

# Open-TI: Open Traffic Intelligence with Augmented Language Model

Longchao Da<sup>1</sup>, Kuanru Liou<sup>1</sup>, Tiejin Chen<sup>1</sup>, Xuesong Zhou<sup>2</sup>, Xiangyong Luo<sup>2</sup>, Yezhou Yang<sup>1</sup> and Hua Wei<sup>1†</sup>

<sup>1</sup>\*School of Computing and Augmented Intelligence, Arizona State University, 350 E Lemon St, Tempe, 85287, AZ, USA.

<sup>2</sup>School of Sustainable Engineering and the Built Environment, Arizona State University, 350 E Lemon St, Tempe, 85287, AZ, USA.

Contributing authors: [longchao@asu.edu](mailto:longchao@asu.edu); [kliou@asu.edu](mailto:kliou@asu.edu); [tchen169@asu.edu](mailto:tchen169@asu.edu); [xzhou74@asu.edu](mailto:xzhou74@asu.edu); [xluo70@asu.edu](mailto:xluo70@asu.edu); [yz.yang@asu.edu](mailto:yz.yang@asu.edu); [hua.wei@asu.edu](mailto:hua.wei@asu.edu);

†Corresponding Author.

## Abstract

Transportation has greatly benefited the cities' development in the modern civilization process. Intelligent transportation, leveraging advanced computer algorithms, could further increase people's daily commuting efficiency. However, intelligent transportation, as a cross-discipline, often requires practitioners to comprehend complicated algorithms and obscure neural networks, bringing a challenge for the advanced techniques to be trusted and deployed in practical industries. Recognizing the expressiveness of the pre-trained large language models, especially the potential of being augmented with abilities to understand and execute intricate commands, we introduce Open-TI. Serving as a bridge to mitigate the industry-academic gap, Open-TI is an innovative model targeting the goal of **Turing Indistinguishable Traffic Intelligence**, it is augmented with the capability to harness external traffic analysis packages based on existing conversations. Marking its distinction, Open-TI is the first method capable of conducting exhaustive traffic analysis from scratch - spanning from map data acquisition to the eventual execution in complex simulations. Besides, Open-TI is able to conduct task-specific embodiment like training and adapting the traffic signal control policies (TSC), explore demand optimizations, etc.

Furthermore, we explored the viability of LLMs directly serving as control agents, by understanding the expected intentions from Open-TI, we designed an agent-to-agent communication mode to support Open-TI conveying messages to ChatZero (control agent), and then the control agent would choose from the action space to proceed the execution. We eventually provide the formal implementation structure, and the open-ended design invites further community-driven enhancements.

**Keywords:** Large Language Models, Traffic Simulation, Traffic Signal Control

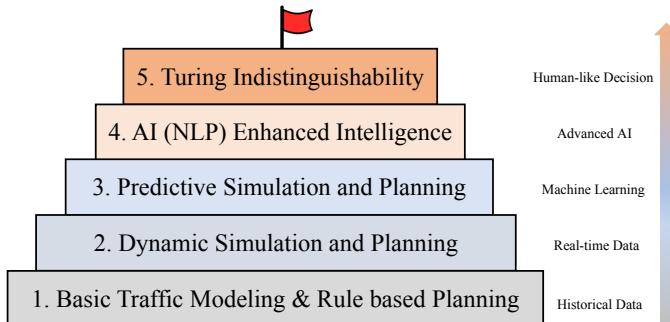
## 1 Introduction

Traffic and Transportation have played an important role in the process of human civilization. With the development of electronic and computer techniques, intelligent transportation is casting hope to further benefit people's daily lives through optimal controlling and scheduling decisions. Transportation depicts vehicles commuting between regions, providing delivery for products and humans. The efficient modern transportation comes from joint efforts from many researchers in various directions like: map modeling [1], traffic simulation [2], schedule optimization [3], etc., and still, there are multiple ongoing challenges regarding the multi-resolution traffic simulation [4], optimal traffic signal control policies [5], dynamic demand dispatch adjustment [6], etc. More specifically, when it comes to vehicle control, the intelligent traffic signal brings hope to city-scale congestion mitigation and energy saving, multiple frontier solutions have been released on different simulators, such as SUMO [7], CityFlow [8], VISSIM [9]. These simulators and algorithms are powerful, and efficient, but hard to operate and implement, thus, introducing a gap from the research to the industry, and leading to a trustworthy problem for practitioners.

The possible solution to bridge that gap includes two steps: 1. Unifying the simulation and analysis process by a standard ecosystem like General Modeling Network Specification (GMNS) [10] to define a common format for sharing routable road network files and is designed for multi-modal static and dynamic transportation planning and operations. 2. Building an intelligent system with self-explain abilities, which is integrated with multiple domain-specific tasks and the corresponding frontier solutions: state-of-the-art algorithms, powerful simulators, etc., and can be easily executed with sufficient explanations in an interactive way.

Transportation intelligence can be divided into 5 stages according to the development of technology as in Fig. 1, with Turing Indistinguishable as the hardest one to provide human-like decisions. With large language models becoming increasingly instrumental in aiding humans, many researchers have leveraged the LLMs to benefit transportation tasks [11, 12] in stage 4, and this has provided a clearer vision on stage 5 as a more Turing Indistinguishable

traffic intelligence. Large-scale pre-trained models such as Llama, GPT-3.0, and ChatGPT, are endowed with the capacity to grasp the context, dissect issues, and discern the logic connecting questions with answers, which can deliver in-depth clarifications on specific topics through a sequence of interactive dialogues. Early explorations are made by leveraging LLMs to benefit domain-specific tasks, such as: Copilot [13], DrugGPT [14], TrafficGPT [11], GraphGPT [15], etc. Due to the limitation of only tackling the context-level questions, researchers managed to augment the language model on their ability to take action and use tools, which significantly broadened the application scenarios and enhanced the beneficial impact [16]. Defined as **Augmented Language Models (ALMs)**, this refers to ‘language models (LMs) that are augmented with reasoning skills and the ability to use tools’ [16]. Inspired by ALMs, we propose to design a rudiment of Turing Indistinguishable Traffic Intelligence: Open-TI, an augmented traffic agent not only able to provide conversational insights, but also able to understand human intentions, conduct intelligent traffic analysis from scratch, answer questions regarding the used techniques or tools, and provide an interpretation of the results. By this, it will be more convenient for the industrial practitioners or any stakeholders to learn about traffic and transportation studies, and cast interesting case analyses.



**Fig. 1:** The 5 Stages of Development of Traffic Intelligence

In this paper, we first provide a background introduction to recent most related research directions, including traffic simulations, traffic domain-specific tasks, and augmented language models. Then we propose a pivotal augmented language agent Open-TI, that is integrated with a neat interface to operate possible tools, thus realizing the language-level operation, it is worth noting that, the Open-TI is able to conduct traffic analysis from scratch (from downloading map to provide simulation on the interested area). Second, we realized multiple domain-specific tasks like traffic signal control policy training and traffic demand optimization under the unified implementation class. Third, we design the ChatLight to realize the meta-control based on the LLM’s inference ability,

by agent-agent communication, the pivotal agent will interact with humans to obtain control requirements and convey the message to the ChatZero control agent, then, the ChatZero will conduct the decision making based jointly on the observation of current traffic situation and described policies.\*

In summary, the **contributions** of this paper are:

- Open-TI is the first paradigm to realize a thorough and stable traffic analysis from scratch: from downloading maps to simulation of interested areas.
- Open-TI is a central agent in charge of multiple task-specific executors, including traffic signal control, demand optimization, etc. It is equipped with standard API for open implementation access of the research community to enhance its capabilities.
- Open-TI could realize meta-control: convey the policy description by communicating with another control agent that directly serves as a controller to output actions based on context understanding.
- Open-TI provides sufficient explanations to any questions during the traffic analysis, from log file interpretation to results comparison, bringing convenience for traffic management and transportation strategy planning.

## 2 Background and Related Work

This section provides concepts of augmented language agents, traffic simulation, and transportation research tasks.

### 2.1 Augmented Language Models

Large Language Models (LLMs) [17–19] have boosted dramatic progress in Natural Language Processing (NLP) and are already core in several products with millions of users, such as the coding assistant Copilot [20], Bert enhanced search engine<sup>†</sup> and 2. ChatGPT and GPT4 [21]. LLMs are able to execute multiple tasks from language understanding to conditional and unconditional text generation relying on memorization [22] combined with compositionality [23] capabilities, thus opening a realistic path towards higher-bandwidth human-computer interactions, or even benefit other research domains by its inference ability [24]. But LLMs are not held solely in the text conversation, when LLMs are equipped with the tools using abilities, it will bring more changes to people’s lives. Some literature shows that by augmenting the LLMs with the tool-using ability, it could realize the advanced automation and bring the intelligence science closer to the goal of being Turing Indistinguishable, such as [25] design the API bank to execute API calls to meet human needs, [26] applied the augmented language models to the medical field to serve as a more flexible knowledge hub for doctors and patients. [27] focuses more on using existing foundation models (as a brain-like central system) and APIs of other AI models and systems (as sub-task solvers) to achieve diversified tasks in both digital

---

\*We have released the code at:<https://github.com/DaRL-LibSignal/OpenTI.git>.

<sup>†</sup><https://blog.google/products/search/search-language-understanding-bert/>

and physical domains. Our work is the first to explore the augmented language agents on automatic traffic intelligence that realize a throughout traffic analysis system.

## 2.2 Traffic Simulation

The continuous growth of urban populations and the increase in vehicular traffic have accentuated the need for efficient traffic management and planning. Traffic simulation provides a good reference for planning strategies, offering insights into traffic patterns, road network efficiencies, and the potential impacts of infrastructural changes as shown in Fig. 2. The utilization of traffic simulation models facilitates the analysis of traffic behavior under various conditions without the need for costly real-world alterations.



**Fig. 2:** The traffic and transportation simulation in cities, (a) is a real-world traffic image, (b) is the simulation of traffic flow in DTALite [28], and (c) is the city simulation from CARLA [29].

Traffic simulation has gone through various stages, from simplistic models to complex systems incorporating real-time data and predictive analytics. Early efforts in traffic simulation were concentrated on microscopic models, which simulate individual vehicle movements. Pioneering work by [30] and others laid the foundation for microscopic traffic simulation, focusing on the behavioral aspects of individual drivers. As computing power increased, mesoscopic and macroscopic models gained popularity. Mesoscopic models, like those developed by [31], provide a balance between simulating details and computational efficiency, ideal for medium-scale traffic networks. Macroscopic models, on the other hand, such as the work by [32], offer insights into broader

traffic flow trends, suitable for large-scale analysis but lacking the granularity of microscopic models.

Recent advancements have shifted focus towards data-driven and AI-based approaches. The integration of machine learning, as seen in the work of [33], has enabled more accurate predictions of traffic patterns by learning from historical data. Additionally, the incorporation of real-time data, such as that from IoT devices and traffic sensors [34], has significantly enhanced the responsiveness of traffic simulation models. Studies by [35] have demonstrated the effectiveness of real-time data in adapting traffic simulations to dynamic conditions.

With the development of autonomous vehicles (AVs), traffic simulation for safety tests and validation become important. The simulator developed by NVIDIA [36] tends to provide a more physically accurate simulation platform and Waymax [37], which is developed by autonomous vehicle company Waymo, provides a multi-agent scene simulator on hardware accelerators to empower the simulation for AVs.

In this work, we provide support to multiple simulators like SUMO [38], CityFlow [8], and DLSim [28], and further present open-implementation instruction to help integrate more advanced ones with the development of the research community.

## 2.3 Traffic Domain Specific Tasks

### *Traffic Signal Control*

Traffic Signal Control (TSC) is crucial for improving traffic flow, reducing congestion in modern transportation systems, and benefiting individuals and societies. Traffic signal control remains an active research topic because of the high complexity of the problem. The traffic situations are highly dynamic and require traffic signal plans to adapt to different situations, making it necessary to develop effective algorithms that can adjust to changing traffic conditions [39]. Recent advances in reinforcement learning (RL) techniques have shown superiority over traditional approaches in TSC [40]. In RL, an agent aims to learn a policy through trial and error by interacting with an environment to maximize the cumulative expected reward over time, when learning the policy, it takes the features like intersection vehicle amount, and current traffic light phase (state) as observations, and change the traffic light phases or duration times (actions), to realize the relief of traffic congestion (reward). The most evident advantage of RL is that it can directly learn how to generate adaptive signal plans by observing the feedback from the environment, so it is beneficial for traffic intelligence to integrate TSC tasks to provide planning insights.

### *OD Matrix Optimization*

Origin-destination (O-D) matrix estimation is a critical component in the field of transportation planning and traffic engineering and involves the creation of a matrix that represents the number of trips between various origins and

### Open-TI for Intelligent Traffic Planning and Simulation

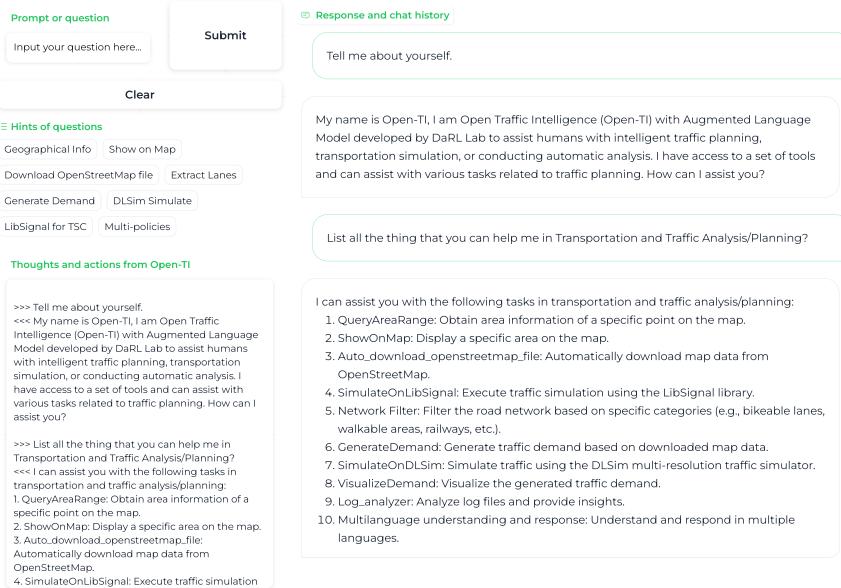


Fig. 3: The Open-TI conversation interface.

destinations in a given area over a specified time period. This matrix is fundamental for understanding travel demand patterns and is important for traffic modeling, infrastructure planning, and policy-making in urban environments. The process of O-D matrix estimation can vary in complexity depending on the size of the area being studied and the granularity of the data required. Traditional methods often involve the use of travel surveys, where individuals report their travel behavior, or the extrapolation of data from limited traffic counts [41–43]. However, these methods can be time-consuming and may not always capture the full spectrum of travel patterns due to their reliance on sampling or incomplete data.

The O-D matrix estimation has evolved in recent years, modern techniques leverage large datasets obtained from a variety of sources, such as traffic sensors, GPS devices, and mobile phone signals [44–54]. These data sources provide a more comprehensive and real-time picture of traffic flows, allowing for more accurate and dynamic O-D matrices. The use of advanced statistical models, machine learning algorithms, and data fusion techniques further enhances the precision of O-D matrix estimations, which not only provide a better understanding of current travel patterns but also enable transportation planners and engineers to predict future trends and make informed decisions for efficient traffic management and sustainable urban planning. Providing O-D matrix optimization would enhance the practicality of the simulation situation.

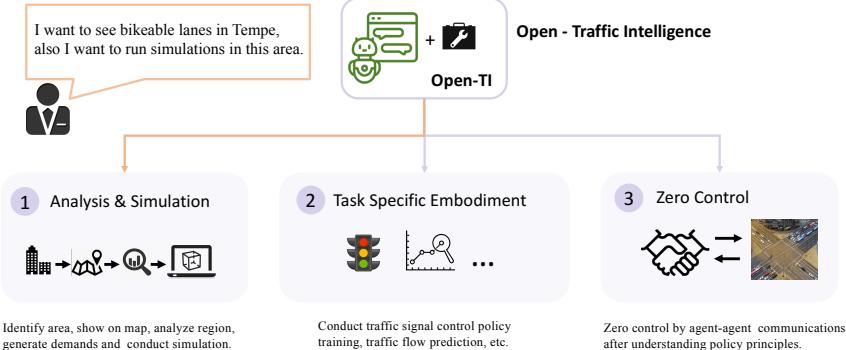


Fig. 4: The overview of the Open-TI functionalities

### 3 The Architecture of Open-TI

#### 3.1 Overview of Open-TI

To take a step forward to more **Turing Indistinguishable Traffic Intelligence**, Open-TI is equipped with human-like semantic ability and goal-oriented execution ability. Human-like semantic ability is realized by convenient conversation sessions between users and the agent, and execution ability is guaranteed by agent augmentations.

For Open-TI, primarily, a user-friendly interface is designed as shown in Fig. 3. It contains four panels: **Prompt or question** (top left), the user can edit input, clear text, and submit request; **Hints of questions** (middle left): user could click on the suggested choices to easily start a conversation; **Thought and action** (bottom left): this panel presents the chain of thought content from Open-TI agent; **Response and chat history** (right): this main panel provides multi-media feedback and execution result from Open-TI, including texts, images, path files, and browser links, etc.

The core of Open-TI mainly incorporates three modules: **Analysis and Simulation**, **Task Specific Embodiment** and **Zero Control** to enhance the intelligent traffic analysis and planning, as shown in Fig. 4.

First, Open-TI can manipulate and help practitioners to conduct analysis and simulation from scratch. Open-TI provides the chance for users to think of a POI (point of interest) or AOI (area of interest) and present the visualization immediately on a map, users can ask for more geology information like latitude and longitude range, and after that, the acquired information can be used to select an analysis range for further investigations like specific lane (e.g., bike lane) filtering and traffic simulation by DTALite (DLSim) [28] or SUMO[38].

Second, the Open-TI supports multiple task-specific embodiments by vague and high-level language explanations, which greatly reduce the professional background requirements of experimental exploration. For example, based on

the current road map, it allows one to conduct traffic light control exploration by either rule-based methods or deep learning policy training [55], it also could easily conduct traffic demand optimization by providing brief task descriptions.

Third, our method leverages the superior understanding capabilities of current LLMs to conduct meta-control by ChatLight agent: LLMs directly serve as a control agent, follow the understanding of the semantic description of the rules, and control the traffic light actions. This explores a higher level traffic light control mode, e.g., the traffic management department may have special requirements on the safety concerns and would like to adjust the traffic signal control policies to reduce the collision rate, only word-level description is required, and the description would be analyzed, extracted, and communicated as a message to the ChatLight agent, which could reduce the complexity of adjusting the control policy and furthermore, provide explanation for its actions by self-reflecting the LLM's own behavior.

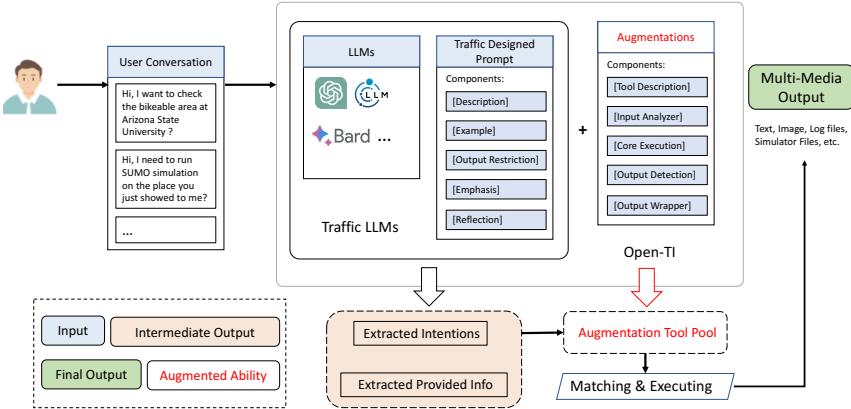


Fig. 5: The design framework of Open-TI

As shown in Fig. 5, the Open-TI consists of two components, the Traffic LLMs and Augmentations. When a user requirement is detected from the conversation, it will be passed to the first components of Traffic LLMs, which use the language agent to extract accurate intentions and useful information from dialogues, then, the extracted intention will be compared with tools in augmentation pool, after matching, if the agent found a possible solution, it will execute with extracted information and generate the output with multimedia data form.

### 3.2 Prompt Design

The Open-TI exploited the LLM's understanding abilities and tuned its behavior by prompt engineering, which is essential for well-organized operation.

**Table 1:** The details of the prompt components for `queryAreaRange`

Name	Purpose	Instances
<b>Description</b>	Core component of the prompt structure, clarifies the basic setup, method, and object of each function.	You are designed to respond with longitudes and latitudes information of a location. Humans might ask similar words of location like position = place = location = geographic info, you can imagine and infer the most possible.
<b>Format Restriction</b>	Specify input format constraints, significantly reducing error rates.	The format of your output longitude and latitude is a query of 4 value array as [min_long, min_lat, max_long, max_lat]
<b>Example</b>	Help LLMs understand the exactly processing of the execution.	Human ask "Where is Arizona State University, Tempe Campus", you need to output [-111.9431, 33.4154, -111.9239, 33.4280].
<b>Reflection</b>	Remind LLMs not to engage in unnecessary tasks and ensure that each process is executed accurately.	You should respond directly with what you know and then stop, do not look for the location or attempt to find it online.
<b>Emphasis</b>	Reinforce the function's objective, significantly reducing API mismatching rates.	You have the valid tool to provide location. You have a specific tool to directly query the location.

We have designed 5 aspects of prompt structure: **[Description]**, **[Example]**, **[Output Restriction]**, **[Emphasis]** and **[Reflection]**. And we verified their effectiveness in cross-task behaviors by ablation experiment. In this section, we provide details of prompt design as shown in Fig. 5. Including the purpose of the prompt and example cases. The examples in Table 1. are from the same augmentation task of `queryAreaRange`.

### 3.3 Execution and Augmentation List

The overall execution process is expressed in Algorithm 1. in pseudo-code. Following the same execution flow, there are different augmented tools that could help users with various requirements as presented in Table 2. The section. 4 will elaborate on the three augmentation modules in detail.

### 3.4 Standard Implementation

In this section, we formalize the API content and provide the structure of the augmentation class. In order to extend the augmentation to the border range with additional capabilities, the keys are: First, make sure the pivotal agent is aware of the functionality of the specific additional tool, so when the requirement is asked by users, the pivotal agent could pick up the most suitable method. Second, the embodiment program should function accurately and effectively, so that when the pivotal agent boots up the process, it can come to the expected result. Based on the two keys, we provide the implementation structure as in the following code. One needs to provide the descriptions on the augmentation class and implement the `Execution ()`, this provides an easy way to understand and neat for other researchers to explore and enforce Open-TI's abilities.

**Table 2:** A list of augmented tools implemented in Open-TI

Augmentation Name	Description
queryAreaRange	Obtain area information, specifically the longitudes and latitudes of a point of interest on the map.
showOnMap	Display the location of interest on the map, such as the ASU campus area.
autoDownloadOpenStreetMapFile	Automatically download map data from OpenStreetMap for a specified area.
simulateOnLibsignal	Execute simulations on the open-source library called Libsignal.
networkFilter	Filter the road network based on required categories, return the file path of a filtered road network that emphasizes lanes of interest.
generateDemand	Generate demand based on OpenStreetMap data.
simulateOnDLSim	Simulate on the DLSim multi-resolution traffic simulator.
simulateOnSUMO	Execute the simulation given arbitrary .osm data.
visualizeDemand	Automatically generate and display visualizations of the demand file.
logAnalyzer	Analyze log or config files and provide comparisons.
resultExplainer	Interpreter results to provide insights.
demandOptimizer	Approximate the origin-destination demand to fit realistic observation.

```

1  class AugmentTask:
2      def __init__(self, params) -> None:
3          self.params = params
4          # set possible pre-defined params, e.g., folder path
5
6          @func_prompt(name="prompt_name",
7                      description="""detailed explanation""")
8
9      def embody(self, target: str) -> str:
10         try:
11             result = Execution(target)
12             # Concrete Execution Implementation
13         except Error as e:
14             print(e)
15             return result

```

**Listing 1:** The Class and API Format

## 4 Sub-module Embodiment

The Open-TI is enhanced by three different modules that eventually realize the superior performance. In this section, we will elaborate on each module with examples and illustrations.

**Algorithm 1** Open-TI Execution Process

---

```

1: INPUT:  $msg \leftarrow UserInputQuery$ 
2: if Intention Needs External Tools then
3:   while Augmented APIs not Found do
4:      $keywords \leftarrow summarize(msg)$ 
5:      $api \leftarrow search(keywords)$ 
6:     if MaximumQueryTime Exceeds then
7:       break
8:     end if
9:   end while
10:  if API found then
11:     $Params \leftarrow extract\_params(msg)$ 
12:    while Params Not Satisfied do
13:      1. Retrospect Expected Form
14:      2. Examine User Input:  $msg$ 
15:      while Missing Info. do
16:        Alert Required Info.
17:      end while
18:       $Response \leftarrow execute\_api\_call(Params)$ 
19:      if MaximumQueryTime Exceeds then
20:        break
21:      end if
22:    end while
23:    while Response not Satisfied do
24:       $api\_call \leftarrow gen\_api\_call(api\_doc, msg)$ 
25:       $Response \leftarrow execute\_api\_call(api\_call)$ 
26:      if MaximumQueryTime Exceeds then
27:        break
28:      end if
29:    end while
30:  end if
31: end if
32: if Response then
33:    $re \leftarrow Construct\_Response(Response)$ 
34: else
35:    $re \leftarrow Query\_Failed()$ 
36: end if
37: return  $re \leftarrow ResponseToUser$ 

```

---

**4.1 Pivotal Agent for Transportation Analysis**

In this module, analysis from scratch is realized by seamless connections between augmented tools and the pivotal operation agent. The supported external tools and packages are shown in Table 3. And when the user asks about related tasks, Open-TI will automatically match the highest probability option and process following Algorithm 1. An example of interaction is shown in Fig. 6.

**Table 3:** The supported external tools and packages

Name	Functions	Versions
osm2gmns	obtain networks from OSM and convert to GMNS	V-0.7.3
grid2demand	Origin-destination trans demand generate	V-0.3.6
DLSim-MRM	Multi-resolution Traffic Simulation	V-0.2.11
Libsignal	Multi-simulator platform for Traffic Signal Control	V-1.0.0

Hi tell me about yourself?

I am Open-TI, an Open Traffic Intelligence with Augmented Language Model (ALM) developed by DaRL Lab. I am designed to assist humans with intelligent traffic planning, traffic simulation, and automatic analysis. I have access to a set of tools and can operate existing packages and open-sourced libraries.

Where is Arizona State University, Tempe Campus?

The longitude and latitude area of Arizona State University, Tempe Campus is [-111.9411651, 33.4102062, -111.9092447, 33.4329786].

Can you show me this area on map?

Sure! The area on map as follow:

I am wondering the bikeable lanes in this place, can you show me that information?

Of course! I have filtered the bikeable lanes in this place, please check below:

Help me to run simulation on SUMO in this area?

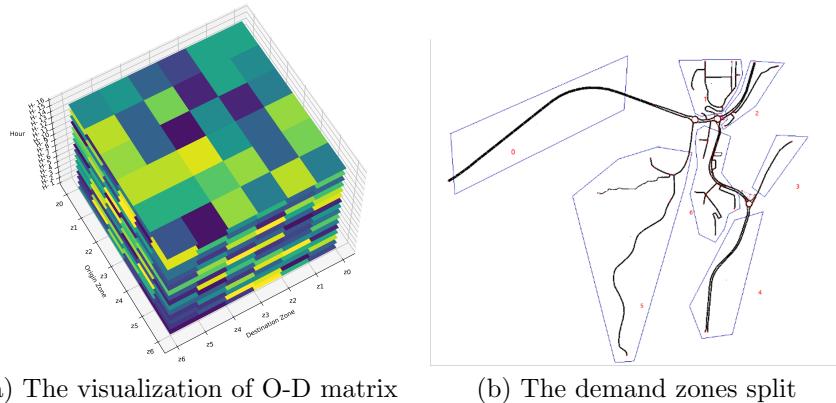
I have set the configs and the successfully opened the sumo for one click simulation:

**Fig. 6:** The demonstration pivotal agent control. (Right: The user messages, Left: The responses from Open-TI). This series of interactions shows how to query geography information of a location, how to visualize on the map, filter the interested lane types, and use the arbitrary map for automatic traffic simulation (SUMO).

## 4.2 Task-Specific Embodiment

The Open-TI is capable of realizing more general research tasks in the traffic domain. Including traffic signal control (TSC), traffic flow prediction, traffic Origin-Destination(O-D) demand optimization, etc. The architecture in Open-TI is well structured and supports extensibility for an open community. We will introduce how the Open-TI achieves the three demonstrating tasks in the following subsections.

### 4.2.1 Traffic O-D Demand Optimization Task



**Fig. 7:** An illustration of the O-D demand optimization task in Sedona, AZ. In (a), it consists of 3 dimensions: origin zone, destination zone, and time (Hour), in each time interval, one slice describes the traffic demand information. (b) is the geography demand zone split of the interested area.

In the OD demand optimization task, the goal is to design an algorithm and learn a model that could help to output an accurate OD matrix, given the partial observation data. In Figure 7, we show an example in Sedona, AZ, USA. When asked to execute an O-D matrix optimization task, users could specify the observation data source, and traffic simulation setting, and then choose the optimization techniques to experiment with. In the example case, the given data is the 16-hour count data at the specific observation point of a roundabout, and we asked the agent to use a genetic algorithm to conduct optimization and provide the result.

### 4.2.2 Traffic Signal Control Task

In the realization of traffic signal control embodiment, we seamlessly integrated the Libsignal [55] that could realize the cross-simulator traffic signal control over the majority of baseline methods, including the rule-based approaches

(Fixed Time and Self-organizing traffic lights - SOTL [56]) and reinforcement-learning-based approaches as shown in Fig. 8. We provide further interaction cases in the Appendix. A.1.

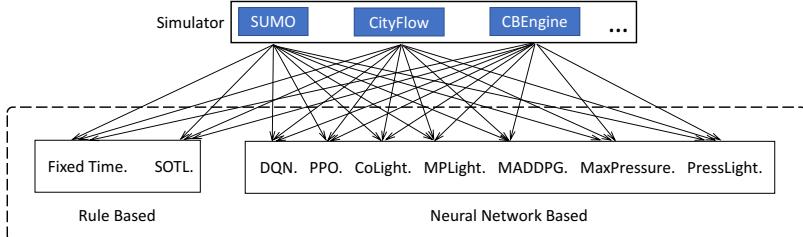


Fig. 8: Current feasible simulators and algorithms for TSC tasks

### 4.3 Agent Meta Control

In the meta-control task, we essentially designed an agent-agent communication and the pivotal agent is in charge of understanding the human descriptive intention of the traffic policy, and the execution agent will take the message as instruction, process, and digest, then connect to the traffic practical control to provide a self-explainable action execution control.

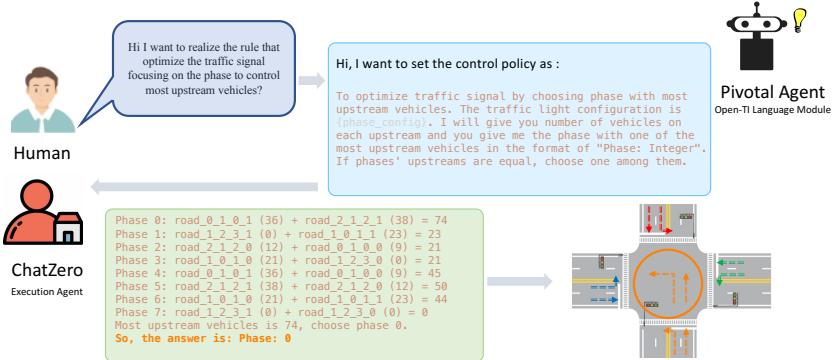


Fig. 9: The demonstration ChatZero Meta Control.

## 5 Experiment

In this section, we conduct extensive experiments to answer the following questions:

- RQ1: How does our proposed approach compare to other state-of-the-art methods in terms of performance?

- RQ2: How do the components impact the performance of our proposed method?
- RQ3: How does ChatZero execute meta-control? How does it perform across various LLMs?

Three aspects of experiments are designed from error rates of API calls, ablation study of Augmented Languages Agents' prompt structure, and Zero-Control agent's performance on LLM agents to verify the effectiveness and stability of the proposed Open-TI. Please note that, for the RQ1 and RQ2, we develop standard Open-TI based on GPT3.5 and for RQ3, we verified on 4 different language models: Llama7B, Llama13B, GPT3.5, and GPT4.0.

## 5.1 Language Agent Analysis on the API Calls

In this section, we conduct the functionality-level experiments of API analysis and compare them with the baseline method known as TrafficGPT[11].

### Experiment Design

First, following the work of [25], we analyze three types of API call abnormal behaviors, namely 'No API Call Rate', 'API Mismatching Rate', and 'Error Raise Rate'. Both Open-TI and TrafficGPT are equipped to handle a range of tasks spanning geographical information, simulation, and traffic signal control. Although the specific functions of Open-TI and TrafficGPT are slightly different, we are still able to evaluate the overall API access stability. We adopted  $T = 6$  similar tasks as shown in the table 4. to conduct the comparison by testing each task 20 times. And calculate the error rate follow the equation 1:

$$\text{Error Rate} = \frac{1}{T} \sum_{t=1}^T \frac{n_t^e}{n_t^c} = \frac{1}{T} \sum_{t=1}^T \frac{1}{n_t^c} \sum (c_t^{no}, c_t^{miss}, c_t^{error}) \quad (1)$$

where  $n_t^e$  represents the number of error occurrences for task  $t$  during total tests,  $n_c$  denotes the number of total testing instances, (i.e.,  $n_c = 20$  for this experiment),  $c_t^{no}$  is the sum of errors caused by the absence of API calls for task  $t$  among all tests, similarly,  $c_t^{miss}$  is the sum of mismatching error times,  $c_t^{error}$  is the number of error raising times, and exist  $n_t^e = c_t^{no} + c_t^{miss} + c_t^{error}$ .

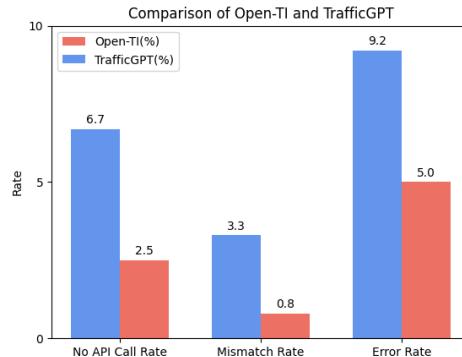
For evaluation of each dimension shown in Fig. 10, denote  $\rho$  as error rate, we have:  $\rho_{no} = \frac{1}{T} \sum_{t=1}^T \frac{c_t^{no}}{n_t^c}$ ,  $\rho_{miss} = \frac{1}{T} \sum_{t=1}^T \frac{c_t^{miss}}{n_t^c}$  and  $\rho_{error} = \frac{1}{T} \sum_{t=1}^T \frac{c_t^{error}}{n_t^c}$ .

As in the top of Table 4, which is an intersection set of functionalities in TrafficGPT and Open-TI. `simulateOnSumo` are commonly integrated by different implementations, `showOnMap` is used to query and show the map of the interested place which is equivalent to the functionality of locating and drawing the intersection on the map. `logAnalyzer` is designed to interpret and help the user understand the log and config files, similar to retrieving data from .XML files in TrafficGPT. `visualizeTrainingCurves` is for visualization of the training process which is equivalent to generating heat graphs by TrafficGPT, `simulateOnLibsignal` applies multiple algorithms to control traffic signal control while the Webster method is used in TrafficGPT, similarly, two

**Table 4:** The compared tasks is an intersection set that exists both in Open-TI and TrafficGPT. The experiment tends to design a fair comparison by identical task goals or similar difficulty.

	<b>Open-TI</b>	<b>TrafficGPT</b>
①	<code>simulateOnSumo</code>	Run the sumo simulation
②	<code>showOnMap</code>	Draw intersections on map
③	<code>logAnalyzer</code>	Retrieve data from the .xml files
④	<code>visualizeTrainingCurves</code>	Generating heat graphs
⑤	<code>simulateOnLibsignal</code>	Optimize intersections by Webster
⑥	<code>resultExplainer</code>	Compare data from the simulation
	<code>queryRangeArea</code>	-
	<code>autoDownloadOpenStreetMapFile</code>	-
	<code>networkFilter</code>	-
	<code>generateDemand</code>	-
	<code>simulateOnDLSim</code>	-
	<code>demandOptimizer</code>	-

systems both provide result explanation as `resultExplainer`, which we take into comparison.



**Fig. 10:** Language agent analysis on the API calls

### Experimental Results

The experiment results can be found in Fig. 10. The comparison is conducted on the average value over 20 times. The sum of the 3 types of error rates in Open-TI and TrafficGPT are 8.3% and 19.2%, calculated by aggregation of the three types of error rates in two systems respectively.

**Table 5:** Error Rate of Open-TI and TrafficGPT

	Tasks	TrafficGPT	Open-TI
①	No API Call	0.00	0.00
	Mismatch	0.05	0.00
	Error Raise	0.10	0.05
②	No API Call	0.05	0.00
	Mismatch	0.00	0.00
	Error Raise	0.00	0.05
③	No API Call	0.15	0.00
	Mismatch	0.05	0.05
	Error Raise	0.10	0.05
④	No API Call	0.05	0.00
	Mismatch	0.10	0.00
	Error Raise	0.15	0.10
⑤	No API Call	0.05	0.05
	Mismatch	0.00	0.00
	Error Raise	0.10	0.05
⑥	No API Call	0.10	0.05
	Mismatch	0.00	0.00
	Error Raise	0.10	0.00

In Fig. 10, we take the *mean* as the evidence to show on the bar chart as from 3 different evaluation dimensions. The x-axis presents individual dimensions of error rates. In each dimension, the two methods are compared side by side, and we can compare their percentage value according to the y-axis.

We could observe that the Open-TI effectively reduced the general API abnormal behaviors across three evaluation aspects. More obviously, Open-TI shows significantly better performance than the baseline method in terms of Error Rate, mainly because the 5 aspects of the prompt components emphasize the task, reactions, and meta-information more explicitly. We also demonstrate the detailed task level side-by-side comparison as shown in Table. 5, from the result, we could notice that Open-TI performs consistently more stable (lower error rate and lighter color) for each task than TrafficGPT, even though for task ②, Open-TI performs slightly worse in Error Raise Rate, it is because `showOnMap` is a more complex task consists of two consecutive sub-tasks, which are 1) identifying the geographic information and 2) request and visualize map data. To understand how each component plays a role in helping the language agent work consistently stable, we conduct further exploration in the next section.

## 5.2 Ablation Study of Open-TI Prompt Structure

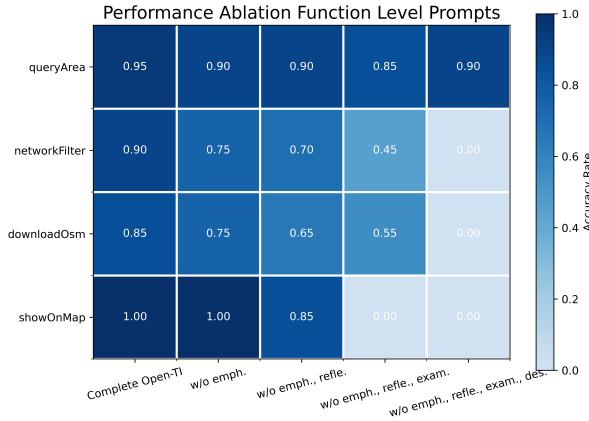
In this section, we conduct the ablation study to unravel the nuanced impact of different components on the functionality of the tasks under consideration.

### Experiment Design

The ablation process is operated by the removal of each prompt component in an order of ***Emphasis***, ***Reflection***, ***Format Restriction***, ***Example***, and ***Description***. The analysis is done across 4 tasks: `queryAreaRange`,

`showOnMap`, `autoDownloadOpenStreetMapFile`, and `simulateOnLibsignal`, which encompasses all five prompt components.

By eliminating the least important component first, followed by the removal of subsequent components in ascending order of importance. This sequential approach enables the examination of the individual influence of each component on the overall function. For each iteration, we conduct an experiment by posing 20 questions related to each function. Then the collected error rates are summarized into a comprehensive heatmap as in Fig. 11.



**Fig. 11:** The ablation study of the prompt components. The x-axis from left to right shows the gradual removal of each prompt module. The y-axis shows the individual task, cell color from dark to light reflects the performance drop.

## Experimental Results

Fig. 11 shows the experimental findings related to the effect of each component. When there is no component removed, the system’s performance is relatively stable and robust. When the **Emphasis** is removed, the accuracy suffers a slight drop across `queryAreaRange` to `autoDownloadOpenStreetMapFile`, because the agent can be confused among similar tasks or tasks with similar keywords then leading to mismatching. For example, when a human asked, “Can you help me find the map of Arizona State University?” The agent may be confused about finding the map and suppose human need the geographical information about Arizona State University. Thus, the agent mismatch `autoDownloadOpenStreetMapFile` question to `queryAreaRange`.

Then if the **Reflection** is further removed, `simulateOnLibsignal` suffers the most, by investigation, we found that it is mainly caused by entering incorrect keywords in questions. However, the agent can memorize the answers to previous questions, preventing error escalation and subsequently providing

**Table 6:** Examples of correct and wrong API usage of Augmented Language Model.

Error Name	Question	Tool	Analysis
No API Call	Can you assist me to download the OSM file for the Sydney Harbour Bridge in Australia?	✓ : It shows the path to the OSM file for Sydney Harbour Bridge.	It finds a similarly named OSM file and shows the wrong answer.
		✗ : It shows the path to the OSM file for the State of Liberty in New York.	
API Mismatch	I'm interested in the OSM file for Dubai Mall; Can you guide me on that?	✓ : It shows the path to the OSM file for Dubai Mall.	The question should match <code>autoDownloadOpenStreetMapFile</code> but it mismatches with <code>queryAreaRange</code> .
		✗ : The longitude and latitude of Dubai Mall is [55.274, 25.194, 55.282, 25.199].	
Error Raise	Can you provide the OSM file for CN Tower in Toronto?	✓ : It shows the path to the OSM file for CN Tower in Toronto.	It mistakenly inputs wrong information into the <code>showOnMap</code> .
		✗ : Error raise and not keep executing.	

inaccurate responses to users. Therefore, the **Reflection** aims to address the problem of incorrect keywords, ensuring that users input the correct keywords in their questions.

Meanwhile, **Format Restriction** mainly affects `showOnMap` and `simulateOnLibsignal`. When **Format Restriction** is removed, their accuracy rates decrease by 25% and 55%, respectively. This is due to the role of **Format Restriction** in input format limit control; Entering incorrect information into a project can lead to errors. As a result, **Format Restriction** significantly affects `showOnMap` and `autoDownloadOpenStreetMapFile` performance.

The component **Example** plays a significant role in helping language agents understand the task, once removed, the accuracy rates decrease by 45% and 55%, respectively. Furthermore, since LLMs can sometimes overlook the middle steps in implementation, the **Example** helps to improve its stability. Thus, the Example is significant in both `showOnMap` and `autoDownloadOpenStreetMapFile`. For the case of overlook behavior, e.g., if I want to download the OSM file of Arizona State University and the agent doesn't have the geographical information for the target place, the agent should first call the `queryAreaRange` function. After obtaining the geographical information, it should then input this data into the `autoDownloadOpenStreetMapFile` function to get the correct answer. We define this embodiment as multi-step sequential actions. However, if the **Example** is removed, the agent might skip calling `autoDownloadOpenStreetMapFile`, resulting in an incorrect answer.

There is no clear influence from each component's removal on `queryAreaRange`. This is because there are solely two steps in the task, match the API request and post to the online map server for response, the format of the request body is well-defined, and only execution is needed, which makes it a simple procedure, not sensitive to the prompt component.

In conclusion, all components contribute to enhancing the performance of Open-TI. Particularly, the impact of **Format Restriction** and examples is notably significant. This reveals that careful attention should be paid to **Format Restriction** and examples when seeking to enhance the execution-related prompt.

### 5.3 Meta Agent Control Analysis

#### Experiment Design

In this section, we conduct experiments to verify the effectiveness of the meta agent control, at the same time, we realized 4 versions of ChatZero on the most well-known LLMs which are: Llama2-7b, Llama2-13b, ChatGPT (GPT-3.5) and GPT-4.0. During the test, the Open-TI pivotal agent will ask the 4 versions of ChatZero to perform traffic signal control tasks across 4 different traffic configurations using the realistic road map data in Hangzhou city. Each traffic control task is conducted 5 times and the reported results are the *mean* values, following the literature in TSC [57], the evaluation metrics are as follows:

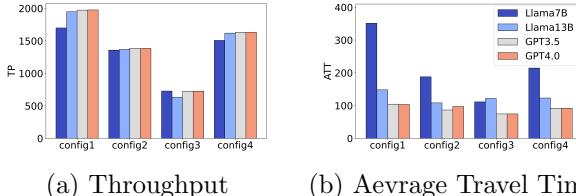
- *Average Travel Time (ATT)* is the average time  $t$  it takes for a vehicle to travel through a specific section of a road network. For a control policy, the smaller  $ATT$ , the better.
- *Throughput (TP)* is the number of vehicles that reached their destinations given amount of time. The larger  $TP$ , the better.
- *Reward* is an RL term that measures the return by taking action  $a_t$  under state  $s_t$ . We use the total number of waiting vehicles as the reward, aligned with Preliminaries. The larger the reward, the fewer waiting vehicles, the better.
- *Queue* is the number of vehicles waiting to pass through a certain intersection in the road network. Smaller is better.
- *Delay* is the average delay per vehicle in seconds and measures the amount of time that a vehicle spends waiting in the network. Smaller is better.

#### Experimental Results

The experiment results are shown in the Fig. 12. From the throughput (TP) and average travel time (ATT), we could find out that ChatZero by GPT 4.0 provides the best control results with the overall highest TP and lowest ATT, reflecting its superior ability to understand the policy description and conduct proper control. Other metrics evaluation are shown in Fig. 13.

## 6 Conclusion

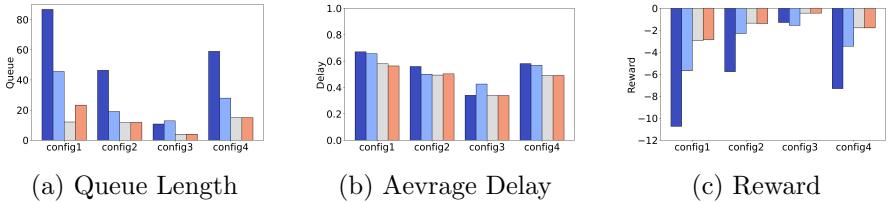
In this paper, we propose Open-TI, an intelligent traffic analysis agent leveraging the large language models' contextual abilities and augmenting them with traffic domain-specific tools, which could provide more than questions consult, but also actual practice on processing raw map data, executing the simulation, training traffic light control policies, and conducting demand optimization, etc. We also explored the meta-control of traffic light by an agent-agent



(a) Throughput

(b) Average Travel Time

**Fig. 12:** The ChatZero performance in TP and ATT across the 4 LLMs. Each LLM is tested to control the traffic signal in 4 different configs of road situation



(a) Queue Length

(b) Average Delay

(c) Reward

**Fig. 13:** The ChatZero performance in TP and ATT across the 4 LLMs. Each LLM is tested to control the traffic signal in 4 different configs of road situation

communication scheme named ChatZero, which could cast insight into the self-explainable control practice.

We also found that in sequential actions practice, it is easier to occur the API mismatch, for future work, it is important to focus on improving the accuracy of the multi-step action embodiment. We have provided the code base and an explicit implementation structure for the research community to enhance and expand the Open-TI's intelligence.

## References

- [1] Yukawa, S., Kikuchi, M.: Coupled-map modeling of one-dimensional traffic flow. *Journal of the Physical Society of Japan* **64**(1), 35–38 (1995)
- [2] Chao, Q., Bi, H., Li, W., Mao, T., Wang, Z., Lin, M.C., Deng, Z.: A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. In: *Computer Graphics Forum*, vol. 39, pp. 287–308 (2020). Wiley Online Library
- [3] Dai, Z., Liu, X.C., Chen, X., Ma, X.: Joint optimization of scheduling and capacity for mixed traffic with autonomous and human-driven buses: A dynamic programming approach. *Transportation Research Part C: Emerging Technologies* **114**, 598–619 (2020)
- [4] Zhou, X.S., Cheng, Q., Wu, X., Li, P., Belezamo, B., Lu, J., Abbasi,

M.: A meso-to-macro cross-resolution performance approach for connecting polynomial arrival queue model to volume-delay function with inflow demand-to-capacity ratio. *Multimodal Transportation* **1**(2), 100017 (2022)

- [5] Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., Li, Z.: Colight: Learning network-level cooperation for traffic signal control. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1913–1922 (2019)
- [6] Osorio, C.: High-dimensional offline origin-destination (od) demand calibration for stochastic traffic simulators of large-scale road networks. *Transportation Research Part B: Methodological* **124**, 18–43 (2019)
- [7] Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation using sumo. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582 (2018). IEEE
- [8] Zhang, H., Feng, S., Liu, C., Ding, Y., Zhu, Y., Zhou, Z., Zhang, W., Yu, Y., Jin, H., Li, Z.: Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In: *The World Wide Web Conference*, pp. 3620–3624 (2019)
- [9] Fellendorf, M., Vortisch, P.: Microscopic traffic flow simulator vissim. *Fundamentals of traffic simulation*, 63–93 (2010)
- [10] Lu, J., Zhou, X.S.: Virtual track networks: A hierarchical modeling framework and open-source tools for simplified and efficient connected and automated mobility (cam) system design based on general modeling network specification (gmns). *Transportation Research Part C: Emerging Technologies* **153**, 104223 (2023)
- [11] Zhang, S., Fu, D., Zhang, Z., Yu, B., Cai, P.: Trafficgpt: Viewing, processing and interacting with traffic foundation models. *arXiv preprint arXiv:2309.06719* (2023)
- [12] de Zarzà, I., de Curtò, J., Roig, G., Calafate, C.T.: Llm multimodal traffic accident forecasting. *Sensors* **23**(22), 9225 (2023)
- [13] Vaithilingam, P., Zhang, T., Glassman, E.L.: Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In: *Chi Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1–7 (2022)

- [14] Li, Y., Gao, C., Song, X., Wang, X., Xu, Y., Han, S.: Druggpt: A gpt-based strategy for designing potential ligands targeting specific proteins. *bioRxiv*, 2023–06 (2023)
- [15] Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., Huang, C.: Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023* (2023)
- [16] Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al.: Augmented language models: a survey. *arXiv preprint arXiv:2302.07842* (2023)
- [17] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018)
- [18] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Advances in neural information processing systems* **33**, 1877–1901 (2020)
- [19] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrman, S., et al.: Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* **24**(240), 1–113 (2023)
- [20] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021)
- [21] Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., et al.: Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, 100017 (2023)
- [22] Tirumala, K., Markosyan, A., Zettlemoyer, L., Aghajanyan, A.: Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems* **35**, 38274–38290 (2022)
- [23] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al.: Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625* (2022)

- [24] Da, L., Gao, M., Mei, H., Wei, H.: Llm powered sim-to-real transfer for traffic signal control. arXiv preprint arXiv:2308.14284 (2023)
- [25] Li, M., Song, F., Yu, B., Yu, H., Li, Z., Huang, F., Li, Y.: Api-bank: A benchmark for tool-augmented llms. arXiv preprint arXiv:2304.08244 (2023)
- [26] Wang, Y., Ma, X., Chen, W.: Augmenting black-box llms with medical textbooks for clinical question answering. arXiv preprint arXiv:2309.02233 (2023)
- [27] Liang, Y., Wu, C., Song, T., Wu, W., Xia, Y., Liu, Y., Ou, Y., Lu, S., Ji, L., Mao, S., et al.: Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. arXiv preprint arXiv:2303.16434 (2023)
- [28] Tong, L., Pan, Y., Shang, P., Guo, J., Xian, K., Zhou, X.: Open-source public transportation mobility simulation engine dtalite-s: A discretized space-time network-based modeling framework for bridging multi-agent simulation and optimization. *Urban Rail Transit* **5**, 1–16 (2019)
- [29] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on Robot Learning, pp. 1–16 (2017). PMLR
- [30] Mullakkal-Babu, F.A., Wang, M., van Arem, B., Shyrokau, B., Happee, R.: A hybrid submicroscopic-microscopic traffic flow simulation framework. *IEEE Transactions on Intelligent Transportation Systems* **22**(6), 3430–3443 (2020)
- [31] de Souza, F., Verbas, O., Auld, J.: Mesoscopic traffic flow model for agent-based simulation. *Procedia Computer Science* **151**, 858–863 (2019)
- [32] Oppe, S.: Macroscopic models for traffic and traffic safety. *Accident Analysis & Prevention* **21**(3), 225–232 (1989)
- [33] Boukerche, A., Tao, Y., Sun, P.: Artificial intelligence-based vehicular traffic flow prediction methods for supporting intelligent transportation systems. *Computer networks* **182**, 107484 (2020)
- [34] Masek, P., Masek, J., Frantik, P., Fujdiak, R., Ometov, A., Hosek, J., Andreev, S., Mlynek, P., Misurec, J.: A harmonized perspective on transportation management in smart cities: The novel iot-driven environment for road traffic modeling. *Sensors* **16**(11), 1872 (2016)
- [35] Maroto, J., Delso, E., Felez, J., Cabanellas, J.M.: Real-time traffic simulation with a microscopic model. *IEEE Transactions on Intelligent*

Transportation Systems **7**(4), 513–527 (2006)

- [36] NVIDIA: Simulation for self-driving vehicles (2023)
- [37] Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., et al.: Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. arXiv preprint arXiv:2310.08710 (2023)
- [38] Behrisch, M., Bieker, L., Erdmann, J., Krauszewicz, D.: Sumo—simulation of urban mobility: an overview. In: Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation (2011). ThinkMind
- [39] Qadri, S.S.S.M., Gökçe, M.A., Öner, E.: State-of-art review of traffic signal control methods: challenges and opportunities. European transport research review **12**, 1–23 (2020)
- [40] Wei, H., Zheng, G., Yao, H., Li, Z.: Intellilight: A reinforcement learning approach for intelligent traffic light control. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2496–2505 (2018)
- [41] Willumsen, L.G.: Estimation of an od matrix from traffic counts—a review (1978)
- [42] Abrahamsson, T.: Estimation of origin-destination matrices using traffic counts—a literature survey (1998)
- [43] Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., Diot, C.: Traffic matrix estimation: Existing techniques and new directions. ACM SIGCOMM Computer Communication Review **32**(4), 161–174 (2002)
- [44] Mahmassani, H.S.: Dynamic network traffic assignment and simulation methodology for advanced system management applications. Networks and spatial economics **1**, 267–292 (2001)
- [45] Mahmassani, H.S., Zhou, X.: In: Abed, E.H. (ed.) *Transportation System Intelligence: Performance Measurement and Real-Time Traffic Estimation and Prediction in a Day-to-Day Learning Framework*, pp. 305–328. Birkhäuser Boston, Boston, MA (2005)
- [46] Zhou, X., Qin, X., Mahmassani, H.S.: Dynamic origin-destination demand estimation with multiday link traffic counts for planning applications. Transportation Research Record **1831**(1), 30–38 (2003)
- [47] Zhou, X., Erdoğan, S., Mahmassani, H.S.: Dynamic origin-destination trip

demand estimation for subarea analysis. *Transportation Research Record* **1964**(1), 176–184 (2006)

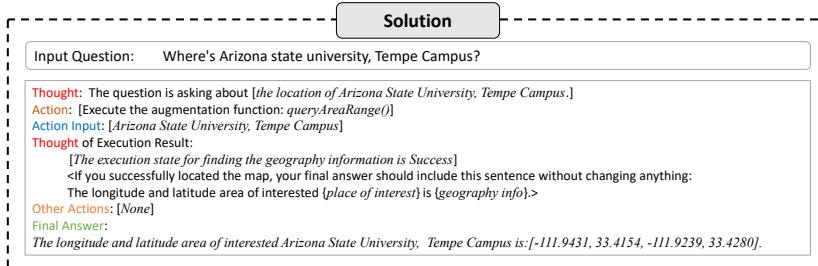
- [48] Zhou, X., List, G.F.: An information-theoretic sensor location model for traffic origin-destination demand estimation applications. *Transportation Science* **44**(2), 254–273 (2010)
- [49] Zhou, X., Lu, C., Zhang, K.: Dynamic origin-destination demand flow estimation utilizing heterogeneous data sources under congested traffic conditions (2013)
- [50] Krishnakumari, P., Van Lint, H., Djukic, T., Cats, O.: A data driven method for od matrix estimation. *Transportation Research Part C: Emerging Technologies* **113**, 38–56 (2020)
- [51] Fedorov, A., Nikolskaia, K., Ivanov, S., Shepelev, V., Minbaleev, A.: Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data* **6**, 1–15 (2019)
- [52] Pamuła, T., Żochowska, R.: Estimation and prediction of the od matrix in uncongested urban road network based on traffic flows using deep learning. *Engineering Applications of Artificial Intelligence* **117**, 105550 (2023)
- [53] Fu, H., Lam, W.H., Shao, H., Kattan, L., Salari, M.: Optimization of multi-type traffic sensor locations for estimation of multi-period origin-destination demands with covariance effects. *Transportation Research Part E: Logistics and Transportation Review* **157**, 102555 (2022)
- [54] Kumarage, S., Yildirimoglu, M., Zheng, Z.: A hybrid modelling framework for the estimation of dynamic origin–destination flows. *Transportation Research Part B: Methodological* **176**, 102804 (2023)
- [55] Mei, H., Lei, X., Da, L., Shi, B., Wei, H.: Libsignal: an open library for traffic signal control. *Machine Learning*, 1–37 (2023)
- [56] Cools, S.-B., Gershenson, C., D’Hooghe, B.: Self-organizing traffic lights: A realistic simulation. *Advances in applied self-organizing systems*, 45–55 (2013)
- [57] Wei, H., Zheng, G., Gayah, V., Li, Z.: Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter* **22**(2), 12–18 (2021)

## Appendix A

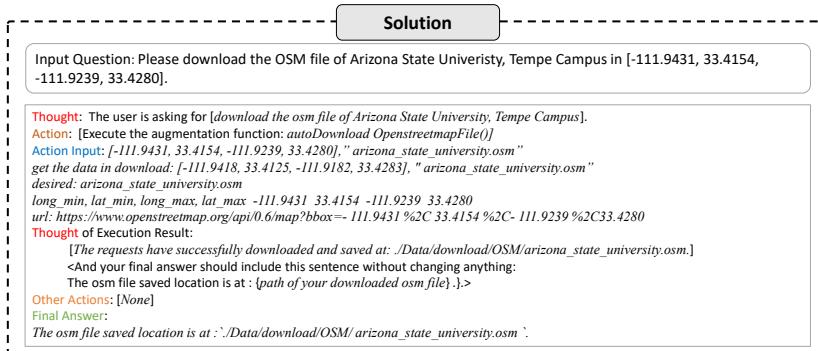
### A.1 Thought Chain Process Examples

In this section, we provide more Chain-of-thought (CoT) process examples, as a reflection on given a task, how the Open-TI thinks and proposes the solutions, and how it searches in the augmentation tools to further provide analysis.

We have shown the requests such as: downloading OSM files of specific locations, interpreting the log files, showing areas on a map, filtering assigned lane types from a given map, generating demand files based on a map file, executing multiple simulations like DLSim, SUMO, etc., running LibSignal for traffic signal control.



**Fig. A1:** Ask Open-TI to get the geographic information of Arizona State University, Tempe Campus.



**Fig. A2:** Ask Open-TI to download OSM data of Arizona State University, Tempe Campus.

**Solution**

Input Question: Please download the OSM file of New York University.

**Thought:** The user is asking to [download the OSM file of New York University]. I don't have the longitude and latitude information of New York University, so I need to [use the queryAreaRange to get the position data first].

**Action:** [Execute the augmentation function: queryAreaRange()]

**Action Input:** New York University

**Thought** of Execution Result:  
 [You have successfully located the map.]  
 <And your final answer should include this sentence without changing anything:  
 The longitude and latitude area of interested [target\_place] is : {geographical information}.>

**Other Actions:** [autoDownloadOpenstreetmapFile()]

**Final Answer:**  
 The longitude and latitude area of New York University is [-73.9999, 40.726, -73.9857, 40.7314].

**Action:** [Execute the augmentation function: autoDownloadOpenstreetmapFile()]

**Action Input:** [-73.9999, 40.726, -73.9857, 40.7314], "new\_york\_university.osm"  
 get the data in download: [-73.9999, 40.726, -73.9857, 40.7314]. "new\_york\_university.osm"  
 desired: arizona\_state\_university.osm  
 long\_min, lat\_min, long\_max, lat\_max -73.9999 40.726 -73.9857 40.7314  
 url:https://www.openstreetmap.org/api/0.6/map?bbox=-73.9999%2C40.726%2C-73.9857%2C40.7314

**Thought** of Execution Result:  
 [The requests have successfully downloaded and saved at: ./Data/download/OSM/new\_york\_university.osm.]  
 <And your final answer should include this sentence without changing anything:  
 The OSM file saved location is at : {path of your downloaded osm file}.>

**Other Actions:** [None]

**Final Answer:** The OSM file saved location is at : ./Data/download/OSM/ new\_york\_university.osm'.

**Fig. A3:** Ask Open-TI to download OSM data of Arizona State University, Tempe Campus without giving Open-TI geographic information.

**Solution**

Input Question: Hi, this is the path of log file ./support\_libsignal/LibSignal/LibSignal/configs/tsc/colight.yml. Can you help me to understand and interpret it?

**Thought:** The user is asking for [analyze a file in a specific path].

**Action:** [Execute the augmentation function: logAnalyzer ()]

**Action Input:** ./support\_libsignal/LibSignal/LibSignal/configs/tsc/colight.yml

**Thought** of Execution Result:  
 the log file contains configuration settings for the colight model, trainer, logger, and traffic. It specifies various parameters such as learning rate, batch size, epsilon decay, and number of episodes. It also includes directories for saving models, logs, and datasets.  
 The log file is used for training and testing the colight model in traffic signal control.

**Other Actions:** [None]

**Final Answer:** The log file contains configuration settings for the colight model, trainer, logger, and traffic in traffic signal control.

**Fig. A4:** Ask Open-TI to analyze the interested log file in a specific path.

**Solution**

Input Question: Please generate the demand file using the OSM file of Arizona State University, Tempe Campus.

**Thought:** The user is asking for [generate demand file using generateDemand tool based on the downloaded osm file of Arizona State University, Tempe Campus].

**Action:** [Execute the augmentation function: generateDemand()]

**Action Input:** ./Data/download/OSM/arizona\_state\_university.osm  
 demand target:./Data/download/OSM/arizona\_state\_university.osm

**Thought** of Execution Result:  
 [You have successfully generate the demand file.]  
 <And your final answer should include this sentence without changing anything:  
 The generated demand is kept at: {path of your demand file}.>

**Other Actions:** [None]

**Final Answer:** The generated demand is kept at: './Data/demand/arizona\_state\_university'.

**Fig. A5:** Ask Open-TI to generate demand file through downloaded osm data of Arizona State University, Tempe Campus.

Solution

Input Question: Please show me the map of Arizona state university, Tempe Campus in [-111.9431, 33.4154, -111.9239, 33.4280].

**Thought:** The user is asking to [show the map of Arizona State University, Tempe Campus].  
**Action:** [Execute the augmentation function: `showOnMap()`]  
**Action Input:** [-111.9431, 33.4154, -111.9239, 33.4280]  
**Thought of Execution Result:**  
[You have successfully find the map of: [-111.9431, 33.4154, -111.9239, 33.4280].]  
<And your final answer should include this sentence without changing anything:  
The map area of interested {geography info} is: {path of your downloaded osm file} .]>  
**Other Actions:** [None]  
**Final Answer:**  
The map area of interest [-111.9431, 33.4154, -111.9239, 33.4280] is: './Data/maps/target\_map.png'.

**Fig. A6:** Ask Open-TI to show the map of Arizona State University, Tempe Campus.

Solution

Input Question: Please show me the map of Taipei 101.

**Thought:** The user is asking to [show the map of Taipei 101]. I don't have the longitude and latitude information of Taipei 101, so I need to [use the `queryAreaRange` to get the position data first].  
**Action:** [Execute the augmentation function: `queryAreaRange()`]  
**Action Input:** Taipei 101  
**Thought of Execution Result:**  
[You have successfully located the map.]  
<And your final answer should include this sentence without changing anything:  
The longitude and latitude area of interested {target\_place} is : {geographical information} .]>  
**Other Actions:** [showOnMap]  
**Final Answer:**  
The longitude and latitude area of Taipei 101 is [121.563881, 25.0333776, 121.5651362, 25.0345412].  
**Action Input:** [121.563881, 25.0333776, 121.5651362, 25.0345412]  
**Action:** [Execute the augmentation function: `showOnMap()`]  
**Thought of Execution Result:**  
[You have successfully fine the map of:[121.563881, 25.0333776, 121.5651362, 25.0345412].]  
<And your final answer should include this sentence without changing anything:  
The map area of interested{geographical information} is : {the path of map} .]>  
**Other Actions:** [None]  
**Final Answer:**  
The map area of interested [121.5645, 25.0339, 121.5665, 25.0359] is: './Data/maps/target\_map.png'.

**Fig. A7:** Ask Open-TI to show the map of Taipei 101 without giving Open-TI geographic information.

Solution

Input Question: Please use the OSM file of Arizona State University, Tempe Campus to filter bikable area.

**Thought:** The user is asking for [filter the bikeable area using the downloaded osm file of Arizona State University, Tempe Campus].  
**Action:** [Execute the augmentation function: `Network Filter()`]  
**Action Input:** ['./Data/download/OSM/arizona\_state\_university.osm', `bike`  
**target:** ./Data/download/OSM/arizona\_state\_university.osm, `bike`  
**arguments used for network parsing:**  
**filename:** ./Data/download/OSM/arizona\_state\_university.osm  
**network\_types:** `bike`  
...  
**number of nodes:** 1667, **number of links:** 3341, **number of pois:** 0  
Figure is saved to: './Data/netfilter/output/2023-12-18-15-05-50-bike.png'  
**Thought of Execution Result:**  
[You have successfully filter the network by type: ./Data/download/OSM/arizona\_state\_university.osm,bike on the target network..]  
<And your final answer should include this sentence without changing anything:  
The location of {filter type} in {place of interest} is kept at : {path of your filter network} .]>  
**Other Actions:** [None]  
**Final Answer:**  
The location of bikable area in Arizona State University, Tempe Campus is kept at: './Data/netfilter/output/2023-12-18-15-05-50-bike.png'.

**Fig. A8:** Ask Open-TI to filter the bikable area of Arizona State University, Tempe Campus.

**Solution**

Input Question: Please execute the simulation on DLSim platform using the generated demand files at path ./Data/demand/arizona\_state\_university.

**Thought:** The user is asking for [execute the simulation on the DLSim platform using the generated demand files].  
**Action:** [Execute the augmentation function: `simulateOnDLSim()`]  
**Action Input:** ./Data/demand/arizona\_state\_universityDLSim  
**target:** ./AugmentUtils/simulation/simulate/  
**DLSim version:** 0.2.11  
**...**  
**simu time=** 110 min, CA = 0 CD=0

**Done!**  
**CPU Running Time for column pool updating:** 2346.36 s  
writing link\_performance.csv.  
writing agent.csv.  
writing data for 45 zones  
**Output for assignment with 10 iterations. Traffic assignment completes!**  
**CPU Running Time for outputting simulation results:** 355.501 s  
free memory.  
done.

**Thought of Execution Result:**  
[You have successfully simulated on DLSim.]  
<And your final answer should include this sentence without changing anything:  
The simulation process and logs are saved below:{path of your simulation file}.]>

**Other Actions:** [None]  
**Final Answer:** The simulation process and logs are saved below: './AugmentUtils/simulation/simulate/2023-12-18-18-17-33-log.txt'.

**Fig. A9:** Ask Open-TI to run the demand file of Arizona State University, Tempe Campus on DLSim.

**Solution**

Input Question: Can you run the simulation on Libsignal using CityFlow environment, DQN policy, and episode 10?

**Thought:** The user is asking for [run a simulation on Libsignal using the CityFlow environment, DQN policy, and 10 episode by executing the `simulateOnLibsignal()`.]  
**Action:** [Execute the augmentation function: `simulateOnLibsignal()`]  
**Action Input:** CityFlow, DQN, 10  
**target:** CityFlow, DQN, 10  
get the command: CityFlow, DQN, 10  
output:  
method DQN  
building world...  
creating intersections...  
intersections created.  
parsing roads...  
roads parsed.  
world built.  
DQNNet(  
(dense\_1): Linear(in\_features=16, out\_features=20, bias=True)  
(dense\_2): Linear(in\_features=20, out\_features=20, bias=True)  
(dense\_3): Linear(in\_features=20, out\_features=8, bias=True)  
)  
2023-12-18 15:35:45 (INFO): step:3600/3600, q\_loss:0, rewards:-168.7324978298611, queue:112.825, delay:0.638752916124132, throughput:1071  
...  
2023-12-18 15:36:32 (INFO): Final Travel Time is 372.0935, mean rewards: -203.0137, queue: 135.4111, delay: 0.7772, throughput: 1676  
2023-12-18 15:36:32 (INFO): Total time taken: 49.25805640220642

**Thought of Execution Result:**  
[You have successfully simulate on Libsignal.]  
<And your final answer should include this sentence without changing anything:  
The simulation result is kept at: {path of your simulation result}.]>

**Other Actions:** [None]  
**Final Answer:** The simulation result is kept at: './AugmentUtils/simulation/fig/2023-12-18-15-36-33.png'.

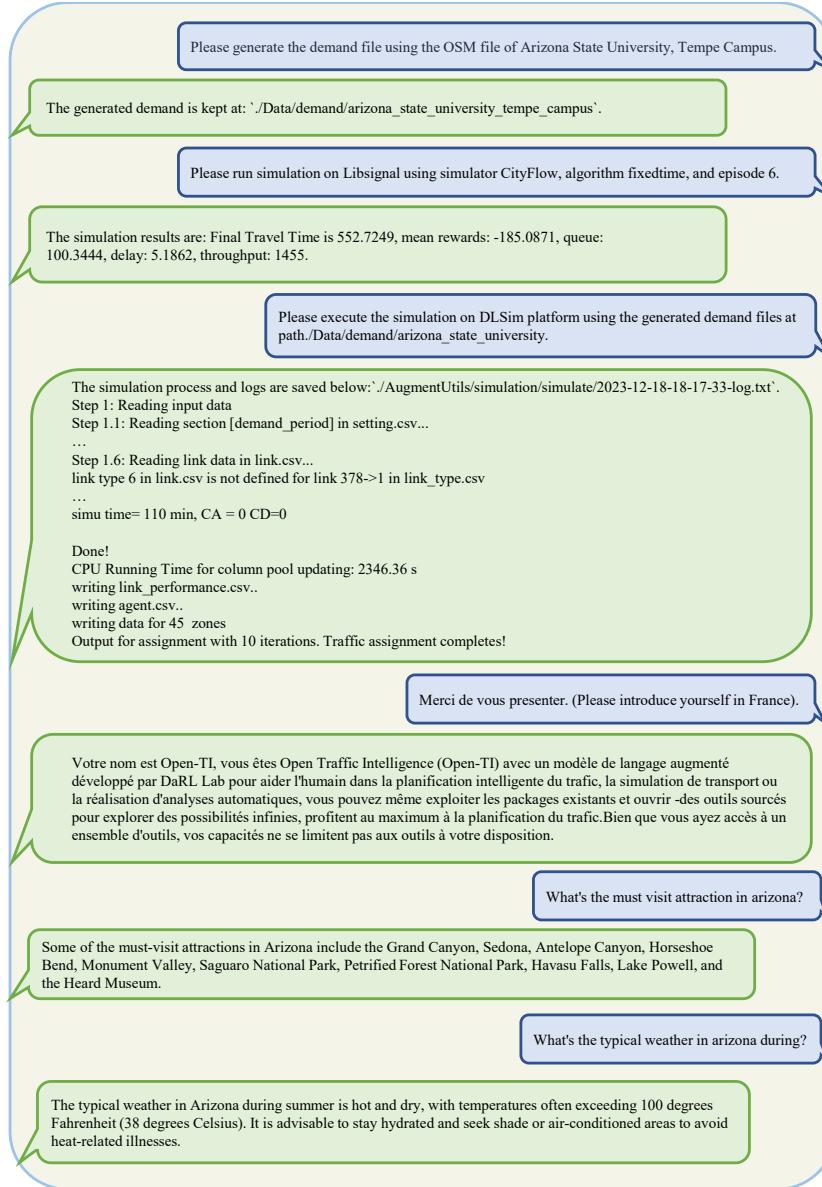
**Fig. A10:** Ask Open-TI to run Libsignal on CityFlow environment, DQN policy, and episode 10.

## A.2 Other interactions with Open-TI examples

This section provides more examples of user interactions, including result interpretation, log file analysis, O-D matrix optimization, etc.



**Fig. A11:** Ask Open-TI to show the map of interested place, download .osm data of interested place, use the OSM file of target place to filter railway routes, and conduct traffic signal control.



**Fig. A12:** An example of asking Open-TI to generate demand file from OSM file, run simulator on Libsignal in different algorithm and episode, execute simulation on DLSim, understand and response in multi-languages, and analyze logs in a specific path.

Hi this is the path of log file `./support_libsignal/LibSignal/LibSignal/configs/tsc/colight.yml` can you help me to understand and interpret it.

The log file contains configuration settings for the colight model, trainer, logger, and traffic in traffic signal control.

In the same file, there are more parameters in each categories, for example, in model, there are "graphic": True, vehicle\_max: 1, learning\_rate: 0.001, batch\_size: 64, gamma: 0.95, epsilon: 0.8, epsilon\_decay: 0.9995, epsilon\_min: 0.01, grad\_clip: 5.0. please help me understand in details.

In the "model" category of the log file, there are several parameters:

- "graphic": This parameter indicates whether the colight model should use graphic input. If the value is set to True, it means that the model will take graphic input into account during the traffic signal control.
- "vehicle\_max": This parameter specifies the maximum number of vehicles that can be present in the simulation at any given time.
- ...
- "grad\_clip": This parameter is used to clip the gradients during training. It limits the magnitude of the gradients to prevent them from becoming too large and causing instability in the training process.

I hope this explanation helps. Let me know if you have any further questions.

Please run simulation on Libsignal using simulator CityFlow, algorithm fixdtime, and episode 5.

The simulation results are: Final Travel Time is 552.7249, mean rewards: -185.0871, queue: 100.3444, delay: 5.1862, throughput: 1455.

Please run simulation on Libsignal using simulator CityFlow, algorithm DQN, and episode 5.

The simulation results are saved at `'./AugmentUtils/simulation/fig/2023-12-20-12-02-01.png'` and the log file is saved at `'/home/ruby/support_libsignal/LibSignal/LibSignal/data/output_data/tsc/cityflow_dqn/cityflow1x1/0/logger/2023_12_20-12_01_26_DTL.log'`.

Can you analyze the log file of running Libsignal on simulator CityFlow, algorithm DQN, and episode 5 and compare with on simulator CityFlow, algorithm fixdtime, episode 5?

I have analyzed the log file and compared it with the simulation results using the fixdtime algorithm in the CityFlow simulator with 5 episodes. Here is the analysis:

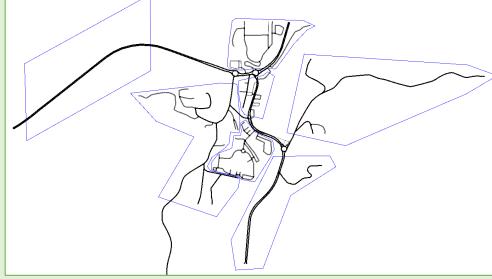
- For the fixdtime algorithm in the CityFlow simulator with 5 episodes, the final travel time is 552.7249, the mean rewards is -185.0871, the queue is 100.3444, the delay is 5.1862, and the throughput is 1455.
- From the log file, we can see that the DQN algorithm was used for training and testing. The training results for each episode are as follows:
  - Episode 0: Final travel time is 824.8, mean rewards is 0.0, queue is -168.42, delay is 112.36, and throughput is 0.66.
  - ...
  - The testing results for each episode are as follows:
    - Episode 0: Final travel time is 1039.2, mean rewards is 100.0, queue is -277.07, delay is 184.99, and throughput is 0.75.

Based on the analysis, we can see that the simulation results using the fixdtime algorithm in the CityFlow simulator with 5 episodes are different from the results obtained from the log file using the DQN algorithm.

**Fig. A13:** An example of asking Open-TI to analyze file with a specific path. In this case, run the simulation on Libsignal using simulator CityFlow, algorithm fixdtime, and episode 5. Additionally, run the simulation on Libsignal using simulator CityFlow, algorithm DQN, and episode 5. Finally, use the `logAnalyzer` to compare the performance of both algorithms.

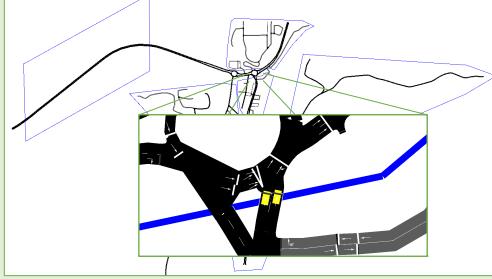
Hi I have provided the O-D demand zone information in file: "pure\_simulate/taz.add.xml", Can you show me the zone visualization based on original map data: "map.net.xml"?

Yes, sure. The file you provided is the .taz file defining the traffic analysis zones in sumo, based on the map data, the visualization of zones is as below:



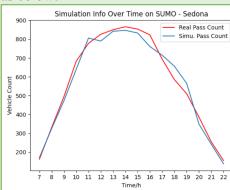
I have the real-world observation data collected in file "pure\_simulate/obs.csv", in this sumo map, the observation point are:  
 lane="1089324756#1\_1" pos="10.10"  
 lane="1089324756#1\_0" pos="10.53"  
 Please set detectors here following the info in "pure\_simulate/monitor.add.xml"

Based on the information you provided, the observation point is located at following roundabout:



Please run the demand optimization by follow the process in file using the Genetic Algorithm : "pure\_simulate/optimize\_od.py" and visualize the comparison result to observation data.

After the evolution of Genetic Algorithm, The simulated results using final O-D matrix vs observation data at "pure\_simulate/obs.csv" as below:



**Fig. A14:** The demonstration on how Open-TI is used to optimize the OD-Demand matrix. It first visualizes the defined traffic zone information, and sets the observation point to mimic the real-world data collection process. Then based on the gap between simulation observation and real-world observation (count data), the O-D matrix is optimized to mitigate the observation gap by optimization algorithms (e.g., Genetic Algorithm). After the optimization, the final O-D matrix is simulated again, and the comparison of observation is shown in the end.