Demo: Lowering the Barrier to Traffic Simulation Through a Prompt-to-Config Agent

William Phong, Jianshu Liu, Patrick Simpauco, Audio Antuna, Logan Moreno
San Diego State University, San Diego, CA, USA
{wphong2900, jliu11, psimpauco6201, aantuna0695, lmoreno3889}@sdsu.edu

***Abstract*—High-fidelity traffic simulators such as CARLA play a critical role in autonomous driving research, enabling controlled and configurable validation and verification across diverse traffic environments. However, scenario configuration often requires domain expertise and specialized tools, creating a usability barrier. In this paper, we present PromptCARLA, a lightweight prompt-to-config interface for CARLA. PromptCARLA allows users to specify traffic scenarios in natural language and translates them into validated CARLA configurations through structured output generation and schema validation. We evaluate PromptCARLA across four commercial LLMs using 59 prompts spanning explicit, descriptive, and adversarial inputs. Our preliminary results show 98.31% schema validation success, and graceful degradation on adversarial inputs.**

***Index Terms*—Traffic simulation, CARLA, Interface**

I. Introduction

Autonomous driving has been an important research direction in intelligent transportation systems, aiming to improve road safety, reduce traffic congestion, and expand mobility for increased independence and accessibility. Traffic simulation has become an essential tool to enable controlled validation and verification for large-scale experimentation across diverse traffic environments. Among the available simulation platforms [1], CARLA [2] stands out, as it is open-source with a supported community, provides a fully customizable environment, including map selection and multiple weather conditions, and diverse types of high-fidelity sensor data. Moreover, CARLA is increasingly used in education, and students can use CARLA to construct traffic scenarios and gain hands-on experience with a tool that bridges classroom learning and industrial engineering practice.

However, constructing and configuring simulation scenarios requires substantial technical expertise. On the one hand, scenario generation requires domain knowledge to design diverse, representative, and challenging test cases. On the other hand, existing scenario generation frameworks, such as OpenSCENARIO [3], aim to enable the concise specification of traffic scenarios but introduce complexity in domain-specific programming languages. Hence, users have to learn specialized syntax and tools before constructing scenarios, which increases the learning efforts for non-technical users and limits the accessibility of simulation platforms for education.

In this paper, we present PromptCARLA, a lightweight prompt-to-configuration interface for CARLA (see Figure 1). Rather than proposing a new scenario generation algorithm, PromptCARLA focuses on lowering the usability barrier of simulator configuration by translating natural language descriptions (e.g., "Stadium area after event in cloudy weather") into schema-validated CARLA parameters. PromptCARLA builds upon LangChain[4] and translates the user's prompt to a concrete set of simulator parameters, including weather preset, map identifier, vehicle count, and pedestrian counts. By combining structured output prompting with Pydantic validation and a lightweight JSON extraction layer, PromptCARLA ensures that every generated configuration is simulator-ready before it is passed to the CARLA backend. PromptCARLA is publicly available at https://github.com/PiscesLab/PromptCARLA.

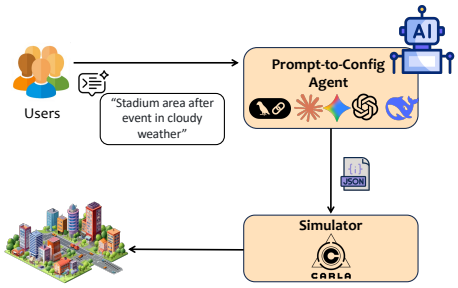

Fig. 1: The architecture of the PromptCARLA.

II. PromptCARLA Design

Figure 2 illustrates the end-to-end pipeline of PromptCARLA, which consists of three phases: prompt construction, LLM invocation, and output validation.

Phase 1: Prompt Construction defines the structure and constraints of the desired configuration for the traffic simulators before invoking LLMs. PromptCARLA employs a Pydantic `CarlaConfig` schema that defines four fields required to initialize a CARLA scenario, including *weather*, *town*, *num_vehicles*, and *num_pedestrians*. The LLM-generated JSON output is parsed into the CarlaConfig schema, where field types, enum membership, and value-range constraints are automatically checked. Invalid map identifiers (e.g., `Town08`), unsupported weather presets (e.g., `SnowHeavy`), or negative counts are automatically rejected. PromptCARLA constructs a system prompt that instructs the LLM to generate structured outputs. The system prompt is identical across LLM backends and contains approximately 680 tokens, providing an ample context window for the user's description.

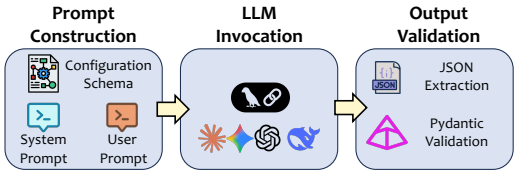

Fig. 2: The overview of the PromptCARLA pipeline.

Phase 2: LLM Invocation. PromptCARLA invokes the LLM through LangChain [4], which provides a unified interface for different LLM providers. During inference, the system prompt is processed as the system message, and the user's scenario description as the user message. PromptCARLA uses commercial LLMs via their standard APIs and requires no model fine-tuning or additional pre-training. The LLM backend is configurable, allowing PromptCARLA to work with different models without modifying the system pipeline.

Phase 3: Output Validation ensures that the generated configuration is both syntactically correct and semantically valid before it is passed to the CARLA simulator. The LLM's raw text response passes through a two-step post-processing with JSON Extraction and Pydantic Validation.

Configurations that pass both stages are guaranteed to be used directly by the CARLA simulator. This validation phase allows PromptCARLA to serve as a reliable intermediary between natural language scenario descriptions and strict configuration requirements of the simulation environments.

III. Preliminary Evaluation

A. Experimental Setup

Testing Prompt Design. We constructed a dataset of 59 unique prompts with varying levels of specificity and difficulty. Each prompt is associated with expected ground-truth values where applicable. The prompts are organized into three tiers:

- **Tier 1 (20 prompts):** Explicit parameter specification, e.g., *"Simulate 20 vehicles and 10 pedestrians in ClearNoon weather in Town01."*. Exact matches are expected for all fields.
- **Tier 2 (20 prompts):** Descriptive natural language, e.g., *"A busy rainy evening with an ambulance crossing downtown."*. Expected values were manually assigned for weather and map based on reasonable interpretations. Vehicle and pedestrian counts were compared only when explicitly stated.
- **Tier 3 (19 prompts):** Adversarial and edge-case inputs, such as invalid map/weather names and negative values. Tier 3 prompts focus on system robustness and expect graceful handling of invalid inputs.

B. Preliminary Results

Our preliminary evaluation includes 236 tests (59 prompts $\times$ 4 models $\times$ 1 repetition) and measures: (1) *constraint satisfaction*, including schema validation success rate, weather/map enum validity, and parameter bound compliance; (2) *semantic accuracy*, measured as the percentage of generated configuration fields that exactly match the expected ground-truth values for a given prompt. Moreover, validation success is defined as successfully producing a configuration that passes both JSON extraction and Pydantic schema validation.

TABLE I: Semantic accuracy by tier and average token usage. Dashes indicate not applicable or not available for that tier.

Tier	Model	Weather	Map	Vehicles	Peds.	InTok	OutTok	TotToks
T1	Claude	100%	100%	100%	100%	1,016	57	1,073
	DeepSeek	100%	100%	100%	100%	872	52	925
	Gemini	100%	100%	100%	100%	—	—	—
	GPT-4.1	100%	100%	100%	100%	861	55	916
T2	Claude	65.0%	75.0%	—	—	1,006	73	1,079
	DeepSeek	50.0%	85.0%	—	—	866	61	927
	Gemini	60.0%	80.0%	—	—	—	—	—
	GPT-4.1	60.0%	75.0%	—	—	855	57	912
T3	Claude	52.6%	21.1%	85.7%	33.3%	1,006	87	1,093
	DeepSeek	10.5%	21.1%	85.7%	33.3%	865	53	918
	Gemini	31.6%	47.4%	85.7%	33.3%	—	—	—
	GPT-4.1	10.5%	10.5%	83.3%	33.3%	854	82	936

First, all four LLM models achieved a 100% rate of producing *parseable* output. Every response contained extractable JSON, even when wrapped in markdown code fences. The overall validation success rate was 98.31%. Second, Table I presents the semantic accuracy results broken down by prompt tier. On Tier 1 prompts, where all parameters are explicitly stated, every model achieved 100% exact match across weather, map, vehicle, and pedestrian count. Tier 2 results reveal interesting variation in how models interpret implicit scenario descriptions. Despite some mismatches, all Tier 2 outputs remained semantically plausible interpretations of the prompts. Tier 3 prompts contain invalid or adversarial inputs, so accuracy is not the primary metric. Instead, the focus is on whether models gracefully degrade. All four models did so, producing valid configurations rather than malformed outputs.

Limitations. This evaluation is preliminary and has several limitations. The prompt dataset is relatively small, and we did not compare against alternative prompt-engineering baselines. In addition, no formal user study was conducted. Future work includes expanding support for additional traffic simulation platforms, conducting classroom usability studies, and extending the configuration schema to support richer scenario parameters. We also plan to investigate interactive multi-round scenario refinement, user feedback-driven correction mechanisms, and model combination strategies.

Acknowledgement. This work was supported in part by the San Diego State University Seed Grant Program.

References

[1] Yueyuan Li, Wei Yuan, Songan Zhang, Weihao Yan, Qiyuan Shen, Chunxiang Wang, and Ming Yang. Choose your simulator wisely: A review on open-source simulators for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 9(5):4861–4876, 2024.

[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[3] Association for Standardisation of Automation and Measuring Systems (ASAM). Openscenario 2.0.0 – a standard for describing dynamic content in driving simulations, 2022. Accessed: 2026-03-01.

[4] Harrison Chase. LangChain: Building applications with LLMs through composability, 2024. Accessed: 2025-06-01.