empirical distillation framework by demonstrating the application in the autonomous driving domain and discuss our key observations and challenges.

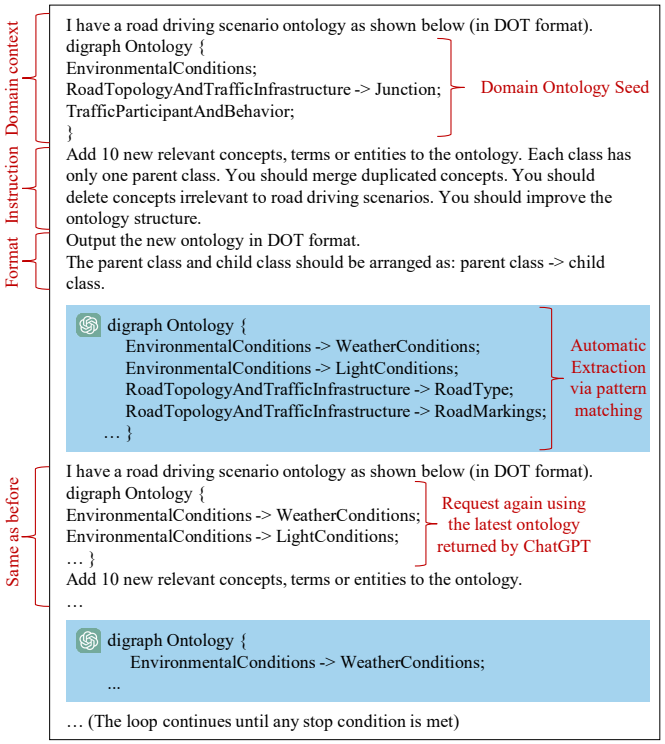

Fig. 3. Concept/hierarchy distillation task chat example.

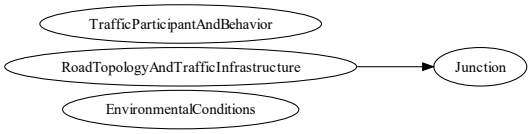

Fig. 4. The seed ontology used in the autonomous driving domain application.

III. Domain Application of the Framework

As mentioned, we apply the framework to construct an ontology in the road traffic domain for scenario-based V&V of autonomous vehicles. Although domain experts (including our team) have already designed such an ontology manually as part of the OpenXOntology framework, we still need similar ontologies for many other transportation domains. This section presents our findings for the ontology distillation lifecycle based on our team's experiences in the OpenXOntology projects.

A. Concept/Hierarchy Distillation

Figure 3 presents a looped execution example of a concept/hierarchy distillation task. We design the seed ontology (Figure 4) to include the three highly abstract seed concepts taken from OpenXOntology [17], i.e., *EnvironmentalCondition*, *RoadTopologyAndTrafficInfrastructure* and *TrafficParticipantAndBehavior*. In addition, the concept *Junction* (a subclass of *RoadTopologyAndTrafficInfrastructure*) is added intentionally as a superclass-subclass example in DOT format. During the looped execution, we use the same prompt template, only updating the ontology description part with the latest refined ontology by ChatGPT automatically extracted from the previous response. Due to limited space, we only present the distillation results after the first (Figure 5) and the tenth (Figure 6) iteration.

In the first iteration, ChatGPT correctly introduces 10 new concepts to the seed ontology in the first iteration result, i.e., "Driver Behavior', "Vehicle Type", "Pedestrian Behavior", etc. However, it gets confused by the concept name "Road Topology And Traffic Infrastructure" and generates four concepts, e.g., "Road Type" and "Road Markings", only related to "roads". As a result, it separates the "Junction" concept from its original category. This result is still considered semantically valid as the definitions of the seed concepts, including the "Road Topology" concept, are absent for ChatGPT.

In the tenth response, we have distilled many new concepts and a remarkable ontology hierarchy compared to the seed ontology. Based on the concept/hierarchy distillation process, we have the following observations:

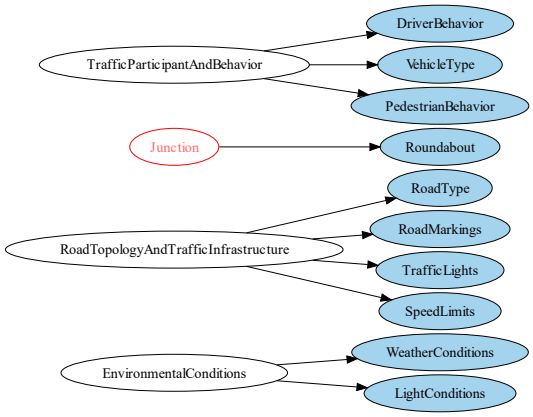

Fig. 5. Concept/hierarchy task result after the first iteration. New concepts compared to the seed ontology are highlighted in blue.

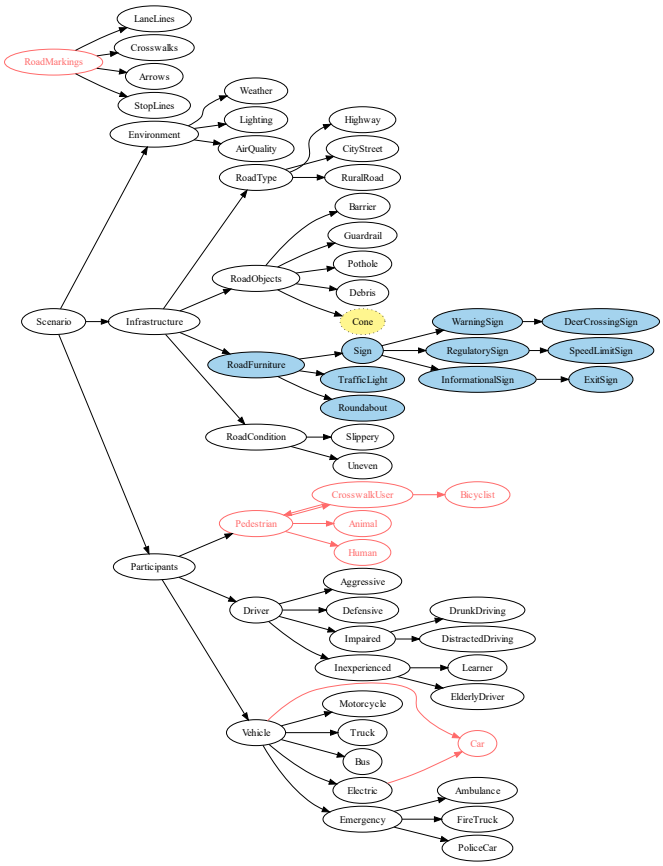

Fig. 6. Concept/hierarchy task result after 10th iteration. New concepts compared to the 9th iteration are highlighted in blue.

Observation 1: ChatGPT may delete highly relevant concepts during iteration.

The equation $C_N = C_0 + 10 \times N$, where C_N is the total number of concepts distilled after N iteration(s), does not necessarily hold as ChatGPT is specifically allowed to merge duplicate concepts and remove irrelevant concepts. Note that it removed the *Cone* concept from the ninth iteration, which we believe should not have done so as *Cones* are valid road objects. In addition, the seed concept *Junction* has been removed from the entire ontology in one of the middle iterations when it becomes "less relevant" (as ChatGPT believes) to the ontology of that iteration.

Observation 2: ChatGPT tends to return highly cohesive concepts in each response.

For example, in the tenth iteration, all the newly distilled concepts belong to the "Road Furniture" category. This behaviour is highly beneficial if, in the later stage, we would like to fine-tune a specific part of the ontology graph, for example, by asking "*Add 10 new relevant concepts under the Car category*".

Observation 3: ChatGPT starts to overlook the details in the prompt as the prompt gets longer.

With a bigger ontology graph comes longer request prompts according to our prompt engineering design, as we need to include the full ontology DOT description. As the prompt gets longer, ChatGPT starts to ignore the specific requirements. For example, we explicitly require that each concept has only one parent class (Figure 3); However, ChatGPT disobeys by setting two parents (i.e., *Vehicle* and *Electric*) for the *Car* concept (Figure 6). This might also be the reason for the undesirable removal of highly relevant concepts, e.g., *Junction*.

Observation 4: It is impractical, if not impossible, to specify all the requirements during prompt engineering.

First, prompt engineering is a closed-loop process where prompts are improved iteratively based on the previous responses. Due to the randomness in the response, the prompt engineering process is also random. For example, in trial T_1, one may need to put one constraint C_1 to fix a response issue Bug_1; while in the rest of the trials T_N, one may never encounter Bug_1 although C_1 is absent in prompts. Imagine that one has collected a considerable number (N) of bug-fixing constraints $\bigcup_{i=1}^{N} C_i$, ChatGPT will likely fail to obey all the constraints as discussed before. Moreover, lengthy constraints would shadow the ontology description content part and thus potentially result in ChatGPT overlooking some of the existing concepts or hierarchical relationships of the

ontology. On the other hand, if one limits the number of constraints in the prompt, ChatGPT will inevitably return undesirable responses. For example, we do not specify that ontology hierarchy should be acyclic. In the tenth response, a loop in the ontology graph is formed between the concepts *Pedestrian* and *Crosswalk User*. Although the DOT language permits loops in directed graphs, it is sometimes confusing and undesirable for domain ontologies.

> **Observation 5**: ChatGPT extends the ontology in a balanced (depth vs breadth) but a random way where the hierarchy grows in both breadth and depth.

Our repeated experiments show that while the ontologies of different experiment trials share many concepts, e.g., *car*, *Pedestrian*, and *Driver*, etc., those concepts have different hierarchical locations and relative distillation orders across the trails. Hierarchies across many trails are also different. This suggests we may need to repeat the automated distillation experiment until we achieve a preferable concept set and hierarchical structure.

With the above observations, we recommend a manual examination and error fixing during or at least immediately after the *Concept/Hierarchy Distillation* task. For example, the ontology after the tenth loop is manually fixed, before the subsequent *Concept Definition Distillation* task, with the following minor modifications: a) set concept *Road Markings* a sub-class of concept *Infrastructure*; b) remove the concept *Electric*; c) remove the concept *Crosswalk User* and set *Bicyclist* a sub-class of *Pedestrian*.

B. Concept Definition Distillation

Figure 7 shows a concept definition distillation example. We first present the previously distilled ontology hierarchy and ask ChatGPT to define a fixed number of concepts such that it can take the entire ontology hierarchy into consideration. This is the primary reason why we distil definitions after hierarchy. We request ten concept definitions per conversation during the experiment to avoid issues such as timeout. By default, ChatGPT prefers to format its output into a markdown table (Figure 8 (a)). However, a markdown formatting bug is often encountered during the loop execution phase, where ChatGPT keeps printing the "- - -" symbols without stopping. To address this issue and eliminate the number of tokens used for formatting purposes (i.e., tokens are wasted by the "- - -" characters), we propose to use comma-separated values (CSV) with the separator "@" as it is unlikely to appear in the concept names and definitions. The distillation loop stops until all the concepts are defined. There are 56 concepts in the revised ontology hierarchy; hence, it stops after six loop iterations.

As we initiate a new conversation (instead of continuing with the previous conversation session), ChatGPT gives the concept definition in different styles. Selected responses of different styles are listed below:

- *Bicyclist*: *Refers to a person who is riding a bicycle on or near the road.* (definition) *Bicyclists have the same rights and responsibilities as other road users and must follow the same traffic laws.* (additional information)
- *Driver*: *Driver is part of the Participants category in the road driving scenario ontology.* (superclass reference) *A driver is a person who operates a vehicle on the road.* (definition) *Drivers can be categorized by their driving behaviour or experience, such as aggressive, defensive, impaired, or inexperienced.* (subclass reference)
- *Aggressive*: *Refers to a type of driver behaviour that involves reckless or hostile actions towards other road users, such as speeding, tailgating, and weaving through traffic.* (definition with examples) *Aggressive driving increases the risk of accidents and can lead to road rage incidents.* (additional information)

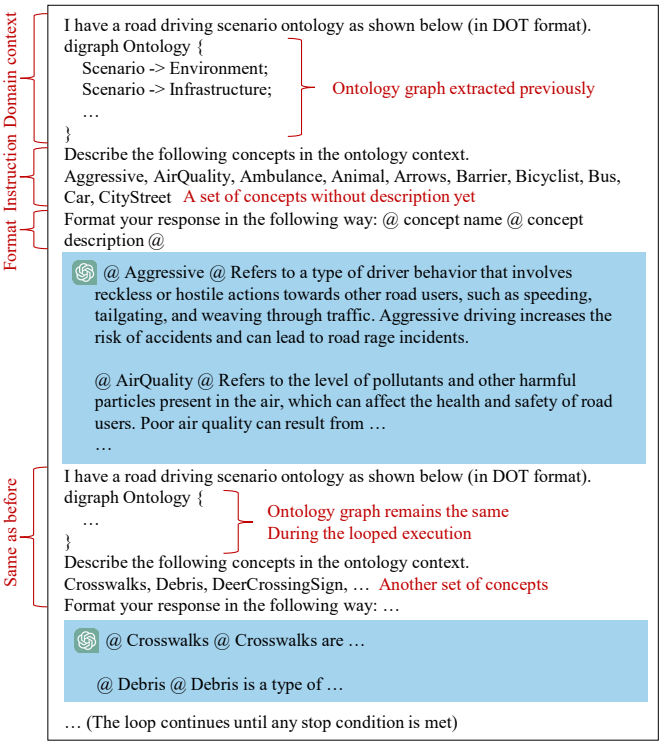

Fig. 7. Concept definition distillation task chat example

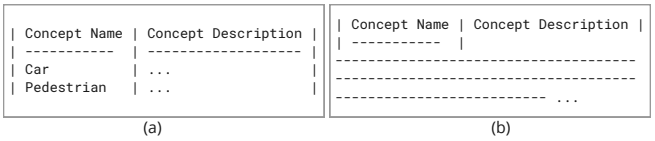

Fig. 8. ChatGPT table formatting bug: (a) normal table format output. (b) incorrect table output with abundant "- - -" causing timeout or render errors

To facilitate discussion, we label (in grey) all the sentences based on their semantic nature. We have the following observations:

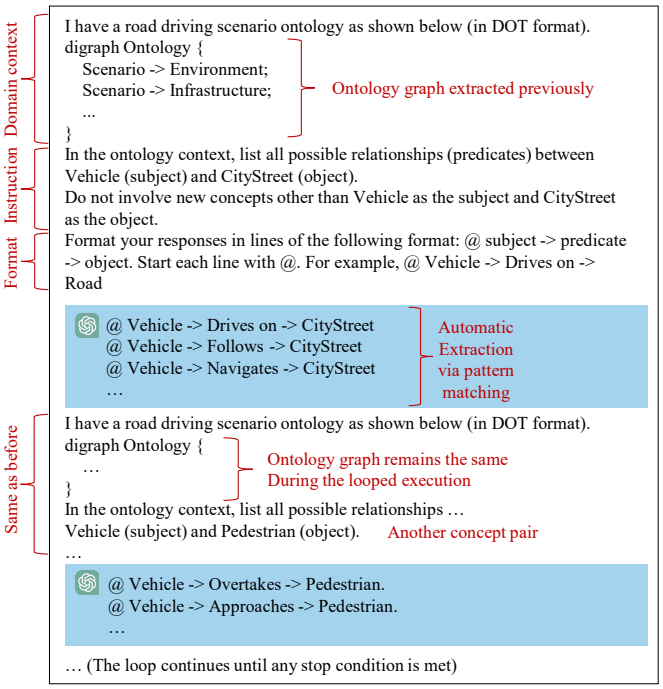

Fig. 9. Concept relationship distillation task chat example

Observation 6: ChatGPT generally defines each concept with a random mixture of key components, i.e., definition, additional information, reference to the concept's superclasses, and reference to the subclasses.

The combination styles remain coherent within the same conversation session and may vary across different conversation sessions. Our further experiments show that the combination style can be customized easily with prompt engineering, e.g., *"in the concept description, describe its definition, its relative position in the ontology hierarchy, and provide any additional relevant information"*.

Observation 7: The definitions with illustrative examples may signal further concept/hierarchy distillation.

In many concept definitions similar to the *Aggressive* concept, ChatGPT also illustrates the concept with concrete examples, while those examples are not present in the ontology yet. This indicates that ChatGPT has additional knowledge regarding the concepts, and further concept/hierarchy distillation can be conducted.

C. Concept Relationship Distillation

This section discusses the results of the non-hierarchical relationship (relationships other than superclass-subclass relationship) distillation task. We aim to distil two main types of relationships, i.e., the **inter-concept relationship** and **intra-concept relationship**. The inter-concept relationships lie between different concepts, e.g., *Vehicle* and *Road*, while intra-concept relationships exist between the same concepts, e.g., *Vehicle* and *Vehicle*. In this study, we limit the scope to pairwise relationships by explicitly specifying only two concepts in the following form: $Subject\ Concept \rightarrow Relationship\ Predicate \rightarrow Object\ Concept$, e.g., *Vehicle* $\rightarrow$ *Drives on* $\rightarrow$ *Road*.

Figure 9 shows an execution example of the relationship distillation task. We adjusted the format request to avoid erroneous responses like: "@ *Emergency* @ *Uses* @ *Ambulance* @ *to respond to incidents affected by poor* @ *AirQuality* @." when "@" is used as the delimiter. Selected relationship responses are listed in Table I. Each relationship distillation result is a union of five independent executions. We have the following observations:

Observation 8: The relationship distillations on any pairs of concepts, regardless of the concepts' relative hierarchical positions, are equally important.

Given a distilled ontology hierarchy of N concepts, there are N^2 unique ordered subject-object concept pairs. We may extract the relationship of every concept pair when N is trivial. However, as N gets bigger, such a complete iteration can become expensive. Ideally, one may consider distilling relationships of concepts at greater heights (if we consider the ontology hierarchy as a tree) only as the relationships of the superclass shall be the union of all the sub-classes relationships, e.g., $Vehicle \rightarrow Vehicle$ include many distilled relationships with $Car \rightarrow Car$ (Table I). However, in practice, this is only partially true for the following reasons:

1) The relationship distillation results are random and independent in different conversation sessions regardless of the concept being a super or sub-class in the ontology context. The results for the superclass may be a part, union, intersection, or mix of the results for the sub-classes. For example, the *Car* $\rightarrow$ *Tows* $\rightarrow$ *Car* relationship may never, although it should, be distilled for the Vehicle $\rightarrow$ *Vehicle* pair.

2) High-level super-class pairs, e.g., *Environment* $\rightarrow$ *Environment* and *Environment* $\rightarrow$ *Infrastructure* tend to induce abstract relationships, such as *Affects*, *Influences*, and *Modifies* etc. While the abstract relationships are valid, they may be less useful in applications. For example, when we design testing scenarios, the term *Influences* can be too broad as we need to specify how to *influence* in concrete scenarios.

3) Concrete sub-class pairs can potentially distil specialized relationships, e.g., *FireTruck* $\rightarrow$ *Turns on* $\rightarrow$ *TrafficLight* is highly unlikely to appear for *Vehicle* $\rightarrow$ *TrafficLight* pair.

Observation 9: The intra-concept relationship and inter-concept relationship distillation are equally important.

Depending on the nature of the concept, ChatGPT may fail to return any intra-concept relationships (e.g., *Aggressive* $\rightarrow$ *Aggressive* as it regards *Aggressive* as "a property

TABLE I
DISTILLED RELATIONSHIP EXAMPLES FOR BOTH INTRA (LEFT) AND INTER (RIGHT)-CONCEPT RELATIONSHIPS

Subject/ Object	Intra-Concept Relationship	Subject/ Object	Inter-Concept Relationship
Environment/ Environment	Affects, Influences, Determines, Modifies, Depends on, Impacts, Changes, Interacts with, Alters	Environment/ Infrastructure	Affects, Alters, Changes, Conditions, Determines, Impacts, Influences, Interacts with, Modifies, Shapes
Vehicle/ Vehicle (highlighted are new compared to Car/Car)	Accelerates past, Blocks, **Brakes suddenly in front of**, **Changes lane**, Changes lane behind, Changes lane in front of, **Changes lane to**, **Collides into**, Collides with, Competes with, **Cooperates with**, **Crosses path with**, Cuts off, Decelerates behind, Drafts behind, **Drives alongside**, **Drives in front of**, Drives next to, Follows, **Gives way to**, Honks at, Overtakes, **Parks behind**, **Parks in front of**, Parks next to, Passes, Pulls over for, Races, Races with, **Shares the road with**, Signals to, Stops behind, Swerves to avoid, Tailgates, **Turns left in front of**, **Turns right in front of**, **Yields to**	Vehicle/ TrafficLight	Approaches, Follows signal, Follows the signal of, Halts before, Ignores, Ignores the signal of, Obeys, Observes, Passes, Proceeds on green, Proceeds through, Runs, Stops at, Waits at, Waits at red, Waits for
Car/ Car (highlighted are new compared to Vehicle/ Vehicle)	Accelerates past, **Avoids**, **Avoids collision with**, Blocks, **Causes traffic jam with**, **Changes lane behind**, **Changes lane in front of**, Changes lanes behind, Changes lanes in front of, Collides with, **Comes into view of**, Competes with, **Crashes into**, **Creates gap for**, Cuts off, Decelerates behind, Drafts behind, **Drives beside**, Drives next to, **Drives past**, **Enters intersection with**, **Exits intersection with**, Follows, **Follows too closely behind**, **Follows too closely to**, **Gets cut off by**, **Gets passed by**, **Gets stuck behind**, Honks at, **Lets in**, **Merges behind**, **Merges in front of**, **Navigates around**, Overtakes, Parks next to, Passes, **Passes by**, **Passes on the left/right of**, Pulls over for, Races, **Races against**, Races with, **Rear-ends**, Signals to, **Signals to turn behind**, **Signals to turn in front of**, Stops behind, **Stops next to**, Swerves to avoid, Tailgates, **Tows**	FireTruck/ TrafficLight	Activates, Affects, Approaches, Changes direction at, Damages, Ignores, Obeys, Passes, Proceeds after stopping at, Stops at, **Turns on**
Aggressive/ Aggressive	"There are no possible relationships between Aggressive and Aggressive because Aggressive is not defined as a concept with any subtypes or attributes ..."	TrafficLight/ Vehicle	Affects, Controls, Determines, Dictates, Dictates actions of, Directs, Governs, Guides, Indicates, Influences, Interacts with, Modifies, Modifies behavior of, Regulates, Signals

dependent on other concepts". However, during inter-concept relationship distillation, ChatGPT can return relationships such as *Aggressive* → *Causes* → *Emergency*, *Aggressive* → *Negatively impacts response time of* → *Emergency* and *Aggressive* → *Interferes with the ability of* → *Emergency (to reach their destination quickly and safely)*.

Observation 10: Post-processing of the responses is often needed.

To keep the ontology concise and organized, we note that post-processing is often necessary, for example, to merge synonyms (e.g., *Races* vs *Races With* vs *Races Against*, *Affects* vs *Influences* vs *Changes*, *Tailgates* vs *Follows too closely behind*), active-passive pairs (e.g., *Passes* vs *Gets passed by*), define relationship groups (e.g., *Parks behind* vs *Parks in front of* vs *Parks next to*, *Turn left in front of* vs *Turn right in front of*) and filter unnecessary relationships (e.g., *Shares the road with*).

D. Concept Property Distillation

The property distillation task shares many common characteristics with previous tasks, e.g., the property inheritance between the superclass and subclass. Due to limited space, similar observations are not discussed in this section.

IV. WEB USER INTERFACE

Based on our empirical study results and observations, a fully automated ontology distillation process is possible. However, it may lead to unpredictable and irrelevant ontology results due to the randomness in the responses and the butterfly effect. Manual supervision and early intervention are still required to guarantee distillation quality, improve efficiency and save potential costs (e.g., from repeated trials). To facilitate this, we develop a web-based domain ontology distillation assistant as shown in Figure 10. The website has four sub-pages corresponding to the four distillation tasks. In the prompt engineering section, all the essential components are rendered as independent editable text areas for maximum flexibility, e.g., the user may change the instruction part from "*Add 10 new relevant concepts, ..., to the ontology*" to "*Add 10 new concepts under the Vehicle class*". The execution log contains the complete history of both prompts and ChatGPT's responses in each iteration. After ChatGPT's response is logged, the entire log is parsed, the ontology is updated, the visualization is refreshed, and the prompt for the next iteration will be generated. To facilitate manual supervision and early intervention, the user can then decide whether to continue the next step or make necessary adjustments to the ontology or prompt during the entire execution loop. Currently, extensive engineering effort is underway to improve the assistant tool's usability and design across transportation application domains, and we are pleased to open-source it soon.

V. CONCLUSION

This paper presents our empirical domain knowledge distillation framework using ChatGPT and discusses our observations from the framework application experiments in the autonomous driving domain. The key finding is that: 1) with proper design of prompt engineering and execution flow, fully automated domain knowledge (in the ontology format) distillation is possible. However, due to the randomness in the response and the butterfly effect, the quality of fully automated distillation results is not guaranteed. To address this, we develop a web-based assistant to enable manual supervision and early intervention at runtime. We hope our findings and tools inspire future research toward revolutionizing the engineering processes of knowledge-based systems across domains.

Manuscript 1418 submitted to 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). Received May 28, 2023.

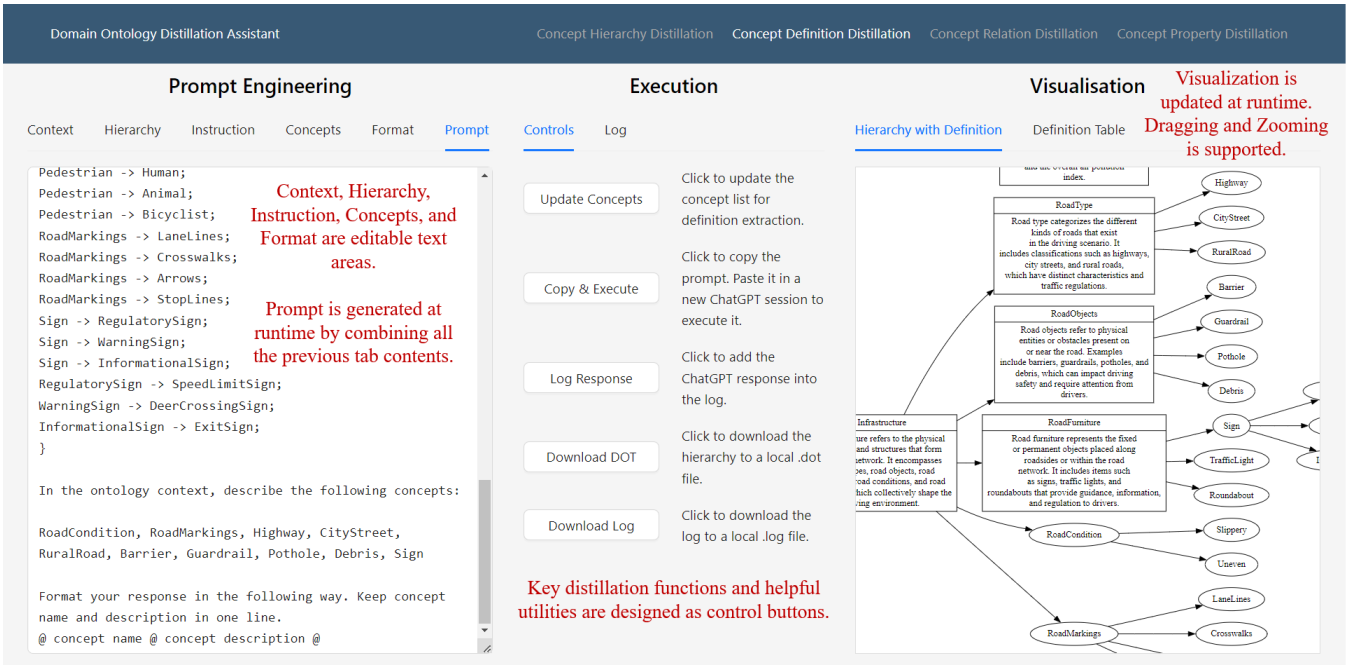

Fig. 10. Website user interface (the concept definition distillation page) of the Domain Ontology Distillation Assistant

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[3] OpenAI, "Introducing chatgpt," 2023. Accessed on Mar 28, 2023.

[4] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," 2023.

[5] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining zero-shot vulnerability repair with large language models," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1–18, IEEE Computer Society, 2022.

[6] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "Adaptive test generation using a large language model," *arXiv preprint arXiv:2302.06527*, 2023.

[7] Z. Zhong, Y. Tang, Y. Zhou, V. d. O. Neves, Y. Liu, and B. Ray, "A survey on scenario-based testing for automated driving systems in high-fidelity simulation," *arXiv preprint arXiv:2112.00964*, 2021.

[8] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y.-F. Li, L. Ma, Y. Xue, *et al.*, "A survey on automated driving system testing: Landscapes and trends," *arXiv preprint arXiv:2206.05961*, 2022.

[9] Y. Tang, Y. Zhou, F. Wu, Y. Liu, J. Sun, W. Huang, and G. Wang, "Route coverage testing for autonomous vehicles via map modeling," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11450–11456, IEEE, 2021.

[10] Y. Tang, Y. Zhou, K. Yang, Z. Zhong, B. Ray, Y. Liu, P. Zhang, and J. Chen, "Automatic map generation for autonomous driving system testing," *arXiv preprint arXiv:2206.09357*, 2022.

[11] Y. Zhou, G. Lin, Y. Tang, K. Yang, W. Jing, P. Zhang, J. Chen, L. Gong, and Y. Liu, "Flyover: A model-driven method to generate diverse highway interchanges for autonomous vehicle testing," *arXiv preprint arXiv:2301.12738*, 2023.

[12] W. Ding, C. Xu, H. Lin, B. Li, and D. Zhao, "A survey on safety-critical scenario generation from methodological perspective," *arXiv preprint arXiv:2202.02215*, 2022.

[13] Y. Tang, Y. Zhou, Y. Liu, J. Sun, and G. Wang, "Collision avoidance testing for autonomous driving systems on complete maps," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 179–185, IEEE, 2021.

[14] Y. Tang, Y. Zhou, T. Zhang, F. Wu, Y. Liu, and G. Wang, "Systematic testing of autonomous driving systems using map topology-based scenario classification," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1342–1346, IEEE, 2021.

[15] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1821–1827, IEEE, 2018.

[16] S. Khastgir, S. Brewerton, J. Thomas, and P. Jennings, "Systems approach to creating test scenarios for automated driving systems," *Reliability engineering & system safety*, vol. 215, p. 107610, 2021.

[17] ASAM, "Asam openxontology," 2023. Accessed on Apr 3, 2023.

[18] P. Weissensteiner, G. Stettinger, S. Khastgir, and D. Watzenig, "Operational design domain-driven coverage for the safety argumentation of automated vehicles," *IEEE Access*, vol. 11, pp. 12263–12284, 2023.

[19] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1813–1820, IEEE, 2018.

[20] E. F. Kendall and D. L. McGuinness, *Ontology engineering*, vol. 18;18.;. San Rafael, California: Morgan & Claypool, 2019.

[21] N. F. Noy, D. L. McGuinness, *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[22] OpenAI, "temperature - api reference - openai api," 2023. Accessed on Apr 3, 2023.

[23] OpenAI, "max_tokens - api reference - openai api," 2023. Accessed on Apr 3, 2023.

[24] Wikipedia, "Dot (graph description language) - wikipedia," 2023. Accessed on Apr 14, 2023.