TRAJTOK: WHAT MAKES FOR A GOOD TRAJECTORY TOKENIZER IN BEHAVIOR GENERATION?

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior generation in autonomous driving aims to simulate dynamic driving scenarios from recorded driving logs. A popular approach is to apply next-token-prediction with discrete trajectory tokenization. In this work, we explore what makes a good trajectory tokenizer from the perspective of logged data usage. We first analyze the four properties (coverage, utilization, symmetry and robustness) of vocabularies of data-driven and rule-based trajectory tokenizers and their impact on performance and generalization. Data-driven tokenizers often build vocabularies with better utilization but suffer from insufficient coverage and sensitivity to noise, while rule-based methods have better coverage but contain too many useless tokens. With these insights, we propose TrajTok, a trajectory tokenizer that combines the two methods with rule-based vocabulary candidate setup and data-driven filtering and selection processes. The tokenizer has balanced coverage and utilization as well as good symmetry and robustness. Furthermore, we propose a spatial-aware label smoothing method for the cross-entropy loss to better model the similarities between the trajectory tokens. Our method wins first place in the 2025 Waymo Open Sim Agents Challenge.

1 INTRODUCTION

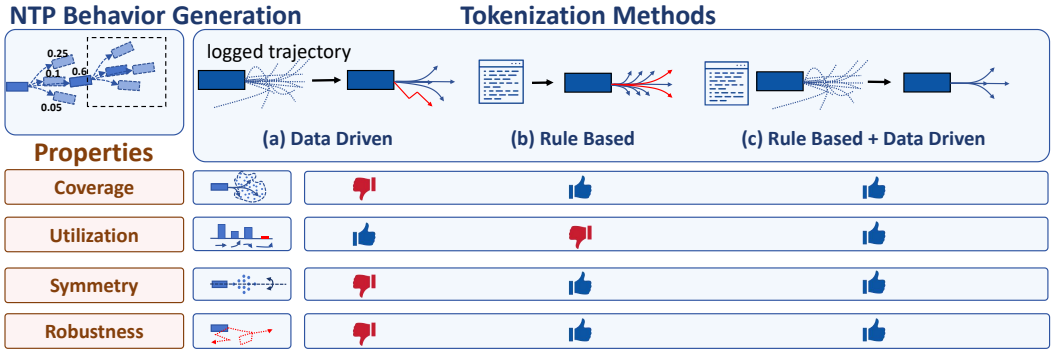

Figure 1: **Comparison of tokenizers with different logged data usage in NTP behavior generation model.** Combining data-driven and rule-based method, the proposed TrajTok balances coverage and utilization while has well symmetry and robustness.

Behavior generation aims to simulate realistic and dynamic chains of actions using logged data. In autonomous driving, behavior generation is a key component in simulators for data collection and evaluation. It typically takes the historical trajectories of agents and environmental information as inputs and generates future multi-agent trajectories auto-regressively (Chen et al., 2024).

Recently, inspired by large language models (LLMs) (Floridi & Chiriatti, 2020; Touvron et al., 2023), a series of behavior generation models adopt the next-token-prediction (NTP) paradigm (Philion et al., 2023; Seff et al., 2023; Wu et al., 2024; Zhao et al., 2024), as shown in Fig. 1 (upper left). These models build a vocabulary, a finite set of trajectories that tries to model the continuous distribution space with trajectory tokenizers. With the trajectory vocabulary, these models transform the

complex trajectory distribution in continuous space into a simpler discrete space for easier multi-modal behavior handling and model fitting.

Due to the low-dimensional nature of trajectories, various methods can be applied for trajectory tokenization, including data-driven methods such as VQ-VAE (Van Den Oord et al., 2017), K-means (Arthur & Vassilvitskii, 2006) and K-disks (Philion et al., 2023), and rule-based methods such as gridding (Seff et al., 2023). However, which method and what properties lead to a good trajectory tokenizer remain unclear to the community.

In this paper, we study what makes a good trajectory tokenizer and how to use the logged data in the tokenization process. First, we analyze the properties of trajectory tokens, including coverage, utilization, symmetry, and robustness, as well as their impact on the performance and generalization of NTP behavior generation models. As shown in Fig. 1 (a), we find that data-driven tokenizers tend to produce vocabularies within the recorded distribution, resulting in high utilization but low coverage. Additionally, their sensitivity to noisy data results in erroneous vocabulary and the asymmetry affects generalization. In contrast, rule-based methods cover a wider region with good symmetry and robustness, which leads to better generalization. However, they rely solely on human priors without adapting to the real data distribution, resulting in vocabularies that contain many redundant trajectories that do not appear in reality and low utilization, as shown in Fig. 1 (b). Under the same vocabulary size, their performance is far inferior to data-driven methods.

Based on these analyses, we propose TrajTok, a tokenizer specifically designed for trajectories that combines data-driven and rule-based methods, as shown in Fig. 1 (c). It first sets up a primary vocabulary candidate with rules, then applies data-driven filtering and expansion. The former ensures the stability and symmetry of the vocabulary, while the latter balances coverage and utilization.

Furthermore, we study loss designs related to vocabularies. A few NTP models use cross-entropy loss with label smoothing (Szegedy et al., 2016) to reduce overfitting and improve generalization. The standard label smoothing mechanism assigns the same weight to all non-ground-truth tokens, ignoring their similarity to ground-truth ones. As the similarity of trajectories is closely related to spatial distance, we propose a spatial-aware label smoothing method that assigns different weights to tokens conditioned on their distances to the ground-truth one. This smoothing method improves the performance of NTP behavior generation models effectively.

Our method wins first place in the Waymo Open Sim Agent Challenge (WOSAC) (Montali et al., 2023) 2025 with a Realism Meta of 0.7852, with good generalization across datasets.

Our contributions are threefold:

- We conduct a detailed investigation of trajectory tokenization in NTP behavior generation from the perspective of logged data usage and analyze the coverage, utilization, symmetry, and robustness of data-driven and rule-based methods.
- We propose TrajTok, a plug-and-play trajectory tokenizer that combines rule-based and data-driven methods, which achieves state-of-the-art performance on the Waymo Open Sim Agent Challenge.
- We explore token-related loss design in NTP behavior generation models and propose a spatial-aware label smoothing method for the cross-entropy loss.

2 Related Work

2.1 Behavior Generation Models

The behavior generation task proposed by Waymo Open Sim Agent Challenge (WOSAC) (Montali et al., 2023) aims to generate next states (including position and heading) of agents step by step realistically conditioned on environmental information and historical steps. The traditional encoder-decoder architectures (Shi et al., 2022; Zhou et al., 2023) in motion prediction can be used in this task, but they suffer from low data utilization and significant out-of-distribution (OOD) problems in the auto-regressive generation process Zhou et al. (2024). Inspired by large language models (LLM), most recent works adapt the next-token-prediction (NTP) paradigm. Trajeglish (Philion et al., 2023) first models the behavior generation task as discrete NTP with a data-driven tokenizer

K-disks. SMART (Wu et al., 2024) introduces a map discrete tokenizer and further analyzes the scalability of the architecture, while KiGRAS (Zhao et al., 2024) factorizes the driving scene in action space with kinematic transformations. Furthermore, CATK (Zhang et al., 2025) introduces a closed-loop fine-tuning strategy to further address the OOD problem.

Despite NTP-based models, diffusion models are another approach for behavior generation. These models focus on generating controllable (Zhong et al., 2022; Huang et al., 2024) or adversarial (Xie et al., 2024; Yin et al., 2024) behaviors with guidance, while their realism is much lower than NTP-based models on WOSAC.

2.2 Trajectory Tokenizers for Behavior Generation

Trajectory tokenizers transform the states of agents from continuous space to discrete space. Various tokenizers are adopted in discrete NTP behavior generation models, including VQ-VAE, K-means, K-disks and naive grid-based methods. Most of them are long-existing general approaches and only the K-disk proposed by Trajeglish (Philion et al., 2023) is specifically designed for trajectory tokens. K-disks randomly select a state in logged data as a trajectory token and exclude all logged states within a certain distance from the selected states, then randomly select the next token in the remaining data. Therefore, it is more like a sampling algorithm rather than a clustering algorithm. Grid-based methods such as that used in MotionLM (Seff et al., 2023) simply select the token with a uniform distribution in a pre-defined region of each state component.

Varying from deep-learning methods, clustering or sampling algorithms and rule-based methods, these tokenizers rely on logged trajectory data at different levels. However, there is a lack of comprehensive analysis of these tokenizers and the data usage preference behind them. Trajeglish (Philion et al., 2023) compares some tokenizers only in terms of discretization error, without studying their impacts on the performance of behavior generation models.

3 Methods

3.1 Problem Formulation

The behavior generation task can be defined as follows: Given an initial scene, including the HD map $\mathcal{M}$ and the past T_h states of all agents $\{\mathcal{S}_{-T_h}, ..., \mathcal{S}_0\}$, the goal is to generate the states of all agents at each future time step within T_f steps, i.e., $\{\mathcal{S}_1, ..., \mathcal{S}_{T_f}\}$.

We focus on the discrete NTP behavior generation models that output trajectories as actions, such as SMART (Wu et al., 2024) and Trajeglish (Philion et al., 2023). We denote the model as $\mathcal{N}_{\theta,V}$ with trainable parameters θ and trajectory vocabulary $\mathcal{V} = \{c_1, c_2, ..., c_{|\mathcal{V}|}\}$. Each trajectory token $c_i \in \mathbb{R}^{L\times 3}$ consists of L points with (x, y, yaw) in the agent-centric coordinate. We denote the number of all agents as N_A, and the model outputs the trajectory tokens $a_t \in \mathbb{R}^{N_A\times L\times 3}$ for each agent in the interval L as:

$$a_t = \mathcal{N}_{\theta,\mathcal{V}}(\mathcal{S}_{-T_h:t\times L}, \mathcal{M}) \tag{1}$$

where $t \in \{0, 1, 2, ..., (T_f//L)\}$ is the end timestamp for each interval. The output trajectory token for each agent is selected from the vocabulary, i.e. $a_t^{(i)} \in \mathcal{V}$. Then, a coordinate transform f is applied to calculate states in the next interval:

$$\mathcal{S}_{t\times L+1:t\times(L+1)+1} = f(a_t, \mathcal{S}_t) \tag{2}$$

The tokenizer $\mathrm{Tok}()$ generates the vocabulary $\mathcal{V}$. Pure rule-based tokenizers only use predefined hyper-parameters θ_{tok}, such as the vocabulary size and sample range as input, i.e.

$$\mathcal{V}_{\text{rule\_based}} = \mathrm{Tok}(\theta_{\text{tok}}) \tag{3}$$

Data-driven tokenizers generate the vocabulary with a dataset $\mathcal{D}$, i.e.

$$\mathcal{V}_{\text{data\_driven}} = \mathrm{Tok}(\theta_{\text{tok}}, \mathcal{D}) \tag{4}$$

3.2 TrajTok

As Fig. 2 shows, TrajTok generates the trajectory vocabulary through the following steps:

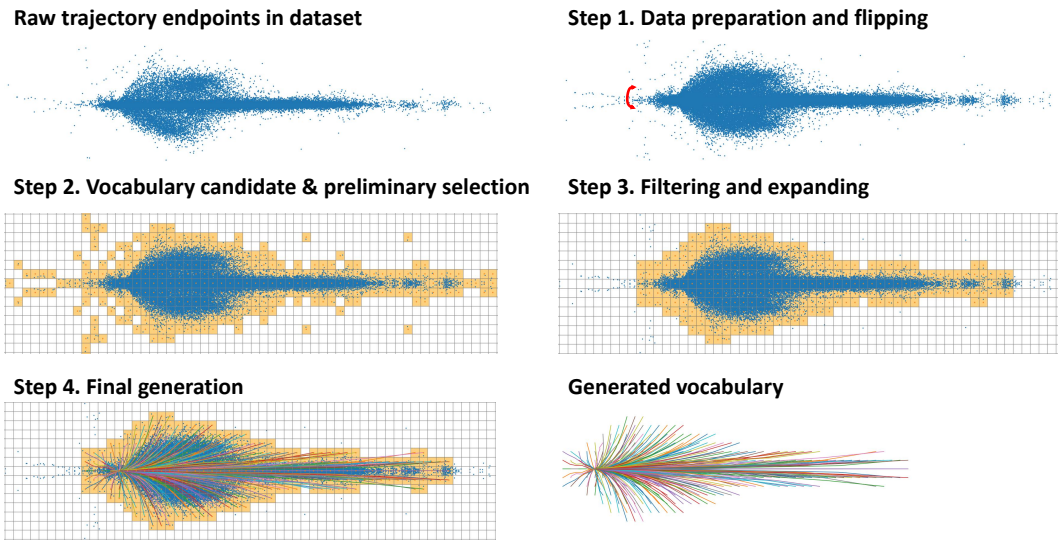

Figure 2: **The vocabulary generation process of TrajTok.** The blue points are endpoints of logged trajectories and the yellow grids are selected in each process.

Step 1: Data preparation and flipping. First, all valid trajectories of length L in the dataset are extracted and normalized to the agent-centric coordinate system. We denote the normalized trajectories as $D \in \mathbb{R}^{N_D \times L \times 3}$, where N_D is the number of trajectories. Then, the trajectories are flipped along the x-axis and concatenated with original trajectories, i.e.

$$\widetilde{D} = \text{Concatenate}(D, \text{Flip}(D)) \tag{5}$$

The flipping operation ensures the symmetry of trajectories, which results in symmetric tokens along the x-axis.

Step 2: Rule-based vocabulary candidate setup and preliminary selection. The vocabulary candidate is set up with a grid. Given ranges x_{min}, x_{max}, y_{min}, y_{max} and intervals x_{interval}, y_{interval}, the size of the grid is $(H, W) = (\frac{y_{\text{max}} - y_{\text{min}}}{y_{\text{interval}}}, \frac{x_{\text{max}} - x_{\text{min}}}{x_{\text{interval}}})$. Each trajectory in flipped data $\widetilde{D}$ is associated with a cell if its endpoint falls within the cell. We denote $\hat{D}_{ij} \in \mathbb{R}^{N_{ij}^{\text{traj}} \times L \times 3}$ as all trajectories associated with cell (i, j) and N_{ij}^{traj} as the number of these trajectories for each grid. For preliminary selection, a grid is marked as valid if N_{ij}^{traj} is larger than a threshold s_p. The validity binary map B is set through

$$B_{ij} = \mathbb{1}_{[N_{ij}^{\text{traj}} \geq s_p]} \tag{6}$$

Step 3: Data-driven filtering and expanding. The preliminary selected grids are within the distribution of logged data, and are sensitive to noise, as shown in Fig 2. To cover more possible trajectories in reality and improve robustness, filtering and expanding are then applied. For a grid (i, j) on the map, we compute the number of its selected neighbors within the distance of k, i.e.

$$N_{ij}^{\text{vb}} = \sum_{m=i-k}^{i+k} \sum_{n=j-k}^{j+k} B_{mn} \tag{7}$$

Then we add an unselected grid from step 2 if the number of its selected neighbors N_{ij}^{vb} is larger than the threshold s_a, and remove a selected grid if N_{ij}^{vb} is less than the threshold s_r through

$$\hat{B}_{ij} = \begin{cases} 1 & \text{if} \quad B_{ij} = 0 \quad \text{and} \quad N_{ij}^{\text{vb}} \geq s_a \\ 0 & \text{if} \quad B_{ij} = 1 \quad \text{and} \quad N_{ij}^{\text{vb}} \leq s_r \end{cases} \tag{8}$$

Step 4: Final generation. Finally, the trajectory vocabulary is generated for each selected grid. The average of logged data trajectories is used if there are trajectories associated with the cell.

$$V_1 = \{ \frac{\sum_{m=0}^{N_{ij}^{\text{traj}}} \hat{D}_{ijm}}{N_{ij}^{\text{traj}}} \quad | \quad i \in [0, W] \wedge j \in [0, H] \wedge \hat{B}_{ij} = 1 \wedge N_{ij}^{\text{traj}} > 0 \} \tag{9}$$

If not, the vocabulary is generated with curve interpolation from the origin point to the center of the grid p_{ij}. This happens if the cell is set valid in the expanding process. The yaw of the endpoint r_{ij} is estimated from the trajectories in nearby grids.

$$V_2 = \{\text{Curve\_Interp}(0, p_{ij}, r_{ij}, L) \quad | \quad i \in [0, W] \wedge j \in [0, H] \wedge \hat{B}_{ij} = 1 \wedge N_{ij}^{\text{traj}} = 0\} \tag{10}$$

The final vocabulary is:

$$V_{\text{TrajTok}} = V_1 \cup V_2 \tag{11}$$

3.3 Spatial-Aware Label Smoothing

NTP models often use cross-entropy loss with label smoothing (Szegedy et al., 2016) to alleviate overfitting and improve generalization. Standard label smoothing assigns the same probability to each non-ground-truth label. Denoting the index of the ground-truth label as j and the parameter of label smoothing as ε, the target probability y for each label i is calculated as:

$$y_i = \begin{cases} 1-\varepsilon & \text{if} \quad i = j \\ \frac{\varepsilon}{|\mathcal{V}|} & \text{if} \quad i \neq j \end{cases} \tag{12}$$

Under this mechanism, the loss is the same for predicting a trajectory token from the ground truth and one next to the ground truth. However, the former impacts the performance of behavior generation more significantly. Thus, we hope the model can be more tolerant of tokens that are spatially close to the ground-truth, while rejecting tokens that are far away. We calculate the average error between each token trajectory and the ground-truth trajectory, and assign the target probability inversely proportional to the square of the error:

$$k_i = \frac{1}{||\mathbf{c_i} - \mathbf{c_j}||^2} \tag{13}$$

$$y_i = \begin{cases} 1-\varepsilon & \text{if} \quad i = j \\ \frac{\varepsilon k_i}{\sum_{m=0, m\neq j}^{|\mathcal{V}|} k_m} & \text{if} \quad i \neq j \end{cases} \tag{14}$$

This spatial-aware label smoothing can both reduce errors and improve generalization.

4 Experiments

4.1 Experimental Setup

Dataset. We conduct experiments on the Waymo Open Motion Dataset (WOMD) (Ettinger et al., 2021). There are 486,995/44,097/44,920 scenarios in the training/validation/test set. Each scenario contains 9 seconds of agent states with an interval of 0.1s and a high-definition map. The behavior generation task is to generate the states in the future 8 seconds of up to 128 agents with states in the past 1 second in an autoregressive manner.

Metrics. We use the metrics in the Waymo Open Sim Agents Challenge (Montali et al., 2023). The main metric is the Realism Meta, which is composed of the Kinematic, Interactive, and Map-based metrics. Notably, it takes several days to compute the metrics on the whole validation set with the official code. Previous works (Zhang et al., 2025) conduct ablation studies with evaluation on a small subset, which may lead to biased results. We improve the code and decrease the computation time to less than 2 hours while strictly aligning the protocol and results with the official version. With the efficient evaluation tool, all our ablation studies are evaluated on the whole validation set.

4.2 Results

WOSAC leaderboard. Table 1 lists the results of the top 15 entries in the 2025 Waymo Open Sim Agents Challenge. TrajTok wins the first place in the Waymo Open Sim Agent Challenge 2025 with a Realism Meta metric of 0.7852. It also reaches state-of-the-art performance on Map-based metrics of 0.9207 and competitive performance on other metrics. Without any fine-tuning processes, our approach has comparable or superior performance compared to other methods that are also

Table 1: **Waymo Open Sim Agents Challenge leaderboard 2025.** Top 15 entries in the Submission Period are presented.

Method	Realism Meta ↑	Kinematic↑	Interactive ↑	Map-based ↑	minADE ↓
SMART-R1	0.7855	0.4940	0.8109	0.9194	1.2990
TrajTok (Ours)	0.7852	0.4887	0.8116	0.9207	1.3179
unimotion	0.7851	0.4943	0.8105	0.9187	1.3036
SMART-tiny-CLSFT (Zhang et al., 2025)	0.7846	0.4931	0.8106	0.9177	1.3065
SMART-tiny-RLFTSim	0.7844	0.4893	0.8128	0.9164	1.3470
comBOT	0.7837	0.4899	0.8102	0.9175	1.3687
AgentFormer	0.7836	0.4906	0.8103	0.9167	1.3422
UniMM (Lin et al., 2025)	0.7829	0.4914	0.8089	0.9161	1.2949
R1Sim	0.7827	0.4894	0.8105	0.9147	1.3593
SimFormer	0.7820	0.4920	0.8060	0.9167	1.3221
SMART-tiny-RLFT	0.7815	0.4853	0.8107	0.9133	1.4266
SMART_topk32	0.7814	0.4854	0.8089	0.9153	1.3931
SMART-tiny-RLFT	0.7780	0.4799	0.8070	0.9109	1.6388
llm2ad	0.7779	0.4846	0.8048	0.9109	1.2827
UniTFormer	0.7776	0.4892	0.7997	0.9140	1.3592

Table 2: **Performance of SMART model with different tokenizers on validation split of WOMD.** The models are trained on a random 20% subset of the training set and the metrics are of WOSAC 2024 version.

Tokenizer	Realism Meta ↑	Kinematic↑	Interactive ↑	Map-based ↑	minADE ↓
VQ-VAE	0.7596	0.4629	0.8101	0.8642	1.3982
K-means	0.7476	0.4375	0.7903	0.8635	1.4797
K-disks	0.7584	0.4602	0.8004	0.8748	1.3532
Grid	0.7527	0.4121	0.8099	0.8737	1.4137
TrajTok	0.7702	0.4867	0.8132	0.8769	1.3428

Table 3: **Performance of K-disks and TrajTok across datasets.**

Tokenizer	Logged Dataset	Realism Meta ↑	minADE ↓
K-disks	Waymo	0.7584	1.3537
K-disks	nuScenes	0.7350 (-0.0234)	1.4074
TrajTok	Waymo	0.7702	1.3428
TrajTok	nuScenes	0.7641 (-0.0061)	1.3681

Table 4: **Performance of K-disks and TrajTok with different sizes of logged data.**

Tokenizer	Logged Data Size	Realism Meta ↑	minADE ↓
K-disks	10^7	0.7584	1.3537
K-disks	10^6	0.7539 (-0.0045)	1.3628
K-disks	10^5	0.7442 (-0.0142)	1.3696
TrajTok	10^7	0.7702	1.3428
TrajTok	10^6	0.7691 (-0.0011)	1.3445
TrajTok	10^5	0.7675 (-0.0027)	1.3511

developed on the SMART-tiny model but need fine-tuning, such as SMART-tiny-CLSFT (Zhang et al., 2025).

Comparison with other tokenizers. We compare TrajTok with other tokenizers, including VQ-VAE, K-means, K-disks and grid method on the SMART (Wu et al., 2024) model, as shown in Table 2. The size of vocabularies of each tokenizer is 2000 for fair comparison. Results show that our method reaches the best performance among all tokenizers that are adopted in behavior generation models.

Generalization. The generalization of the tokenizer is reflected in its ability to handle diverse real-world data with vocabularies derived from any data record. We investigate generalization of TrajTok through two experimental setups: ❶ Building vocabularies with nuScenes (Caesar et al., 2020) data and training and evaluating on WOMD to evaluate the tokenizer's generalization across datasets. The results are shown in Table. 3. ❷ Building vocabularies with a small subset of WOMD to evaluate the tokenizer's ability to infer the overall distribution from few sampled data. The results are shown in Table. 4.

Qualitative results. Fig. 3 shows the vehicle trajectory vocabularies generated by different tokenizers. The data-driven methods K-means and K-disks produce trajectory tokens that are asymmetric and contain kinematically implausible noisy trajectories. The purely rule-based grid method generates many invalid trajectories (e.g., lateral shifts of over 1 meter within 0.5 seconds) and has lower density at the same vocabulary size. In contrast, TrajTok demonstrates good symmetry and stability, with generated trajectories that largely adhere to kinematic principles. Fig. 4 illustrates the behavior generated by TrajTok in two scenarios. The first is a busy intersection scene where

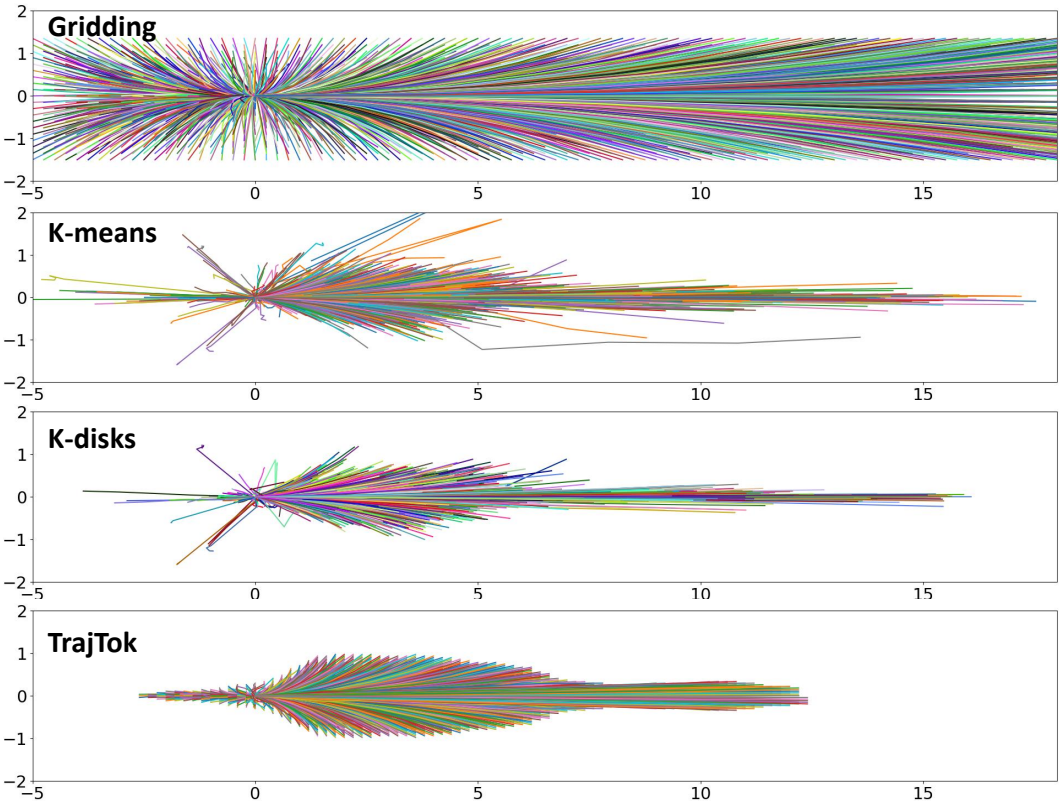

Figure 3: **Trajectory vocabularies from TrajTok and other tokenizers.** Each colored line represents a trajectory token within 0.5 seconds in agent-centric coordinate system. The size of all four vocabularies is 2000.

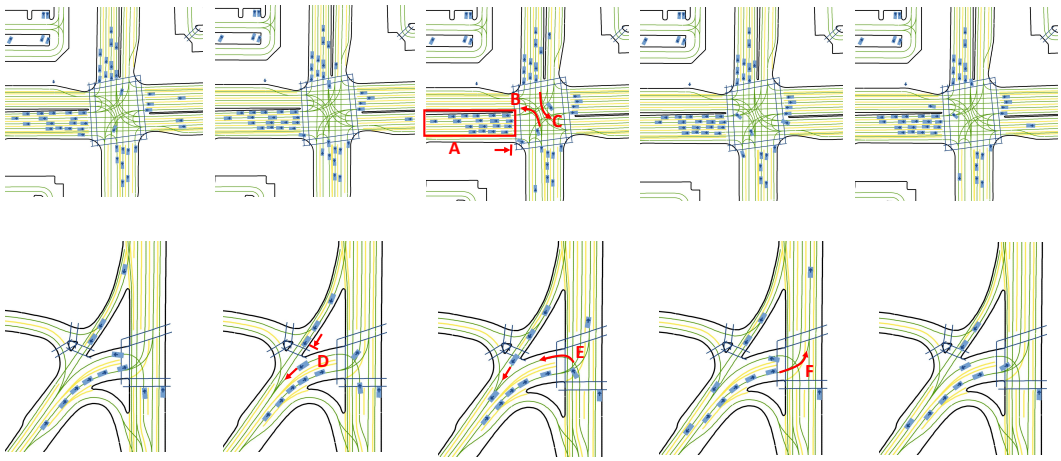

Figure 4: **Generated behavior in two scenarios.**

traffic in direction A is queued, maintaining appropriate distances between vehicles, while vehicles in directions B and C make left turns. This demonstrates TrajTok's capability to handle interactions among a large number of vehicles precisely. The second scenario features an atypical intersection where vehicles in direction E need to turn left and backward, alternating passage with vehicles from directions D and F. Benefiting from its superior coverage, TrajTok effectively manages vehicle interactions in uncommon scenarios.

4.3 Ablation

Vocabulary Size. We ablate the vocabulary size in Fig. 5. Increasing the vocabulary size improves the ability to represent complex distributions but may lead to model underfitting, which is also

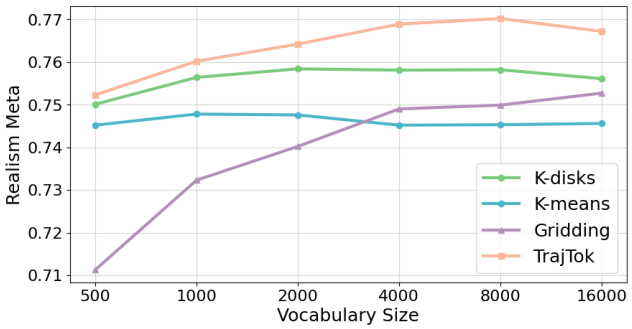

Figure 5: **Ablation of vocabulary size of vehicles.**

Table 5: **Ablation of Spatial-Aware label smoothing.**

Tokenizer	Label Smoothing Type	Realism Meta↑	minADE ↓
K-disks	default	0.7443	1.4230
K-disks	spatial-aware	0.7584	1.3537
TrajTok	default	0.7597	1.3797
TrajTok	spatial-aware	0.7702	1.3428

observed in LLMs (Tao et al., 2024). Rule-based gridding methods require a larger vocabulary size to reach optimal performance, while data-driven methods need a smaller size. The optimal vocabulary size for TrajTok lies between data-driven and rule-based methods. Additionally, TrajTok achieves the best performance across different vocabulary sizes.

Spatial-Aware label smoothing. Table 5 presents the ablation study for Spatial-Aware Label Smoothing. Whether using K-disks or TrajTok as tokenizers, Spatial-Aware Label Smoothing improves performance compared to standard label smoothing.

4.4 Analysis and Discussions

Coverage. Qualitatively, Fig. 6 plots the endpoints of real trajectories and vocabulary trajectories. Data-driven methods generate vocabularies within the distribution of real trajectories and leave several insufficiently covered regions. The vocabulary generated by rule-based methods extends far beyond the real distribution range. In contrast, the vocabulary of TrajTok covers most of the valued trajectories. Quantitatively, Fig. 7 shows the missing rate of tokenization at different distances, which is defined as the proportion of trajectories with discretization errors larger than a given distance. TrajTok has a better missing rate at larger distances. It shows that TrajTok has appropriate coverage and is better at dealing with the long tail effect in tokenization.

Utilization. Fig. 8 shows the frequency of each token in the vocabulary. Rule-based gridding methods have a large number of tokens with zero frequency. Although the frequency of uncommon

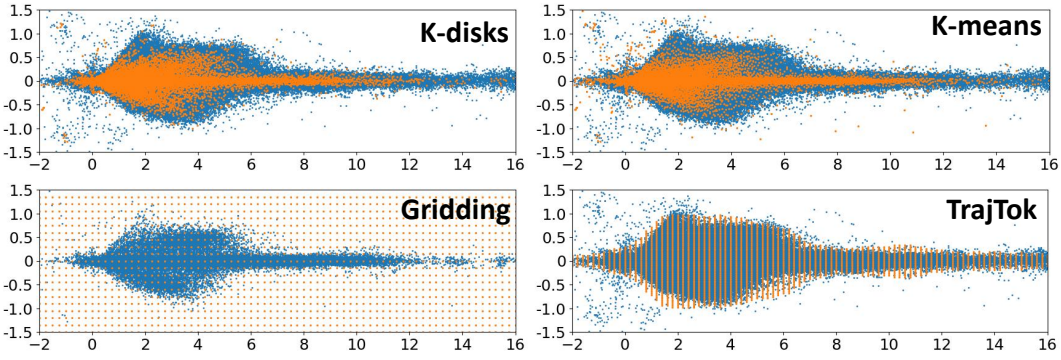

Figure 6: **The distribution of logged trajectories and vocabularies of vehicles.** Orange dots indicate the endpoints of logged trajectories, while blue dots indicate those of vocabularies.

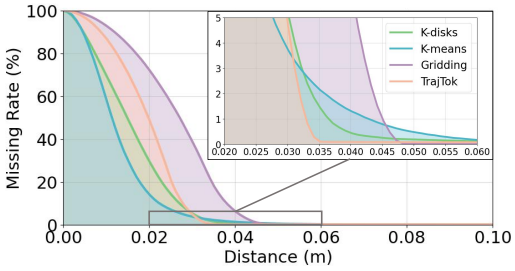

Figure 7: **The missing rate of tokenization for cyclists.**

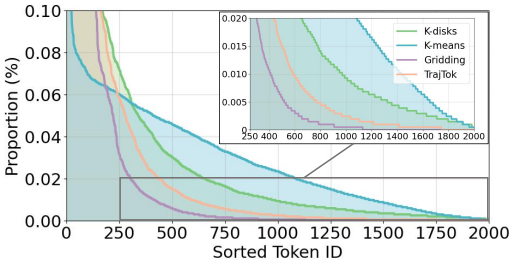

Figure 8: **The frequency of vocabularies for cyclists.**

Table 6: **Symmetric tokenizers have better performance.**

Tokenizer	Realism Meta↑	minADE ↓
K-disks w/ symmetry	0.7610	1.3541
K-disks w/o symmetry	0.7584	1.3532
K-means w/ symmetry	0.7526	1.3519
K-means w/o symmetry	0.7476	1.3597
TrajTok w/ symmetry	0.7702	1.3428
TrajTok w/o symmetry	0.7670	1.3611

Table 7: **The average discretization error of tokenizers for vehicles.** The vocabulary size is 2000.

Tokenizer	Realism Meta	average discretization error (m)
K-means	0.7476	0.0224
K-disks	0.7584	0.0204
Grid	0.7527	0.0813
TrajTok	0.7702	0.0520

tokens of TrajTok is lower than that of data-driven methods, most of them match with several logged trajectories. With the increase of tokens for rare trajectories, there is a trade-off between coverage and utilization. TrajTok balances these two properties.

Symmetry. Due to the symmetry of vehicle kinematic models, the diversity of real-world traffic scenarios, and the symmetry of vehicle kinematic models, the possible distribution boundary of trajectories in reality is nearly symmetric. In the experiments, we remove the flipping operation in the TrajTok generation process, resulting in a performance decline. Data-driven methods, due to the randomness of data distribution and algorithms, can not guarantee the generation of a symmetric vocabulary. We flip the trajectories below the x-axis to above the x-axis, generate a vocabulary of half size on the flipped trajectories, and then flip it back to obtain the complete vocabulary. The vocabularies generated in this manner show improved performance, as shown in Table 6. Both aspects of the experiments demonstrate the importance of symmetry.

Robustness. Fig. 3 and Fig. 6 show that TrajTok is less sensitive to noise. In contrast, K-disks may directly add noisy trajectories to the vocabulary during random sampling. K-means may also pick outliers since they have large distances to other samples.

Does lower average discretization error lead to a better tokenizer? Average discretization error is used to evaluate the performance of a tokenizer in previous work (Philion et al., 2023). However, as shown in Table 7, the average discretization error of K-disks is lower than that of TrajTok, yet its performance is inferior to TrajTok. The indicator cannot entirely reflect the distance between the vocabulary and distribution in the real world. For example, it cannot well indicate the long-tail effect in Fig. 7. When the average discretization error is already low, the performance of the tokenizer is influenced mostly by other factors.

5 Conclusion

In this paper, we analyze data-driven and rule-based tokenizers from four perspectives including coverage, utilization, symmetry, and robustness. Based on the analysis, we combine data-driven and rule-based methods and introduce TrajTok. Additionally, we propose a spatial-aware label smoothing method to better model the similarities between the trajectory tokens. Experiments demonstrate the effectiveness of our methods.

6 Ethics Statement

The research conducted in the paper conforms with the ICLR Code of Ethics.

7 Reproducibility Statement

We describe the proposed tokenizer and loss in Sec. 3 and implementation details in Appendix A.1. The datasets are public accessible. The code will be open-sourced for reproduction.

References

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Di Chen, Meixin Zhu, Hao Yang, Xuesong Wang, and Yinhai Wang. Data-driven traffic simulation: A comprehensive review. *IEEE Transactions on Intelligent Vehicles*, 9(4):4730–4748, 2024.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9710–9719, 2021.

Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and machines*, 30(4):681–694, 2020.

Zhiyu Huang, Zixu Zhang, Ameya Vaidya, Yuxiao Chen, Chen Lv, and Jaime Fernández Fisac. Versatile scene-consistent traffic scenario generation as optimization with diffusion. *arXiv preprint arXiv:2404.02524*, 3, 2024.

Longzhong Lin, Xuewu Lin, Kechun Xu, Haojian Lu, Lichao Huang, Rong Xiong, and Yue Wang. Revisit mixture models for multi-agent simulation: Experimental study within a unified framework, 2025. URL https://arxiv.org/abs/2501.17015.

Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, et al. The waymo open sim agents challenge. *Advances in Neural Information Processing Systems*, 36:59151–59171, 2023.

Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction. *arXiv preprint arXiv:2312.04535*, 2023.

Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8579–8590, 2023.

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543, 2022.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *Advances in Neural Information Processing Systems*, 37:114147–114179, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction. *Advances in Neural Information Processing Systems*, 37: 114048–114071, 2024.

Yuting Xie, Xianda Guo, Cong Wang, Kunhua Liu, and Long Chen. Advdiffuser: Generating adversarial safety-critical driving scenarios via guided diffusion. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9983–9989. IEEE, 2024.

Yuan Yin, Pegah Khayatan, Éloi Zablocki, Alexandre Boulch, and Matthieu Cord. Regents: Real-world safety-critical driving scenario generation made stable. In *European Conference on Computer Vision*, pp. 262–276. Springer, 2024.

Zhejun Zhang, Peter Karkus, Maximilian Igl, Wenhao Ding, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Closed-loop supervised fine-tuning of tokenized traffic models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jianbo Zhao, Jiaheng Zhuang, Qibin Zhou, Taiyu Ban, Ziyao Xu, Hangning Zhou, Junhe Wang, Guoan Wang, Zhiheng Li, and Bin Li. Kigras: Kinematic-driven generative model for realistic agent simulation. *IEEE Robotics and Automation Letters*, 2024.

Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. *arXiv preprint arXiv:2210.17366*, 2022.

Zikang Zhou, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction. *arXiv preprint arXiv:2306.10508*, 2023.

Zikang Zhou, HU Haibo, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. *Advances in Neural Information Processing Systems*, 37:79597–79617, 2024.

A Appendix

A.1 Implementation Details

Model Settings. We use the SMART-tiny (Wu et al., 2024) model as base NTP behavior generation model for experiments with TrajTok. Follow original settings, the the number of decoder layers (including Map-Agent Cross-Attention Layer, Agent Interaction Self-Attention Layer and Temporal Self-Attention Layer) is 6 and the hidden dim is 128. The interval L is set to 5 and the re-plan frequency is 2 Hz, which means the model predict the 0.5-second trajectory at 10Hz every 0.5s. The original model build different trajectory vocabularies for each type of agents (vehicle, bicycle and pedestrian) but use the same head to predict classification logits for all types. We use separate prediction heads instead and set their output dim the same as the vocabulary size each. For spatial-aware label smoothing, the total target probability of all non-ground-truth labels ε is 0.1, which is the same as standard label smoothing used in original model.

Tokenizer. For TrajTok, we set the grid range and interval for each type of agents as Table 8. We extract trajectories that last 0.5s from the Waymo Open Motion Dataset(WOMD). In the submit version, the sizes of vocabularies for vehicle, bicycle and pedestrian are 8040, 2798, 3001 separately.

Training Details. We train the model with 8×A100 80GB GPUs for 32 epochs on the training split of the WOMD with the AdamW optimizer. The total batch size is 48. The initial learning rate to 5×10^{-4} and is decayed to 5×10^{-6} based on the cosine annealing schedule.

Agent Type	x_{min}	x_{max}	$x_{interval}$	y_{min}	y_{max}	$y_{interval}$
Vehicle	-5	20	0.1	-1.5	4.5	0.05
Bicycle	-1	8	0.05	-1	1	0.05
Pedestrian	-1.5	4.5	0.05	-2	2	0.05

Table 8: Detailed hyper-parameters of the submit version of TrajTok. The unit of all parameters is meter.

B LLM USAGE

LLMs are used in writing for improving grammar and correcting typos.