

OPEN-TI: OPEN TRAFFIC INTELLIGENCE WITH AUGMENTED LANGUAGE MODEL

Longchao Da, Kuanru Liou, Tiejin Chen, Xuesong Zhou, Xiangyong Luo, Yezhou Yang & Hua Wei *
Arizona State University, Tempe, AZ, USA
{longchao, kliou, tchen169, xzhou74, xluo70, yz.yang, hua.wei}@asu.edu

ABSTRACT

Transportation has greatly benefited the cities' development in the modern civilization process. Intelligent transportation, leveraging advanced computer algorithms, further increases commuting efficiency, however, as a cross-discipline, it 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 to understand and execute intricate commands, we introduce **Open Traffic Intelligence**, **Open-TI**. As a bridge to mitigate the industry-academic gap, **Open-TI** is an innovative language agent augmented with the capability to harness external analysis packages according to conversations. Marking its distinction, **Open-TI** is the first method capable of conducting exhaustive traffic analysis from scratch - from map acquisition to execution in complex simulations. Besides, **Open-TI** can 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 to choose from the action space by understanding the expected intentions of human descriptive policy. We provide the code including the implementation structure and will invite further community-driven enhancements, the demo video is available: <https://youtu.be/QLiMvyXaQeM>.

1 INTRODUCTION

Traffic and Transportation play important roles in the process of human civilization. Intelligent transportation is casting hope to further benefit people's daily lives through optimal controlling and scheduling decisions. The efficient modern transportation comes from joint efforts from many researchers in various directions like: map modeling (Yukawa & Kikuchi, 1995), traffic simulation (Chao et al., 2020), schedule optimization (Dai et al., 2020), etc., and still, there are multiple ongoing challenges regarding the multi-resolution traffic simulation (Zhou et al., 2022), optimal traffic signal control policies (Wei et al., 2019b), dynamic demand dispatch adjustment (Osorio, 2019), 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 (Lopez et al., 2018), CityFlow (Zhang et al., 2019), VISSIM (Fellendorf & Vortisch, 2010). 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.

To bridge the gap, two steps are proposed: 1. Unify the simulation and analysis process using General Modeling Network Specification (GMNS) (Lu & Zhou, 2023), providing a common format for sharing road network files suitable for multi-modal transportation planning. 2. Develop an intelligent system with self-explanatory abilities, integrating domain-specific tasks, state-of-the-art algorithms, and powerful simulators. This system should allow easy execution with interactive explanations. Leveraging large language models (LLMs) has become crucial in aiding humans, benefiting transportation tasks (Zhang et al., 2023; de Zarzà et al., 2023). Large-scale pre-trained models

*Corresponding Author

such as Llama (Touvron et al., 2023), GPT-4 (Achiam et al., 2023), 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 (Vaithilingam et al., 2022), DrugGPT (Li et al., 2023b), TrafficGPT (Zhang et al., 2023), GraphGPT (Tang et al., 2023), 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 Mialon et al. (2023). Defined as **Augmented Language Models (ALMs)**, this refers to ‘language models (LMs) that are augmented with reasoning skills and the ability to use tools’ (Mialon et al., 2023).

Inspired by ALMs, we propose to design a rudiment of general 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.

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

- 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, Open-TI is able to conduct traffic analysis from scratch (from downloading map to provide simulation on the interested area).
- We realized multiple domain-specific tasks like traffic signal control policy training and traffic demand optimization under the unified implementation class.
- We design the ChatZero 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, and the ChatZero will conduct the decision making based jointly on the observation of current traffic situation and described policies.
- Open-TI is an open-ended platform and encourages researchers to explore more extensions and augmentation for future study, We have released the code at repository ¹.

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) (Devlin et al., 2018; Brown et al., 2020; Chowdhery et al., 2023) represent dramatic progress in Natural Language Processing (NLP) and are already core in several products with millions of users, such as the coding assistant Copilot (Chen et al., 2021), Bert enhanced search engine², ChatGPT and GPT4 (Achiam et al., 2023). LLMs can execute multiple tasks from language understanding to text generation, thus paving a path towards more intelligent human-computer interactions, or even benefit other research domains by its inference ability (Da et al., 2023). 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, such as Li et al. (2023a) design the API bank to execute API calls to meet human needs, Wang et al. (2023) applied the augmented language models to the medical field to serve as a more flexible knowledge hub for doctors and patients. Liang et al. (2023) 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 and physical domains. Our work is the first to explore the augmented language agents on automatic traffic intelligence that realize a throughout traffic analysis.

¹Code released at: <https://github.com/DaRL-LibSignal/OpenTI>

²<https://blog.google/products/search/search-language-understanding-bert/>

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 reference for planning strategies, offering insights into traffic patterns, road network efficiencies, and the potential impacts of infrastructural changes as shown in Figure 1. The utilization of traffic simulation models facilitates the analysis of traffic behavior under various conditions without the need for costly real-world alterations.



Figure 1: The traffic and transportation simulation in cities, (a) is a real-world traffic image, (b) is the simulation of traffic flow in DTALite (Tong et al., 2019).

With the development of autonomous vehicles (AVs), traffic simulation for safety tests and validation become important. The simulator developed by NVIDIA (NVIDIA, 2023) tends to provide a more physically accurate simulation platform and Waymax (Gulino et al., 2023), 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 (Behrisch et al., 2011), CityFlow (Zhang et al., 2019), and DLSim (Tong et al., 2019), 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 and reducing congestion in modern transportation systems. Recent advances in reinforcement learning (RL) techniques have shown superiority over traditional approaches in TSC (Wei et al., 2018). 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, 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. This paper presents Open-TI can conduct multiple RL algorithms for TSC problems and provide insights analysis from the results.

OD Matrix Optimization Origin-destination (O-D) matrix optimization is a critical component in the field of transportation planning and traffic engineering. It involves the creation and optimization of a matrix to represent the number of trips between various origins and destinations in a given area over a specified time period. Modern techniques tackle this task in two directions, enhance data representation (Zhou et al., 2013; Krishnakumari et al., 2020a; Fedorov et al., 2019; Pamuła & Żochowska, 2023; Fu et al., 2022; Kumarage et al., 2023) and improve O-D matrix precision (Krishnakumari et al., 2020b; Oliveira et al., 2020). This paper implemented a genetic algorithm-based O-D matrix optimization process that can be controlled by Open-TI.

3 THE ARCHITECTURE OF OPEN-TI

3.1 OVERVIEW OF OPEN-TI

To take a step forward to more general 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

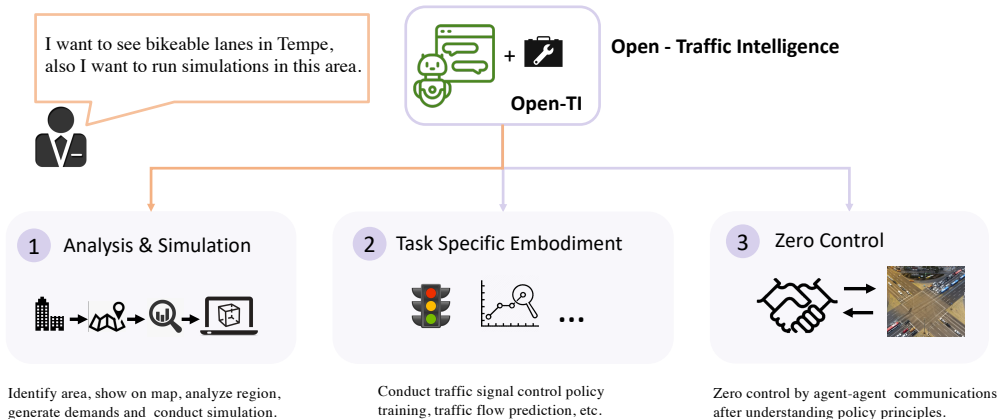


Figure 2: The overview of Open-TI functionalities.

by agent augmentations. Primarily, a user-friendly interface is designed as shown in Appendix Figure 7. 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 Figure 2.

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) (Tong et al., 2019) or SUMO(Behrisch et al., 2011).

Second, 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 (Mei et al., 2023), 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.

As shown in Figure 3, Open-TI consists of two components, the Traffic LLMs and Augmentations. When a user’s 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 information from dialogues, then, the information will be compared with tools in the augmentation pool. After matching, if the agent finds a possible solution, it will execute with extracted information and generate the output with a multimedia data form.

3.2 PROMPT DESIGN

Open-TI exploited the LLM’s understanding abilities and tuned its behavior by prompt engineering, which is essential for well-organized operation. We have designed 5 aspects of prompt structure: **[Description]**, **[Example]**, **[Format Restriction]**, **[Emphasis]** and **[Reflection]**. And we verified their effectiveness in cross-task behaviors by ablation experiment. In this section, we provide

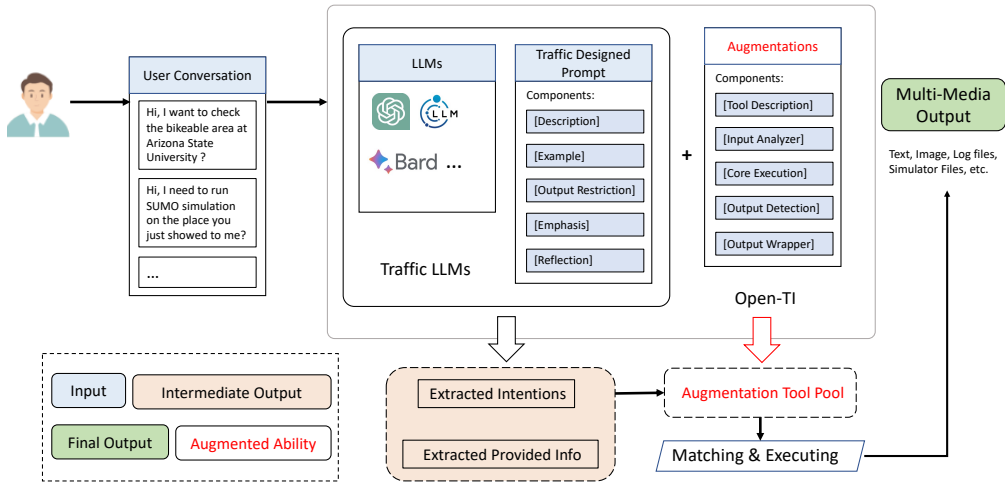


Figure 3: The design framework of Open-TI.

Table 1: The details of the prompt components for queryAreaRange

| Name | Purpose | Instance |
|---------------------------|---|--|
| <i>Description</i> | 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. |
| <i>Format Restriction</i> | 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] |
| <i>Example</i> | 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]. |
| <i>Reflection</i> | Remind LLMs not to engage in unnecessary tasks and ensure that each process is executed precisely. | You should respond directly with what you know and then stop, do not look for the location or attempt to find it online. |
| <i>Emphasis</i> | 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. |

details of prompt design as shown in Figure 3 including the purpose of the prompt and example cases. The **Instances** in Table 1. are from the task of queryAreaRange.

3.3 EXECUTION AND AUGMENTATION LIST

The overall execution process is expressed in Appendix 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 4 of Appendix. 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.

```

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

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.

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 2. And when the user asks about related tasks, Open-TI will automatically match the highest probability option and process following Algorithm 1 in Appendix A.3. An example of interaction is shown in Appendix Figure 18.

Table 2: 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 |

4.2 TASK-SPECIFIC EMBODIMENT

Open-TI is capable of realizing more general research tasks in the traffic domain. Including traffic signal control (TSC), 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 Open-TI achieves the three demonstrating tasks in the following subsections.

Traffic O-D Demand Optimization Task 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. 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. For the example case in Appendix Figure 22, the given data is realistic 16-hour count data of a roundabout area at Sedona, AZ, USA, and we embodied the agent to use a genetic algorithm to conduct optimization and provide the result.

Traffic Signal Control Task In the realization of traffic signal control embodiment, we seamlessly integrated the Libsignal (Mei et al., 2023) 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 (Cools et al., 2013)) and reinforcement-learning-based approaches:

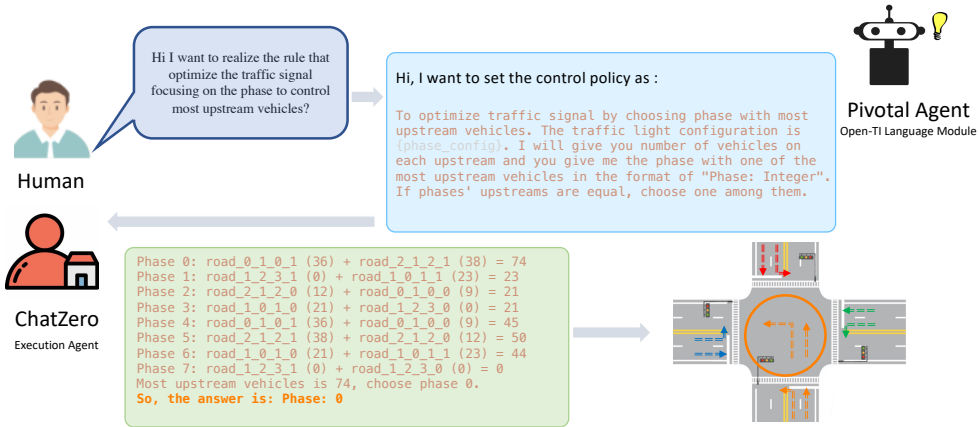


Figure 4: The demonstration ChatZero Meta Control.

DQN (Mnih et al., 2013), PPO (Schulman et al., 2017), CoLight (Wei et al., 2019b), MPLight Chen et al. (2020), MADDPG (Lowe et al., 2017), MaxPressure (Varaiya, 2013), PressLight (Wei et al., 2019a). We provide further interaction cases in the Appendix. A.7.

4.3 AGENT META CONTROL

In the meta-control task, we 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 into instructions, process, digest, and provide a self-explainable traffic signal action. As shown in Figure 4, the pivotal agent extracts human’s descriptive policy (objectives), combined with knowledge of traffic phase setting and current lane-level traffic states (Phase indicates the traffic light set, lane ID follows with the current vehicle amount on the specific lane) as the input for the ChatZero, and acquires traffic light control actions, the quality of control will be evaluated by traffic indicators like average travel time, throughput, etc.

5 EXPERIMENT

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

- RQ1: How does Open-TI compare to other baseline methods in terms of stability performance?
- RQ2: How do the embodiment components contribute to Open-TI performance?
- 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 augmentation 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 EXPERIMENT DESIGN

RQ1: Language Agent Analysis on the API Calls Following the work of (Li et al., 2023a), we analyze three types of API call abnormal behaviors, namely ‘No API Call Rate’, ‘API Mismatching Rate’, and ‘Error Raise Rate’ and compare with the baseline method known as TrafficGPT (Zhang et al., 2023). We adopted $T = 6$ similar tasks as shown in the top of table 3. to conduct the comparison by testing each task 20 times. And calculate the error rate, we have: $Error Rate = \frac{1}{T} \sum_{t=1}^T \frac{n_t^e}{n_t^c} = \frac{1}{T} \sum_{t=1}^T \frac{1}{n_t^c} \sum (c_t^{no}, c_t^{miss}, c_t^{error})$ where n_t^e represents the number of error occurrences for task t during total tests, n_t^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 $n_t^c = c_t^{no} + c_t^{miss} + c_t^{error}$. For evaluation of each dimension shown in Figure 5, denote ρ 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 $\rho_{error} = \frac{1}{T} \sum_{t=1}^T \frac{c_t^{error}}{n_t^c}$.

RQ2: Ablation Study of Embodiment Prompt Structure In this experiment, we tend to verify the contribution of each component in the current design prompt structure. By gradually removing the key component in an order of *Emphasis*, *Reflection*, *Format Restriction*, *Example*, and *Description*, we investigate the drop of performance, here we adopt accuracy as measurement, the *Accuracy Rate = 1 - Error Rate*.

RQ3: Analyze the MetaControl Performance in Various Versions of LLMs To verify the effectiveness of the meta agent control, 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. 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 (Wei et al., 2021), the evaluation metrics are *Average Travel Time (ATT)*, *Throughput (ATT)*, *Reward*, *Queue*, *Delay*. Detailed explanations are shown in the Appendix A.2.

Table 3: 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.

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

5.1.1 EXPERIMENTAL RESULTS

Results for RQ1: Analysis on the API Calls As shown in Figure 5 (a). 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. We also provide results in three different evaluation dimensions in Appendix , Open-TI consistently performs more stable (lower error rate and lighter color) than the baseline method. We notice Open-TI performs slightly non-ideal enough in Error Raise Rate, it is because showOnMap is a more complex task consisting of two consecutive sub-tasks, which are 1) identifying the geographic information and 2) request and visualize map data, this may cause connection problem between sequential executions.

Results for RQ2: Ablation Study on Open-TI Figure 5 reflects the effect of each component to Open-TI. We can observe that each part plays an important role in its final robust operation. When the *Emphasis* is removed, the accuracy suffers a slight drop across queryAreaRange to autoDownloadOpenStreetMapFile, because the agent can be confused among similar keywords task descriptions, then leading to mismatching. Removing *Reflection* slightly impacts the overall stability mainly due to the given deceptive description, the agent conducting extra steps searching around but instead, it should concisely ask for clearer requests. *Format Restriction* notably impacts showOnMap and simulateOnLibsignal. Its removal results in a 25% accuracy drop for showOnMap and 55% for simulateOnLibsignal. Because *Format Restriction* controlling input format limits, preventing errors caused by entering incorrect

information into a project. The *Example* significantly aids language agents in task comprehension. Its removal leads to a decrease in accuracy rates by 45% and 55% for showOnMap and autoDownloadOpenStreetMapFile, respectively.

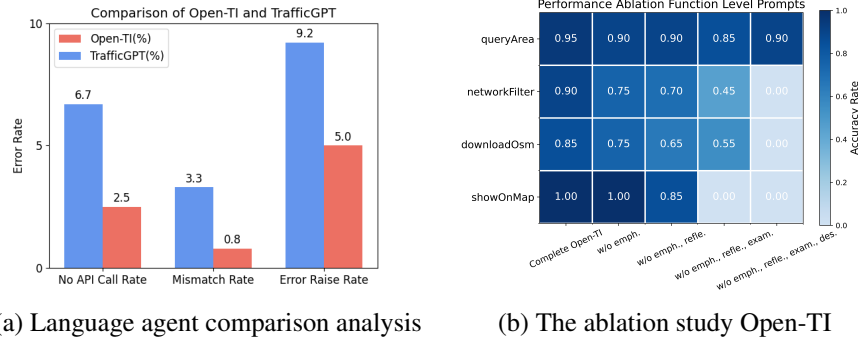


Figure 5: An illustration of Open-TI performance. In (a), three types of error rate comparisons are considered. In (b), ablation study on Open-TI. The x-axis from left to right shows the gradual removal of each prompt module. The y-axis shows the individual task, cell color from dark to light reflects the accuracy drop.

RQ3: MetaControl Performance in Various Versions of LLMs The experiment results are shown in the Figure 6. From 5 metrics: throughput (TP) and average travel time (ATT) and etc., 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.

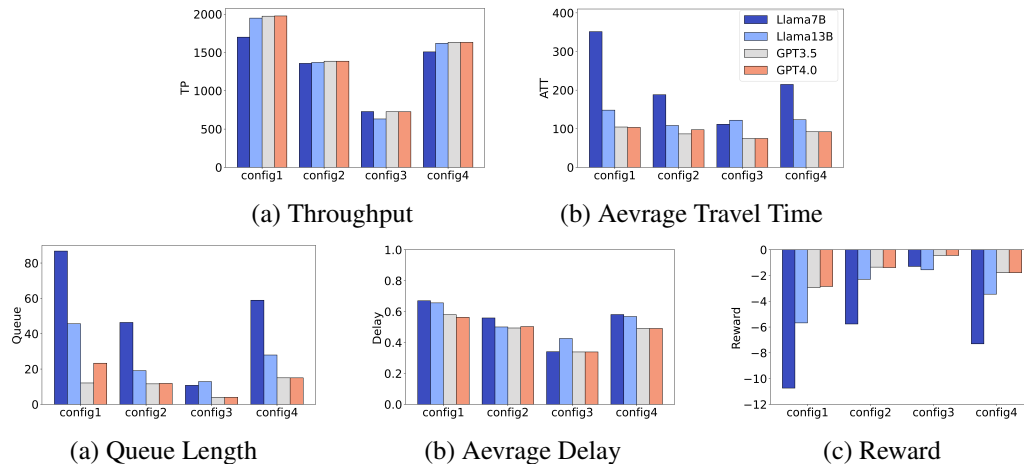


Figure 6: The ChatZero performance in TP and ATT across the 4 LLMs. Each LLM is tested to control the traffic signal in 4 different configs of road situation

6 CONCLUSION

In this paper, we propose Open-TI, an intelligent traffic analysis agent leveraging the large language models' contextual abilities and augmenting with traffic domain-specific tools, which could provide not only 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 signals by an agent-agent communication scheme named ChatZero, which casts insight for the self-explainable control practice. We also found that sequential action practice is vulnerable to action error, it is a direction for future exploration to focus on improving the accuracy of the multi-step action embodiment. The demo video is publicly available on site <https://youtu.be/QLiMvyXaQeM>.

Acknowledgement The work was partially supported by NSF award #2153311. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo—simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Qianwen Chao, Huikun Bi, Weizi Li, Tianlu Mao, Zhaoqi Wang, Ming C Lin, and Zhigang Deng. A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. In *Computer Graphics Forum*, volume 39, pp. 287–308. Wiley Online Library, 2020.
- Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3414–3421, 2020.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.
- Seung-Bae Cools, Carlos Gershenson, and Bart D’Hooghe. Self-organizing traffic lights: A realistic simulation. *Advances in applied self-organizing systems*, pp. 45–55, 2013.
- Longchao Da, Minchiuan Gao, Hao Mei, and Hua Wei. Llm powered sim-to-real transfer for traffic signal control. *arXiv preprint arXiv:2308.14284*, 2023.
- Zhuang Dai, Xiaoyue Cathy Liu, Xi Chen, and Xiaolei Ma. 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.
- I de Zarzà, J de Curtò, Gemma Roig, and Carlos T Calafate. Llm multimodal traffic accident forecasting. *Sensors*, 23(22):9225, 2023.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Aleksandr Fedorov, Kseniia Nikolskaia, Sergey Ivanov, Vladimir Shepelev, and Alexey Minbaleev. Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data*, 6:1–15, 2019.
- Martin Fellendorf and Peter Vortisch. Microscopic traffic flow simulator vissim. *Fundamentals of traffic simulation*, pp. 63–93, 2010.
- Hao Fu, William HK Lam, Hu Shao, Lina Kattan, and Mostafa Salari. 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.

- Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *arXiv preprint arXiv:2310.08710*, 2023.
- Panchamy Krishnakumari, Hans Van Lint, Tamara Djukic, and Oded Cats. A data driven method for od matrix estimation. *Transportation Research Part C: Emerging Technologies*, 113:38–56, 2020a.
- Panchamy Krishnakumari, Hans Van Lint, Tamara Djukic, and Oded Cats. A data driven method for od matrix estimation. *Transportation Research Part C: Emerging Technologies*, 113:38–56, 2020b.
- Sakitha Kumarage, Mehmet Yildirimoglu, and Zuduo Zheng. A hybrid modelling framework for the estimation of dynamic origin–destination flows. *Transportation Research Part B: Methodological*, 176:102804, 2023.
- Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023a.
- Yuesen Li, Chengyi Gao, Xin Song, Xiangyu Wang, Yungang Xu, and Suxia Han. Druggpt: A gpt-based strategy for designing potential ligands targeting specific proteins. *bioRxiv*, pp. 2023–06, 2023b.
- Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023.
- Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pp. 2575–2582. IEEE, 2018.
- Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.
- Jiawei Lu and Xuesong Simon Zhou. 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.
- Hao Mei, Xiaoliang Lei, Longchao Da, Bin Shi, and Hua Wei. Libsignal: an open library for traffic signal control. *Machine Learning*, pp. 1–37, 2023.
- Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- NVIDIA. Simulation for self-driving vehicles. 2023. URL <https://www.nvidia.com/en-us/self-driving-cars/simulation/>.
- Andréa Leda Ramos de Oliveira, Monique Filassi, Bruna Fernanda Ribeiro Lopes, and Karina Braga Marsola. Logistical transportation routes optimization for brazilian soybean: an application of the origin-destination matrix. *Ciência Rural*, 51:e20190786, 2020.
- Carolina Osorio. 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.

- Teresa Pamuła and Renata Żochowska. 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.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*, 2023.
- Lu Tong, Yuyan Pan, Pan Shang, Jifu Guo, Kai Xian, and Xuesong Zhou. 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.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 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.
- Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.
- Yubo Wang, Xueguang Ma, and Wenhui Chen. Augmenting black-box llms with medical textbooks for clinical question answering. *arXiv preprint arXiv:2309.02233*, 2023.
- Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 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.
- Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1290–1298, 2019a.
- Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. 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, 2019b.
- Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18, 2021.
- Satoshi Yukawa and Macoto Kikuchi. Coupled-map modeling of one-dimensional traffic flow. *Journal of the Physical Society of Japan*, 64(1):35–38, 1995.
- Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*, pp. 3620–3624, 2019.
- Siyao Zhang, Daocheng Fu, Zhao Zhang, Bin Yu, and Pinlong Cai. Trafficgpt: Viewing, processing and interacting with traffic foundation models. *arXiv preprint arXiv:2309.06719*, 2023.
- Xuesong Zhou, C Lu, and Kuilin Zhang. Dynamic origin-destination demand flow estimation utilizing heterogeneous data sources under congested traffic conditions, 2013.
- Xuesong Simon Zhou, Qixiu Cheng, Xin Wu, Peiheng Li, Baloka Belezamo, Jiawei Lu, and Mohammad Abbasi. 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.

A APPENDIX

A.1 OVERVIEW OF OPEN-TI

In this section, we present the interface of Open-TI. The interface comprises four parts: **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.

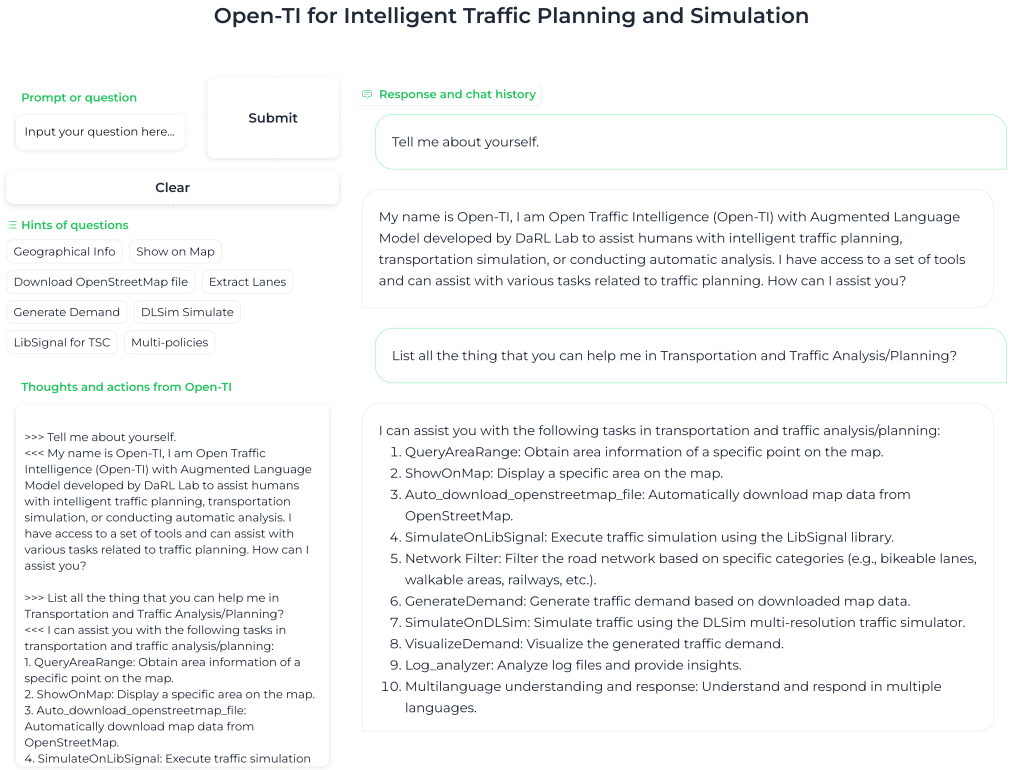


Figure 7: Open-TI conversation interface

A.2 METRICS EXPLANATION FOR TRAFFIC SIGNAL CONTROL

In this section, we provide a detailed explanation of the metrics we adopted in the experiment of ChatZero traffic signal control performance. • *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.

A.3 EXECUTION PROCESS

We provide the pseudo-code for Open-TI execution process in Algorithm 1.

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  $re \leftarrow ResponseToUser$ 

```

A.4 DETAILS OF AUGMENTED TOOLS IMPLEMENTED IN OPEN-TI

In this section, we will provide a list of augmented tools implemented in Open-TI. There are 12 augmented tools in total and each of them can do different tasks to help Open-TI answer user’s question. We show the detailed 12 augmented tools in Table 4

A.5 API ANALYSIS AND CASE STUDY

In this section, we will present definition of the meaning of each kind of API error in Table 5. Three types of incorrect API usage will be discussed: No API Call, API Mismatch, and Error Raise. The table includes questions, correct and incorrect answers, and an analysis of the reasons behind each error.

Table 4: 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. |

Table 5: 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 sometimes has no awareness of existence of specific functionality. |
| | | ✗ : Sorry, I don't have the valid tool to download OSM file. | |
| 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. | |

A.6 DETAILED RESULTS ACROSS THREE DIMENSIONS API EVALUATION

In this section, we will provide more detailed experimental data regarding the API evaluation. There are six tasks and three kinds of API evaluation. We will compare Open-TI with baseline model TrafficGPT and find out performance of each model.

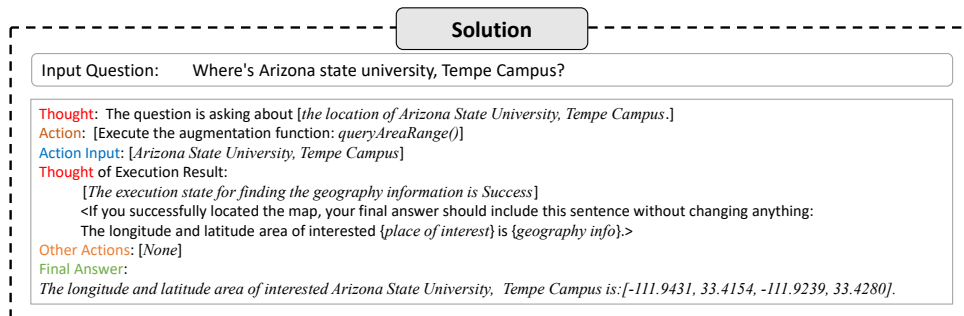
Table 6: Error Rate of Open-TI and TrafficGPT

| 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 |

A.7 THOUGHT CHAIN PROCESS EXAMPLES

In this section, we provide more Chain-of-thought (CoT) process examples, as a reflection on given a task, how 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.

**Figure 8:** Ask Open-TI to get the geographic information of Arizona State University, Tempe Campus.

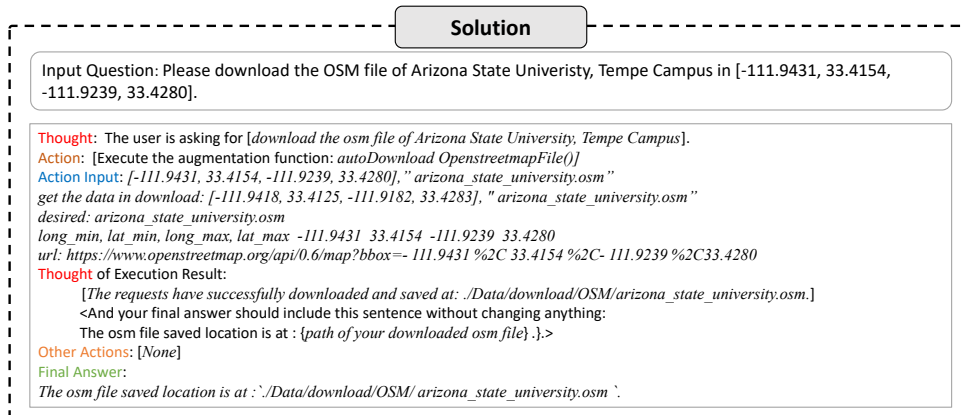


Figure 9: Ask Open-TI to download OSM data of Arizona State University, Tempe Campus.

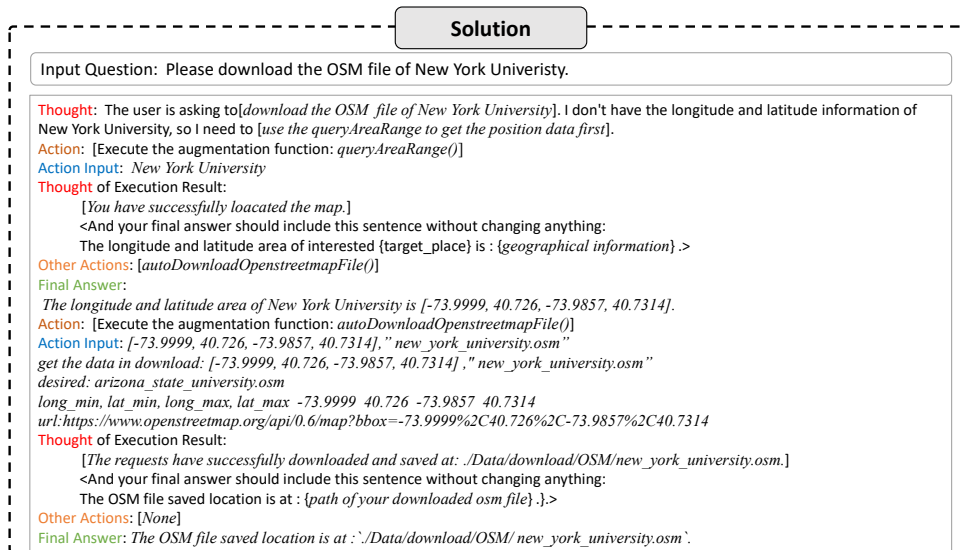


Figure 10: Ask Open-TI to download OSM data of Arizona State University, Tempe Campus without giving Open-TI geographic information.

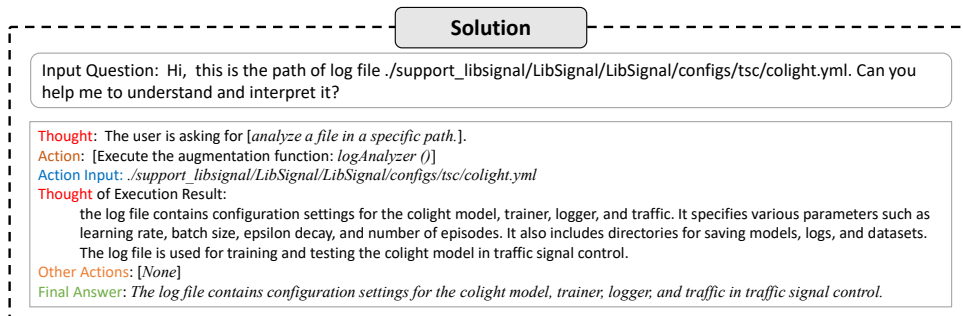


Figure 11: Ask Open-TI to analyze the interested log file in a specific path.

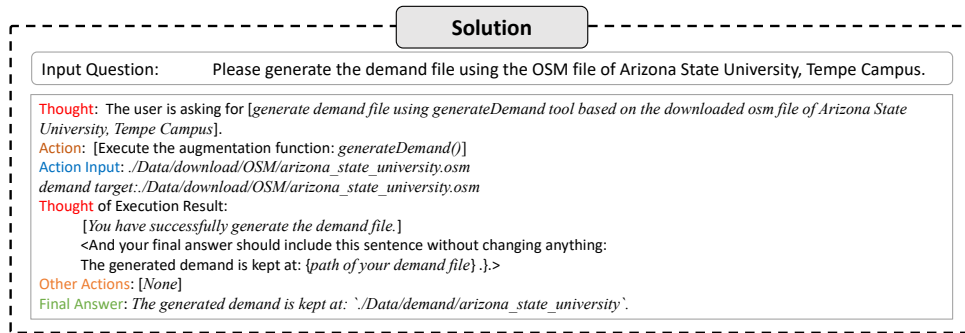


Figure 12: Ask Open-TI to generate demand file through downloaded osm data of Arizona State University, Tempe Campus.

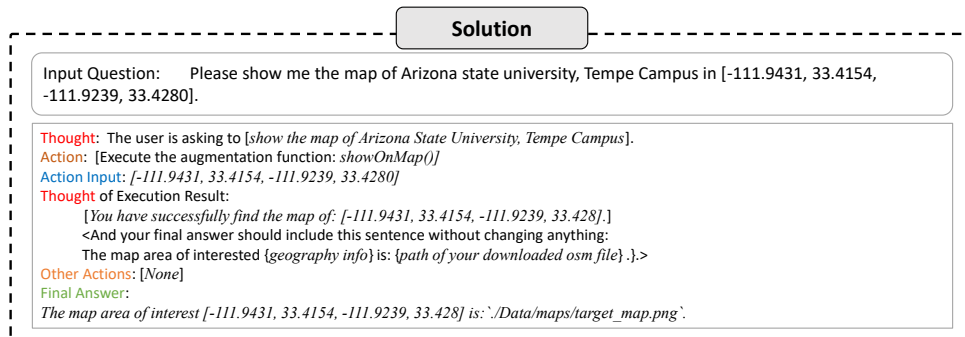


Figure 13: Ask Open-TI to show the map of Arizona State University, Tempe Campus.

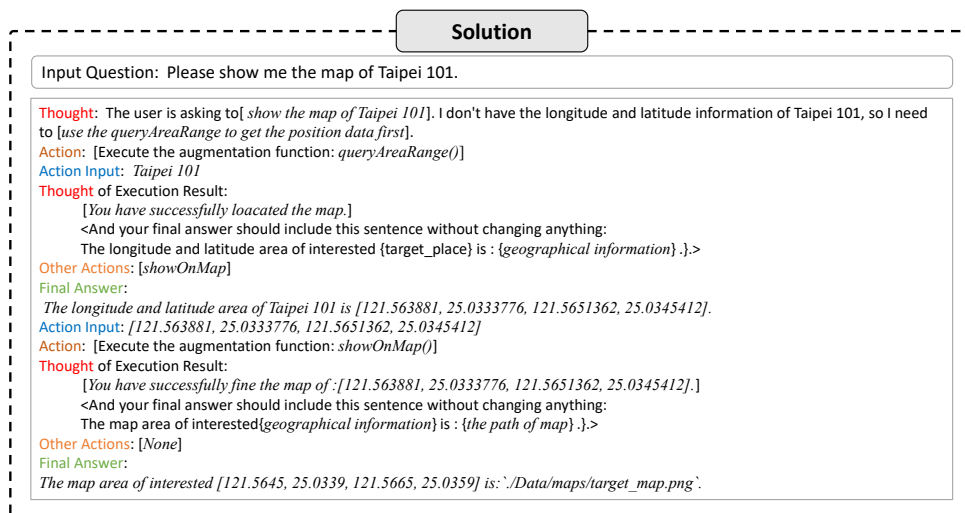


Figure 14: Ask Open-TI to show the map of Taipei 101 without giving Open-TI geographic information.

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`.

Figure 15: Ask Open-TI to filter the bikable area of Arizona State University, Tempe Campus.

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`.

Figure 16: Ask Open-TI to run Libsignal on CityFlow environment, DQN policy, and episode 10.

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`.*

Figure 17: Ask Open-TI to run the demand file of Arizona State University, Tempe Campus on DLSim.

A.8 INTERACTIONS WITH OPEN-TI EXAMPLES

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

Hi tell me about yourself?

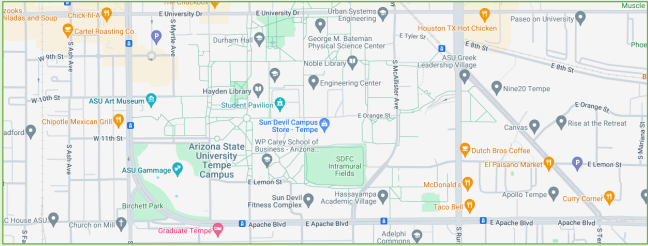
I am Open-TI, an Open Traffic Intelligence with Augmented Language Model (ALM) developed by DaRL Lab. I am designed to assist humans with intelligent traffic planning, traffic simulation, and automatic analysis. I have access to a set of tools and can operate existing packages and open-sourced libraries.

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].

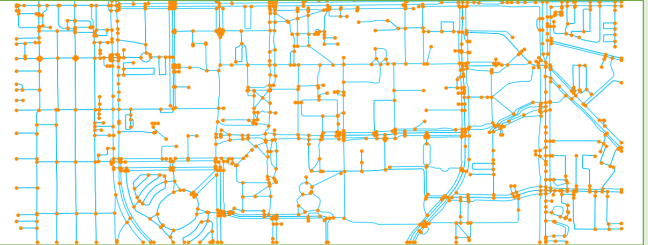
Can you show me this area on map?

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:

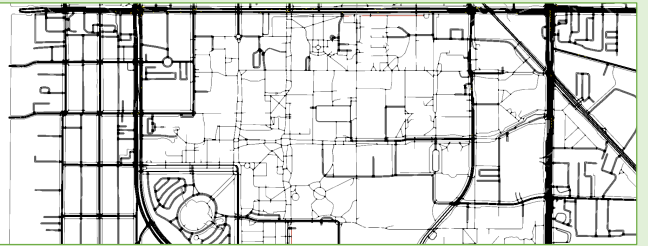


Figure 18: The demonstration pivotal agent control. (Right: The user messages, Left: The responses from Open-TI). This series of interactions shows how to query geography information of a location, how to visualize on the map, filter the interested lane types, and use the arbitrary map for automatic traffic simulation (SUMO).

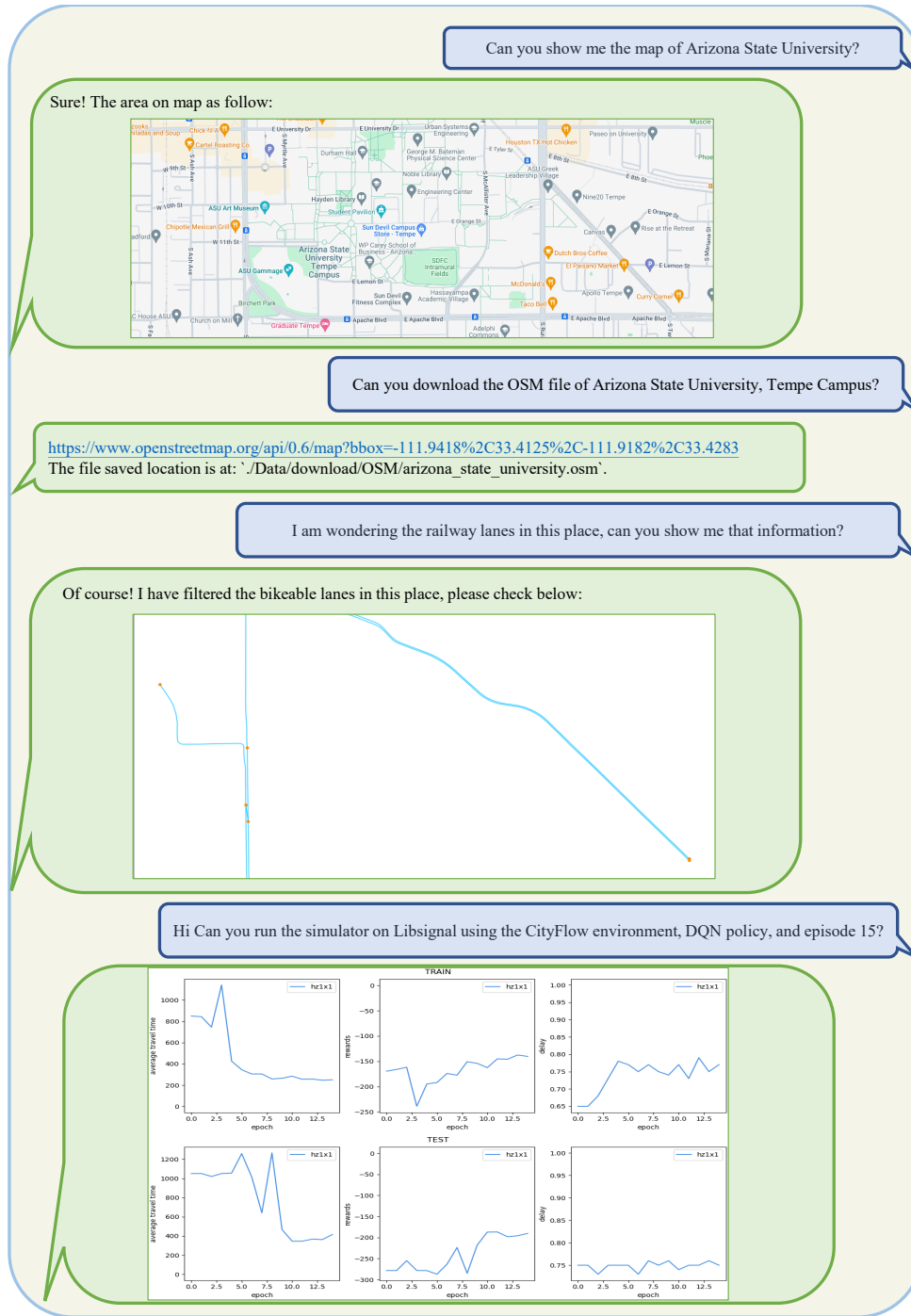


Figure 19: Ask Open-TI to show the map of the interested place, download .osm data of the interested place, use the OSM file of the target place to filter railway routes, and conduct traffic signal control.

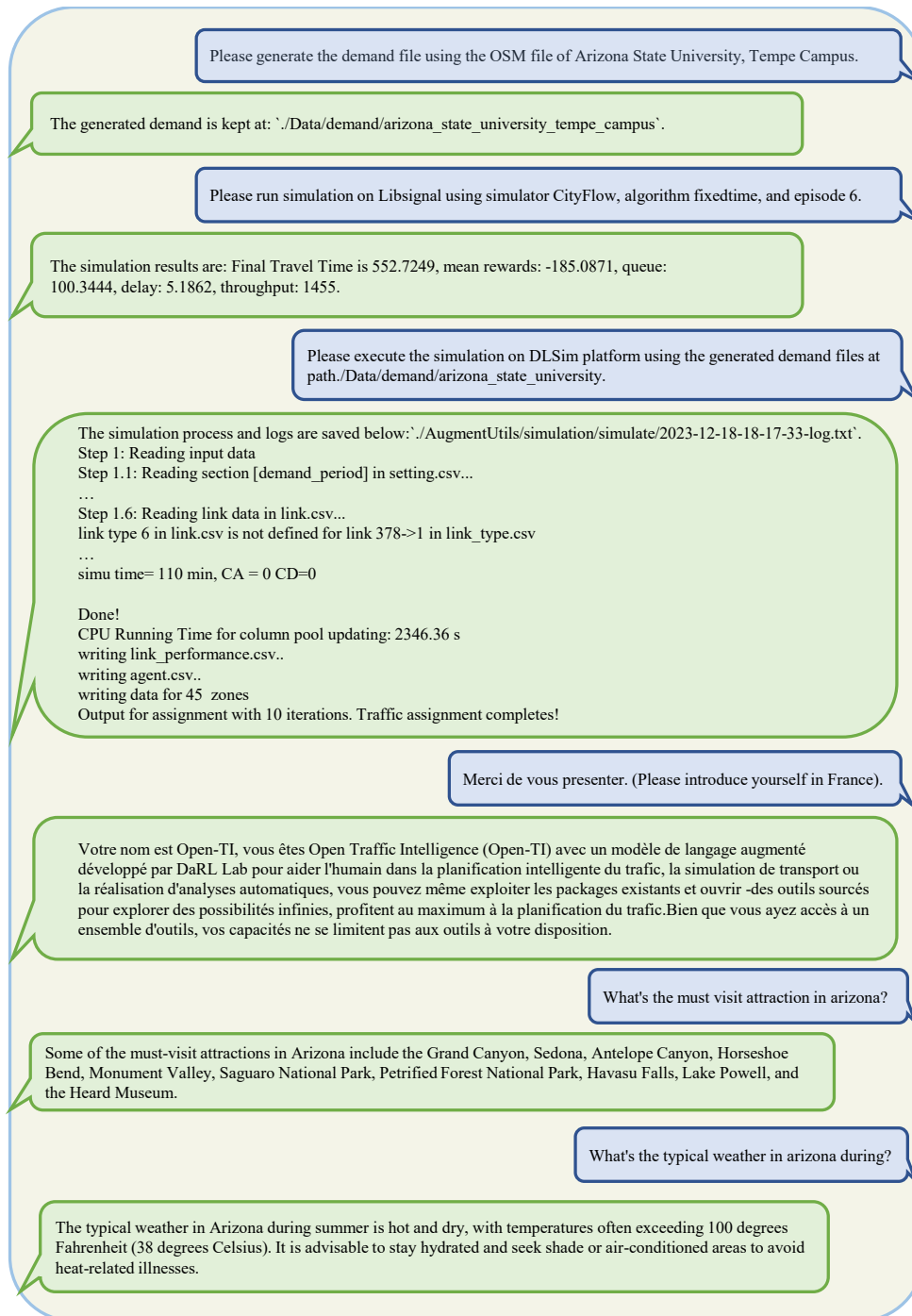


Figure 20: An example of asking Open-TI to generate demand file from OSM file, run simulator on Libsignal in different algorithm and episode, execute simulation on DLSim, understand and response in multi-languages, and analyze logs in a specific path.

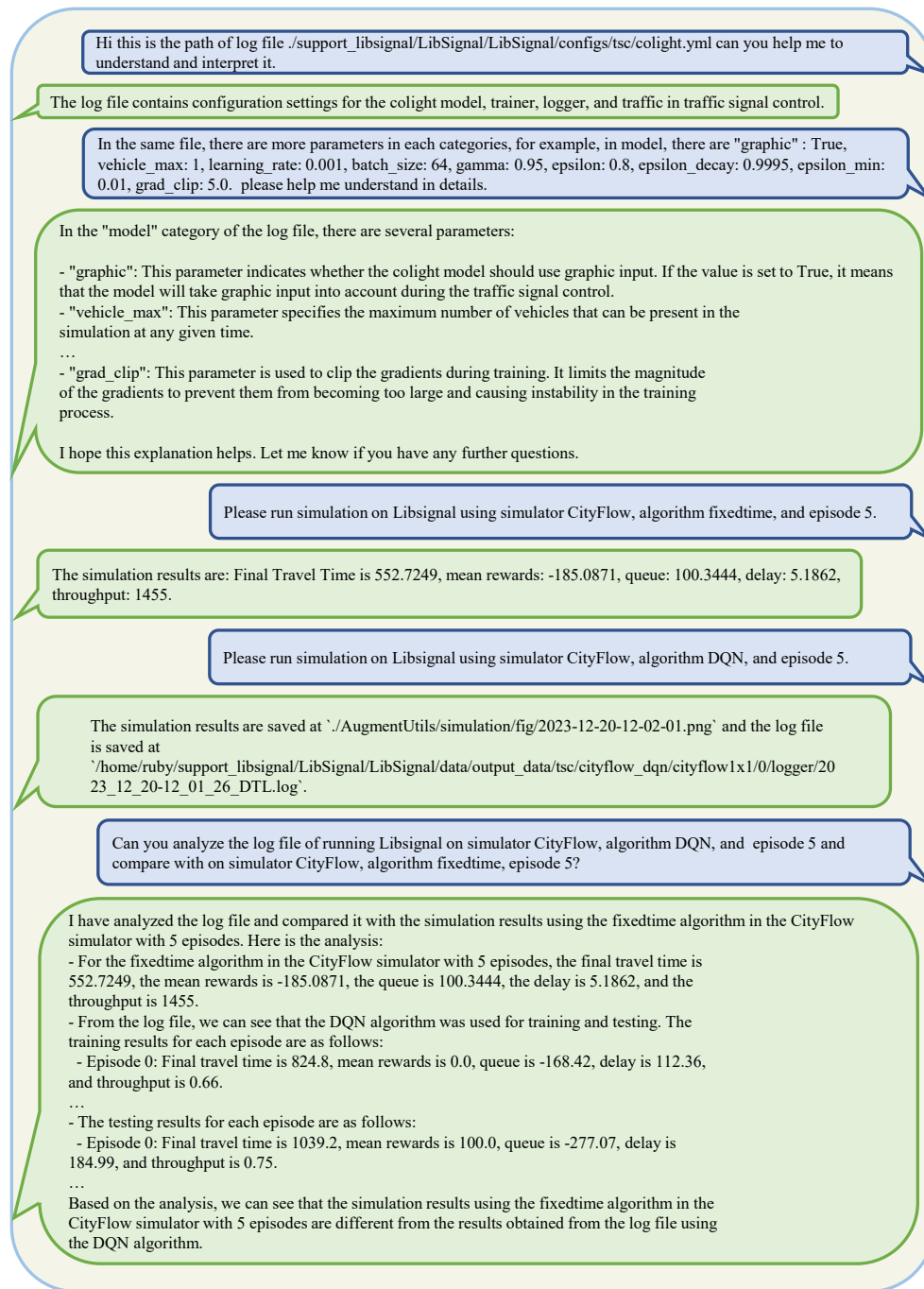


Figure 21: An example of asking Open-TI to analyze file with a specific path. In this case, run the simulation on Libsignal using simulator CityFlow, algorithm fixedtime, and episode 5. Additionally, run the simulation on Libsignal using simulator CityFlow, algorithm DQN, and episode 5. Finally, use the `logAnalyzer` to compare the performance of both algorithms.

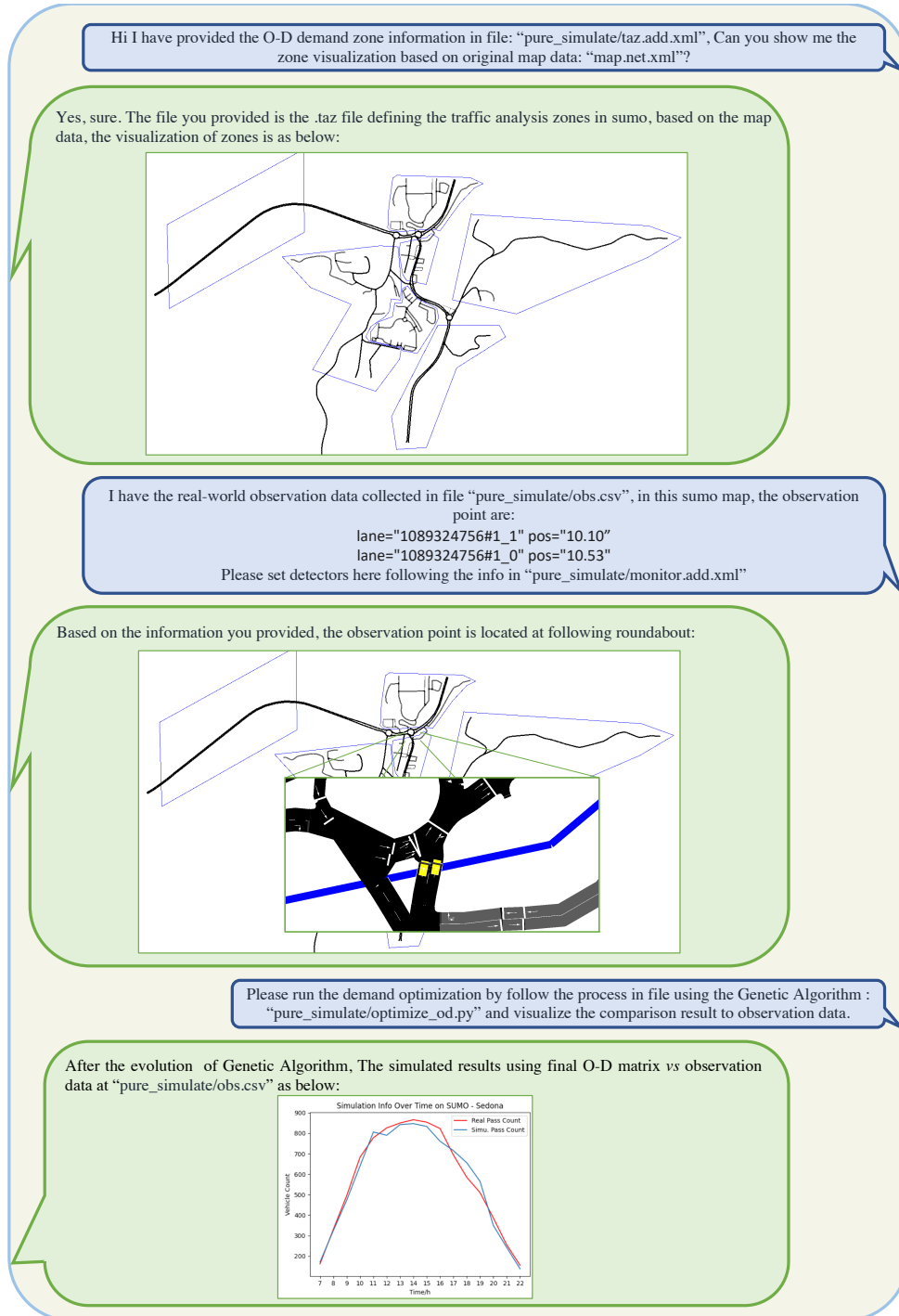


Figure 22: The demonstration on how Open-TI is used to optimize the OD-Demand matrix. It first visualizes the defined traffic zone information, and sets the observation point to mimic the real-world data collection process. Then based on the gap between simulation observation and real-world observation (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.