PlannerForge: LLM Agents for Scenario-Based Testing of Motion Planners in Autonomous Driving*

Yuan Gao[1] Sebastian Müller[1] Mattia Piccinini[1] Marc Kaufeld[1]
Yuchen Zhang[1] Finn Rasmus Schäfer[1] Qunying Song[2] Johannes Betz[1]
[1]Professorship of Autonomous Vehicle Systems, TUM School of Engineering and Design,
Technical University of Munich, 85748 Garching, Germany;
Munich Institute of Robotics and Machine Intelligence (MIRMI)
[2]University College London, London, United Kingdom

Abstract

Ensuring the safety of autonomous driving is a critical challenge. Scenario-based testing is a systematic process used to validate Autonomous Driving Systems (ADSs), but it remains a fragmented modular pipeline in which scenario generation, retrieval, modification, ADS execution, and results analysis are performed by separate tools with little interaction. Large Language Model (LLM) agents have shown promise across ADS sub-systems such as perception, planning, and control. However, no prior work covers the whole scenario-based testing pipeline for ADSs with a unified LLM-agent framework. We present **PlannerForge**, an LLM-agent framework that extends all scenario-based testing stages (from Scenario Generation to ADS Assessment) and adds two further LLM-enhanced stages: ADS Enhancement and ADS Benchmarking. We evaluate PlannerForge with 10 off-the-shelf LLMs across all tasks (Generation, Selection, Modification, Module Routing, Planner Testing, and Enhancement) under 5 prompt conditions. Best-per-task scores range from 0.88 to 1.00, and open-source 20–35B backends match commercial APIs on most tasks. Open-source models such as Qwen3.6:35B match commercial APIs on three of the five tasks. Chaining the modules end-to-end retains 83% / 78% of seed queries (commercial / open). It outperforms Scenario Factory 2.0 (Finkeldei et al., 2025) on natural-language generation (193 vs. 144 executable of 200) and realises 92–96% of requested city, road and vehicle attributes. It outperforms BM25 (Robertson and Zaragoza, 2009) at rank 1 selection (92.0% vs. 67.5%) and From-Words-to-Collisions (Gao et al., 2025) on physically valid edits ($\geq$94% vs. 31%). At N=400, cost-tuning lifts planner success from 50.4% to 70.2% and cuts collisions from 19.0% to 8.4%, without domain-specific fine-tuning.

*Code and data: https://github.com/TUM-AVS/PlannerForge

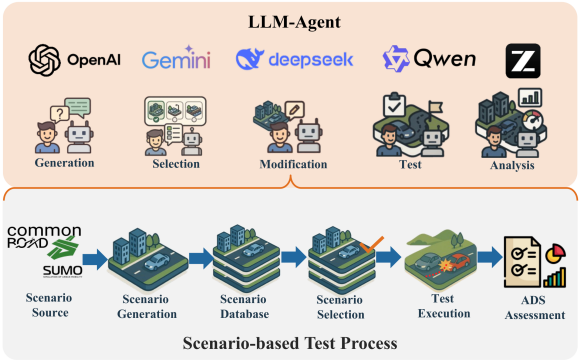


Figure 1: **PlannerForge** overview: LLM agents automate the scenario-based testing pipeline.

1 Introduction

The rapid advancement of Autonomous Driving Systems (ADSs) to SAE Level 4 (Waymo, 2018; International, 2021) hinges on rigorous validation (Betz et al., 2024). Because real-world testing of rare edge cases is prohibitively expensive (Winner et al., 2019), the industry relies heavily on scenario-based testing in simulation (Riedmaier et al., 2020; Song et al., 2024). While recent advances in Large Language Models (LLMs) have begun to enhance the realism and scalability of such testing, their application has been largely confined to scenario generation (Gao et al., 2026b). However, an ADS assessment pipeline requires significantly more: it must seamlessly integrate the selection of relevant cases, the execution of tests, and the analysis of results (Song et al., 2026).

In current motion-planner testing practice, these stages remain highly fragmented and manual. Scenario generation relies on GUI editors or scripts, selection depends on hand-crafted database filters, and ad-hoc scenario modification is largely unsupported. Furthermore, testing pipelines are rigidly scripted rather than guided by user intent, planner cost weights are manually tuned, and cross-planner comparisons require separately scripted batch runs

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 14522–14556
October 24-29, 2026 ©2026 Association for Computational Linguistics

with offline aggregation. To overcome these bottlenecks, there is a clear need for a unified, LLM-powered framework that spans and automates the entire lifecycle of scenario-based testing.

In this paper, we introduce **PlannerForge**, an LLM-powered scenario-based testing framework for motion planners that closes this gap (Figure 1). We target motion planners because they are the core decision-making module of an ADS and their behavior is particularly sensitive to complex, safety-critical traffic scenarios. Unlike prior work that focuses narrowly on scenario generation, PlannerForge integrates the full testing pipeline: scenario *generation* from real-world OpenStreetMap (OSM)[1] maps, scenario retrieval from a curated open-source scenario database, scenario modification via traffic object and behavior changes, motion planner execution, and performance analysis. A chatbot-oriented interface abstracts away technical complexity while enabling the seamless integration of diverse motion planners, providing researchers and practitioners with a scalable, benchmark-ready platform for ADS assessment.

The key contributions of this paper are:

1. **PlannerForge**, to our knowledge, the first full-lifecycle LLM framework that unifies all scenario-based-testing stages (Riedmaier et al., 2020) (Scenario Source, Generation, Database, Selection, Test Execution, ADS Assessment; together with two further LLM-era stages, ADS Enhancement and ADS Benchmarking) in a single chatbot-driven pipeline.
2. An **empirical evaluation** of ten off-the-shelf LLMs backends (five commercial, five open-source, spanning reasoning and non-reasoning models) across the five core framework tasks under five prompt conditions, showing both module-level performance and the effectiveness of off-the-shelf LLMs agents without domain-specific fine-tuning.
3. A **unified multi-planner interface** for comparative evaluation on generated and modified scenarios.

2 Related Work

Scenario-based testing provides a systematic methodology for validating ADSs by structurally evaluating operational conditions and safety-critical situations. This section reviews classical and LLM-powered approaches and positions **PlannerForge** within this landscape.

2.1 Classical Scenario-based Testing

Industry initiatives such as Pegasus (Winner et al., 2019) and SAKURA (Nakamura et al., 2022), alongside foundational surveys (Riedmaier et al., 2020), have established an influential six-component taxonomy for scenario-based testing: (1) *Scenario Source*, (2) *Scenario Generation*, (3) *Scenario Database*, (4) *Scenario Selection*, (5) *Test Execution*, and (6) *ADS Assessment*. Prior literature has extensively explored these individual components. For *Scenario Generation*, research covers knowledge-driven and data-driven approaches (Nalic et al., 2020) as well as adversarial and deep generative methods (Ding et al., 2023). Work on *Scenario Databases* includes reviews comparing dataset sensor modalities and annotations (Ding et al., 2023). *Scenario Selection* strategies typically involve knowledge-driven, data-driven, or falsification-based prioritization (Riedmaier et al., 2020). Finally, comprehensive surveys have examined scenario-based accelerated testing for ISO 21448 Safety of the Intended Functionality (SOTIF)[2] (Tang et al., 2025) and *ADS Assessment* through on-road performance metrics (Sharath and Mehran, 2021).

2.2 LLM-powered Scenario-based Testing

With the emergence of LLMs, the scenario-based testing process has been augmented with their reasoning capabilities across generation, analysis, and downstream execution stages.

LLM-powered Scenario Generation: Existing systems are split along simulator class and input source. *Autonomous Driving Simulation* (CARLA (Dosovitskiy et al., 2017)): ChatScene (Zhang et al., 2024), TTSG (Ruan et al., 2024), Aasi et al. (Aasi et al., 2024), NL2Scenic (Bauerfeind et al., 2025), Chat2Scenic (Gao et al., 2026a), Petrovic et al. (Petrovic et al., 2024), and Text2Scenario (Cai et al., 2026) synthesise safety-critical or branching Out-of-Distribution (OOD) scenarios from natural-language prompts; LCTGen (Tan et al., 2023) generates language-conditioned traffic on real maps. *Crash-report reconstruction*: SoVAR (Guo et al., 2024) and LeGEND (Tang et al., 2024) recover simulator assets from accident reports.

[1] https://www.openstreetmap.org/

[2] https://www.iso.org/standard/77490.html

Traffic rules and datasets: TARGET (Deng et al., 2025) compiles traffic rules into a Domain Specific Language (DSL); Chat2Scenario (Zhao et al., 2024) extracts scenarios from naturalistic logs. *Adversarial generation*: LLM-attacker (Mei et al., 2025) optimises attacker trajectories in closed loop. *Traffic flow Simulation* (SUMO (Lopez et al., 2018)): ChatSUMO (Li et al., 2025) couples LLMs with OSM (Haklay and Weber, 2008) import scripts, while LLMScenario (Chang et al., 2024) composes safety-critical HighD (Krajewski et al., 2018) trajectories in MetaScenario (Chang et al., 2022) through in-context demonstrations.

LLM-powered ADS Enhancement: Recent research integrates LLMs into autonomous driving systems as planners or controllers. The Language-Agent line of work (Mao et al., 2023) is a tool-using LLM decision agent for the planner, while MPC×LLM (Baumann et al., 2025) is a Model Predictive Control parameter tuner that adapts costs and constraints from natural-language context while preserving the underlying optimization. DualAD (Wang et al., 2024) overlays an LLM reasoning layer that issues speed decisions from textual scene encodings, while LeAD (Zhang et al., 2025) employs a dual-rate architecture in which low-frequency LLMs modules supplement high-frequency end-to-end systems in challenging scenarios via chain-of-thought reasoning.

Recent surveys of LLMs in ADS testing and scenario generation (Song et al., 2026; Gao et al., 2026b) confirm that most reviewed papers focus on scenario generation.

2.3 Critical Summary

Across these works, prior LLM-powered systems remain highly fragmented, typically focusing in isolation on either *Scenario Generation* or *ADS Enhancement*. The critical research gap is the absence of a comprehensive, full-pipeline framework for scenario-based testing of ADSs. To close this gap, **PlannerForge** unifies the classic six-component taxonomy (Riedmaier et al., 2020) into a single full-lifecycle framework and extends it with two further stages: *ADS Enhancement* (LLM-guided planner tuning) and *ADS Benchmarking* (cross-planner comparative evaluation under shared scenarios).

3 Problem Formulation

We formalize LLM-powered scenario-based testing as a sequence of language-to-structured-output decisions. Let $\mathcal{U}$ be the space of natural-language utterances, $\mathcal{S}$ that of 2D scenarios produced by an open-source motion-planning simulator, $\mathcal{D} \subseteq \mathcal{S}$ a curated database, Θ the space of motion-planner configurations, $\mathcal{H}$ conversation histories, $\mathcal{O}$ execution outcomes, and $\mathcal{Y}$ natural-language analyses. At dialogue turn t the agent observes $x_t = (u_t, s_t, \theta_t, h_t)$, where $u_t \in \mathcal{U}$, $s_t \in \mathcal{S}$, $\theta_t \in \Theta$, and $h_t \in \mathcal{H}$.

A session begins with Generation or Selection to populate the initial scenario s_0:

$$f_{\text{gen}} : \mathcal{U} \to \mathcal{S} \quad \text{or} \quad f_{\text{sel}} : \mathcal{U} \times \mathcal{D} \to \mathcal{D} \tag{1}$$

Subsequent turns are dispatched by the Module Router, which at a high level selects the next phase in the scenario-based testing pipeline based on the user prompt u_t and dialogue history h_t. Formally, it acts as an intent classifier predicting $\hat{a}_t = \arg\max_{a \in \mathcal{A}} f_{\text{router}}(a \mid u_t, h_t)$ over $\mathcal{A} = \{\text{MODIFY}, \text{TUNE}, \text{TEST}, \text{ANALYSE}, \text{QA}\}$ (the additional QA action returns a free-form answer without invoking any downstream operator; see §4). The router then invokes the corresponding action-conditional maps:

$$\begin{aligned} f_{\text{mod}} &: \mathcal{S} \times \mathcal{U} \to \mathcal{S}, && \text{where output } s' \models \Sigma_{\mathcal{D}} \\ f_{\text{tune}} &: \Theta \times \mathcal{U} \to \Theta, && \text{where output } \theta' \models \Sigma_{\Theta} \\ f_{\text{test}} &: \mathcal{S} \times \Theta \to \mathcal{O} \\ f_{\text{eval}} &: \mathcal{O} \times \mathcal{U} \to \mathcal{Y} \end{aligned}$$

where $\Sigma_{\mathcal{D}}$ is the curated scenario XML dataset and Σ_{Θ} the planner configuration. Crucially, f_{gen}, f_{mod}, and f_{tune} are *constrained generators*: their outputs (denoted s' and θ' above) must satisfy these respective schemas. Producing schema-conformant XML is the central linguistic challenge.

4 Methodology

PlannerForge is a unified framework that integrates LLMs across the entire scenario-based testing pipeline for motion planners, as illustrated in Figure 2. The six modules summarised in the caption (*Generation*, *Selection*, *Module Router*, *Modification*, *Testing*, and *Analysis*) are detailed in the subsections below; the *Module Router* (§4.2) acts as the intent dispatcher that unlocks flexible post-selection navigation.

4.1 Framework Setup

The PlannerForge framework features a chatbot interface (Figure 3) built with a Gradio[3] frontend and

[3] https://gradio.app/

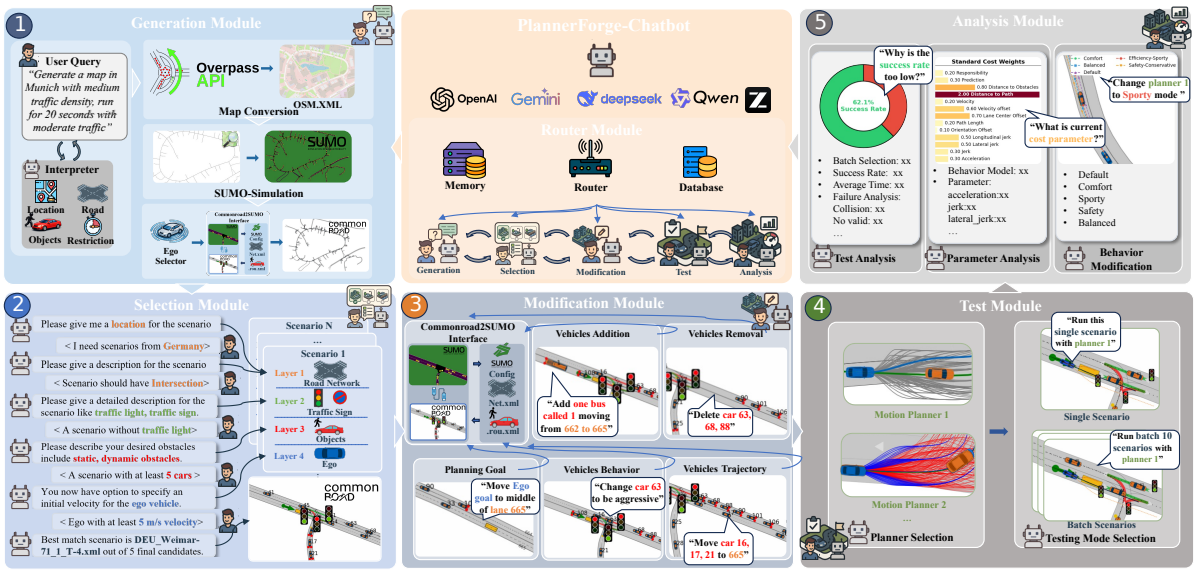


Figure 2: PlannerForge framework with six modules: **Router** (classifies user intent post-selection), **Generation** (OSM+SUMO synthesis), **Selection** (dialogue-guided retrieval from the CommonRoad DB), **Modification** (LLM-guided SUMO edits), **Testing** (Frenetix / MP-RBFN execution), and **Analysis** (LLM-powered result interpretation).

a LangChain[4] backend. To support coherent multi-turn interactions, it manages state across three levels: *Conversational Memory* (LangChain retains recent exchanges and summarises history exceeding 125k tokens), *UI Chat History* (Gradio maintains an unmodified visual log of the conversation), and *Session State* (in-RAM storage for user-specific context and intermediate module outputs).

Scenario Database: Open-source driving scenarios from CommonRoad (Althoff et al., 2017) are stored as XML files augmented with a structured `<Metadata>` element covering four scenario layers (location, roadside constructs, participants, ego vehicle) and indexed in a Chroma vector database (Chroma Team, 2023). Further implementation details and the exact metadata schema are provided in Appendix A.1.

Prompting Techniques. All modules in the framework implement the following prompting techniques (Figure 4), so that pretrained LLMs can be adjusted to our specific tasks (Gao et al., 2026b): *Contextual Prompt (CP)* injects the structured output schema, syntactic constraints, and available operators into the prompt. For instance, the Generation module receives the JSON intent schema with required keys location, road classes, density, vehicle mix, and duration. *Chain-of-Thought (CoT)* structures generation into explicit reasoning steps per module. The Modification scaffold reads: identify target, enumerate route changes, preserve connectivity, and emit the SUMO edit. *In-Context Learning (ICL)* adds a few-shot demonstration examples: positive natural-language to output pairs, plus, where applicable, negative refusal examples that anchor edge-case behavior.

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

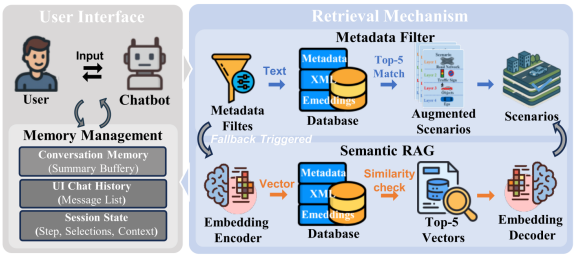


Figure 3: Chatbot front-end of PlannerForge. The Gradio UI exposes the natural-language query box, session state, conversation memory, and scenario retrieval mechanism. The LangChain backend routes user utterances to the six modules of Figure 2.

We denote the prompt used by module M as $\mathbf{P}_M$ (e.g. $\mathbf{P}_{\text{GEN}}, \mathbf{P}_{\text{SEL}}, \mathbf{P}_{\text{ROUTER}}, \mathbf{P}_{\text{MOD}}, \mathbf{P}_{\text{TUNE}}, \mathbf{P}_{\text{EVAL}}$); full prompts for each module are released with the code (Appendix A.8).

4.2 Module Router

Traditional testing frameworks follow a rigid *generation* → *selection* → *modification* → *testing* → *analysis* workflow. PlannerForge breaks this linearity through the **Module Router**. Following the formalisation in §3, after the initial scenario genera-

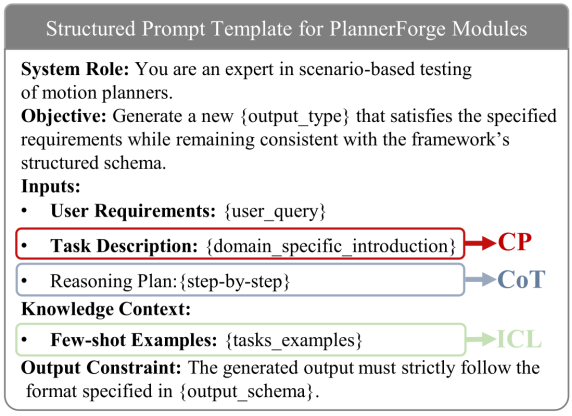


Figure 4: Structured prompt template shared across PlannerForge modules.

tion or selection, the router acts as the intent classifier $f_{\text{router}}(a \mid u_t, h_t)$, mapping the user utterance u_t to an action $\hat{a}_t \in \mathcal{A}$. These actions correspond to five categories: *Scenario Modification* (MODIFY), *Parameter Tuning* (TUNE), *Test Execution* (TEST), *Result Analysis* (ANALYSE), and *General Question & Answer* (QA). This is implemented via two-stage LLM function calling: In the first stage, guided by the router prompt P_{ROUTER}, the LLM identifies the corresponding module and extracts the required arguments `args`, returning both as a structured JSON object. The second stage's Process Engine dispatches `args` to the corresponding module. Full dispatch pseudocode is given in Algorithm 1 (Appendix A.5.1). The router is the key architectural mechanism that distinguishes PlannerForge from prior LLM-assisted testing tools, and the per-module implementations are detailed in the following subsections.

4.3 Scenario Generation Module

When database scenarios are insufficient, PlannerForge generates new CommonRoad scenarios from scratch via a two-stage pipeline. An LLM parses natural-language requests into structured intents, combining real-world road topologies with procedurally simulated traffic.

Stage 1: Map and traffic synthesis. The user describes the desired scenario in natural language (e.g., "Munich intersection with light traffic, focus on a turning truck"). An LLM parses this with prompt $\mathbf{P}_{\text{GEN}}$ into a structured JSON intent specifying location (city or bounding box), drivable road classes, traffic density, vehicle mix, and simulation duration. The bounding box drives an OpenStreetMap query via the Overpass API (Haklay and Weber, 2008); the returned road network is simulated in SUMO (Lopez et al., 2018) and then converted to CommonRoad (Althoff et al., 2017) format. A microscopic SUMO simulation populates the network with vehicles, trucks, buses, and other configurable actor types, producing trajectories that respect car-following and lane-changing dynamics.

Stage 2: Planning problem synthesis. From the populated scenario, the user selects an ego vehicle and a goal region; the LLM may also suggest an ego candidate using strategies such as *first car*, *by type*, or *by index*. A planning problem is then synthesized by attaching an initial state (the ego's current pose) and a goal region (either a chosen lanelet or a forward offset along the ego's trajectory). The result is saved as a standard CommonRoad scenario file ready for downstream modification, testing, and analysis.

4.4 Scenario Selection Module

The Scenario Selection Module facilitates the retrieval of test cases from large-scale databases by abstracting low-level representations into an LLM-guided natural-language dialogue.

We structure retrieval around the layer-based taxonomy introduced by Riedmaier et al. (Riedmaier et al., 2020), indexing scenarios across four metadata layers: *location*, *roadside constructs* (split into the `tags` and `road_net` extractors below), *participants*, and *ego vehicle*. During a five-step dialogue (Figure 2), the LLM extracts a structured slot $\hat{\ell}_k$ from the user's utterance at step k (geographical codes, discrete keywords, kinematic ranges) using a per-slot prompt $\mathbf{P}_{\text{SEL}}^{(k)}$. Let $\mathcal{D}^{(0)} = \mathcal{D}$ denote the full database. For $k = 1, \ldots, 5$ corresponding to (`location`, `tags`, `road_net`, `obstacles`, `velocity`),

$$\mathcal{D}^{(k)} = \mathcal{D}^{(k-1)} \cap \texttt{match}(\hat{\ell}_k), \tag{2}$$

monotonically pruning the candidate set. The module returns top-$k(\mathcal{D}^{(5)})$ when non-empty, and otherwise falls back to a SentenceTransformer (Reimers and Gurevych, 2019) semantic-similarity search over $\mathcal{D}$ keyed by the concatenated dialogue $u_{1:5}$, guaranteeing retrieval by contextual meaning when exact metadata matches fail. Ablation results for the retrieval pipelines are detailed in Appendix A.3.3.

4.5 Scenario Modification Module

CommonRoad scenarios encode fixed pre-recorded trajectories. To make edits tractable for the LLM, we route them through the CommonRoad–SUMO interface (Klischat et al., 2019): scenarios are converted to a `.net.xml` (network topology) plus `.vehicles.rou.xml` (routes and behaviour) pair, the LLM (invoked with a per-task prompt $\mathbf{P}^{\tau}_{\text{MOD}}$ for $\tau \in \{T, B, P, G\}$) emits a modified SUMO file, and the round-trip back to CommonRoad produces kinematically feasible trajectories. We support four edit categories:
(1) **Trajectory (T)** modifications redirect vehicles by updating edge sequences (two-stage prompt: the network topology is summarised into valid routes, then the edit is generated against the route file);
(2) **Behaviour (B)** modifications swap each vehicle's `<vType>` against six car-following presets (*Aggressive*, *Cautious*, *Emergency*, *Eco*, *Balanced*, *Speeder*) while preserving `vClass`;
(3) **Population (P)** modifications add or remove vehicle entries with type, departure, and valid routes derived from the topology summary, as shown in Figure 10 (Appendix A.4.3) with vehicle removal and addition examples;
(4) **Goal (G)** modifications edit the planning problem by updating the goal region of the Ego Vehicle in place without a SUMO round-trip.

4.6 Planner Testing and Enhancement Module

This module implements the test executor f_{test} and parameter tuner f_{tune}. The executor f_{test} abstracts the simulation environment, parameter parsing, and logging via a unified interface:

$$f_{\text{test}}(s, \theta) \triangleq \pi_P(s, \theta) = (\tau, c, m) \in \mathcal{O}, \qquad (3)$$

where $\mathcal{O}$ comprises a trajectory τ, collision flag $c \in \{0, 1\}$, and cost log m. The tuner f_{tune} enables natural-language ADS Enhancement. Users state qualitative presets (e.g., *Safety-Conservative*) or explicit weight adjustments (e.g., "increase `distance_to_obstacles`"). The LLM, invoked with the tuning prompt $\mathbf{P}_{\text{TUNE}}$, interprets utterance u_t and emits a schema-conformant YAML override, updating the configuration $\theta \to \theta'$ in place while preserving formatting.

To enable ADS Benchmarking (see Appendix A.7), PlannerForge wraps two classical motion planners, sampling-based Frenetix (Trauth et al., 2024) and learning-based MP-RBFN (Kaufeld et al., 2025), under this π_P interface. Execution operates in *single-scenario* mode for individual analysis, or *batch* mode (preset sizes, query-driven, or custom sets) for parallel statistical evaluation.

4.7 Result Analysis Module

The Analysis Module realizes the operator $f_{\text{eval}} : \mathcal{O} \times \mathcal{U} \to \mathcal{Y}$ defined in §3, translating batch outcomes into natural-language feedback. Given a batch of B outcomes $\{o_i = (\tau_i, c_i, m_i)\}_{i=1}^{B}$ produced under planner configuration θ (each o_i as defined in Eq. 3: trajectory τ_i, collision flag c_i, per-step cost log m_i) and a user utterance u_t (e.g. "why is the success rate low?"), the module assembles the analysis prompt $\mathbf{P}_{\text{EVAL}}$ over four context blocks: (i) batch-level statistics aggregated from $\{c_i\}$ and $\{m_i\}$ (success rate, mean trajectory length, collision count, mean cost); (ii) chronological per-scenario logs; (iii) the active configuration θ; and (iv) the underlying CSV log path. The LLM returns a response $y \in \mathcal{Y}$ comprising quantitative metrics, a failure-mode breakdown (*collision*, *timeout*, *kinematic infeasibility*), the cost configuration used, and qualitative correlations between outcomes and scenario characteristics, together with parameter-adjustment recommendations that close the loop with the tuner f_{tune}. A complementary *behavior-comparison* path retains the chronological history of past batches with their configurations $\{(\theta^{(k)}, \{o_i^{(k)}\}_i)\}_k$ and lets the LLM reason about which cost weights changed between runs and how those changes shifted the success/failure profile.

5 Results & Discussion

In this section, we present the performance of PlannerForge with quantitative results. Five cloud API models (Qwen3.6-plus (Qwen Team, 2026b), Deepseek-v3.2 (Liu et al., 2025), Glm-5 (Zeng et al., 2026), Gemini-3-flash[5], Gpt-5.4-mini[6]) and open-source models (Qwen3.6:35b (Qwen Team, 2026a) (think/no-think), Gemma4:31b[7] (think/no-think), Gpt-oss:20b (Agarwal et al., 2025) (think) from Ollama[8] are evaluated. Metric definitions and scoring procedures for each task are documented

[5] https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/
[6] https://openai.com/index/introducing-gpt-5-4-mini-and-nano/
[7] https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/
[8] https://ollama.com/

in the Supplementary Material.

5.1 Quantitative Evaluation

We evaluate PlannerForge on all 5 tasks (Fig. 5), where the modification task spans the 4 sub-tasks T/B/P/G, yielding 8 task slices. We use N=200 queries per (sub-)task cell, 10 model variants, and 5 prompt conditions, from a bare `baseline` (zero shot) to `cp_icl_cot` (context prompting + in-context examples + chain-of-thought). For each task, we report the best-performing model under the prompt condition that achieves it, while per-cell ablations are in the appendix.

Generation: Glm-5 with `cp_cot` (context prompting + chain-of-thought) achieves the best score, 0.957, improving over its bare baseline of 0.783 by +0.174. This suggests that CP+CoT is sufficient to nearly saturate intent parsing, while adding ICL provides no further gain. Because this task uses the Overpass API to interface with OpenStreetMap, the most difficult part is simply interpreting the user's free-form query into a structured intent. **Selection:** Qwen3.6-plus with `cp_cot` achieves the best `sat_all` (joint satisfaction of all five retrieved scenarios) score, 0.880, compared with a baseline of 0.180 (+0.700). Selection is the hardest task because the strict five-stage filter fails whenever any extracted slot is incorrect (per-slot extract in Appendix A.3.3). Adding ICL on top of CoT (`cp_icl_cot`, 0.835) can further hurt performance by encouraging over-confident slot guesses. **Modification:** Under `cp_icl_cot`, Qwen3.6-plus reaches ∼100% on the headline checks for all four sub-tasks: *ends %* (redirected route terminates at target edge) for T, *preset %* (behaviour vector matches preset) for B, *count_match %* (vehicle count changes) for P, and *edge-Ground Truth (GT) %* (extracted target lanelet matches GT) for G. Because these modifications target SUMO configuration files, basic structural edits (T, P, G) are straightforward with high zero-shot baselines (99.0%, 99.0%, 88.5%). In contrast, behavior modification (B) requires injecting precise parameter vectors, since it fails completely zero-shot (0%) but reaches 100% once advanced prompting provides the necessary context. **Module Router:** Gemma4:31b with `cp_icl_cot` achieves 0.997, improving over its baseline of 0.722 by +0.275. In comparison, a hand-crafted regex router reaches only 45.5% on the same corpus. This highlights that conversational intent is too diverse for rigid keyword matching, but is easily solved by LLMs via function calling given a clear JSON schema. **Planner Testing and Enhancement:** Gpt-5.4-mini with `cp_icl` achieves a perfect score of 1.000, compared with a baseline of 0.675 (+0.325), and outperforms a schema-constrained YAML editor baseline of 72.5%. This shows that translating abstract user requests (e.g., "drive safely") into precise YAML parameter adjustments requires semantic understanding that rule-based editors lack. As downstream validation, 155 of 191 emitted YAML configurations (81.2%) run end-to-end in Frenetix.

Per-module scores do not by themselves show that the stages compose. Table 1 therefore chains them on N=200 seed queries, each stage consuming the previous stage's actual output (Generation → Database → Selection → Modification → Test → Enhancement), for a commercial backend (`qwen3.6-plus`) and an open-source one (`qwen3.6:35b`) under `cp_icl_cot`. Both start at 96% after Generation; Selection and Modification are the leak points, and the funnel ends at 83% / 78% cumulative success (commercial / open) with mean cost ≈126 s / 99 s and ≈42.5k / 32.6k tokens per scenario.

Take-aways. Schema-constrained tasks (Generation, Router, Planner) saturate with CP+CoT or CP+ICL, while semantically dense tasks (Selection, Modification) require advanced promptings. Chained end-to-end, the pipeline retains 83% / 78% of seed queries (commercial / open). Open-source 20–35B models perform slightly lower than commercial APIs but are fully capable of driving the entire pipeline. Notably, applying additional prompting techniques to open-source models with native reasoning (e.g., Think variants) disrupts their internal reasoning (Fig. 5), increasing latency and token consumption while degrading performance.

5.2 Comparison with Prior Scenario-Testing Tools

The modules above are reliable at schema-checked execution. We next show that they also beat or lift the strongest available baseline at each stage, including Enhancement versus the hand-set Default planner configuration.

Generation (Table 3). CommonRoad has no natural-language DSL, so alternative sourcing relies on GUI drawing. Against Scenario Factory 2.0 (Finkeldei et al., 2025), the rule-based state of the art, PlannerForge is an order of magnitude slower per scenario, because it runs an LLM where SF 2.0 runs a procedure. In exchange it delivers more ex-

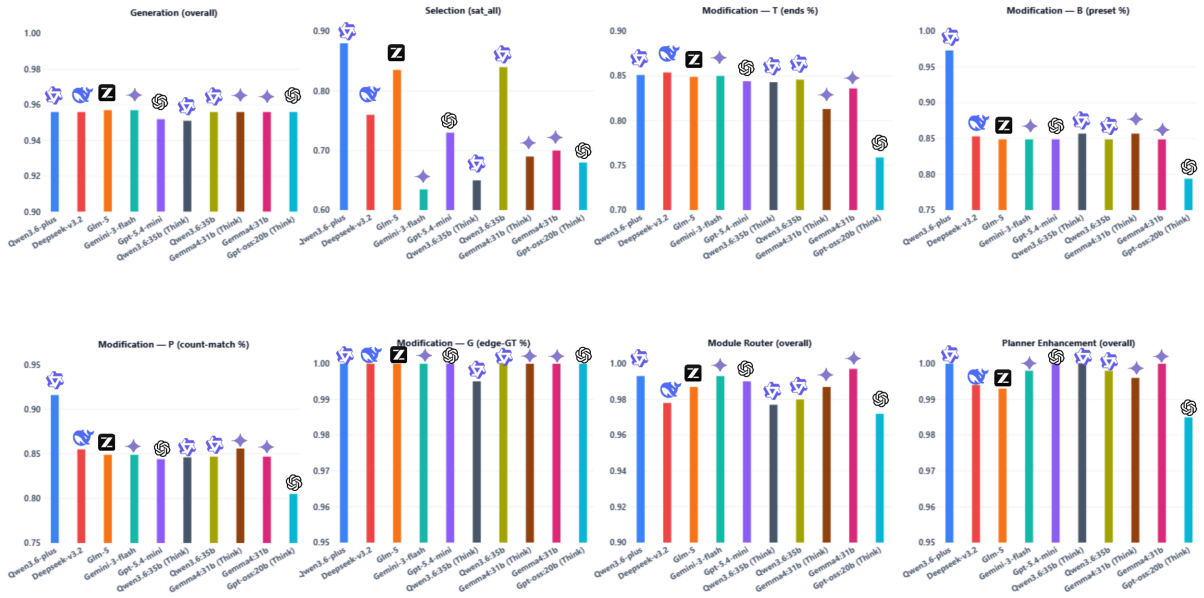


Figure 5: Best overall score per model on eight evaluated task slices (maximum across the prompt conditions).

Table 1: End-to-end pipeline success (N=200 seed queries). Each stage consumes the previous stage's actual output: Generation → Database → Selection → Modification → Test → Enhancement. Values are Commercial/Open (C/O), using `qwen3.6-plus` and `qwen3.6:35b` with the `cp_icl_cot` prompt. **FR** is the failure rate of that stage; **Cum. SR** is cumulative success up to it. Latency and token cost are means per scenario.

Stage	in→out (C/O)	FR ↓	Cum. SR ↑	Latency ↓	Token ↓	Hardware
① Generation	200→192 / 200→192	4%/4%	96%/96%	21.6 s/21.0 s	4.5k/4.6k	API/27G
② Database	192→192	0%/0%	96%/96%	1.8 s/1.8 s	0/0	CPU
③ Selection	192→181 / 192→170	6%/11%	91%/85%	29.3 s/26.7 s	8.5k/8.5k	API/27G
④ Modification	181→165 / 170→156	9%/8%	83%/78%	44 s/23 s	29k/19k	API/27G
– G (goal)	46→46 / 43→43	0%/0%		3.7 s/2.5 s	7.5k/7.5k	API/27G
– B (behaviour)	45→42 / 43→41	7%/5%		56.9 s/31.2 s	33k/22k	API/27G
– P (add/remove)	45→38 / 42→37	16%/12%		56.6 s/31.7 s	40k/21k	API/27G
– T (trajectory)	45→39 / 42→35	13%/17%		63.1 s/29.2 s	34k/23k	API/27G
⑤ Test	165→165 / 156→156	0%/0%	83%/78%	24.6 s/23.9 s	0/0	CPU
⑥ Enhancement	165→165 / 156→156	0%/0%	83%/78%	4.5 s/2.6 s	0.5k/0.5k	API/27G
→ End-to-end	**200→165 / 200→156**		**83%/78%**	**≈126 s/99 s**	**≈42.5k/32.6k**	API/27G

Table 2: Cost-tuning across batch sizes, paired per scenario. Each batch is tuned by three independent LLM calls (qwen3.6-plus, `cp_icl_cot`, temperature 0.0); we report mean ± SD over the three rounds. The planner is deterministic for a fixed (scenario, configuration) pair, so the LLM call is the only stochastic component. All 15 rounds returned the identical configuration.

N	Success ↑ (b→a)	Δ (pp)	Collision ↓ (b→a)	Δ (pp)
50	46.7%→68.0±9.1%	+21.3 ± 6.8	20.7%→7.3±2.5%	−13.3 ± 5.2
100	51.0%→68.6±4.2%	+17.6 ± 2.6	20.3%→9.8±0.9%	−10.5 ± 1.7
200	51.3%→71.5±2.4%	+20.1 ± 0.5	20.6%→8.2±1.0%	−12.4 ± 1.5
300	51.6%→69.8±0.5%	+18.2 ± 0.7	18.9%→7.8±0.6%	−11.1 ± 1.3
400	50.4%→70.2±0.4%	+19.8 ± 0.3	19.0%→8.4±0.4%	−10.6 ± 0.3

ecutable scenarios (193 vs. 144 of 200), realises 92–96% of the requested city, road and vehicle attributes that SF 2.0 cannot target at all, produces seven traffic-participant classes rather than four, and induces 3.3× more planner collisions (20.0% vs. 6.1%).

Table 3: **Generation.** PlannerForge vs. the rule-based state of the art, Scenario Factory 2.0 (Finkeldei et al., 2025), on 200 queries across 50 cities. PlannerForge receives the full natural-language query; SF 2.0 receives the extracted target city, its native input. **Exec. S** = scenarios that generate and execute in the planner. ✗ = attribute not targetable. †SF 2.0 is given the city directly.

Method	Time ↓	Exec. S ↑	City ↑	Road ↑	Vehicle ↑	Diverse ↑	Coll ↑
SF 2.0	**2.2 s**	144/200	72%†	✗	✗	4	6.1%
PlannerForge	21.6 s	**193/200**	**96.0%**	**92.0%**	**95.6%**	**7**	**20.0%**

Selection (Table 4). The CommonRoad GUI supports only manual parameter filters. Against BM25 keyword search (Robertson and Zaragoza, 2009), which is effectively free at <0.01 s and zero tokens, PlannerForge costs 21.6 s and 8.7k tokens per query. BM25 already finds a valid scenario somewhere in the top five for 86.0% of queries, so the gap at Any@5 is modest (96.5%). The gap at

Table 4: **Selection.** PlannerForge vs. BM25 keyword search (Robertson and Zaragoza, 2009) on 200 natural-language queries over a 500+ scenario database; the retrieval backend is shared. **Satisfy@1 / Any@5** = the request is satisfied at rank 1 / anywhere in the top 5.

Retrieval (top-5)	Latency ↓	Token ↓	Satisfy@1 ↑	Any@5 ↑
Keyword search / BM25	<**0.01 s**	**0**	67.5%	86.0%
PlannerForge (LLM)	21.6 s	8.7k	**92.0%**	**96.5%**

Table 5: **Modification.** PlannerForge (PF) four edit types vs. From-Words-to-Collisions (Gao et al., 2025) on the same 200 base scenarios. **Exec. S** = planner-runnable; **Phy. Val.** = physically valid share; **New Coll.** = valid new collisions; **min_risk** = mean base→modified risk (0 = collision, 5 = safe).

Method	Time ↓	Tokens ↓	Exec. S ↑	Phy. Val. ↑	New Coll. ↑	Goal ↓	min_risk ↓
FWtC	51 s	17.6k	200/200	31.0%	16	24.2%	1.84→1.69
PF (Behaviour)	56 s	24.2k	200/200	98.0%	31	47.2%	1.84→1.61
PF (Trajectory)	60 s	22.1k	194/200	97.9%	32	50.5%	1.84→1.51
PF (Participant)	63 s	21.7k	192/200	94.8%	**58**	35.7%	**1.84→1.20**
PF (Goal)	**8 s**	**7.1k**	199/200	**100%**	45	**22.6%**	1.84→1.36

rank 1 is what matters for an interactive tool: 67.5% vs. 92.0%, because LLM slot extraction resolves paraphrase, location ambiguity and implicit range constraints that keyword matching cannot.

Modification (Table 5). Against From-Words-to-Collisions (Gao et al., 2025), a recent LLM-based safety-critical modification tool, the decisive difference is physical validity. FWtC writes raw coordinates without vehicle dynamics, so roughly 70% of its edits are kinematically impossible, and it offers a single edit type. Because PlannerForge routes every edit through SUMO, all four of its edit types stay above 94% valid, and Participant (1.84→1.20, 58 new collisions) and Goal (1.84→1.36, 45) stress the planner considerably harder than FWtC's valid edits (1.84→1.69, 16).

Enhancement (Table 2). The LLM retunes cost weights against the hand-set Default configuration across five batch sizes (N=50–400), three independent calls each, scored paired per scenario. Success rises in every batch (+17.6 to +21.3 pp) and collisions fall (10.5 to 13.3 pp); at N=400, 50.4%→70.2%. Spread shrinks with N (±6.8 pp at 50 vs. ±0.3 pp at 400). A Frenetix vs. MP-RBFN dispatch is in Appendix A.7.

Taken together, the LLM buys attribute control in Generation, rank-1 precision in Selection, physically valid edits in Modification, and a lift over Default in Enhancement, at a cost in seconds and tokens that the classical tools do not pay.

5.3 Discussion: Transferable Insights

Three findings generalise to other structured-output agent tasks. **Prompt techniques match distinct failure modes.** CP saturates closed vocabularies (`Gpt-5.4-mini`: Planner full-YAML 36.1%→100%, Router 43.5%→91.0%). ICL is required for refusals (out-of-vocab parameters: 0% baseline, 44.4% CP, 100% only with ICL). CoT helps joint constraints (Selection `sat_all` 46.0%→72.0% from CP to CP+CoT). Match the prompt to the failure mode rather than stacking every technique. **External CoT can conflict with native thinking.** On Selection `sat_all`, adding CoT on top of CP hurts every reasoning-enabled model (65.0→40.0%, 67.0→30.0%, 58.0→32.0%) while lifting non-thinking `Qwen3.6:35b` (63.5→83.0%). Treat reasoning models as a distinct prompting regime. **Reliability needs an executable harness.** Code around each LLM call parses, schema-validates, and scores both form and downstream execution. The prompt raises the hit rate; the harness makes the stage dependable.

6 Conclusion and Future Work

We presented **PlannerForge**, a full-lifecycle LLM-agent framework for scenario-based testing of motion planners, with a Module Router, schema-checked modules, and a unified motion planner interface. Across 80,000 off-the-shelf calls with no fine-tuning, modules score from 0.88 (Selection) to 1.00 (Planner Testing), with Generation at 0.957, Modification near 100% on the headline checks, and the Router at 0.997. End-to-end chaining from Generation through Enhancement retains 83% / 78% of seed queries (commercial / open). Against Scenario Factory 2.0, BM25, and From-Words-to-Collisions, it is more attribute-faithful in generation (193 vs. 144 executable), more precise at rank 1 selection (92.0% vs. 67.5%), and physically valid in modification (≥94% vs. 31%). On planner safety-critical performance, generated scenarios induce 20.0% collisions versus 6.1% for Scenario Factory 2.0, and cost-tuning against Default at N=400 lifts success from 50.4% to 70.2% while cutting collisions from 19.0% to 8.4%. Future work will close the planner loop and extend the framework to simulators such as CARLA.

Limitations

Measured failure modes. In the end-to-end chain, Trajectory and Population edits drop 13–17% and 12–16% of surviving queries on simulation round-trip, not on headline semantics (Table 1, Appendix A.4.2). Selection is the other leak. Tag over-prediction drives the 6%/11% commercial/open drop, and the best `sat_all` is 0.880 (Appendix A.3.3). The Analysis Module is not scored against ground truth.

Scope and domain. PlannerForge runs open-loop. Other agents follow recorded or SUMO-exported trajectories and do not react to the ego vehicle. Closed-loop falsification is left to future work. Quantitative planner, collision, and cost-tuning results use Frenetix. The MP-RBFN comparison is qualitative (Appendix A.7). Evaluation uses CommonRoad, and the modification corpus is Germany-dominated (DEU is 80–92 queries per task among 15 country codes).

Ethical Considerations

PlannerForge is driven by off-the-shelf LLM agents that can hallucinate structured outputs. In our evaluation this appears as invented scenario tags, invalid map or vehicle identifiers, misrouted module calls, and Analysis claims that are not supported by the run logs. Unchecked, such errors can produce invalid tests or misleading planner diagnostics. Every scored module therefore parses, schema-validates, and executes the model output before it is accepted. Residual risk remains where that check is incomplete, in particular the unscored Analysis module.

Acknowledgements

The authors wrote the initial draft and used LLMs only to improve grammar, clarity, and readability. They reviewed every suggestion and take responsibility for the paper.

References

Erfan Aasi, Phat Nguyen, Shiva Sreeram, Guy Rosman, Sertac Karaman, and Daniela Rus. 2024. Generating out-of-distribution scenarios using language models. *arXiv preprint arXiv:2411.16554*.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Matthias Althoff, Markus Koschi, and Stefanie Manzinger. 2017. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE.

Philipp Bauerfeind, Amir Salarpour, David Fernandez, Pedram MohajerAnsari, Johannes Reschke, and 1 others. 2025. David vs. goliath: A comparative study of different-sized llms for code generation in the domain of automotive scenario generation. *arXiv preprint arXiv:2510.14115*.

Nicolas Baumann, Cheng Hu, Paviththiren Sivasothilingam, Haotong Qin, Lei Xie, Michele Magno, and Luca Benini. 2025. Enhancing autonomous driving systems with on-board deployed large language models. In *Robotics: Science and Systems XXI*.

Johannes Betz, Melina Lutwitzi, and Steven Peters. 2024. A new taxonomy for automated driving: Structuring applications based on their operational design domain, level of automation and automation readiness. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7.

Xuan Cai, Xuesong Bai, Zhiyong Cui, Danmu Xie, Daocheng Fu, and 1 others. 2026. Text2Scenario: Text-driven scenario generation for autonomous driving test. *Automotive Innovation*.

Cheng Chang, Dongpu Cao, Long Chen, Kui Su, Kuifeng Su, Yuelong Su, Fei-Yue Wang, Jue Wang, Ping Wang, Junqing Wei, and 1 others. 2022. Metascenario: A framework for driving scenario data description, storage and indexing. *IEEE Transactions on Intelligent Vehicles*, 8(2):1156–1175.

Cheng Chang, Siqi Wang, Jiawei Zhang, Jingwei Ge, and Li Li. 2024. Llmscenario: Large language model driven scenario generation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Chroma Team. 2023. Chroma: The ai-native open-source embedding database. `https://www.trychroma.com/`. Accessed: 2026-01-18.

Yao Deng, Zhi Tu, Jiaohong Yao, Mengshi Zhang, Tianyi Zhang, and Xi Zheng. 2025. Target: Traffic rule-based test generation for autonomous driving via validated llm-guided knowledge extraction. *IEEE Transactions on Software Engineering*, 51(7):1950–1968.

Wenhao Ding, Chejian Xu, Mansur Arief, Haohong Lin, Bo Li, and Ding Zhao. 2023. A survey on safety-critical driving scenario generation—a methodological perspective. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):6971–6988.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. Carla: An open urban driving simulator. In *Conference on robot learning*. PMLR.

Florian Finkeldei, Christoph Thees, Jan-Niklas Weghorn, and Matthias Althoff. 2025. Scenario factory 2.0: Scenario-based testing of automated vehicles with CommonRoad. *Automotive Innovation*, 8(2):207–220.

Yuan Gao, Wenting Miao, Mattia Piccinini, Haoyu Wang, Qunying Song, and Johannes Betz. 2026a. Chat2scenic: An iterative RAG-based framework for scenario generation in autonomous driving. In *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. To appear.

Yuan Gao, Mattia Piccinini, Korbinian Moller, Amr Alanwar, and Johannes Betz. 2025. From words to collisions: Llm-guided evaluation and adversarial generation of safety-critical driving scenarios. In *2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2134–2141.

Yuan Gao, Mattia Piccinini, Yuchen Zhang, Dingrui Wang, Korbinian Moller, Roberto Brusnicki, Baha Zarrouki, Alessio Gambi, Jan Frederik Totz, Kai Storms, Steven Peters, Andrea Stocco, Bassam Alrifaee, Marco Pavone, and Johannes Betz. 2026b. Foundation models in autonomous driving: A survey on scenario generation and scenario analysis. *IEEE Open Journal of Intelligent Transportation Systems*, pages 1–1.

An Guo, Yuan Zhou, Haoxiang Tian, Chunrong Fang, Yunjian Sun, Weisong Sun, Xinyu Gao, Anh Tuan Luu, Yang Liu, and Zhenyu Chen. 2024. Sovar: Build generalizable scenarios from accident reports for autonomous driving testing. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 268–280.

Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18.

SAE International. 2021. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE J3016*.

Marc Kaufeld, Mattia Piccinini, and Johannes Betz. 2025. Mp-rbfn: Learning-based vehicle motion primitives using radial basis function networks. In *2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1262–1269.

Moritz Klischat, Octav Dragoi, Mostafa Eissa, and Matthias Althoff. 2019. Coupling sumo with a motion planning framework for automated vehicles. In *SUMO User Conference*, pages 1–9.

Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. 2018. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, page 2118–2125. IEEE.

Shuyang Li, Talha Azfar, and Ruimin Ke. 2025. Chatsumo: Large language model for automating traffic scenario generation in simulation of urban MObility. *IEEE Transactions on Intelligent Vehicles*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. 2023. A language agent for autonomous driving. *arXiv preprint arXiv:2311.10813*.

Yuewen Mei, Tong Nie, Jian Sun, and Ye Tian. 2025. LLM-Attacker: Enhancing closed-loop adversarial scenario generation for autonomous driving with large language models. *IEEE Transactions on Intelligent Transportation Systems*.

Hiroki Nakamura, Husam Muslim, Ryosuke Kato, Sandra Préfontaine-Watanabe, H Nakamura, H Kaneko, Hisashi Imanaga, Jacobo Antona-Makoshi, Sou Kitajima, Nobuyuki Uchida, and 1 others. 2022. Defining reasonably foreseeable parameter ranges using real-world traffic data for scenario-based safety assessment of automated vehicles. *IEEE Access*, 10:37743–37760.

Demin Nalic, Tomislav Mihalj, Maximilian Bäumler, Matthias Lehmann, Arno Eichberger, and Stefan Bernsteiner. 2020. Scenario based testing of automated driving systems: A literature survey. In *FISITA web Congress*, volume 10, page 1.

Nenad Petrovic, Krzysztof Lebioda, Vahid Zolfaghari, André Schamschurko, Sven Kirchner, Nils Purschke, Fengjunjie Pan, and Alois Knoll. 2024. Llm-driven testing for autonomous driving scenarios. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 173–178. IEEE.

Qwen Team. 2026a. Qwen3.6-35B-A3B: Agentic coding power, now open to all.

Qwen Team. 2026b. Qwen3.6-Plus: Towards real world agents.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. 2020. Survey on scenario-based safety assessment of automated vehicles. *IEEE access*, 8:87456–87477.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Bo-Kai Ruan, Hao-Tang Tsui, Yung-Hui Li, and Hong-Han Shuai. 2024. Traffic scene generation from natural language description for autonomous vehicles with large language model. *arXiv preprint arXiv:2409.09575*.

Mysore Narasimhamurthy Sharath and Babak Mehran. 2021. A literature review of performance metrics of automated driving systems for on-road vehicles. *Frontiers in Future Transportation*, 2:759125.

Qunying Song, Emelie Engström, and Per Runeson. 2024. Industry practices for challenging autonomous driving systems with critical scenarios. *ACM Trans. Softw. Eng. Methodol.*, 33(4).

Qunying Song, He Ye, Mark Harman, and Federica Sarro. 2026. Generative ai for testing of autonomous driving systems: A survey. *ACM Transactions on Software Engineering and Methodology*.

Shuhan Tan, Boris Ivanovic, Xinshuo Weng, Marco Pavone, and Philipp Kraehenbuehl. 2023. Language conditioned traffic generation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2714–2738. PMLR.

Lei Tang, Ruijie Wang, Zhanwen Liu, Yunji Liang, Yuanyuan Niu, Wei Zhu, and Zongtao Duan. 2025. Scenario-based accelerated testing for SOTIF in autonomous driving: A review. *IEEE Internet of Things Journal*, 12(2):1453–1470.

Shuncheng Tang, Zhenya Zhang, Jixiang Zhou, Lei Lei, Yuan Zhou, and Yinxing Xue. 2024. Legend: A top-down approach to scenario generation of autonomous driving systems assisted by large language models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 1497–1508.

Rainer Trauth, Korbinian Moller, Gerald Würsching, and Johannes Betz. 2024. Frenetix: A high-performance and modular motion planning framework for autonomous driving. *IEEE Access*.

Dingrui Wang, Marc Kaufeld, and Johannes Betz. 2024. Dualad: Dual-layer planning for reasoning in autonomous driving. *arXiv preprint arXiv:2409.18053*.

Waymo. 2018. Waymo one: The next step on our self-driving journey.

Hermann Winner, Karsten Lemmer, Thomas Form, and Jens Mazzega. 2019. Pegasus—first steps for the safe introduction of automated driving. In *Road Vehicle Automation 5*, pages 185–195, Cham. Springer International Publishing.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Jiawei Zhang, Chejian Xu, and Bo Li. 2024. Chatscene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15459–15469.

Yuhang Zhang, Jiaqi Liu, Chengkai Xu, Peng Hang, and Jian Sun. 2025. Lead: The llm enhanced planning system converged with end-to-end autonomous driving. *arXiv preprint arXiv:2507.05754*.

Yongqi Zhao, Wenbo Xiao, Tomislav Mihalj, Jia Hu, and Arno Eichberger. 2024. Chat2scenario: Scenario extraction from dataset through utilization of large language model. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 559–566. IEEE.

A Appendix

A.1 Implementation Details

Hardware. All experiments are conducted on a single workstation equipped with an NVIDIA GeForce RTX 5090 GPU (Blackwell architecture, 32 GB GDDR7, 21,760 CUDA cores, 1,792 GB/s memory bandwidth, 575 W TGP). The host CPU is an Intel Core i9-14900K (24 cores / 32 threads), paired with 128 GB DDR5 system memory and a 2 TB NVMe SSD for scenario storage and SUMO simulation caches.

Large Language Models. The LLMs used in the experiments are listed in Table 6. Models are grouped by provider, and the "Think" column indicates whether the model's internal reasoning mode is enabled at inference time (`ON`) or disabled (`OFF`). All cloud-hosted closed-weight models run with reasoning disabled to keep their structured-output behavior comparable. For local open-weight reasoning models (`gpt-oss`, `gemma-4`) reasoning is on by default, and for `qwen3.6:35b` and `gemma4:31b` we evaluate both modes.

Table 6: LLM backends used in the experiments, grouped by provider. "Think" = internal reasoning mode at inference time. "VRAM" = measured peak resident GPU memory of the local Ollama backend (`Q4_K_M` weights, 32k-token context window) on the RTX 5090; cloud models are served over a provider API and use no local VRAM.

Model	Provider	Think	VRAM
Cloud API			
Qwen3.6-plus	DashScope	OFF	API
Deepseek-v3.2	DashScope	OFF	API
Glm-5	DashScope	OFF	API
Gemini-3-flash	Google	OFF	API
Gpt-5.4-mini	OpenAI	OFF	API
Local Ollama (RTX 5090, Q4_K_M)			
Qwen3.6:35b	Ollama	ON	27 GB
Qwen3.6:35b	Ollama	OFF	27 GB
Gemma4:31b	Ollama	ON	27 GB
Gemma4:31b	Ollama	OFF	27 GB
Gpt-oss:20b	Ollama	ON	14 GB

Pinned identifiers (reproducibility). Commercial models are invoked by their exact provider model-id: `qwen3.6-plus`, `deepseek-v3.2`, `glm-5` (DashScope), `gemini-3-flash-preview` (Google), `gpt-5.4-mini` (OpenAI). Open backends are pinned Ollama image tags: `qwen3.6:35b @ 07d3521`, `gemma4:31b @ 6316f06`, `gpt-oss:20b @ 17052f9`. A fully commercial-API-free run uses only the five open backends.

Framework components. PlannerForge integrates various data sources and simulation tools into a unified pipeline:

- **Frontend.** The framework is exposed as a web application built with **Gradio**, organized as two browser tabs: a *Main* tab hosting the chatbot interface for query, modification, testing, and analysis, and a separate *Generate* tab for the OSM-based scenario-generation pipeline. The chatbot widget streams LLM responses and renders generated/simulated scenarios as inline animated GIFs.

- **Conversational memory.** To preserve chat history context across multi-turn interactions, the Python runtime leverages Langchain's[9] `ConversationSummaryBufferMemory`, which automatically summarises older messages to fit within token limits.

- **Simulation backends.** PlannerForge operates on the CommonRoad scenario format (Althoff et al., 2017), with traffic dynamics provided by the SUMO microscopic traffic simulator (Lopez et al., 2018) via the CommonRoad-SUMO interface (Klischat et al., 2019). We integrate two motion planners: **Frenetix** (Trauth et al., 2024) (sampling-based, weighted cost functions) and **MP-RBFN** (Kaufeld et al., 2025) (learning-based, radial basis function networks).

- **Scenario database.** We index 500+ curated CommonRoad scenarios in a persistent **ChromaDB** (Chroma Team, 2023) vector store (`PersistentClient`), using SentenceTransformer (Reimers and Gurevych, 2019) embeddings for semantic retrieval. Each scenario includes a structured metadata dictionary covering location (country, road network type), roadside infrastructure (e.g., traffic light presence), participant characteristics (dynamic and static obstacle counts and types), and ego-vehicle properties (initial velocity), enabling exact-match filtering. The hybrid metadata-filtering and RAG strategy is described in the Methodology.

- **OSM data sources.** The Generation Module retrieves real-world road networks from

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

the OpenStreetMap project (Haklay and Weber, 2008) via two public APIs: (i) the **Overpass API** for raw `.osm` XML fetching. We use four mirror endpoints (`overpass-api.de`, `overpass.kumi.systems`, `overpass.private.coffee`, `maps.mail.ru`) as automatic fallbacks for resilience; and (ii) **Nominatim** (accessed through the `osmnx` library) for geocoding city names into bounding boxes. A polite delay between Overpass requests respects the public-endpoint rate limits.

A.2 Scenario Generation Module

A.2.1 LLM Scope and Hallucination Isolation

The Generation Module separates *language-level* tasks (handled by the LLM) from *geometric- and dynamic-level* tasks (handled by deterministic tools). This split is a deliberate design choice: it preserves the flexibility of natural-language scenario authoring while preventing LLM hallucinations from propagating into map topology or vehicle dynamics.

LLM responsibilities. The LLM is invoked at two well-typed boundaries:

- **Stage 1 — intent parsing.** The free-form utterance is parsed into a JSON intent with fixed keys: location (city or explicit bounding box), drivable road classes (subset of OSM `highway` tags), traffic density (categorical: `low/medium/high`), vehicle mix (counts per `vClass`), and simulation duration. Each key has a schema-defined default that is applied when the LLM omits or emits an invalid value. The per-key defaults-compliance rates are reported in Table 7.
- **Stage 2 — ego selection.** From the populated traffic, the LLM picks one of three strategies (*first car*, *by type*, or *by index*) to identify an ego vehicle. The goal region is then attached deterministically (*chosen lanelet* or *forward offset along the ego trajectory*).

Deterministic-tool responsibilities. Three downstream stages execute without LLM involvement:

- **OSM/Overpass fetch** takes the bounding box as input and returns the corresponding `.osm` data, with mirror-endpoint failover (§A.1).
- **CR–SUMO conversion** (Althoff et al., 2017; Lopez et al., 2018) transforms the OSM map into a CommonRoad scene (`.cr.xml`) and a SUMO road network (`.net.xml`).
- **Microscopic SUMO simulation** populates the network with background traffic according to the vehicle mix and density parsed by the LLM; the resulting trajectories satisfy car-following and lane-changing dynamics by construction.

Hallucination isolation. Because the LLM never emits map geometry, road-network topology, or vehicle trajectories directly, three classes of hallucination that would otherwise cause silent downstream failure are structurally ruled out:

- invalid lanelet IDs or topologies (only the converter produces these);
- kinematically infeasible trajectories (only SUMO produces these);
- references to non-existent map fragments (only the OSM fetch produces these).

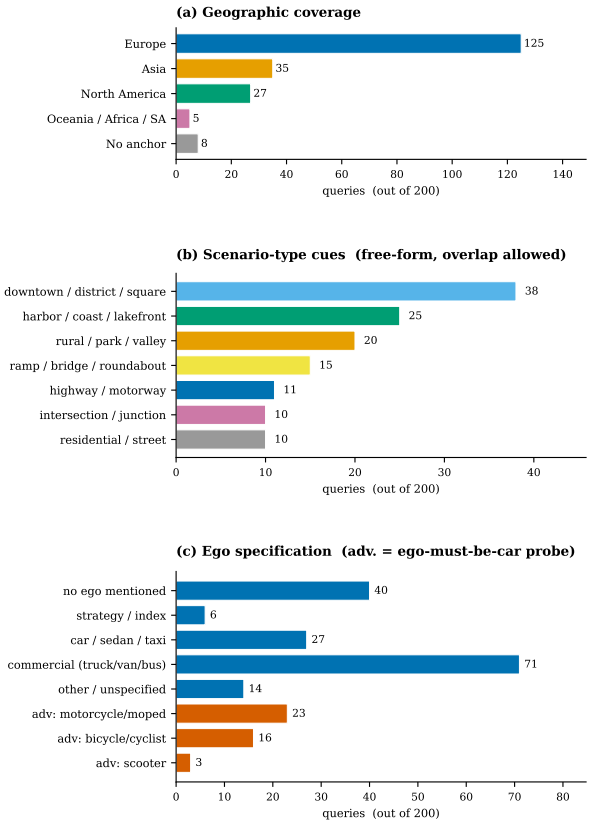
The remaining LLM-side failure modes (misparsed location, wrong traffic-density level, missing ego specifier) are caught at schema-validation time and either default-applied or surfaced as an error. Per-key pass rates appear in the *loadable* column of Table 7.

A.2.2 Query Corpus

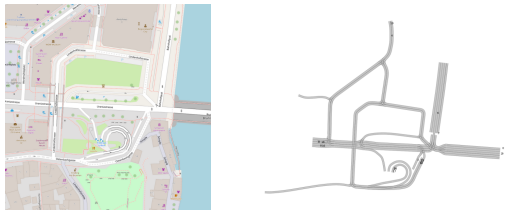
The Generation Module benchmark uses $N = 200$ natural-language queries with a median length of 8 words (p10 = 5, p90 = 11, max = 23). Frequently requested cities include Shanghai, Madrid, Cologne, Seoul, and London. The corpus splits into two reporting buckets:

- **CLEAN** ($n = 158$) – standard car-ego queries. These exercises the canonical pipeline and dominate the headline numbers.
- **ADVERSARIAL** ($n = 42$) – queries that explicitly request a non-car ego, split as motorcycle/moped (23), bicycle/cargo bike/cyclist (16), and scooter/e-scooter (3). This bucket probes the system rule that the Frenetix-planned ego *must* be a car: the CP rules teach the LLM to keep the requested vehicle type in the surrounding traffic while downgrading the ego itself to the first car.

A separate cross-cutting count: 55 queries mention a traffic-density adjective (system-fixed default: `low`) and 21 request an explicit duration



(d) Q013: Rome roundabout, run for 60 seconds with moderate traffic, ego is the emergency vehicle.



(e) Q004: Sparse rural road outside Zurich; include cars and motorcycles, and the motorcycle is the ego.

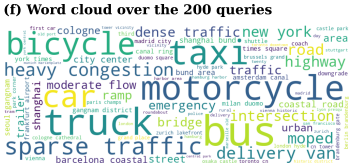


Figure 6: Composition of the 200-query Scenario Generation corpus. (a) Geographic coverage. (b) Scenario-type cues (free-form, overlapping). (c) Ego specification, with non-car ego queries in vermillion. (d, e) Two example queries rendered through the pipeline: Q013 (clean commercial ego) and Q004 (adversarial non-car ego). (f) Word cloud over the queries.

(system-fixed default: `20 s`), probing the defaults-compliance metric defined in §A.2.3.

Figure 6 summarises the geographic coverage, scenario-type cues, and ego specification. It also includes a word cloud of the queries.

A.2.3 Evaluation Metrics

We evaluate the generation chain (*query* → *JSON* → *Scenario* → *Planner*) across three buckets: **ALL** (n=200), **CLEAN** (n=158), and **ADVERSARIAL** (n=42; requiring ego-must-be-car rule). Table 7 tracks nine metrics clustered into three groups:

- **Cost:** Mean *Tokens* (prompt+completion+reasoning) and *Latency* (s) per query.
- **Pipeline:** Success rate at each stage: *JSON valid %* (schema-compliant output); *Defaults compliance* (adherence to fixed settings like `sim.duration_s=20`); *Load %* (CommonRoad scene parses successfully); *Traffic %* (loadable with SUMO ≥ 1 background actor); *Runnable %* (Frenetix completes a trajectory, even if colliding); and *Planner %* (Frenetix reaches the goal safely).
- **Overall:** The composite *Intent overall score* aggregating LLM and pipeline metrics, serving as the primary metric to rank (model, condition) pairs.

A.2.4 Evaluation Results

Each cell aggregates 200 generation runs end-to-end (geocode → OSM fetch → CR convert → SUMO sim → Frenetix). We surface the full results matrix on the **ALL** bucket in Table 7.

Ablation discussion. The prompt-condition ladder reveals that intent parsing is gated almost entirely by the schema injection (`cp`): the *defaults-compliance* column jumps from ≈0.50 to 1.00 for nine out of ten models the moment the JSON schema and the defaults are spelled out. Once the schema is in the prompt, adding CoT or ICL yields only marginal gains, and the four ALL-bucket overall scores converge to a tight 0.95–0.96 band across model families. The implication is that for structured-output tasks with a narrow grammar, contextual prompting alone is sufficient. The extra latency cost of CoT scaffolding (1.5–2× tokens per query) buys negligible additional reliability here, which is why small open-source models such as Qwen3.6:35b and Gemma4:31b match commercial APIs on this task.

A.3 Scenario Selection Module

A.3.1 Query Corpus

The Selection Module benchmark uses 200 natural-language scenario-selection queries (median length: 16 words), each targeting a unique source scenario from the CommonRoad database. The structured GT targets are notably narrow, reflecting realistic, specific user requests: 29.5% resolve to exactly one matching scenario, and 32.5% resolve to a tight set of 4–10 alternatives.

Each fixture's GT JSON defines 5 extractor slots (as detailed in Table 8), corresponding to the fields the LLM pipeline attempts to extract. For each query, we count how many of these 5 slots are populated:

- **5/5 fields (14 queries):** The user specifies all five aspects (e.g., "On a 2-lane road in Munich at ∼25 km/h, ego approaches a pedestrian crossing with 2 vehicles ahead").
- **4/5 fields (111 queries):** Most common; one slot is omitted (typically obstacles or velocity).
- **3/5 fields (74 queries):** Three slots specified (e.g., location, tags, and road network).
- **2/5 fields (1 query):** An outlier with only two slots populated.

Per-slot field coverage is reported in Table 8. Figure 7 visualizes the corpus along three axes: country distribution, GT `scenarioTags` frequency, and a word cloud over the queries.

Retrieval methods. Given the structured extraction (the five GT slots in Table 8) and the original natural-language (Natural Language (NL)) query, we evaluate four retrieval strategies that combine these signals against the ChromaDB scenario index in increasingly hybrid ways:

- **Funnel pipeline:** A strict 5-stage *AND* filter over the extracted GT slots (`location` → `tags` → `road_net` → `obstacles` → `velocity`). It applies predicates as a left-to-right intersection over the scenario index, where each stage's output feeds the next, and empty intermediate results abort the pipeline. Velocity uses a tolerant range comparison (single-element bounds widen to ±2 m/s);

Table 7: Performance of the Generation Module on the ALL bucket (N=200 per cell).

Model	Prompt	Cost		Pipeline					Overall ↑
		tokens ↓	latency (s) ↓	JSON % ↑	defaults ↑	load % ↑	traffic % ↑	runnable % ↑	
Cloud API									
Qwen3.6-plus	baseline	749	25.8	100.0	0.497	90.5	90.5	90.5	0.780
	cp	1970	27.0	100.0	1.000	96.0	96.0	96.0	0.955
	cp_cot	2827	33.7	100.0	1.000	96.5	96.5	96.5	0.956
	cp_icl	3750	27.8	100.0	1.000	96.5	96.5	96.5	0.956
	cp_icl_cot	4480	32.1	100.0	1.000	96.5	96.5	96.5	0.956
Deepseek-v3.2	baseline	724	29.1	100.0	0.497	94.0	94.0	94.0	0.784
	cp	1912	28.3	100.0	1.000	96.0	96.0	96.0	0.955
	cp_cot	2684	32.9	100.0	1.000	96.5	96.5	96.5	0.956
	cp_icl	3556	28.8	100.0	1.000	96.5	96.5	96.5	0.956
	cp_icl_cot	4318	34.3	100.0	1.000	96.5	96.5	96.5	0.956
Glm-5	baseline	671	28.6	100.0	0.500	92.0	91.5	92.0	0.783
	cp	1850	30.0	100.0	1.000	95.0	95.0	95.0	0.954
	cp_cot	2601	36.5	100.0	1.000	97.0	97.0	97.0	**0.957**
	cp_icl	3436	36.6	100.0	1.000	96.0	96.0	95.5	0.955
	cp_icl_cot	4204	41.1	100.0	1.000	96.5	96.5	96.5	0.956
Gemini-3-flash	baseline	763	29.7	100.0	0.497	95.0	95.0	95.0	0.787
	cp	2026	26.7	100.0	1.000	95.5	95.5	95.5	0.954
	cp_cot	2822	25.1	100.0	1.000	97.0	97.0	97.0	**0.957**
	cp_icl	3890	23.7	100.0	1.000	96.5	96.5	96.5	0.956
	cp_icl_cot	4675	24.6	100.0	1.000	96.5	96.5	96.5	0.956
Gpt-5.4-mini	baseline	609	39.9	100.0	0.495	75.5	75.5	75.5	0.763
	cp	1780	34.2	99.5	0.892	94.0	94.0	94.0	0.912
	cp_cot	2517	40.5	100.0	0.995	95.0	95.0	95.0	0.950
	cp_icl	3382	31.0	100.0	1.000	89.5	89.5	89.5	0.937
	cp_icl_cot	4122	34.5	100.0	1.000	95.0	95.0	95.0	0.952
Local Ollama									
Qwen3.6:35b (Think)	baseline	3322	53.4	100.0	0.497	49.5	49.5	49.5	0.735
	cp	3352	46.5	97.0	0.995	94.0	94.0	94.0	0.927
	cp_cot	4159	52.0	94.0	1.000	90.5	90.5	90.5	0.899
	cp_icl	4528	36.8	99.5	1.000	96.0	96.0	96.0	0.951
	cp_icl_cot	5469	43.3	99.0	1.000	95.5	95.5	95.5	0.947
Qwen3.6:35b	baseline	753	20.1	100.0	0.495	44.0	44.0	44.0	0.728
	cp	1972	25.5	100.0	1.000	96.5	96.5	95.5	0.956
	cp_cot	2778	29.0	100.0	1.000	96.5	96.5	96.5	0.956
	cp_icl	3741	26.8	100.0	1.000	96.0	96.0	96.0	0.956
	cp_icl_cot	4638	33.0	100.0	1.000	96.0	96.0	96.0	0.955
Gemma4:31b (Think)	baseline	1585	40.3	100.0	0.497	93.0	93.0	93.0	0.784
	cp	2813	38.9	100.0	1.000	94.5	94.5	94.5	0.950
	cp_cot	3572	43.1	100.0	1.000	96.0	96.0	96.0	0.955
	cp_icl	4408	35.7	100.0	1.000	95.0	95.0	95.0	0.953
	cp_icl_cot	5004	37.2	100.0	1.000	96.5	96.5	96.5	0.956
Gemma4:31b	baseline	785	28.8	100.0	0.497	94.0	94.0	94.0	0.785
	cp	2044	27.8	100.0	1.000	92.0	92.0	92.0	0.946
	cp_cot	2790	31.2	100.0	1.000	96.0	96.0	96.0	0.955
	cp_icl	3914	28.0	100.0	1.000	91.0	91.0	91.0	0.942
	cp_icl_cot	4649	31.7	100.0	1.000	96.5	96.5	96.5	0.956
Gpt-oss:20b (Think)	baseline	1894	33.6	100.0	0.485	86.0	86.0	86.0	0.771
	cp	2545	32.5	100.0	1.000	96.5	96.5	96.5	0.956
	cp_cot	3472	32.7	100.0	1.000	96.0	96.0	96.0	0.955
	cp_icl	4016	31.4	100.0	1.000	96.5	96.5	96.0	0.956
	cp_icl_cot	4816	31.1	100.0	1.000	96.5	96.5	96.5	0.956

Note: Metrics are split into Cost, Pipeline, and a standalone Overall summary column. Arrows mark preferred directions (↑/↓). In the Overall column, **bold** marks the best score across the table and underline marks the second-best. Models are grouped by provider and ordered within each provider by their peak `cp_icl_cot` overall score. Overall and default values are bounded in [0, 1]; tokens and latency (s) are means per query.

Table 8: The 5 extractor slots in the Scenario Selection GT. "Pop." is the non-empty count across $N = 200$ fixtures.

Slot	Description (e.g.)	Pop.
`location`(req)	Country/city ({"DEU"})	200
`tags`(req)	Scenario tags (["urban"])	200
`road_net`(opt)	Topology ({lanes: 2})	68
`obstacles`(opt)	Actors ({car: [1, 2]})	190
`velocity`(opt)	Ego speed ([10, 20])	80

tags use case-insensitive matching against the controlled taxonomy; and locations fall back to country-only matching if the city specifier yields zero hits. This approach offers high precision but is brittle to noisy extraction.

- **Semantic search:** Computes ChromaDB cosine similarity over SentenceTransformer embeddings of the NL query. It is language-tolerant but provides no structural guarantees.
- **Hybrid pre-filtering:** Semantic retrieval restricted to candidates whose metadata matches the extracted `country_code`, ensuring grounded recall and country-safe results.
- **Reciprocal-rank fusion (RRF):** Fuses the top-k results from both the `funnel` and `semantic` pipelines to balance strict metadata matching with semantic similarity.

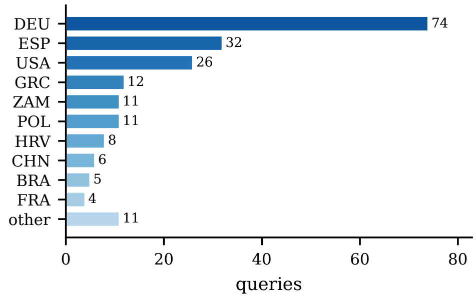
A.3.2 Evaluation Metrics

We evaluate the selection chain (*query* $\rightarrow$ *structured extraction* $\rightarrow$ *retrieval* $\rightarrow$ *top-k scenarios*) on the N=200 fixtures with cached `gt_valid_ids` (the programmatic answer key derived from the GT structured conditions). Reported metrics fall into three groups: **Cost**, **Retrieval quality**, and **Extraction quality**.

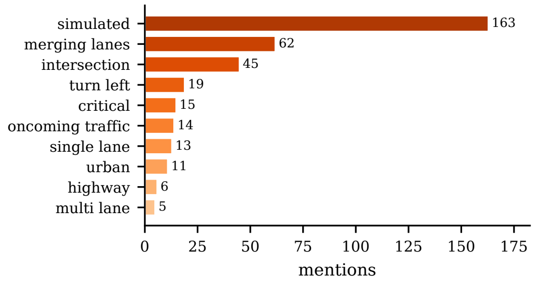
Cost. Mean *Tokens* per query (prompt+completion+reasoning) and *Latency* (s).

Retrieval quality. The three retrieval-quality metrics share a common test ("*does the returned scenario satisfy the GT structured conditions?*") but answer different operational questions:

- `sat_any@5` $\rightarrow$ production-faithful (top-k UI, one valid match = success).
- `sat_top1` $\rightarrow$ strictest user-visible (single-result UIs).
- **`sat_all@returned`** $\rightarrow$ precision-strict headline (deployed scoring contract; ranks variants in Table 9).



(a) Country distribution (`DEU` dominates).



(b) Top-10 GT `scenarioTags`.

(c) Word cloud (stopwords/function words removed).

Figure 7: Composition of the 200-query Scenario Selection Module. Per-slot field coverage is reported separately in Table 8.

By default, these are reported on the `funnel` pipeline. Both `sat_any@5` and `sat_all@returned` are additionally broken down across the four retrieval pipelines (`funnel`, `semantic`, `hybrid_prefilter`, `rrf`) in Figure 8 to isolate structured filtering from semantic search.

Extraction quality. *Extract* is the mean of five per-slot extractor scores in $[0, 1]$, computed by comparing the LLM's structured output to the GT slot-by-slot. It is *independent of retrieval* and isolates pure LLM extraction quality. The per-slot scoring functions are:

- `location` $\rightarrow$ exact `country_code` + fuzzy `specifier`.
- `tags` $\rightarrow$ Jaccard set overlap.

- road_net → field-wise topology agreement.
- obstacles → range/count overlap.
- velocity → numeric range overlap (±2 m/s for single-element bounds).

Diagnosing failures with *Extract* vs. *sat_all*. The two columns decouple two distinct failure modes (see Table 9): high *Extract* with low *sat_all* indicates correct extraction followed by a low-precision retrieval stage that admits spurious items into the returned list (a candidate for retrieval-side refinement). Low *Extract* with high *sat_all* indicates a noisy LLM whose extraction errors are masked by a permissive downstream filter. We therefore report both columns. We rank **(model, condition)** pairs by funnel *sat_all@returned*, the precision-strict headline metric: every scenario the system returns must satisfy the user's structured request.

A.3.3 Evaluation Results

Experimental design. We evaluate a grid of ten model variants, five prompt conditions, and four retrieval methods over the same N=200 fixtures, with gt_valid_ids cached as the programmatic answer key. Since the four retrieval methods consume the same LLM-extracted slots per (model, condition) call, the experiment requires 10,000 LLM extractions but yields 40,000 (cell, fixture) retrieval outcomes. The two reporting artifacts surface complementary slices: Table 9 gives every (model, condition) pair on the production funnel pipeline (50 cells), while Figure 8 reports sat_any@5 across all four pipelines at each model's best prompt condition (40 cells). Each cell aggregates 200 end-to-end runs: NL query → slot extraction → retrieval → top-k scoring against the cached answer key.

Reading Table 9. The metrics are split into two clusters: **Cost** (tokens, latency) and **Retrieval (funnel)**, where the latter also embeds the LLM-only *Extract* column so that the LLM-vs.-retrieval failure-mode comparison is visible at a glance (cf. §A.3.2). The shaded column *sat_all* (sat_all@returned) is the precision-strict headline we use to rank variants: every returned scenario must satisfy the GT structured conditions. Models are grouped by provider and listed in the same order as Table 6. Within each model family, the (Think) (reasoning-on) variant precedes the default. The cross-pipeline comparison across all four retrieval methods is shown in Figure 8.

Slot-level extract. Table 9 reports a single *extract* mean that hides which of the five slots fails. Table 10 decomposes that mean. The bottleneck is tags (e.g. urban): it is the lowest slot in all five prompt conditions. Once the schema is injected it sits at 0.64–0.67, while the other four slots range 0.75–0.97. The cause is over-prediction, not misses (recall 99.7%): the model adds plausible unsupported descriptors such as traffic_jam (invented 139 times although no scenario in the database carries it). CP alone recovers most of the slot (0.205→0.665); no further technique moves it. Under ICL without CoT, velocity drops 0.957→0.745 and road_net drops 0.937→0.864, likely from copying exemplar values; adding CoT restores both. On Qwen3.6-plus, cp_cot has the lower *extract* (0.903) yet the paper's best sat_all (88.0), while cp_icl_cot raises *extract* to 0.927 and lowers sat_all to 83.5, because the funnel is a strict AND over slots.

Cross-pipeline analysis. Figure 8 reports both sat_any@5 (recall) and sat_all@returned (precision) across all four retrieval pipelines at each model's best prompt. Under recall (panel a), the ranking is essentially universal (rrf ≈ funnel ≫ hybrid_prefilter ≫ semantic), with funnel at 70.0–96.5% and hybrid_prefilter flat near 75%. Any LLM-consulting pipeline can usually surface one valid match in five. Under precision (panel b), only the strict-AND funnel survives at 63.5–88.0%. The semantic-based pipelines collapse: semantic returns 0% (no structured constraints), hybrid_prefilter reaches only 5.5–6.0% (country pre-filter too weak), and rrf stays below 1.0% (semantic candidates dilute the funnel list). The Selection Module ships the precision view, so funnel is the production pipeline. Rrf's parity with funnel under recall is an artefact of the top-5 hit-rate metric.

A.4 Scenario Modification Module

A.4.1 Query Corpus

The Modification Module benchmark uses N = 200 natural-language queries per task across four task types: **T** (trajectory redirection), **B** (behavior preset), **P** (population edit), and **G** (goal extraction), for a total of 800 queries. Task P is a synthetic union of the population-insertion (P_{add}) and population-deletion (P_{remove}) sub-tasks: 100 queries from each side, yielding N=200 for P and matching the per-cell sample size used elsewhere.

Table 9: Performance of the Selection Module on the `funnel` pipeline (N=200 per cell). *sat_all %* denotes the joint satisfaction rate across all five GT slots (location → tags → road network → obstacles → velocity) for the returned top-k scenario list; we report it as the precision-strict headline metric.

Model	Prompt	Cost		Fail% ↓	Retrieval (funnel)			Overall ↑
		tokens ↓	latency (s) ↓		sat_any@5 % ↑	sat_top1 % ↑	extract ↑	**sat_all % ↑**
Cloud API								
Qwen3.6-plus	baseline	900	6.5	72.5	27.5	22.5	0.748	18.0
	cp	4845	7.1	26.5	73.5	64.0	0.870	57.0
	cp_cot	7029	21.9	3.5	96.5	93.0	0.903	88.0
	cp_icl	6621	6.4	27.5	72.5	63.5	0.858	56.0
	cp_icl_cot	8707	21.6	3.5	96.5	90.5	0.927	83.5
Deepseek-v3.2	baseline	827	17.5	51.0	49.0	34.5	0.694	26.0
	cp	4615	18.2	41.0	59.0	53.5	0.831	48.0
	cp_cot	6602	32.8	19.5	80.5	80.0	0.902	76.0
	cp_icl	6242	16.6	25.0	75.0	66.0	0.838	56.0
	cp_icl_cot	8158	36.4	15.0	85.0	80.5	0.903	74.5
Glm-5	baseline	812	14.7	85.0	15.0	12.5	0.674	9.0
	cp	4559	13.3	29.5	70.5	62.5	0.840	56.0
	cp_cot	6482	29.1	13.0	87.0	83.0	0.865	77.5
	cp_icl	6209	16.3	47.5	52.5	51.0	0.758	47.0
	cp_icl_cot	7974	29.1	9.5	90.5	88.5	0.886	83.5
Gemini-3-flash	baseline	861	5.7	76.5	23.5	18.0	0.685	14.0
	cp	4835	6.4	32.5	67.5	58.0	0.824	51.0
	cp_cot	6821	8.1	32.5	67.5	61.0	0.861	56.0
	cp_icl	6630	5.2	35.5	64.5	58.0	0.813	53.5
	cp_icl_cot	8548	7.9	30.0	70.0	67.0	0.858	63.5
Gpt-5.4-mini	baseline	834	5.8	76.5	23.5	22.5	0.612	21.5
	cp	4569	5.6	49.5	50.5	48.0	0.802	46.0
	cp_cot	6139	8.7	25.0	75.0	73.0	0.892	72.0
	cp_icl	6185	6.1	51.5	48.5	43.5	0.789	38.0
	cp_icl_cot	7772	8.1	21.0	79.0	76.5	0.889	73.0
Local Ollama								
Qwen3.6:35b (Think)	baseline	5642	34.5	59.0	41.0	35.0	0.666	26.5
	cp	9703	36.2	26.5	73.5	68.0	0.857	65.0
	cp_cot	13356	82.2	37.0	63.0	52.5	0.777	40.0
	cp_icl	11843	65.6	27.0	73.0	66.0	0.826	60.0
	cp_icl_cot	16857	77.7	30.5	69.5	50.0	0.763	33.5
Qwen3.6:35b	baseline	913	1.3	47.5	52.5	48.5	0.679	46.5
	cp	4857	2.2	33.5	66.5	64.0	0.792	63.5
	cp_cot	6949	7.5	9.0	91.0	87.5	0.884	83.0
	cp_icl	6822	3.9	44.0	56.0	54.5	0.779	54.0
	cp_icl_cot	8574	6.8	8.5	91.5	88.5	0.895	84.0
Gemma4:31b (Think)	baseline	2915	59.6	77.0	23.0	16.0	0.645	11.0
	cp	6487	29.1	27.0	73.0	70.5	0.830	67.0
	cp_cot	8318	36.8	60.5	39.5	34.0	0.884	30.0
	cp_icl	8275	29.6	27.5	72.5	72.0	0.829	69.0
	cp_icl_cot	10055	36.5	62.0	38.0	35.5	0.883	33.5
Gemma4:31b	baseline	958	2.7	97.0	3.0	2.0	0.537	1.5
	cp	4928	4.2	22.5	77.5	70.5	0.809	61.0
	cp_cot	6763	12.3	20.0	80.0	72.0	0.877	65.0
	cp_icl	6728	4.0	28.5	71.5	70.5	0.830	68.0
	cp_icl_cot	8558	12.1	27.5	72.5	71.5	0.880	70.0
Gpt-oss:20b (Think)	baseline	3121	16.3	91.0	9.0	8.0	0.703	7.0
	cp	5891	7.1	27.5	72.5	64.5	0.852	58.0
	cp_cot	7234	9.5	49.0	51.0	37.5	0.790	32.0
	cp_icl	8123	9.6	25.5	74.5	72.0	0.862	68.0
	cp_icl_cot	10044	14.4	48.5	51.5	46.5	0.857	39.5

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query.

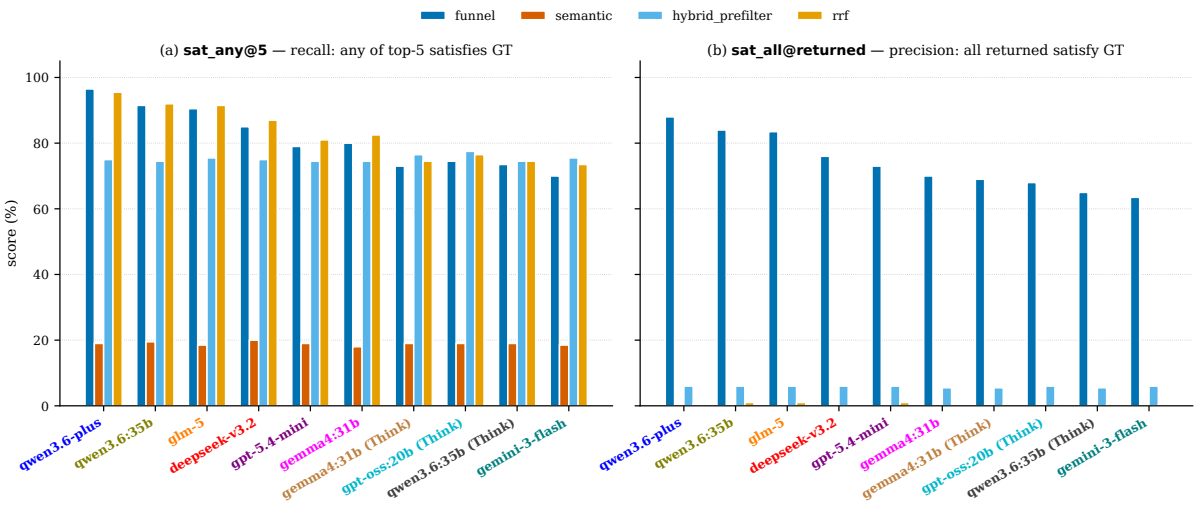


Figure 8: Cross-pipeline retrieval comparison under two scoring contracts: **(a)** recall-oriented `sat_any@5`; **(b)** precision-strict `sat_all@returned`. Each model shown at its best funnel condition; x-axis order shared across panels; colours match Table 9.

Table 10: Slot-level *extract* score by prompt condition (mean over the ten models, N=200). The bottom row equals the *extract* column of Table 9 averaged across models.

Extractor slot	baseline	cp	cp_cot	cp_icl	cp_icl_cot
location	0.881	0.914	0.857	0.911	0.879
tags	**0.205**	**0.665**	**0.648**	**0.642**	**0.658**
road_net	0.825	0.878	0.937	0.864	0.951
obstacles	0.745	0.947	0.918	0.928	0.910
velocity	0.665	0.749	0.957	0.745	0.974
Mean = *extract*	0.664	0.831	0.864	0.818	0.874

The 800 queries are anchored in 345 unique CommonRoad scenarios spanning 15 country codes (DEU dominates: 80–92 queries per task). The three scenario counts in the paper denote distinct sets: the Selection Module's persistent ChromaDB store indexes 500+ curated scenarios (Appendix §A.1). The Modification corpus draws from a smaller 245-scenario SUMO-cached pool (with the 5/245 round-trip exclusion below, leaving 240 clean seeds), and the 345 unique anchor scenarios above are the union across all five Modification tasks, larger than the 240 simulable seeds because Task G only edits the planning problem and can use scenarios outside the SUMO-cached pool. Five of 245 cached CommonRoad scenarios (2.0%) were excluded from the source pool because their *unmodified* form fails to round-trip through the CR↔SUMO interface: four due to a missing `ObstacleType.MOTORCYCLE` mapping in the simulator-to-CommonRoad converter, and one for an unrelated SUMO route-validity issue. These are infrastructure limits of the bridge, not failures of LLM-generated modifications. The LLM never touches them in the sweep. Replacement queries are drawn from the same clean 240-scenario pool to keep the per-task count at exactly N=200.

The GT schema is task-specific because each modification target requires different anchor fields: **T** carries the source vehicle, source edge, target edge, and BFS depth. **B** carries the target-vehicle list and the expected behavior preset. **P** carries the ±1 vehicle-count delta together with the target edge/vehicle and vehicle-type fields. **G** carries the target edge and the position keyword (start/quarter/middle/three-quarter/end). The canonical GT shapes feed the per-task scorers (§A.4.2). The principal corpus-stratification dimensions per task are visualized in Figure 9.

To illustrate the diversity of query phrasings across tasks, one representative GT row per task is shown below (the first query from each task's query set. For P, we list both sub-task examples since the synthetic task interleaves them:

- **T** (*POL_Krakow-22_1_T-4*): "Redirect vehicle 14 to edge 595." GT: `{vid:14, source_edge:332, target_edge:595, bfs_depth:1}`
- **B** (*GRC_NeaSmyrni-102_1_T-6*): "Set vehicles 20026 and 20019 to urgent driving." GT: `{target_vids:[20026,20019], expected_preset:emergencyType, descriptor_kind:multi}`

- **P (add)** (*DEU_Hanover-44_29_T-1*): "Please add a new truck, starting on edge 874, to the simulation." GT: `{expected_delta:+1, target_edge:874, target_vtype:truck}`
- **P (remove)** (*DEU_Bremen-5_5_T-1*): "Could you please remove vehicle with ID 30243 from the simulation?" GT: `{expected_delta:-1, target_vid:30243, target_vClass:passenger}`
- **G** (*RUS_Bicycle-3_2_T-1*): "Set ego goal to 3/4 of lanelet 7." GT: `{target_edge:7, position:three_quarter}`

Figure 9 visualizes the corpus along three axes per task: principal stratification (row 1), query-length distribution (row 2), and a word cloud over the 200 NL queries per task (row 3). The behavior-preset distribution (panel b) and the goal-position distribution (panel d) are intentionally near-uniform to ensure the prompt techniques cover every preset and every position keyword. Task T concentrates on short-hop reroutes (1–2 BFS hops cover 86.5% of queries), reflecting the typical local-edit semantics of trajectory modification requests. Task P's combined panel (c) makes the asymmetry between the add and remove sub-tasks visible: the add side is balanced across car/truck/bus while the remove side is dominated by passenger targets (the first 100 P_{remove} queries are all passenger), because passenger vehicles are by far the most common non-ego actors in CommonRoad scenarios. The word-cloud row (i–l) highlights the verbs and adjectives that characterize each task: "redirect/route/target" for T, "preset/aggressive/comfort" for B, "add/remove/truck/passenger" for P, "goal/lanelet/quarter/middle" for G.

A.4.2 Evaluation Metrics

We evaluate each (model, condition) pair on the N=200 queries per task with a two-stage scoring funnel: per-task **semantic correctness** (does the modified scenario reflect the requested change?), followed by **downstream simulability** (does the modified scenario still produce a valid SUMO trace that round-trips back into CommonRoad?). All checks are boolean. The headline metric *overall* is the unweighted mean of these boolean checks. Per-query and per-cell telemetry (latency and tokens) is auto-recorded alongside.

Shared funnel stages (every task). The Cost columns (mean tokens, mean latency) and the syntactic/simulability checks below apply uniformly to T, B, P, and G. Task G omits SUMO % and CR % because it only edits the planning problem, so there is no traffic round-trip.

- *tokens*: Mean total tokens per query (Cost).
- *latency (s)*: Mean LLM wall-clock per query (Cost).
- *XML % / JSON %*: Output parses as well-formed SUMO route XML (or JSON for G).
- *SUMO %*: Modified route file simulates end-to-end without runtime errors or stuck vehicles.
- *CR %*: SUMO output round-trips back into a valid CommonRoad scenario via the CR→SUMO bridge.

The optional Frenetix-runnability gate is disabled by default in this sweep (the cross-planner Frenetix evaluation is instead reported as the separate batch comparison summarised in Figure 5), so the column does not appear in Tables 11–13.

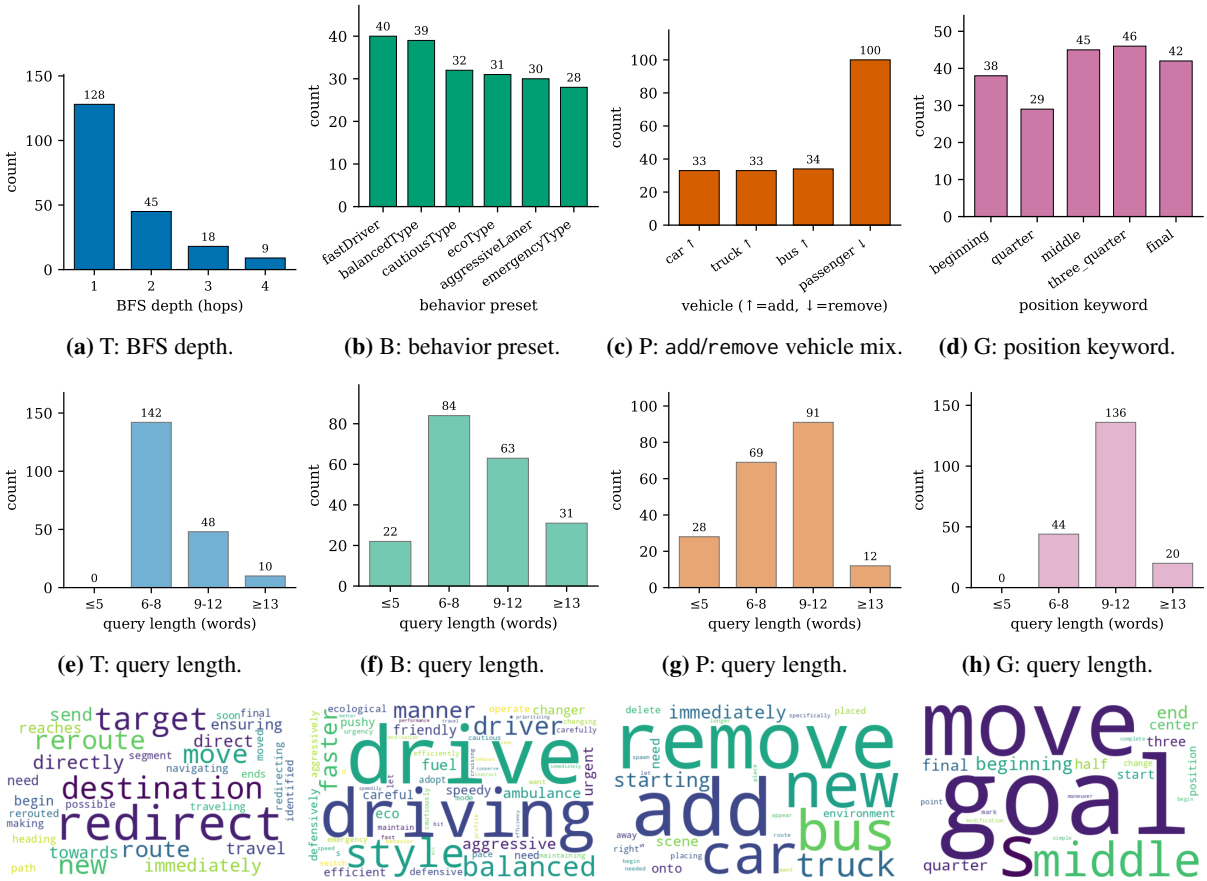
Per-task semantic checks. Each task carries its own semantic checks against the per-task GT schema described above. For each task, we identify the strictest semantic gate as the **HEADLINE** metric (the check whose failure most directly indicates that the LLM has not performed the requested edit). The headline column is rendered in red in the corresponding result table.

Task T — Trajectory Redirection (Table 11).

- *tgt %*: The named vehicle survives the edit.
- *ends %* — **HEADLINE**: The redirected route terminates at the requested target edge.
- *preserve %*: Non-target vehicles' routes remain byte-equivalent to baseline.

Task B — Behaviour Preset (Table 12).

- *tgt %*: The named vehicle survives the edit.
- *preset %* — **HEADLINE**: Modified behaviour-parameter vector matches the requested preset within relative tolerance 10^{-3}.
- *vClass %*: Vehicle class is preserved (e.g. a bus stays a bus).



(a) T: BFS depth. **(b)** B: behavior preset. **(c)** P: add/remove vehicle mix. **(d)** G: position keyword.

(e) T: query length. **(f)** B: query length. **(g)** P: query length. **(h)** G: query length.

(i) T: word cloud. **(j)** B: word cloud. **(k)** P: word cloud. **(l)** G: word cloud.

Figure 9: Per-task corpus composition for the Scenario Modification benchmark (N=200 queries each). **Row 1 (a–d):** principal stratification axis per task. **Row 2 (e–h):** query-length distribution. **Row 3 (i–l):** per-task word cloud.

Task P — Population Edit ($P_{\text{add}} \cup P_{\text{remove}}$; Table 13). The add and remove sub-task scorers share an indistinguishable arithmetic semantics (both grade the model on producing the requested ± 1 delta while leaving the rest of the route file unchanged) so the combined P overall is the unweighted mean of the boolean checks across all 200 queries.

- *count_match %* — **HEADLINE**: Vehicle count changes by exactly $+1$ (add) or -1 (remove). Without this, the requested edit has not happened.
- *tgt-chg %*: New vehicle starts on the requested edge (add) or the named vehicle is absent from the modified file (remove).
- *others %*: Non-target route entries remain byte-equivalent to baseline.

Task G — Goal Extraction (Table 14).

- *edge %*: A target edge ID is emitted.
- *pos-enum %*: Position keyword belongs to the allowed enumeration.
- *edge-GT %* — **HEADLINE**: Extracted target lanelet matches the GT — the structural goal decision.
- *pos-GT %*: Extracted position keyword matches the GT.
- *lanelet %*: Lanelet ID resolves in the CommonRoad scenario.

Two further G checks (the planning-problem-accept check and the shared CR-reload check) are computed but not displayed in Table 14 because they are perfectly correlated with *lanelet %* in this sweep. Both still contribute to *Overall*.

Auto-recorded telemetry. For each LLM invocation, we record prompt, completion, and (where the provider exposes it) reasoning token counts, the total token count, the per-invocation API latency, and the full per-query wall-clock (which also includes parsing and simulation overhead). These columns appear unaltered in the per-task KPI summaries and let us decouple "the LLM did its job correctly" (semantic checks) from "the LLM was fast enough to be deployable" (latency/token cost).

Diagnosing failures with the funnel. The two stages decouple distinct failure modes: high semantic correctness with low SUMO simulability indicates a semantically correct edit that produces a structurally fragile route (e.g. a target edge that is connected but disallows the vehicle's class), while uniformly low semantic checks indicate a model that either fails to identify the right anchor field or emits malformed XML/JSON. The two paths, therefore, call for different fixes (network-aware prompting vs. stricter output-format prompting), which the per-task per-check tables in §A.4.3 surface independently.

A.4.3 Evaluation Results

Qualitative examples. Before reporting quantitative outcomes, Figure 10 shows one representative successful modification per task, all from `gemini-3-flash-preview` under the `cp_icl_cot` condition. Each row compares the baseline scenario (left) and the LLM-modified scenario (right) rendered at the same mid-trajectory timestep so the modification's effect on the dynamic traffic is directly visible. The four examples surface the visible footprint of each task: in T, the redirected vehicle has joined a different downstream lanelet by t=32. In B, vehicles 30293 and 30296 have already pulled ahead under the faster behavior preset by t=122. The P (remove) sub-task shows that vehicle 30242 is removed. G renders the ego's modified goal region on the requested lanelet at t=15 (G's modification only edits the planning problem, so the surrounding traffic is unchanged by design).

Quantitative results. Tables 11, 12, 13, and 14 report the per-task funnel KPIs for all ten models across the five prompt conditions. Each table covers N=200 queries per (model, condition) cell (Task P combines the first-100 of P_{add} with the first-100 of P_{remove}). Cells marked "N/A" correspond to open-source models still in the data-collection phase as of submission. The table-generation pipeline will replace them with values for the camera-ready version.

Application: safety-criticality. As a downstream application of the B-task modification, we verify that it can systematically shift the *safety criticality* of a scenario. On N=100 paired (baseline, modified) scenarios run through Frenetix under identical default cost weights, where the modification (`gemini-3-flash-preview`/`cp_icl_cot`) sets the three NPCs closest along ego's executed

Table 11: Task T (trajectory redirection) per-cell funnel KPIs (N=200 queries per cell). The rightmost column *ends %* is the headline semantic check (the LLM-edited route terminates on the requested target edge); *preserve %* verifies non-target vehicles' routes stay byte-equivalent to baseline.

Model	Prompt	Cost		Fail% ↓	Pipeline stage pass rates					
		tokens ↓	latency (s) ↓		XML % ↑	tgt % ↑	preserve % ↑	SUMO % ↑	CR % ↑	**ends % ↑**
Cloud API										
Qwen3.6-plus	baseline	18324	52.6	6.5	99.5	99.5	99.0	93.5	93.5	99.0
	cp	19336	55.2	3.5	99.5	99.5	99.0	96.5	96.5	98.5
	cp_cot	21011	61.9	2.0	100.0	100.0	100.0	98.0	98.0	99.5
	cp_icl	21611	50.5	4.0	100.0	100.0	100.0	96.0	96.0	98.5
	cp_icl_cot	24336	61.1	2.5	100.0	100.0	100.0	97.5	97.5	98.0
Deepseek-v3.2	baseline	14722	69.5	12.5	99.5	99.5	99.0	87.5	87.5	99.0
	cp	15274	67.5	4.5	99.5	99.5	99.5	95.5	95.5	94.0
	cp_cot	16962	83.8	0.5	99.5	99.5	99.0	99.5	99.5	97.0
	cp_icl	17067	58.5	5.0	100.0	100.0	99.5	95.0	95.0	99.0
	cp_icl_cot	18350	68.7	0.5	100.0	100.0	100.0	99.5	99.5	99.0
Glm-5	baseline	14223	36.3	23.5	99.5	99.5	99.5	76.5	76.5	98.0
	cp	15559	43.8	3.0	99.5	99.5	99.0	97.0	97.0	97.0
	cp_cot	16568	48.8	2.5	99.5	99.5	99.0	97.5	97.5	99.5
	cp_icl	17844	42.4	1.5	99.5	99.5	99.0	98.5	98.5	98.0
	cp_icl_cot	19004	51.5	2.0	99.5	99.5	99.5	98.0	98.0	99.5
Gemini-3-flash	baseline	18658	10.0	2.5	99.5	99.5	99.5	97.5	97.5	99.0
	cp	19757	10.4	1.5	99.5	99.5	99.5	98.5	98.5	99.5
	cp_cot	20831	12.4	2.5	98.5	98.5	98.5	97.5	97.5	98.5
	cp_icl	22225	10.4	2.0	99.5	99.5	98.5	98.0	98.0	99.5
	cp_icl_cot	23398	12.1	2.0	99.5	99.5	99.5	98.0	98.0	99.5
Gpt-5.4-mini	baseline	14727	11.3	13.0	99.5	99.5	99.0	87.0	87.0	98.5
	cp	15839	11.7	9.5	98.0	98.0	98.0	90.5	90.5	94.5
	cp_cot	16866	13.3	3.5	99.5	99.5	99.5	96.5	96.5	98.5
	cp_icl	16348	10.8	6.5	99.5	99.5	99.5	93.5	93.5	95.5
	cp_icl_cot	17446	11.5	2.0	99.5	99.5	99.0	98.0	98.0	97.0
Local Ollama										
Qwen3.6:35b (Think)	baseline	22700	55.9	11.5	98.0	98.0	97.5	88.5	88.5	97.0
	cp	25990	68.4	5.0	100.0	100.0	100.0	95.0	95.0	100.0
	cp_cot	26216	66.3	4.5	99.5	99.5	99.5	95.5	95.5	99.5
	cp_icl	27796	66.7	7.5	98.0	98.0	98.0	92.5	92.5	98.0
	cp_icl_cot	28681	68.8	8.0	98.5	98.5	98.5	92.0	92.0	98.5
Qwen3.6:35b	baseline	19169	24.0	10.0	99.5	99.5	99.5	90.0	90.0	98.0
	cp	20492	26.2	4.5	100.0	100.0	100.0	95.5	95.5	99.5
	cp_cot	21664	30.4	7.0	99.5	99.5	99.5	93.0	93.0	98.5
	cp_icl	23758	32.5	7.5	100.0	100.0	100.0	92.5	92.5	99.0
	cp_icl_cot	23676	27.8	3.5	100.0	100.0	99.5	96.5	96.5	100.0
Gemma4:31b (Think)	baseline	19840	88.7	6.5	98.5	98.5	98.5	93.5	93.5	98.0
	cp	21374	97.3	4.5	99.5	99.5	99.5	95.5	95.5	99.5
	cp_cot	22479	104.1	5.5	99.0	99.0	99.0	94.5	94.5	98.5
	cp_icl	23446	104.8	7.5	96.1	96.1	96.1	89.6	89.6	96.1
	cp_icl_cot	22346	118.9	6.0	98.0	98.0	98.0	94.0	94.0	97.0
Gemma4:31b	baseline	17864	41.2	8.0	100.0	100.0	100.0	92.0	92.0	99.5
	cp	18868	48.6	7.5	99.5	99.5	99.5	92.5	92.5	99.5
	cp_cot	20225	60.1	5.0	99.0	99.0	99.0	95.0	95.0	98.5
	cp_icl	21264	50.0	7.5	99.0	99.0	99.0	92.5	92.5	99.0
	cp_icl_cot	22420	57.7	5.5	99.0	99.0	99.0	94.5	94.5	99.0
Gpt-oss:20b (Think)	baseline	18348	36.6	31.5	89.0	89.0	88.0	68.5	68.5	88.0
	cp	18362	30.2	14.0	90.0	90.0	90.0	86.0	86.0	89.0
	cp_cot	19784	35.3	14.5	85.5	85.5	85.0	85.5	85.5	82.0
	cp_icl	23048	46.3	31.5	71.5	71.5	71.5	68.5	68.5	70.0
	cp_icl_cot	31955	96.6	24.0	80.0	80.0	79.0	76.0	76.0	73.0

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query (errored / rate-limited zero-token rows excluded).

Table 12: Task B (behaviour preset) per-cell funnel KPIs (N=200 per cell). The rightmost column *preset %* is the headline check (the modified vType feature vector matches the expected preset within $rtol$=10^{-3}); *vClass %* confirms the LLM did not silently switch the vehicle class.

Model	Prompt	Cost		Fail% ↓	Pipeline stage pass rates					
		tokens ↓	latency (s) ↓		XML % ↑	tgt % ↑	vClass % ↑	SUMO % ↑	CR % ↑	**preset % ↑**
Cloud API										
Qwen3.6-plus	baseline	19149	53.8	8.5	100.0	100.0	92.5	91.5	91.5	0.0
	cp	20667	56.3	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	20784	61.5	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	21882	46.8	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	24666	55.5	0.5	100.0	100.0	100.0	99.5	99.5	100.0
Deepseek-v3.2	baseline	14152	73.8	9.5	98.0	98.0	95.5	90.5	90.5	0.0
	cp	15801	73.2	1.0	99.5	99.5	99.5	99.0	99.0	99.5
	cp_cot	17074	84.9	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl	17027	59.2	0.5	99.5	99.5	99.5	99.5	99.5	99.5
	cp_icl_cot	19543	74.0	2.5	97.5	97.5	97.5	97.5	97.5	97.5
Glm-5	baseline	14233	40.2	9.5	99.0	99.0	88.5	90.5	90.5	0.0
	cp	15531	43.9	1.0	99.0	99.0	99.0	99.0	99.0	96.5
	cp_cot	16723	51.9	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl	18382	45.4	3.0	99.0	99.0	97.0	97.0	97.0	99.0
	cp_icl_cot	19756	53.1	2.0	99.0	99.0	98.0	98.0	98.0	99.0
Gemini-3-flash	baseline	18266	11.4	17.0	99.0	99.0	87.0	83.0	83.0	0.5
	cp	19732	11.5	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_cot	21028	13.1	1.0	99.0	99.0	99.0	99.0	99.0	98.5
	cp_icl	22326	10.6	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl_cot	23915	12.7	2.0	99.0	99.0	99.0	98.0	98.0	99.0
Gpt-5.4-mini	baseline	14756	13.3	12.5	99.0	99.0	88.5	87.5	87.5	0.0
	cp	16121	12.4	2.0	98.5	98.5	98.0	98.0	98.0	98.0
	cp_cot	17223	14.2	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl	18447	12.0	1.5	99.0	99.0	99.0	98.5	98.5	99.0
	cp_icl_cot	19521	13.3	1.0	99.0	99.0	99.0	99.0	99.0	99.0
Local Ollama										
Qwen3.6:35b (Think)	baseline	23582	64.9	14.0	98.0	98.0	85.0	86.0	86.0	0.0
	cp	28174	84.4	0.5	100.0	100.0	100.0	99.5	99.5	99.5
	cp_cot	28190	79.4	0.0	100.0	100.0	100.0	100.0	100.0	99.5
	cp_icl	29818	80.0	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	31863	89.0	0.5	99.5	99.5	99.5	99.5	99.5	99.5
Qwen3.6:35b	baseline	18980	28.3	22.5	98.0	98.0	84.5	77.5	77.5	0.5
	cp	20650	28.7	3.0	97.5	97.5	97.0	97.0	97.0	97.5
	cp_cot	21437	30.9	2.0	98.0	98.0	98.0	98.0	98.0	97.5
	cp_icl	23511	33.1	4.0	98.5	98.5	96.0	96.0	96.0	98.5
	cp_icl_cot	23546	28.3	1.0	99.0	99.0	99.0	99.0	99.0	99.0
Gemma4:31b (Think)	baseline	19462	94.2	3.5	99.0	99.0	94.0	96.5	96.5	1.0
	cp	21564	100.7	0.0	100.0	100.0	100.0	100.0	100.0	99.5
	cp_cot	22031	104.2	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	23799	92.8	1.5	98.5	98.5	98.5	98.5	98.5	98.5
	cp_icl_cot	24478	106.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0
Gemma4:31b	baseline	17312	44.8	9.0	99.0	99.0	90.5	91.0	91.0	0.5
	cp	18902	49.5	1.5	98.5	98.5	98.5	98.5	98.5	98.5
	cp_cot	20397	63.7	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl	21448	53.1	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl_cot	22730	61.9	1.0	99.0	99.0	99.0	99.0	99.0	99.0
Gpt-oss:20b (Think)	baseline	19056	39.1	12.5	93.5	93.5	87.0	87.5	87.5	0.0
	cp	19023	32.1	8.5	91.5	91.5	91.0	91.5	91.5	87.5
	cp_cot	22174	46.8	7.0	93.0	93.0	93.0	93.0	93.0	91.0
	cp_icl	28777	76.7	18.5	82.0	82.0	82.0	81.5	81.5	81.5
	cp_icl_cot	30613	79.2	15.0	85.0	85.0	85.0	85.0	85.0	83.0

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query (errored / rate-limited zero-token rows excluded).

Table 13: Task P (population edit, combined add+remove) per-cell funnel KPIs (first-100 of P_{add} + first-100 of P_{remove} = N=200 per cell). The rightmost column *count_match %* is the headline check (the requested $+1/-1$ vehicle-count delta is present); *tgt-chg %* fires when the new vehicle starts on the requested edge (add) or the named vehicle is gone (remove); *others %* confirms non-target route entries are byte-equivalent.

Model	Prompt	Cost		Fail% ↓	Pipeline stage pass rates					
		tokens ↓	latency (s) ↓		XML % ↑	tgt-chg % ↑	others % ↑	SUMO % ↑	CR % ↑	**count_match % ↑**
Cloud API										
Qwen3.6-plus	baseline	17948	50.9	33.0	99.0	90.5	98.5	67.0	67.0	99.0
	cp	19066	53.2	4.0	99.0	99.0	99.0	96.0	96.0	99.0
	cp_cot	19958	58.5	2.0	99.0	99.0	99.0	98.0	98.0	99.0
	cp_icl	21371	47.3	3.5	99.0	98.5	98.5	96.5	96.5	98.5
	cp_icl_cot	23369	55.6	0.0	100.0	100.0	99.5	100.0	100.0	100.0
Deepseek-v3.2	baseline	14410	70.0	34.5	99.5	98.5	99.5	65.5	65.5	99.0
	cp	14993	67.3	1.0	99.5	99.5	99.5	99.0	99.0	99.5
	cp_cot	17068	77.7	0.5	100.0	100.0	99.5	99.5	99.5	100.0
	cp_icl	16418	56.6	2.0	99.5	99.5	98.5	98.0	98.0	99.5
	cp_icl_cot	17683	69.4	1.0	99.5	99.5	99.0	99.0	99.0	99.5
Glm-5	baseline	14635	36.4	39.0	99.0	98.0	99.0	61.0	61.0	99.0
	cp	15763	41.6	2.0	99.0	99.0	98.5	98.0	98.0	99.0
	cp_cot	16505	49.6	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl	17122	41.5	3.0	99.0	99.0	98.5	97.0	97.0	99.0
	cp_icl_cot	18640	51.2	1.0	99.0	99.0	99.0	99.0	99.0	99.0
Gemini-3-flash	baseline	18992	10.7	51.0	98.5	96.0	98.5	49.0	49.0	98.5
	cp	19996	11.0	1.5	99.0	99.0	99.0	98.5	98.5	99.0
	cp_cot	21198	12.4	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_icl	21946	10.5	1.5	99.0	99.0	99.0	98.5	98.5	99.0
	cp_icl_cot	23271	12.0	1.0	99.0	99.0	99.0	99.0	99.0	99.0
Gpt-5.4-mini	baseline	15085	11.9	12.0	97.5	96.5	97.5	88.0	88.0	97.5
	cp	16124	12.5	3.0	99.0	98.5	98.0	97.0	97.0	98.5
	cp_cot	17215	13.7	3.0	99.0	99.0	99.0	97.0	97.0	99.0
	cp_icl	17642	11.2	1.5	99.0	98.0	98.0	98.5	98.5	98.5
	cp_icl_cot	18676	13.4	2.5	99.0	99.0	99.0	97.5	97.5	99.0
Local Ollama										
Qwen3.6:35b (Think)	baseline	23502	56.2	12.5	100.0	95.5	98.0	87.5	87.5	98.0
	cp	26284	69.4	2.0	99.5	99.0	97.0	98.0	98.0	97.5
	cp_cot	27253	71.5	3.0	98.5	98.5	95.0	97.0	97.0	95.0
	cp_icl	26972	62.8	3.0	99.5	99.5	97.0	97.0	97.0	96.5
	cp_icl_cot	29308	74.2	0.5	100.0	100.0	96.5	99.5	99.5	96.5
Qwen3.6:35b	baseline	19200	24.0	6.0	100.0	95.5	99.0	94.0	94.0	99.0
	cp	20452	25.8	3.0	100.0	98.0	100.0	97.0	97.0	100.0
	cp_cot	21402	28.6	9.0	100.0	100.0	96.5	91.0	91.0	96.5
	cp_icl	23033	32.6	1.5	100.0	98.0	99.0	98.5	98.5	99.0
	cp_icl_cot	23296	30.3	2.5	100.0	99.5	95.5	97.5	97.5	95.5
Gemma4:31b (Think)	baseline	20408	95.3	6.0	99.5	98.5	99.5	94.0	94.0	99.5
	cp	21321	89.7	0.5	100.0	100.0	100.0	99.5	99.5	100.0
	cp_cot	22424	98.1	0.5	100.0	100.0	100.0	99.5	99.5	100.0
	cp_icl	23181	90.8	1.5	98.7	98.7	98.7	98.7	98.7	98.7
	cp_icl_cot	24394	97.7	1.0	99.3	99.3	99.3	98.7	98.7	99.3
Gemma4:31b	baseline	17710	45.7	38.0	99.0	96.0	99.0	62.0	62.0	99.0
	cp	19090	52.2	2.0	99.0	99.0	99.0	98.0	98.0	99.0
	cp_cot	20452	58.9	2.0	99.0	99.0	99.0	98.0	98.0	99.0
	cp_icl	20726	46.0	1.5	99.0	99.0	99.0	98.5	98.5	99.0
	cp_icl_cot	22078	54.5	1.5	99.0	99.0	99.0	98.5	98.5	99.0
Gpt-oss:20b (Think)	baseline	21892	53.2	12.0	90.0	90.0	90.0	85.0	85.0	90.0
	cp	22965	49.1	10.0	90.0	90.0	90.0	88.0	88.0	90.0
	cp_cot	25822	74.1	10.5	92.9	92.9	92.9	90.5	90.5	92.9
	cp_icl	28929	102.9	18.5	82.0	82.0	81.5	81.5	81.5	81.5
	cp_icl_cot	30534	86.3	20.5	80.0	80.0	79.5	79.5	79.5	80.0

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query (errored / rate-limited zero-token rows excluded).

Table 14: Task G (goal extraction) per-cell funnel KPIs (N=200 per cell). The rightmost column *edge-GT %* is the headline semantic check (the extracted target lanelet matches the GT); *pos-GT %* is the secondary semantic check (the position keyword inside that lanelet matches the GT). G has no Frenetix or SUMO column because the edit modifies only the planning problem, not traffic.

Model	Prompt	Cost		Fail% ↓	Pipeline stage pass rates					
		tokens ↓	latency (s) ↓		JSON % ↑	edge % ↑	pos-enum % ↑	pos-GT % ↑	lanelet % ↑	**edge-GT % ↑**
Cloud API										
Qwen3.6-plus	baseline	6089	2.4	11.5	100.0	88.5	100.0	94.5	88.5	88.5
	cp	6345	2.2	1.0	100.0	99.0	100.0	100.0	99.0	99.0
	cp_cot	6695	3.6	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	6746	2.2	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	7097	3.3	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Deepseek-v3.2	baseline	4732	9.0	11.0	100.0	89.0	100.0	98.5	89.0	89.0
	cp	4966	8.2	8.0	100.0	92.0	100.0	100.0	92.0	92.0
	cp_cot	5280	8.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	5326	10.8	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	5631	9.8	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Glm-5	baseline	4958	3.6	11.0	100.0	89.0	98.5	98.5	89.0	89.0
	cp	5194	3.3	3.0	100.0	97.0	99.5	99.5	97.0	97.0
	cp_cot	5521	4.3	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	5559	3.0	7.5	100.0	92.5	100.0	100.0	92.5	92.5
	cp_icl_cot	5883	3.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Gemini-3-flash	baseline	6037	1.0	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp	6301	0.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	6646	1.3	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	6710	0.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	7047	1.3	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Gpt-5.4-mini	baseline	4687	0.7	0.0	100.0	100.0	100.0	91.5	100.0	100.0
	cp	4927	0.7	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	5262	1.1	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	5281	0.7	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	5575	0.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Local Ollama										
Qwen3.6:35b (Think)	baseline	6745	5.9	0.5	99.5	99.5	99.5	99.5	99.5	99.5
	cp	6955	5.6	1.0	99.0	99.0	99.0	99.0	99.0	99.0
	cp_cot	8074	11.5	5.0	95.0	95.0	95.0	95.0	95.0	95.0
	cp_icl	7161	4.5	0.5	99.5	99.5	99.5	99.5	99.5	99.5
	cp_icl_cot	8012	8.3	9.0	91.0	91.0	91.0	91.0	91.0	91.0
Qwen3.6:35b	baseline	6085	1.7	0.5	100.0	99.5	100.0	94.0	99.5	99.5
	cp	6346	1.7	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	6697	2.2	0.0	100.0	100.0	100.0	98.0	100.0	100.0
	cp_icl	6747	1.7	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	7105	2.3	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Gemma4:31b (Think)	baseline	6325	7.4	1.5	100.0	98.5	100.0	100.0	98.5	98.5
	cp	6483	5.1	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	6734	4.8	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	6840	4.6	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	7140	5.6	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Gemma4:31b	baseline	6055	2.5	2.0	100.0	98.0	100.0	100.0	98.0	98.0
	cp	6320	2.5	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	6657	3.7	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl	6730	2.6	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	7067	3.7	0.0	100.0	100.0	100.0	100.0	100.0	100.0
Gpt-oss:20b (Think)	baseline	4851	1.2	1.0	100.0	99.0	100.0	99.0	99.0	99.0
	cp	5078	1.0	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_cot	5415	1.6	6.0	94.0	94.0	94.0	94.0	94.0	94.0
	cp_icl	5432	1.2	0.0	100.0	100.0	100.0	100.0	100.0	100.0
	cp_icl_cot	5825	1.6	0.0	100.0	100.0	100.0	100.0	100.0	100.0

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query (errored / rate-limited zero-token rows excluded).

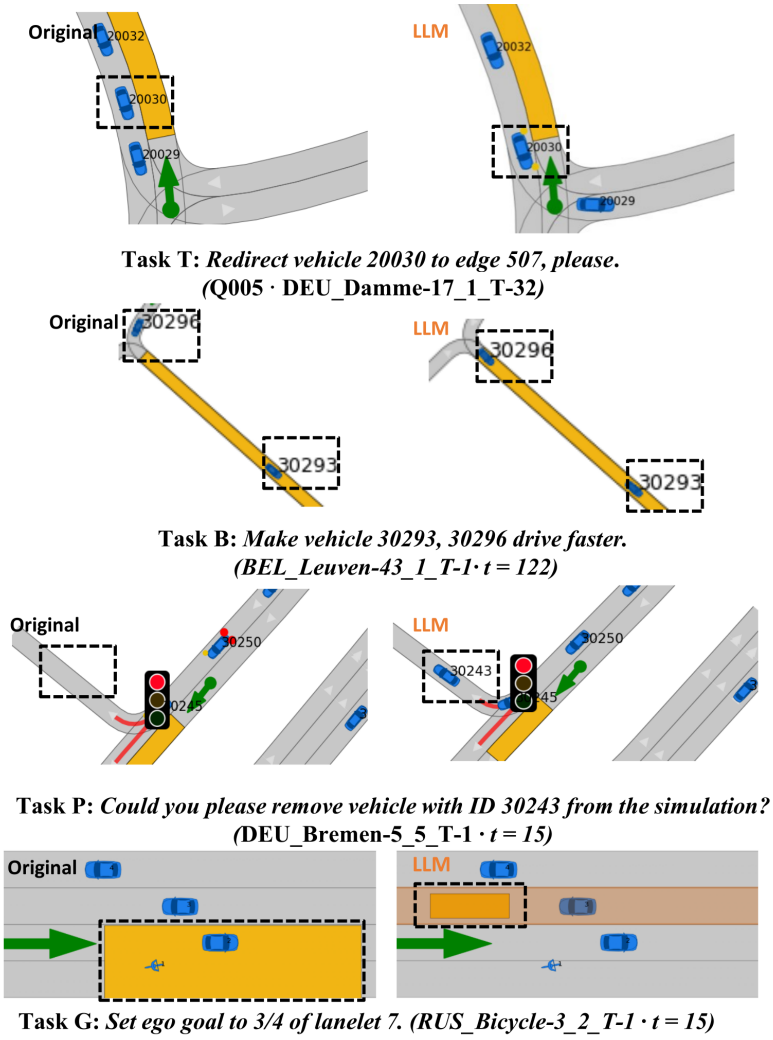


Figure 10: Per-task qualitative diff grid (four rows = four tasks). Each row compares baseline (left) and LLM-modified (right) scenarios at the same mid-trajectory timestep (annotated in the row header). All examples from `gemini-3-flash-preview` under `cp_icl_cot`.

baseline trajectory to the `aggressiveLaner` preset, we measure per-scenario `min_risk_score` (integer 0–5; 0 = collision risk, 5 = safe) using the criticality toolkit *From-Words-to-Collisions* (Gao et al., 2025), computed directly on each side's Frenetix-executed trajectory (no re-planning required). The score distribution shifts measurably toward more critical under modification: the 0-bucket (collision risk) doubles from 10 to 20 and $23/84$ paired scenarios move to a strictly more critical bucket vs. 18 less critical (paired Wilcoxon $count\_match = -0.095$, p=0.44). The B-task modification thereby provides a natural-language interface for shifting downstream scenario safety profiles, a building block for safety-critical scenario generation.

A.5 Module Router

A.5.1 Dispatch Pseudocode

Algorithm 1 formalises the per-turn dispatch. The router observes (u_t, h_t), computes the *argmax* over the action set $\mathcal{A}$ defined in §3, and invokes the action-conditional operator. The five branches cover the five-action $\mathcal{A}$. The actual implementation expands MODIFY into the four modification sub-tasks (T/B/P/G), TEST into single- and batch-execution variants, and QA into general/parameter/batch Q&A specialisations.

A.5.2 Query Corpus

The Module Router benchmark uses 200 queries spanning 9 action classes (the eight supported actions plus the `fail` class that the router emits when a request is unsupported). Table 15 summarises the action-class distribution and, for the `vehicle_mod`

Algorithm 1 Module Router Dispatch

Require: database $\mathcal{D}$, planner config θ, action set $\mathcal{A}$

$s \leftarrow f_{\text{gen}}(u_0)$ **or** $f_{\text{sel}}(u_0, \mathcal{D})$ ▷ initial population
$h \leftarrow \emptyset$; $o \leftarrow \perp$ ▷ empty history, no outcome yet
while dialogue is active **do**
  observe utterance u_t
  Stage 1 (LLM): $(\hat{a}_t, \texttt{args}) \leftarrow$ JSON output with $\hat{a}_t = \arg\max_{a \in \mathcal{A}} f_{\text{router}}(a \mid u_t, h)$
  Stage 2 (Process Engine): dispatch on $\hat{a}_t$ with operand u_t
  if $\hat{a}_t = \text{MODIFY}$ **then**
    $s \leftarrow f_{\text{mod}}(s, u_t)$; $r \leftarrow s$
  else if $\hat{a}_t = \text{TUNE}$ **then**
    $\theta \leftarrow f_{\text{tune}}(\theta, u_t)$; $r \leftarrow \theta$
  else if $\hat{a}_t = \text{TEST}$ **then**
    $o \leftarrow f_{\text{test}}(s, \theta)$; $r \leftarrow o$ ▷ $o = (\tau, c, m)$
  else if $\hat{a}_t = \text{ANALYSE}$ **then**
    $r \leftarrow f_{\text{eval}}(o, u_t)$ ▷ requires prior TEST
  else if $\hat{a}_t = \text{QA}$ **then**
    $r \leftarrow$ LLM Q&A on (u_t, h)
  end if
  $h \leftarrow h \cup \{(u_t, \hat{a}_t, r)\}$
end while

class, the sub-tasks each query exercises. Composite `vehicle_mod` codes (e.g. `T+B+P`) test that the router enumerates multiple letters and respects the canonical $T{\rightarrow}B{\rightarrow}P{\rightarrow}G$ ordering rule.

Table 15: Composition of the Module Router corpus.

Action Class	#	Mod Sub-tasks	#
vehicle_mod	40	Goal (G)	9
qa	23	Behavior (B)	9
param_qa	21	Add/Rem (P)	7
batch_qa	20	T+B+P	3
batch_sim	20	Trajectory (T)	3
param_mod	20	T+B	2
analysis	19	T+P	2
batch_analysis	19	P+G	2
fail	18	B+P	2

A.5.3 Evaluation Metrics

The Module Router is graded by a five-stage funnel computed against the GT JSON action emitted by each prompt:

- **JSON %**: the model output parses as a JSON object with exactly one top-level key.
- **key %**: the top-level key matches the GT class (e.g. the composite vehicle mod or the refusal class fail).
- **value %** (**headline**): the value is a strict string match against GT. For vehicle mod composites, this metric is sensitive to letter ordering.
- **set %**: the letter set matches GT, relaxing the canonical $T{\rightarrow}B{\rightarrow}P{\rightarrow}G$ ordering rule. Reported only for vehicle mod composites; isolates the ordering signal from the underlying classification accuracy.
- **order %**: the emitted letter sequence honours the canonical order. Reported on the composite-only subset.

The composite *Overall* score is the mean of the per-query boolean checks above. We report mean tokens and per-query latency as the cost cluster.

A.5.4 Evaluation Results

Table 16 reports the full results matrix ($N = 200$ per cell, all 50 (model, condition) cells fully populated). The prompt ladder lifts strict-match accuracy substantially for every model: `gpt-5.4-mini` climbs from 43.5% (baseline) to 98.5% under `cp_icl_cot`. `Gemini-3-flash-preview` climbs from 62.0% to 99.0%. The `cp` condition delivers the biggest single jump (the categorical action schema and the ordering rule both fit cleanly into a curated-prompt block). Adding ICL exemplars (`cp_icl`) and CoT scaffolding (`cp_icl_cot`) yield smaller but consistent gains. The relaxed *set %* metric is at or near 100% for almost every (model, condition) cell once any prompting is applied, confirming that the residual error under the headline metric is dominated by letter-ordering mistakes rather than misclassification of the underlying subtasks.

A.6 Planner Testing and Enhancement Module

A.6.1 Query Corpus

The Planner Testing and Enhancement benchmark uses 200 queries that test the LLM's ability to emit a complete updated Frenetix `cost.yaml`. Table 17 summarises the request-type distribution and per-parameter coverage. Queries are partitioned into four request types (*pure preset* = name a preset only; *pure explicit* = name one or more parameters with target values; *mixed* = preset + explicit overrides; *out of vocab* = name a parameter that

Table 16: Performance of the Module Router on the action-classification benchmark (N=200 per cell). *value %* is a strict string match against the GT JSON action (ordering-sensitive for `vehicle_mod` composites); *set %* relaxes the canonical $T \rightarrow B \rightarrow P \rightarrow G$ ordering rule and is reported only on the `vehicle_mod` composite subset.

Model	Prompt	Cost		Fail% ↓	Pipeline stage pass rates					Overall ↑
		tokens ↓	latency (s) ↓		JSON % ↑	key % ↑	value % ↑	set % ↑	order % ↑	
Cloud API										
Qwen3.6-plus	baseline	274	1.7	0.0	100.0	49.5	47.0	0.0	—	0.651
	cp	1013	1.6	0.0	100.0	95.0	94.5	100.0	91.7	0.965
	cp_cot	1446	3.4	0.0	100.0	97.0	96.5	97.5	100.0	0.978
	cp_icl	2026	1.6	0.0	100.0	99.0	99.0	100.0	100.0	0.993
	cp_icl_cot	2459	3.5	0.0	100.0	97.5	97.0	97.5	100.0	0.981
Deepseek-v3.2	baseline	261	7.1	0.0	100.0	62.5	48.5	22.2	100.0	0.681
	cp	983	2.3	0.0	100.0	92.5	91.5	95.0	100.0	0.946
	cp_cot	1397	4.4	0.0	100.0	91.0	90.5	97.5	100.0	0.938
	cp_icl	1943	6.4	0.0	100.0	97.0	96.5	97.4	100.0	0.978
	cp_icl_cot	2353	4.3	0.0	100.0	95.5	95.5	100.0	100.0	0.970
Glm-5	baseline	261	3.1	0.5	99.5	64.5	51.5	17.2	100.0	0.699
	cp	979	2.7	0.0	100.0	94.5	94.5	100.0	100.0	0.963
	cp_cot	1386	5.1	0.0	100.0	96.5	96.0	97.5	100.0	0.974
	cp_icl	1919	2.4	0.0	100.0	97.0	96.5	97.5	100.0	0.978
	cp_icl_cot	2320	3.8	0.0	100.0	98.0	98.0	100.0	100.0	0.987
Gemini-3-flash	baseline	256	0.9	0.0	100.0	73.0	62.0	29.0	100.0	0.767
	cp	1019	0.9	0.0	100.0	97.5	97.5	100.0	100.0	0.983
	cp_cot	1429	1.3	0.0	100.0	97.0	96.5	97.5	100.0	0.978
	cp_icl	2064	1.0	0.0	100.0	97.5	97.0	97.5	100.0	0.981
	cp_icl_cot	2479	1.3	0.0	100.0	99.0	99.0	100.0	100.0	0.993
Gpt-5.4-mini	baseline	272	1.2	0.0	100.0	57.0	43.5	22.9	100.0	0.649
	cp	982	1.0	0.0	100.0	91.0	91.0	100.0	100.0	0.940
	cp_cot	1309	1.1	0.0	100.0	90.5	90.5	100.0	100.0	0.937
	cp_icl	1913	1.0	0.0	100.0	98.0	98.0	100.0	100.0	0.987
	cp_icl_cot	2243	1.1	0.0	100.0	98.5	98.5	100.0	100.0	0.990
Local Ollama										
Qwen3.6:35b (Think)	baseline	5110	35.9	26.5	73.5	48.0	45.5	37.5	100.0	0.552
	cp	1291	2.1	0.0	100.0	93.0	92.0	94.9	100.0	0.949
	cp_cot	2743	10.2	4.5	95.5	91.0	91.0	100.0	100.0	0.925
	cp_icl	2271	2.2	0.0	100.0	96.0	96.0	100.0	100.0	0.973
	cp_icl_cot	2874	4.0	0.0	100.0	96.5	96.5	100.0	100.0	0.977
Qwen3.6:35b	baseline	289	0.5	21.0	79.0	46.0	30.5	18.4	100.0	0.494
	cp	1018	0.4	0.0	100.0	94.5	93.0	97.5	83.3	0.957
	cp_cot	1465	1.2	0.0	100.0	95.5	95.5	100.0	100.0	0.970
	cp_icl	2028	0.6	0.0	100.0	96.5	95.5	95.0	100.0	0.973
	cp_icl_cot	2448	1.2	0.0	100.0	97.0	97.0	100.0	100.0	0.980
Gemma4:31b (Think)	baseline	1035	12.6	0.0	100.0	69.5	65.5	33.3	75.0	0.778
	cp	1319	4.9	0.0	100.0	96.5	95.0	92.5	100.0	0.971
	cp_cot	1704	5.6	0.0	100.0	97.0	96.0	95.0	100.0	0.976
	cp_icl	2347	4.8	0.0	100.0	97.0	97.0	100.0	100.0	0.980
	cp_icl_cot	2721	5.5	0.0	100.0	98.0	98.0	100.0	100.0	0.987
Gemma4:31b	baseline	278	0.5	0.0	100.0	60.0	57.5	0.0	—	0.722
	cp	1038	0.4	0.0	100.0	97.0	96.5	97.5	100.0	0.978
	cp_cot	1440	1.4	0.0	100.0	98.0	98.0	100.0	100.0	0.987
	cp_icl	2080	0.4	0.0	100.0	98.5	97.0	100.0	75.0	0.984
	cp_icl_cot	2488	1.5	0.0	100.0	99.5	99.5	100.0	100.0	0.997
Gpt-oss:20b (Think)	baseline	696	2.1	0.5	99.5	53.0	42.5	32.3	62.5	0.633
	cp	1135	0.6	18.5	81.5	78.0	78.0	100.0	100.0	0.792
	cp_cot	1656	1.6	0.0	100.0	95.5	95.5	100.0	100.0	0.970
	cp_icl	2110	0.7	6.0	94.0	91.0	91.0	100.0	100.0	0.920
	cp_icl_cot	2581	1.3	0.5	99.5	96.0	96.0	100.0	100.0	0.972

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query.

does not exist in the schema, which the LLM must refuse).

Table 17: Composition of the Planner Testing and Enhancement configuration corpus.

Request Type	#	Top Parameters	#
Pure preset	71	dist to ref path	21
Pure explicit	69	dist to obstacles	18
Mixed	51	orientation offset	16
Out of vocab	9	accel / jerk	14
		respons. / path len	13

A.6.2 Evaluation Metrics

The Planner Testing and Enhancement task is graded by a six-stage funnel against the GT YAML, plus a downstream simulator-runnability check:

- **YAML %**: the model output parses as a YAML mapping.
- **struct %**: the 13 cost weights keys and 3 external cost weights keys are present, with identical names and ordering to the input `cost.yaml`.
- **preset %**: on queries whose GT carries a named preset, every cost weight matches the preset's tabulated value within relative tolerance 10^{-3}.
- **explicit %**: on queries whose GT carries explicit (key, value) pairs, each named parameter matches GT within relative tolerance 10^{-3}.
- **full-YAML %** (**headline**): the strictest stage: every value in the emitted YAML matches GT within relative tolerance 10^{-3} (combines the previous two stages plus untouched defaults).
- **refused %**: on the 9-query out-of-vocab slice, the model declines to emit a YAML and instead surfaces a refusal token (e.g. "do not", "not a valid parameter").

The composite *Overall* score is the mean of the per-query boolean checks. As an additional downstream check we run Frenetix on every emitted YAML against a paired baseline-simulable scenario; an outcome of either reaching the goal or hitting the simulation horizon without a crash counts as runnable.

A.6.3 Evaluation Results

Table 18 reports the full results matrix (N=191 scored queries per (model, condition) cell, with the remaining 9 out-of-vocab queries reported separately under *refused %*. All 50 cells are fully populated). The headline *full-YAML %* is 36% across all baseline cells. This is the floor produced by leaving the input `cost.yaml` untouched, which matches GT only on $\sim 36\%$ of queries that did not require a change. Any prompting (`cp` onwards) lifts almost every model to $\geq 98\%$ full-YAML match. The *refused %* column reveals that out-of-vocab refusals are the hardest sub-task: only the strongest models hit 100% under `cp` alone. Adding ICL exemplars (`cp_icl`, `cp_icl_cot`) is what eliminates the remaining false-positive YAML emissions for those out-of-vocab queries. The Frenetix-validation footnote shows that 81.2% of the YAMLs emitted by `gpt-5.4-mini` under `cp_icl_cot` produced a runnable planner configuration end-to-end. The remaining 18.8% are scenarios in which the modified cost weights cause the planner to collide before reaching the goal, an additional planning-robustness signal that the YAML-match metric cannot detect. Figure 11 shows a representative qualitative example.

A.7 Cross-Planner Qualitative Comparison

PlannerForge exposes the same five driving-mode presets (*Default*, *Comfort*, *Balanced*, *Sporty*, *Safety*) for both the sampling-based Frenetix planner and the learning-based MP-RBFN planner. Figure 13 compares each planner's per-mode cost-weight profile. The two planners operate on different cost vocabularies: Frenetix exposes thirteen named weights (acceleration, jerk axes, path-length, lane-centre offset, velocity offset, distances to reference path and to obstacles, prediction, responsibility), while MP-RBFN exposes seven normalised channels keyed on its radial-basis representation (distance-to-boundary, distance-to-reference-path, orientation/velocity offsets, obstacle prediction). Despite the schema differences, the five shared presets produce the same relative shape in both planners: *Sporty* drives the velocity-offset weight up while loosening obstacle prediction. *Safety* drives the obstacle and prediction weights to their maxima; *Comfort* balances orientation and lateral-jerk weights. *Balanced* lies between them. A head-to-head quantitative Frenetix vs MP-RBFN comparison on framework-generated scenarios is part of an ongoing extension. The qualitative correspondence in Figure 13 establishes that the framework's natural-language preset interface generalises across planner families. Figure 12 shows

Table 18: Performance of the Planner Testing and Enhancement module (N=200 per cell). *full-YAML %* is a strict element-wise match against the GT `cost.yaml` (rtol 10^{-3}); *preset %* and *explicit %* are reported on the query subsets carrying a preset or explicit assignment respectively; *refused %* is the 9-query out-of-vocab refusal slice.

Model	Prompt	Cost		Fail% ↓	Pipeline stage pass rates						Overall ↑
		tokens ↓	latency (s) ↓		YAML % ↑	struct % ↑	preset % ↑	explicit % ↑	full-YAML % ↑	refused % ↑	
Cloud API											
Qwen3.6-plus	baseline	532	4.3	0.0	100.0	100.0	0.0	98.3	35.6	0.0	0.672
	cp	1479	4.4	0.0	100.0	100.0	98.4	99.2	99.0	88.9	0.990
	cp_cot	1966	7.1	0.0	100.0	100.0	100.0	100.0	100.0	88.9	0.995
	cp_icl	2964	4.3	0.0	100.0	100.0	99.2	100.0	99.5	100.0	0.998
	cp_icl_cot	3444	7.1	0.0	100.0	100.0	100.0	100.0	100.0	100.0	1.000
Deepseek-v3.2	baseline	532	9.8	0.0	100.0	100.0	0.0	99.2	36.1	0.0	0.674
	cp	1498	8.3	0.5	99.5	99.5	100.0	99.2	99.0	88.9	0.989
	cp_cot	1965	11.2	0.5	99.5	99.5	99.2	100.0	99.0	100.0	0.994
	cp_icl	3003	6.9	0.5	99.5	99.5	99.2	99.2	99.0	100.0	0.993
	cp_icl_cot	3465	10.1	0.0	100.0	100.0	97.5	100.0	98.4	100.0	0.993
Glm-5	baseline	511	5.6	0.0	100.0	100.0	0.0	97.5	35.6	0.0	0.671
	cp	1437	5.0	0.5	99.5	99.5	98.4	97.5	98.4	100.0	0.989
	cp_cot	1910	8.0	0.5	99.5	99.5	99.2	99.2	99.0	100.0	0.993
	cp_icl	2883	4.9	1.0	99.0	99.0	97.5	99.2	98.4	100.0	0.987
	cp_icl_cot	3353	7.7	1.0	99.0	99.0	96.7	97.5	96.3	100.0	0.978
Gemini-3-flash	baseline	561	1.5	0.0	100.0	100.0	0.0	99.2	36.1	0.0	0.674
	cp	1581	2.3	0.0	100.0	100.0	100.0	99.2	99.0	100.0	0.996
	cp_cot	2099	6.2	0.0	100.0	100.0	100.0	100.0	99.0	100.0	0.998
	cp_icl	3202	1.6	0.0	100.0	100.0	100.0	100.0	99.0	100.0	0.998
	cp_icl_cot	3697	2.3	0.0	100.0	100.0	98.4	99.2	97.9	100.0	0.992
Gpt-5.4-mini	baseline	521	1.7	0.0	100.0	100.0	0.0	100.0	36.1	0.0	0.675
	cp	1456	1.7	0.0	100.0	100.0	100.0	100.0	100.0	44.4	0.975
	cp_cot	1884	2.4	0.0	100.0	100.0	100.0	100.0	100.0	77.8	0.990
	cp_icl	2924	1.6	0.0	100.0	100.0	100.0	100.0	100.0	100.0	1.000
	cp_icl_cot	3246	1.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0	1.000
Local Ollama											
Qwen3.6:35b (Think)	baseline	2459	13.9	0.0	100.0	99.0	0.0	100.0	36.1	0.0	0.673
	cp	3252	13.1	0.5	99.5	99.0	95.9	99.2	96.9	88.9	0.979
	cp_cot	4140	16.9	0.0	100.0	100.0	95.9	100.0	97.4	100.0	0.990
	cp_icl	4129	9.0	0.0	100.0	100.0	96.7	100.0	97.9	100.0	0.991
	cp_icl_cot	5206	14.1	0.0	100.0	100.0	100.0	100.0	100.0	100.0	1.000
Qwen3.6:35b	baseline	533	1.3	0.0	100.0	100.0	0.0	100.0	36.1	0.0	0.675
	cp	1480	1.4	0.0	100.0	100.0	99.2	100.0	99.5	88.9	0.993
	cp_cot	1961	2.5	0.0	100.0	99.5	97.5	99.2	97.9	100.0	0.990
	cp_icl	2964	1.3	0.0	100.0	100.0	99.2	100.0	99.5	100.0	0.998
	cp_icl_cot	3410	2.2	0.5	99.5	99.5	98.4	100.0	99.0	100.0	0.993
Gemma4:31b (Think)	baseline	2918	40.9	0.0	100.0	100.0	0.0	99.2	36.1	0.0	0.674
	cp	2541	18.2	0.5	99.5	99.5	98.4	99.2	98.4	100.0	0.991
	cp_cot	3019	20.9	0.0	100.0	100.0	99.2	100.0	99.0	100.0	0.996
	cp_icl	3967	16.0	0.5	99.5	99.5	97.5	99.2	97.9	100.0	0.989
	cp_icl_cot	4416	17.8	0.0	100.0	100.0	99.2	100.0	99.0	100.0	0.996
Gemma4:31b	baseline	578	3.1	0.0	100.0	100.0	0.0	99.2	36.1	0.0	0.674
	cp	1598	3.1	0.0	100.0	100.0	100.0	100.0	100.0	100.0	1.000
	cp_cot	2075	5.4	0.0	100.0	100.0	99.2	100.0	99.5	100.0	0.998
	cp_icl	3219	3.1	0.0	100.0	100.0	99.2	100.0	99.5	100.0	0.998
	cp_icl_cot	3725	5.9	0.0	100.0	100.0	100.0	100.0	100.0	100.0	1.000
Gpt-oss:20b (Think)	baseline	1632	5.4	0.5	99.5	99.0	0.0	99.2	36.1	0.0	0.671
	cp	2210	4.2	1.0	99.0	99.0	96.7	99.2	97.9	100.0	0.985
	cp_cot	2877	6.0	1.0	99.0	99.0	96.7	99.2	97.9	100.0	0.985
	cp_icl	3567	3.8	3.1	96.9	96.9	94.3	100.0	95.8	100.0	0.966
	cp_icl_cot	4198	5.3	1.6	98.4	98.4	95.1	100.0	96.9	100.0	0.978

Frenetix validation (`gpt-5.4-mini`, `cp_icl_cot`, N=191)*:* ***155/191 (81.2%)*** *runnable (goal_reached + max_steps)*

Note: Arrows mark preferred directions (↑/↓). Tokens and latency (s) are means per query.

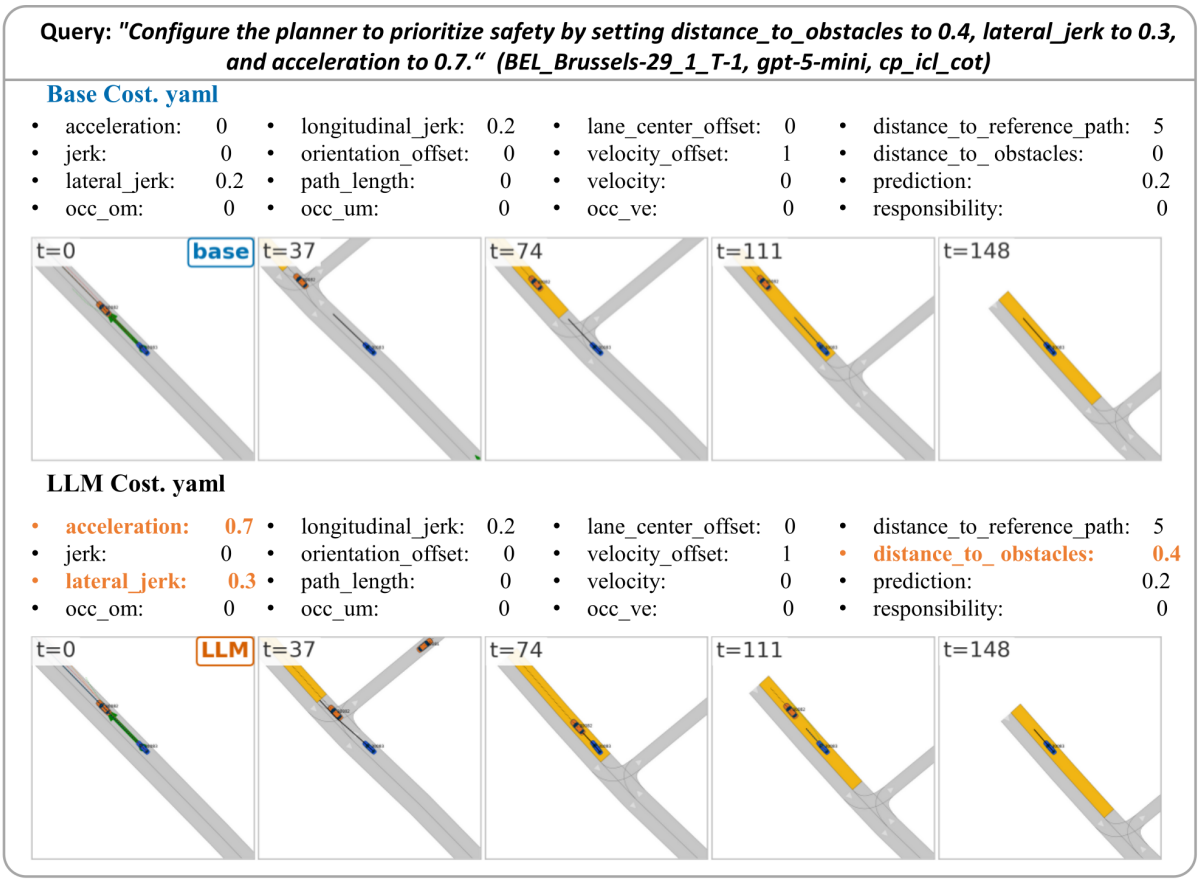


Figure 11: Pure-explicit Planner Testing and Enhancement example (Q003, scenario `BEL_Brussels-29_1_T-1`, `gpt-5.4-mini` × `cp_icl_cot`). The three weights modified by the LLM are highlighted in orange in the YAML; five matched-timestep frames are shown for the base and modified rollouts.

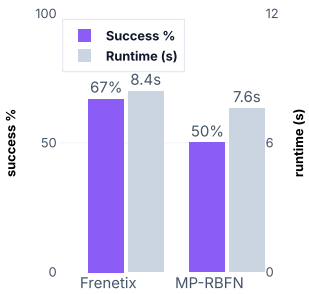
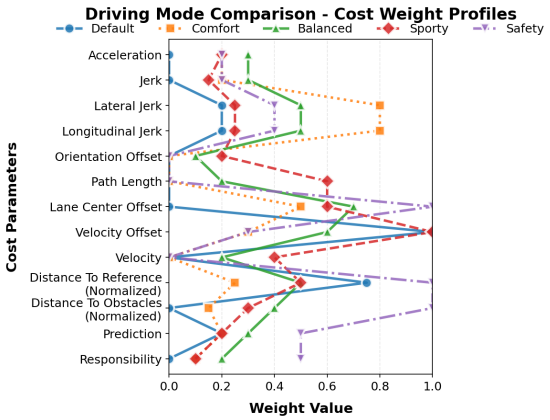


Figure 12: Cross-planner batch comparison. A single prompt dispatches the same scenario batch to both Frenetix and MP-RBFN and returns success rate and runtime for each.

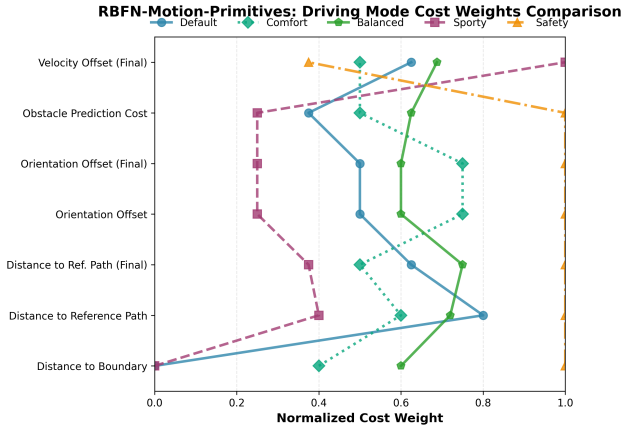
the chatbot workflow: a single prompt dispatches both Frenetix and MP-RBFN on N=100 shared scenarios from the `batch_100` preset. The Analysis Module reports success rate and mean runtime side-by-side, and can further diagnose specific failure reasons (e.g., collisions vs. timeouts) for each planner. This confirms that the unified π_P interface supports efficient, intuitive cross-planner benchmarking.

A.8 Overview Prompts

Each module uses a curated-prompt (`cp`) header that specifies the system role, the JSON/YAML/XML output schema, and the main constraints. The five prompt conditions (`baseline`, `cp`, `cp_cot`, `cp_icl`, `cp_icl_cot`) share this header; ICL and CoT variants add demonstrations and reasoning scaffolds on top. Full prompt files for Generation, Selection, Modification (T/B/P/G), Module Router, Planner Testing and Enhancement, and Batch Analysis are released with the code at `https://github.com/TUM-AVS/PlannerForge`.



(a) Frenetix — thirteen cost weights across the five preset modes.



(b) MP-RBFN — seven normalised cost channels across the five preset modes.

Figure 13: Qualitative cross-planner comparison of the five driving-mode presets exposed by PlannerForge. Despite the schema difference, both planners realise the same intent space (Sporty → velocity-up, lower obstacle weight; Safety → obstacle/prediction maxed; Comfort → jerk-and-orientation flattened) through their respective cost vocabularies.