Enhancing Urban Traffic Systems through LLM Integration with SUMO Simulations

1st Matheus Gomes Diniz Andrade
Postgraduate Program in Electrical and Computer Engineering
Federal University of Rio Grande do Norte
Natal, Rio Grande do Norte, Brazil
matheus.diniz.122@ufrn.edu.br

2nd Ivanovitch Silva
Postgraduate Program in Electrical and Computer Engineering
Federal University of Rio Grande do Norte
Natal, Rio Grande do Norte, Brazil
ivanovitch.silva@ufrn.br

***Abstract*—This study explores the integration of Large-Scale Language Models (LLMs) with the SUMO (Simulation of Urban MObility) simulator to improve urban traffic systems. By using the LangChain library to create intelligent agents and Streamlit for user interaction, the research demonstrates how LLMs can interpret traffic data and make informed decisions within SUMO simulations. The system simplifies simulation configurations, enhances data analysis, and supports efficient traffic management. Key findings highlight the potential for LLMs to automate and optimize urban mobility, contributing to smarter and more sustainable city planning.**

***Index Terms*—LLM, SUMO, urban mobility, traffic simulation, LangChain, Streamlit, AI agents**

I. Introduction

The evolution of Large-Scale Language Models (LLMs) has brought a significant transformation in interacting with computational systems, making them more accessible and intuitive [1], [2]. These models, based on advanced neural network architectures, possess the capability to comprehend and generate text at a level comparable to human language [3]. With the ability to process large volumes of data and make contextually accurate inferences, LLMs have been applied in various fields, from virtual assistants to biomedicine and autonomous vehicles [3].

Simultaneously, in the field of urban mobility, the use of simulators such as SUMO (Simulation of Urban MObility)[1] has proven fundamental for the study and development of urban transportation systems [4]–[6]. Developed by the Institute of Transportation Systems at the German Aerospace Center, SUMO offers a robust platform for simulating traffic flows and mobility, allowing for the modeling of complex traffic scenarios and the evaluation of the impact of various transportation policies [6]. The flexibility and precision of SUMO make it a useful tool for analyzing urban mobility systems.

The combination of these two promising technologies, LLMs and simulators like SUMO, opens up new possibilities for application. The integration of LLMs with agents enables a wide range of applications, due to the vast world knowledge embedded in these models, making them inherently versatile in various scenarios [7]. In the context of the SUMO simulator, this integration opens new possibilities for optimizing and managing transportation systems, as well as improving data interpretability. For instance, these agents can enhance traffic light management [3] and urban navigation [8].

This article explores the synergy between LLMs and SUMO through agents, using the LangChain library[2], demonstrating how the combination of these technologies can result in advanced and intelligent systems. The implementation of agents that use LLMs to interpret traffic data and make informed decisions, integrating these capabilities into SUMO through an interface developed in Streamlit[3], will be discussed. This study aims to highlight the benefits and challenges of this approach, as well as its implications for the future of urban mobility.

One of the main benefits of integrating LLMs and SUMO is the ability to perform simulations without the need to specify many parameters for the simulation. The interface developed in Streamlit facilitates user interaction with the system. This intuitive interface makes the model's responses more accessible and easily allows for simulation analysis.

The results of this article highlight the importance and relevance of applying language models in conducting simulations and interpreting the generated data. These findings emphasize the utility of LLMs in automating and optimizing processes, such as urban traffic simulation. In this way, these models can contribute to understanding and improving urban transportation systems, as well as promoting the development of more effective and sustainable solutions for smart cities.

Finally, the remainder of this article is organized as follows: Section II presents related works; Section III specifies the proposed approach; Section IV describes the case study conducted to compare the solutions; Section V discusses the main results obtained; and, finally, Section VI presents the potential limitations and challenges of this integration. Future research directions, including algorithm improvement, are also addressed.

II. Related Works

This section provides an overview of related works that influenced the research conducted and contributed to the

[1]https://eclipse.dev/sumo/

[2]https://www.langchain.com/

[3]https://streamlit.io

development of the proposed solution. Various studies have been developed applying LLMs in different fields such as education [9], medicine [10], and the automotive industry [11].

Similarly, the use of simulations has proven essential in various industries, especially in the automotive sector. The work of Schumann et al. [8] proposed VELMA (Verbalization Embodiment of LLM Agents), an embedded LLM agent that uses verbalization of trajectory and visual environment observations as a contextual prompt for the next action. Visual information is converted into verbal descriptions through a pipeline that extracts landmarks from human-written navigation instructions and uses the CLIP (Contrastive Language-Image Pre-Training) model to determine their visibility in the current panoramic view.

Simulation plays a crucial role in accelerating the development of vehicle platforms, offering a safe, cost-effective, and efficient alternative to traditional testing methods [12], [13]. Furthermore, the integration of LLMs can enhance the modeling and execution of simulations, providing new perspectives for data analysis and the generation of realistic traffic scenarios [3], [14].

In the field of urban traffic management, optimizing traffic signals on arterial roads is essential to reduce congestion and improve road efficiency. Tang et al. [3] investigated the use of LLMs to design and implement green wave control on these roads. This study proposed a workflow that integrates LLMs with traffic signal control policies, using the SUMO simulation software to validate the approach. The results showed that LLMs could generate traffic signal control policies interactively and efficiently, improving the average speed on arterial roads and reducing the data analysis and calculation burden for traffic managers. Additionally, the LLM (GPT-4) demonstrated the ability to extract key information from complex road network files and efficiently design green wave control schemes based on the parameters provided by the road network.

Similarly, the work of Güzay et al. [14] presents a solution that utilizes the capabilities of LLMs to overcome difficulties in creating traffic scenarios. The proposed approach allows for the linguistic generation of traffic scenarios, eliminating the need for user interfaces of editors or writing scenario files in specific formats. To achieve this, the authors designed and developed an application that allows user inputs for scenarios and used OpenAI's GPT-4 API [4] for LLM processing. The model's responses are then processed and converted into simulation files in XML (Extensible Markup Language) format.

Thus, the presented works addressing the use of LLMs with SUMO highlight the effectiveness and innovation of these technologies in urban traffic management and simulations. However, there is still room to explore new approaches that increase the accessibility and functionality of these solutions.

This work proposes integrating agents, built using the `LangChain` library, with the SUMO simulator to enhance the execution of simulations and the retrieval of data from them. Utilizing the intuitive Streamlit interface, the developed solution allows not only the configuration and execution of simulations based on the provided parameters but also the ability to answer specific questions about simulated vehicles and generate charts for them.

[4]https://openai.com

III. Approach

The proposed approach aims to provide a methodology to automate urban simulations through the use of LLMs and SUMO. It is composed of three modules (see Figure 1):

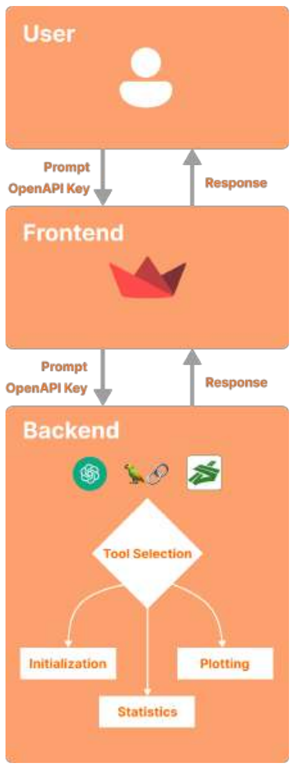


Fig. 1. Metodology

A. User

In the user module, the responsible party must initially acquire the necessary files to run the simulation. This can be done using the OpenStreetMap Web Wizard[5], which is available when downloading SUMO. This tool allows the generation of scenarios from user-provided specifications, facilitating the initial configuration of the simulation.

After generating the scenario, the user needs to place the generated files in a specific directory. This way, the files will be accessible to the application, allowing both the agent, built using the `LangChain` library, to perform its functions, and the `TraCI`[6] (Traffic Control Interface) library to interact with the files.

The `LangChain` library is a robust tool for building natural language agents, enabling the creation of complex workflows

[5]https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html
[6]https://sumo.dlr.de/docs/TraCI.html

that can process, interpret, and act based on natural language commands. The `TraCI` library is a traffic control interface that allows direct interaction with the SUMO simulator using the `Python` language, providing a flexible and efficient way to monitor and manipulate traffic simulations.

B. Frontend

The frontend was built using the Streamlit library, which provides an interactive and accessible user interface. This choice allows users to configure and run traffic simulations intuitively, simply by typing commands on how they want the simulation to proceed.

In this interface, the user provides their OpenAI key and the desired prompt. This way, the user interacts with advanced natural language models. This facilitates the execution of simulations without requiring deep knowledge about simulation configuration, making the process more accessible and practical for various types of users.

C. Backend

In the backend, the interpretation of the user's prompt is carried out through the LLM model. Given the LLM's interpretation, the agent uses one of the created tools to perform the task proposed in the prompt. For the present work, three tools were developed (see Table I).

TABLE I
TOOLS DEVELOPED FOR THIS WORK

Tool	Description
Sumo Initialization	Performs the simulation based on the specifications provided by the user.
Simulation Data Analysis	Provides information about specific vehicles in the simulation.
Simulation Data Plotting	Generates graphical visualizations of data from specific vehicles in the simulation.

To facilitate the recognition of which tool in Table I should be used, a prompt engineering technique known as few-shot learning is applied. Few-shot learning is an approach that allows obtaining a high-performance model by learning from a few samples in new tasks, providing a solution for many scenarios that lack samples [15]. This technique enables the LLM model to understand the context and select the appropriate tool based on a small number of provided examples, enhancing the accuracy in task execution.

For the present work, a few-shot learning approach was specified as follows:

1) If the user's input is related to the execution of the simulation, the LLM should classify it as ”initialization.”
2) If the user's input is related to information about a specific vehicle after the simulation, the LLM should classify it as ”statistics.”
3) If the user's input is related to generating a chart of a specific vehicle after the simulation, the LLM should classify it as ”plotting.”

After the command specifying how the LLM should classify the input, some examples of inputs and their respective classifications are provided.

The agent, in turn, upon receiving the classification from the model, selects the appropriate tool to perform the requested task. This ensures that each request is handled according to its specific nature, whether it is related to the initialization of the simulation, obtaining vehicle statistics, or generating charts.

IV. Case Study

The case study aims to evaluate the effectiveness of the proposed approach by exploring the use of agents with custom tools from the `LangChain` library to automate urban traffic simulations using SUMO and facilitate the interpretation of information obtained after the simulation.

This section will be divided into the subsections preparation, which details the necessary steps to configure the environment, implementation, which describes the development and integration of agents with the tools from the `LangChain` library, and execution, which will focus on performing the prompts.

A. Preparation

To configure the environment, it is initially necessary to access the OpenStreetMap Web Wizard tool. This can be done by running the `osmWebWizard.py` file located in the SUMO installation directory. Once the file is executed, a webpage will open in the web browser (see Figure 2).

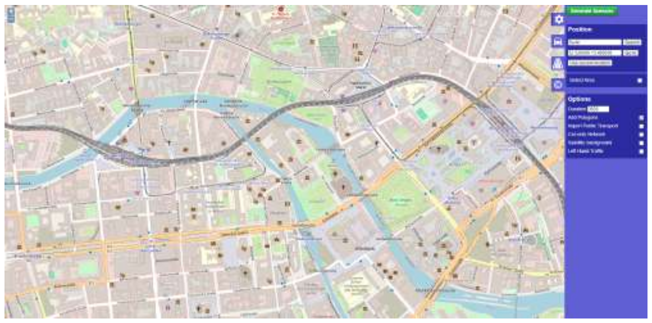


Fig. 2. OpenStreetMap Web Wizard

In the tool illustrated in Figure 2, it is possible to select a specific region for scenario generation. Additionally, various scenario parameters can be defined, such as the types of vehicles that will be present and the inclusion of elements like railway, aeroway, and waterway, among others. For the present work, the region of the city of Natal, Rio Grande do Norte, Brazil, was selected, as shown in Figure 3.

Additionally, the scenario configurations illustrated in Figure 4 were considered.

With the scenario generated, it was necessary to place the files in a specific folder called `sumo_files`, where both the `TraCI` library and the agent can interact with these files.

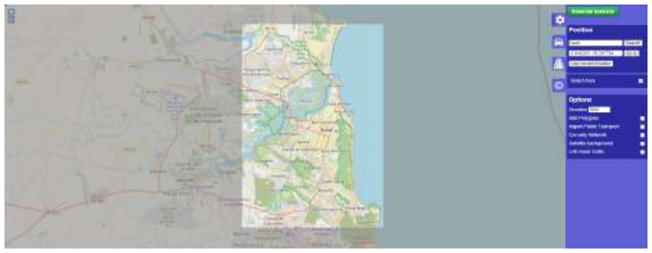


Fig. 3. Selected area for Natal, Rio Grande do Norte, Brazil

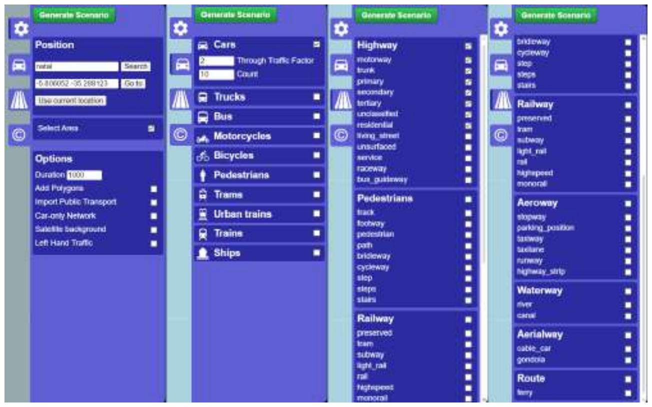


Fig. 4. OpenStreetMap Web Wizard

B. Implementation

For the implementation, the `Python` programming language was used in conjunction with the `LangChain` and `TraCI` libraries. The `LangChain` library was employed in creating the tools that will be selected by the agent. To achieve this, the `BaseModel` class from the `Pydantic` library[7] was used to define and validate the input data for the tools in a structured manner.

Next, the `BaseTool` class from the `LangChain` library was used in the development of the tools. In this implementation, the previously implemented validation, using `BaseModel`, is used as the tool's input schema. Following this, the special method `_run` is implemented, defining the main logic of the tool. This method is responsible for processing the input data and executing the necessary actions as specified by the user.

With the tools implemented, it is necessary to select the model to be used when making requests to OpenAI. This process involves configuring the API key and specifying the desired model. The choice of the model is crucial as it directly influences the quality and relevance of the responses generated by the agent, as well as the cost per token.

For the present work, the models in the GPT-3.5 Turbo family were analyzed, which are fast and inexpensive for simple tasks. The comparison is presented in Table II.

Given the values in Table II, the `gpt-3.5-turbo-0125` model was selected for being a more economical model. With the selected model, it is possible to instantiate it using the `ChatOpenAI` class from the `LangChain` library. Next, the following few-shot prompt was created:

[7]https://docs.pydantic.dev/latest/

TABLE II
PRICES FOR GPT-3.5 TURBO FAMILY

Model	Input (1M tokens)	Output (1M tokens)
gpt-3.5-turbo-0125	\$0.50	\$1.50
gpt-3.5-turbo-instruct	\$1.50	\$2.00

Few-shot Prompt

You are an urban traffic simulation agent. Your tasks include:

1) Initializing the simulation with specific parameters.
2) Providing detailed statistics about specific vehicles after the simulation.
3) Generating graphs for a vehicle.

Example input and classification:

- ”Start simulation with 50 vehicles” - Classification: initialization
- ”Show data for vehicle with ID 2” - Classification: statistics
- ”Generate speed graph for vehicle with ID 2” - Classification: plotting

The tools are then added to the agent, which will be responsible for receiving the model's classification and executing the appropriate tool, using the `AgentExecutor` class from the `LangChain` library.

C. Execution

For code execution, prompts were created to test the system's functionality. To test the `SumoInitializationTool`, which is the tool responsible for running a simulation, two prompts were developed:

- ”Initialize a simulation with the config file in ./data/sumo_files/osm.sumocfg”
- ”Initialize a simulation with the config file in ./data/sumo_files/osm.sumocfg with 8 threads and starts at time 0 and end at time 50, with a max of 20000 vehicles simultaneously. Get from the vehicles the speed, acceleration, fuel consumption, geolocation, odometer and co2 emissions.”

The purpose of creating these prompts is to evaluate the tool's ability to start simulations with different levels of complexity and extract relevant information from the vehicles. For the `SimulationDataAnalysisTool`, responsible for calculating statistics using the `Pandas` library[8], the following prompt was used:

- ”Get the statistics of the vehicle with id 1.”

Finally, to test the tool responsible for generating graphs, `SimulationDataPlottingTool`, which uses both the

[8]https://pandas.pydata.org

`Pandas` and `Matplotlib`[9] libraries, the following prompt was used:

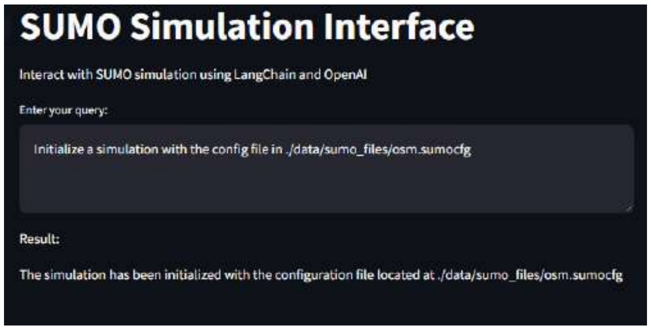
- "Plot the data of speed for vehicle with id 5."
- "Plot the data of fuel consumption for vehicle with id 5."

These tests aim to ensure that each tool performs its functions correctly, from running the simulation to analyzing and visualizing the resulting data. Validating these components is essential to ensure that the system operates correctly, meeting the requirements of the case study.

V. Results and Discussion

This section aims to discuss the results obtained in the application of the case study presented in Section IV. Thus, it seeks to analyze the responses obtained by introducing the prompts and evaluate the effectiveness of the developed functionalities.

Initially, the simulation execution tool (`SumoInitializationTool`) was analyzed. For the first prompt, a simulation was requested to run without specifying many parameters, only specifying the location of the simulation configuration file. The execution is shown in Figure 5.


SUMO Simulation Interface

Interact with SUMO simulation using LangChain and OpenAI

Enter your query:

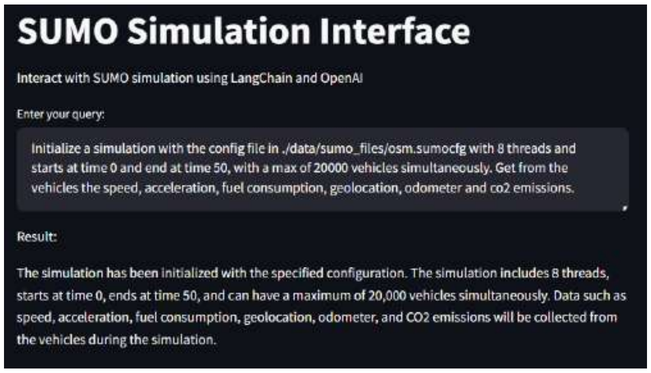
Initialize a simulation with the config file in ./data/sumo_files/osm.sumocfg

Result:

The simulation has been initialized with the configuration file located at ./data/sumo_files/osm.sumocfg

Fig. 5. Streamlit interface for a simple simulation prompt

Next, a more complex prompt was provided (see Figure 6), involving more input parameters and specifying the columns of the final dataset.


SUMO Simulation Interface

Interact with SUMO simulation using LangChain and OpenAI

Enter your query:

Initialize a simulation with the config file in ./data/sumo_files/osm.sumocfg with 8 threads and starts at time 0 and end at time 50, with a max of 20000 vehicles simultaneously. Get from the vehicles the speed, acceleration, fuel consumption, geolocation, odometer and co2 emissions.

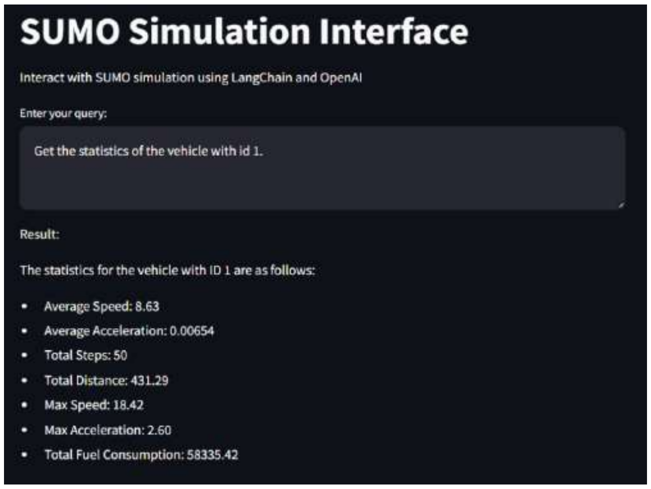
Result:

The simulation has been initialized with the specified configuration. The simulation includes 8 threads, starts at time 0, ends at time 50, and can have a maximum of 20,000 vehicles simultaneously. Data such as speed, acceleration, fuel consumption, geolocation, odometer, and CO2 emissions will be collected from the vehicles during the simulation.

Fig. 6. Streamlit interface for a complex simulation prompt

[9]https://matplotlib.org

These tests allowed us to verify the system's ability to handle different levels of complexity in simulations, demonstrating its flexibility. Next, the prompt for using the analysis tool (`SimulationDataAnalysisTool`) was tested. The result is presented in Figure 7.


SUMO Simulation Interface

Interact with SUMO simulation using LangChain and OpenAI

Enter your query:

Get the statistics of the vehicle with id 1.

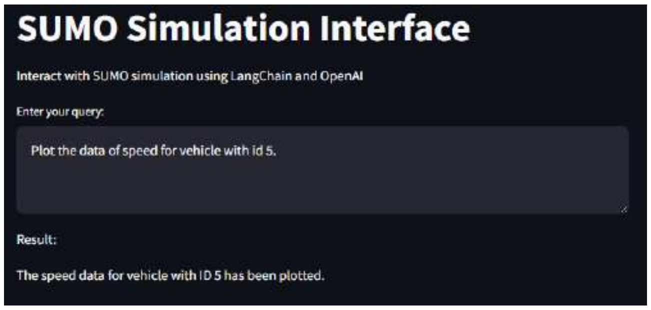
Result:

The statistics for the vehicle with ID 1 are as follows:

- Average Speed: 8.63
- Average Acceleration: 0.00654
- Total Steps: 50
- Total Distance: 431.29
- Max Speed: 18.42
- Max Acceleration: 2.60
- Total Fuel Consumption: 58335.42

Fig. 7. Streamlit interface for prompt of statistics for vehicle with ID 1

Finally, to evaluate the graph generation tool (`SimulationDataPlottingTool`), the generation of a graph was initially tested. The prompt shown in Figure 8 aimed to generate a speed graph for the vehicle with ID 5.


SUMO Simulation Interface

Interact with SUMO simulation using LangChain and OpenAI

Enter your query:

Plot the data of speed for vehicle with id 5.

Result:

The speed data for vehicle with ID 5 has been plotted.

Fig. 8. Streamlit interface for prompt for plotting the speed for the vehicle with ID 5

In response, a graph was generated and saved in the local files, as illustrated in Figure 9.

Similarly, the prompt shown in Figure 10 was tested, which aimed to generate a fuel consumption graph for the vehicle with ID 5.

Just like in the previous example, a graph was generated and saved in the local files, which can be seen in Figure 11.

VI. Conclusion

This article proposed a system to automate urban traffic simulations, integrating LLMs, agents, and the SUMO simulator. The case study demonstrated the methodology's effectiveness, highlighting its potential to improve urban transportation systems. The developed prompts allowed for the

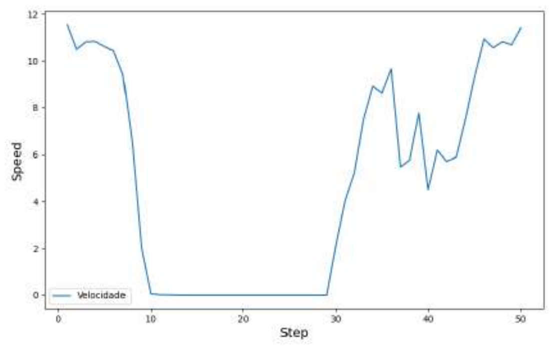
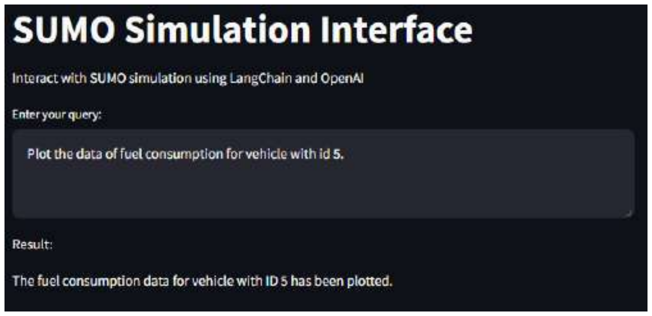


Fig. 9. Speed plot for the vehicle with ID 5



SUMO Simulation Interface

Interact with SUMO simulation using LangChain and OpenAI

Enter your query:

Plot the data of fuel consumption for vehicle with id 5.

Result:

The fuel consumption data for vehicle with ID 5 has been plotted.

Fig. 10. Streamlit interface for prompt for plotting the fuel consumption for the vehicle with ID 5

precise and efficient configuration of simulations, showcasing the system's ability to interpret commands and direct them to the appropriate functionalities.

The proposed approach significantly simplifies the process of configuring and running urban simulations, making it more accessible for users with different levels of technical knowledge. The use of tools such as `LangChain` and `TraCI`, along with the interactive interface provided by Streamlit, proved effective in automating the tasks involved in traffic simulations.

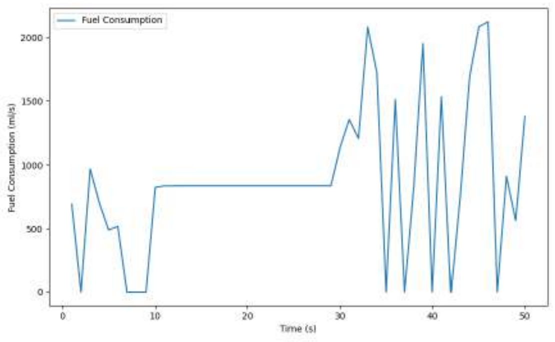


Fig. 11. Fuel consumption plot for the vehicle with ID 5

The results obtained with the different prompts demonstrated that the system can handle various levels of complexity in simulations, from simple initialization to analysis and graph generation. This flexibility is crucial to meet the diverse needs of users in urban traffic studies.

Future work includes, but is not limited to: improving the user interface to make it even more intuitive, including drag-and-drop functionality for file insertion and real-time options for viewing simulation results; creating new tools that can handle other simulation variables; and implementing more advanced few-shot learning techniques to further improve the accuracy in classification and interpretation of prompts by the LLM.

References

[1] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3544548.3581388

[2] Güzay, E. Özdemir, and Y. Kara, "A generative ai-driven application: Use of large language models for traffic scenario generation," in *2023 14th International Conference on Electrical and Electronics Engineering (ELECO)*, 2023, pp. 1–6.

[3] Y. Tang, X. Dai, and Y. Lv, "Large language model-assisted arterial traffic signal control," *IEEE Journal of Radio Frequency Identification*, vol. 8, pp. 322–326, 2024.

[4] T. Garg, G. Kaur, and P. S. Rana, "Real-time traffic light optimization using simulation of urban mobility," *SN Computer Science*, vol. 4, no. 526, 2023. [Online]. Available: https://doi.org/10.1007/s42979-023-01916-9

[5] C. Gao, "Simulating and validating the traffic of blackwall tunnel using tfl jam cam data and simulation of urban mobility (sumo) (short paper)," in *12th International Conference on Geographic Information Science (GIScience 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 277. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 34:1–34:8. [Online]. Available: https://doi.org/10.4230/LIPIcs.GIScience.2023.34

[6] A. M. Menin, L. Soares, D. R. Bisconsini, J. Trombetta, and N. L. Tabalipa, "Aplicação do software simulation of urban mobility para a avaliação de uma interseção urbana: estudo de caso na cidade pato branco, brasil," *OBSERVATÓRIO DE LA ECONOMÍA LATINOAMERICANA*, vol. 21, no. 6, pp. 4618–4639, Jun. 2023. [Online]. Available: https://ojs.observatoriolatinoamericano.com/ojs/index.php/olel/article/view/754

[7] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang, "Expel: Llm agents are experiential learners," *AAAI*, vol. 38, no. 17, pp. 19 632–19 642, Mar. 2024.

[8] R. Schumann, W. Zhu, W. Feng, T.-J. Fu, S. Riezler, and W. Y. Wang, "Velma: Verbalization embodiment of llm agents for vision and language navigation in street view," *AAAI*, vol. 38, no. 17, pp. 18 924–18 933, Mar. 2024.

[9] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, M. Stadler, J. Weller, J. Kuhn, and G. Kasneci, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1041608023000195

[10] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan *et al.*, "Large language models in medicine," *Nature Medicine*, vol. 29, pp. 1930–1940, 2023. [Online]. Available: https://doi.org/10.1038/s41591-023-02448-8

[11] J. Ge, C. Chang, J. Zhang, L. Li, X. Na, Y. Lin, L. Li, and F.-Y. Wang, "Llm-based operating systems for automated vehicles: A new perspective," *IEEE Transactions on Intelligent Vehicles*, pp. 1–5, 2024.

[12] S.-T. Liu, C. Chang, Y.-H. Huang, T.-H. Lin, J. Chiu, and J.-L. Lee, "Development and test of abs/tcs controller with dual-axis dynamometer hil platform," SAE Technical Paper, Tech. Rep., 2023.

[13] H. Alghodhaifi and S. Lakshmanan, "Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches," *IEEE Access*, vol. 9, pp. 151 531–151 566, 2021.

[14] Ç. Güzay, E. Özdemir, and Y. Kara, "A generative ai-driven application: Use of large language models for traffic scenario generation," in *2023 14th International Conference on Electrical and Electronics Engineering (ELECO)*. IEEE, 2023, pp. 1–6.

[15] J. Wang, K. Liu, Y. Zhang *et al.*, "Recent advances of few-shot learning methods and applications," *Science China Technological Sciences*, vol. 66, pp. 920–944, 2023. [Online]. Available: https://doi.org/10.1007/s11431-022-2133-1