Scenario Mining with LLMs

Xiaohong Zhang[1], Yuanqi Li[1], Xianfei Li[2], Peng Pai[2]
[1]Nanjing University,[2]Cowa Robot

Abstract

The Argoverse 2 Scenario Mining Challenge at the CVPR 2025 Workshop on Autonomous Driving involves converting textual queries into executable scripts to filter safety-critical scenarios from extensive driving logs. This report details our 3rd-place solution, which centers on a comparative analysis of leading Large Language Models (LLMs) for this code generation task. We used a batch-prompting strategy to efficiently process all scenario descriptions in a single pass, leveraging the contextual understanding of models like GPT-4o, Claude Sonnet 4, Gemini 2.5 Pro, and Grok. Our findings show that Gemini 2.5 Pro delivered the highest performance, achieving a HOTA-Temporal score of 52.09. This work provides a direct comparison of foundation models on a practical, domain-specific code synthesis task and demonstrates the viability of using off-the-shelf LLMs for complex autonomous driving workflows.

1. Introduction

The development of safe and reliable autonomous vehicles is contingent on rigorous testing and validation across a wide spectrum of driving scenarios. Manually identifying the full spectrum of driving scenarios is labor-intensive and unscalable. The Argoverse 2 (AV2) Scenario Mining Challenge [7] addresses this by providing a benchmark to automate the discovery of scenarios based on natural language descriptions. The core of the challenge is to translate a human-readable query, such as "large truck blocking view of a vehicle that is about to turn left", into a programmatic script that can query a database of vehicle trajectory data.

This translation task is a perfect application for large language models (LLMs) [1, 4–6], which have demonstrated remarkable capabilities in understanding language and generating code. Our approach directly tackles this challenge by using LLMs as code synthesizers. The goal is to generate a Python script that correctly utilizes a set of predefined atomic functions (e.g., $get\_objects\_of\_category()$, $is\_turning()$) to isolate the exact scenario described. Our method relies on a comprehensive prompt that bundles the function API, object categories, and few-shot examples, enabling batch code generation for all descriptions. By comparing the performance of GPT-4o, Claude Sonnet 4, Gemini 2.5 Pro, and Grok3, we identify the most effective model for this task and analyze the results. Our work provides a direct empirical comparison of four leading foundation models, quantifying their effectiveness on the AV scenario mining task.

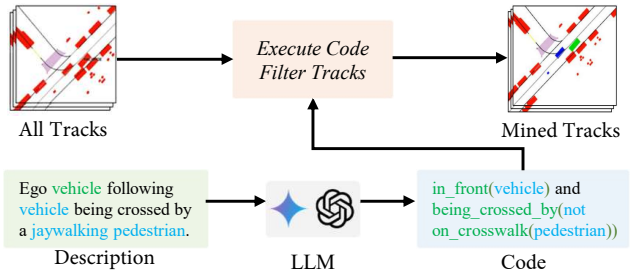

Figure 1. **Scenario Mining Pipeline.** Natural language descriptions are fed into an LLM, which generates a Python script using a predefined API. This script is then executed on the trajectory data to extract the target scenarios.

2. Methodology

Our solution is built upon the official competition baseline [3], which already provides tracked object data from Le3DE2E [2]. The overall pipeline of scenario mining is illustrated in Fig. 1.

2.1. Problem Formulation

The task is to find specific scenarios within a large dataset of driving logs. Each scenario is defined by a natural language description, d. The competition provides a predefined library of atomic functions, $\mathcal{F} = \{f_1, f_2, ..., f_k\}$, which operate on trajectory data. The goal is to generate a Python script, s, for each description d. This script s must compose functions from $\mathcal{F}$ to create a filter that, when executed, returns the log and track IDs corresponding to scenarios that match d. For example, for the description "*passenger vehicle turning left at intersection*", the target script is shown in Listing 1.

Table 1. **Performance Comparation of different LLMs.** Higher is better for all metrics. The best performance is highlighted in **bold**.

Model	HOTA-Temp.	HOTA-Track	Timestamp BA	Log BA
GPT-4o	39.19	42.14	69.39	64.43
Claude Sonnet 4	45.64	46.68	72.33	**69.09**
Grok	50.40	50.12	74.99	68.17
Gemini 2.5 Pro	**52.09**	**50.24**	**76.12**	66.52

```
description = "passenger vehicle turning left at intersection"
subject = get_objects_of_category(log_dir, category='VEHICLE')
near_inter = near_intersection(subject, log_dir, threshold=12)
scenario_candidate = turning(subject, log_dir, direction='left')
scenario_final = scenario_and([near_inter, scenario_candidate])
output_scenario(scenario_final, description, log_dir, output_dir)
```

Listing 1. Python code for vehicle turning scenario.

```
Please use the following functions to find instances of a referred object in an autonomous driving dataset. Be precise to the description, try to avoid returning false positives.
atomic functions:
{refav_context}
categories:
{av2_categories}
natural language descriptions:
{description_file}
Here is a list of examples:
{prediction_examples}
Output all the description and code pairs. Wrap all code in one python block and do not provide alternatives. Output code even if the given functions are not expressive enough to find the scenario.
```

Listing 2. Instruction for LLMs.

2.2. LLM-based Code Synthesis

We frame this problem as a direct code synthesis task for an LLM. The model is prompted with all the necessary information to understand the task and generate the correct code. This approach avoids complex intermediate representations or multi-agent systems, relying instead on the powerful in-context learning and reasoning abilities of modern foundation models.

To improve efficiency and leverage the LLM's ability to recognize patterns across the entire task set, we employed a batch-prompting strategy. We aggregated all descriptions into a single request. This approach has two main benefits: it significantly reduces the costs of calling LLMs, and it provides the model with a global view of the task distribution, potentially improving consistency. The core instruction given to the LLM is shown in Listing 2.

3. Experiments and Results

3.1. Experimental Setup

Dataset and Metrics. We evaluated our method on the AV2 Scenario Mining Challenge dataset. Performance is measured using four official metrics: HOTA-Temporal, HOTA-Track, Timestamp-level Binary Accuracy (Timestamp BA), and Log-level Binary Accuracy (Log BA).

Models. We conducted a comparative study of four prominent LLMs:

- **GPT-4o [1]**
- **Claude Sonnet 4 [4]**
- **Gemini 2.5 Pro [6]**
- **Grok3 [5]**

For each model, we used the same batch prompt to generate the full set of scenario mining scripts.

3.2. Results Analysis

The performance of the different LLMs is summarized in Tab. 1. A clear performance hierarchy emerges from the results. Gemini 2.5 Pro stands out as the top performer, achieving the highest scores on the primary HOTA-Temporal metric (52.09) as well as HOTA-Track and Timestamp BA. Grok also delivers a strong performance, closely trailing Gemini. Both models significantly outperform Claude Sonnet 4 and GPT-4o on this specific task.

This suggests that the models' capabilities in logical reasoning and adherence to complex, structured instructions (like a programming API) vary considerably. The superior performance of Gemini and Grok may be attributed to better underlying reasoning engines or more extensive training on code-related data. Based on these validation results, we select the scripts generated by Gemini 2.5 Pro for our final submission to the competition.

4. Conclusion

This report presents our 3rd-place solution to the AV2 Scenario Mining Challenge. Our approach centered on using Large Language Models to directly translate natural language descriptions into executable scenario-finding scripts. By conducting a comparative analysis of four leading foundation models, we found that Gemini 2.5 Pro provided the best performance for this domain-specific code generation task. We validate the potential of leveraging off-the-shelf LLMs to automate complex and critical workflows in the autonomous driving industry.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1, 2

[2] Feng Chen, Kanokphan Lertniphonphan, Yaqing Meng, Ling Ding, Jun Xie, Kaer Huang, and Zhepeng Wang. Le3de2e solution for av2 2024 unified detection, tracking, and forecasting challenge. 1

[3] Cainan Davidson, Deva Ramanan, and Neehar Peri. Refav: Towards planning-centric scenario mining. *arXiv preprint arXiv:2505.20981*, 2025. 1

[4] Claude Team. Introducing claude 4. *https://www.anthropic.com/news/claude-4*, 2024. 1, 2

[5] Grok Team. Grok. *https://x.ai/grok*, 2024. 2

[6] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1, 2

[7] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023. 1