TrajAgent: An Agent Framework for Unified Trajectory Modelling

Anonymous Author(s)

ABSTRACT

Trajectory modeling, which includes research on trajectory data pattern mining and future prediction, has widespread applications in areas such as life services, urban transportation, and public administration. Numerous methods have been proposed to address specific problems within trajectory modelling. However, due to the heterogeneity of data and the diversity of trajectory tasks, achieving unified trajectory modelling remains an important yet challenging task. In this paper, we propose *TrajAgent*, a large language model-based agentic framework, to unify various trajectory modelling tasks. In *TrajAgent*, we first develop *UniEnv*, an execution environment with a unified data and model interface, to support the execution and training of various models. Building on *UniEnv*, we introduce *TAgent*, an agentic workflow designed for automatic trajectory modelling across various trajectory tasks. Specifically, we design *AutOpt*, a systematic optimization module within *TAgent*, to further improve the performance of the integrated model. With diverse trajectory tasks input in natural language, *TrajAgent* automatically generates competitive results via training and executing appropriate models. Extensive experiments on four tasks using four real-world datasets demonstrate the effectiveness of *TrajAgent* in unified trajectory modelling, achieving an average performance improvement of 15.43% over baseline methods.

ACM Reference Format:
Anonymous Author(s). 2018. TrajAgent: An Agent Framework for Unified Trajectory Modelling. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the rapid development of web services and mobile devices [4, 53], large-scale trajectory data, such as check-in data from social network [43], have been collected, greatly facilitating research in trajectory modelling. Trajectory modelling involves the processing, mining and prediction of trajectory data, with widespread applications in urban transportation, location services and public management. The typical areas of trajectory modelling [4, 11] can be classified into three main categories: trajectory classification [19], trajectory prediction [45], and trajectory generation [37]. Each category encompasses various sub-tasks; for instance, the trajectory prediction task can be further divided into next location prediction task [20], final destination prediction task [51], and travel time estimation task [34], among others. Given the huge value of trajectory modelling in diverse practical applications, various algorithms and models [11] have been proposed to address these tasks, particularly deep learning-based models in recent years. This has facilitated significant advancements in the field, with many tasks achieving a high level of performance.

However, existing methods are designed for specific tasks and datasets, making it difficult to share them across different tasks and data sources. For example, TrajFormer [19] is tailored for trajectory classification and cannot be applied in trajectory prediction or trajectory generation. Flash-back [42] is designed for sparse check-in trajectory prediction and is not suitable for dense GPS trajectory or road network-based trajectory modelling. In other words, due to the heterogeneity of application scenario and the diverse nature of trajectory data–varying in resolution, format and geographical regions–existing methods can only be applied in limited task with specific data for specific regions. Without a unified model, the deployment and application of these advanced models become costly and inefficient, hindering further development of trajectory modelling area. Therefore, developing a unified model for various trajectory modelling tasks is both emergent and fraught with significant challenges.

In last two years, the rapid development of large language models (LLMs) [23, 33] with extensive commonsense and powerful reasoning abilities presents great potential for solving challenging tasks, such as mathematics [27] and logic games [47]. Furthermore, LLM based agent [32, 40] have been proposed as a new paradigm, offering more robust and intelligent task-solving abilities. These agents have demonstrated stable and promising performance in tackling complex real-world tasks, such as web-navigation [22, 46, 55] and software development [12, 25, 44]. Based on the powerful understanding and reasoning abilities of LLM based agents, researchers explore the potential of utilizing LLM based agent unified modelling of diverse machine learning tasks [14, 29, 38, 52]. For example, HuggingGPT [29] utilize LLM as core manager to solve various machine learning tasks with existing AI models, VisionLLM [38] explores the unified modelling of vision tasks in different domains, Mora [50] designs a multi-agent framework for replicating Sora's functionalities with combining existing open source models.

Inspired by these works, we explore the potential of solving the unified trajectory modeling problem with the LLM based agent framework. However, designing LLM based agent for solving unified trajectory modelling problem is not easy due to the following challenges. Firstly, trajectory data and existing models for various trajectory modelling tasks are diverse and often inconsistent, which makes the integration and deployment of these models difficult. Secondly, the processing steps from raw trajectory data to final output results are lengthy and cumbersome [4], which introducing large action space for the planning and execution of whole process. Third, the performance of existing models highly rely on the data adaptation and parameter optimization and the direct usage of existing trained models are infeasible.

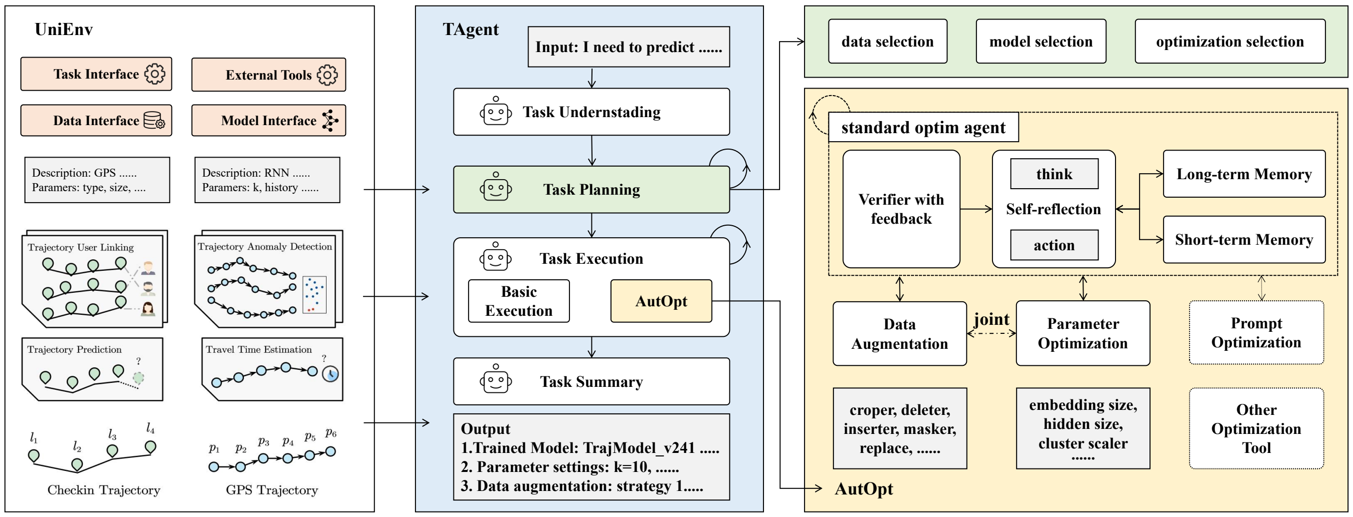


Figure 1: Framework of *TrajAgent*, including *UniEnv* with unified data and model interface for running experiment, *TAgent* an agentic workflow for executing the whole trajectory modeling procedure, and *AutOpt* for the model enhancement and adaptation on selected data.

In this paper, we propose *TrajAgent*, a systematic agent framework for unified trajectory modelling and analytics. In *TrajAgent*, we first design a unified environment, *UniEnv*, for processing diverse trajectory data and preparing unified running environment for various trajectory modelling tasks. In *UniEnv*, we define unified data and model interface to support the easy execution of various methods for trajectory modelling. Then, we design an agentic workflow, *TAgent*, for automatic planning and execution of trajectory modelling tasks. In *TAgent*, we decompose the task workflow into four steps, including task understanding, task planning, task execution and task summary. In each step, we design a expert agent to conduct specific operations of it. Specifically, we design *AutOpt* to enhancing the task execution module for adapting the performance of models with diverse automatic optimization mechanisms, including data augmentation, parameter optimization and prompt optimization. In summary, our contributions are as follows,

- To the best of our knowledge, *TrajAgent* is the first attempt to employ an LLM-based agentic framework for unified trajectory modeling across diverse trajectory data and various trajectory tasks.
- In *TrajAgent*, we design a agentic workflow to decompose the whole trajectory modelling task into several sub-tasks, all carried out within a unified environment *UniEnv*. Finally, we develop a comprehensive model optimization module to further enhance the model performance on targeted data.
- Extensive experiments on four representative trajectory modelling tasks across four real-world datasets demonstrate the effectiveness of proposed framework, which the optimized model achieving an average performance gain of 15.43%. The codes and data can be accessed via https://anonymous.4open.science/r/TrajAgent-63CC

2 METHODS

Figure 1 presents the whole framework of *TrajAgent*. It contains three components: 1) *UniEnv*, an environment with unified data and model interface which supports the following module to execute and train various models; 2) *TAgent*, an agentic workflow for completing the diverse trajectory modelling task automatically, which accept requirements in forms of language as input and directly output adapted models; 3) *AutOpt*, an extra optimization module for *TAgent* to enhance the performance of specific model with data and parameter adaptation. Details of each component are introduced as follows.

2.1 UniEnv

As shown in the left of Figure 1, *UniEnv* is a comprehensive and integrated environment that bridges trajectory data, tasks, and models, providing a foundational platform for trajectory modelling and analysis. It is designed to support the entire lifecycle of trajectory modelling workflows, from data preparation to task execution and model optimization. *UniEnv* comprises four key components: a rich set of *datasets* accompanied by processing tools, a comprehensive *task collection* that defines and manages various task types, a extensive *model library* with available source code, and an external *tools pool* for extending the capabilities of *TrajAgent*. Each component is seamlessly connected through a unified interface, enabling agents to plan and execute trajectory modeling tasks with minimal complexity.

2.1.1 *Task Interface.*

We summarize the supported trajectory modelling tasks in *UniEnv* as follows.

- Trajectory prediction task: next location prediction and travel time estimation.
- Trajectory classification task: trajectory user linking and trajectory anomaly detection.

- Trajectory generation task: trajectory recovery and trajectory generation.

To enhance the clarity and effectiveness of understanding users' language queries, we provide a detailed language description for each task. This description helps extract precise task requirements from user queries and facilitates subsequent data and model selection processes.

2.1.2 *Data Interface.*

UniEnv supports two commonly used trajectory data formats, namely Checkin trajectory (i.e., sequence of visited POIs) and GPS trajectory (i.e., sequence of gps points). These datasets, which come from different cities and with distinct forms, are pre-processing through a standard pipeline that ensures compatibility across the system. Pre-processing steps are done by generated code scripts from LLMs, include data cleaning, normalization, format transformation, which are crucial for handling inconsistencies between real-world datasets and task models. After processing, we will add a description for each dataset to support efficient data selection in the subsequent stage.

2.1.3 *Model Interface.*

To support various model training in *TrajAgent*, we select at least one well-known model for each task and adapt them to match the running environments in *UniEnv*. Furthermore, we extract the semantic context from original paper of each model with the txyz.ai APIs [1] to generate detailed description of model. In this description, the verified data information and supported trajectory task information are recorded which supports the data and model selection in the subsequent trajectory planning stage of *TrajAgent*.

2.1.4 *External Tools.*

To extend the capabilities of *TrajAgent*, we collect several external tools in *UniEnv*, including paper context extraction tool txyz.ai, hyperparameter optimization tool optuna [2], processing and visualization tool for trajectory data movingpandas [3], open street map data processing tool osmnx [4].

Here, we also regard the LLM APIs utilized in the agentic workflow as one of the interface in the *UniEnv*, including the ChatGPT API [5] and DeepInfra [6] for open-source LLMs.

2.2 TAgent

As shown in the middle of Figure 1, the architecture of TAgent is organized into four key modules: Task Understanding, Task Planning, Task Optimization, and Task Summary, which form an automated processing chain from user query to final result, eliminating the need for human-in-the-loop. Specifically, the task understanding module first receives user instructions in natural language form, and analyzes and identifies the type, name, and other key information of the tasks involved. Then, the task planning module will plan for the identified task, including dataset matching and model selection. Next, the task execution module executes the planning task and cooperate with the additional model optimization module *AutOpt* to further improve the task performance from both data augmentation and parameter selection perspectives. Last, the task summary module generate an analytical report of the task based on historical interactions and decisions of TAgent. Following the common practice[30], each module in *TAgent* can be regarded as a small agent, which consists of a function module for executing its core function, a memory for recording the history interaction, and a reflection module for learning practice from the memory.

2.2.1 *Task Understanding Module.*

As the first module of *TAgent*, task understanding module is designed to interact with user and extract detailed task information to launch subsequent stages. Given the user query, understanding module recognize the potential task name from it with the predefined supported tasks as additional input. If users ask for the out-of-scope tasks which has not been supported in the *UniEnv*, we will directly recommend user to select task from the supported list.

2.2.2 *Task Planning Module.*

Follow the task understanding module is the planning module which is designed to generate the subsequent execution plan for efficient experiments of trajectory modelling. The input of the planning module is the task name and description from the undertanding module, the supported data and model with brief description from *UniEnv*. With the carefully designed prompt, the generated execution plan will contain the data name and model name for the given task, and also the detailed model optimization plan.

Due to the characteristics of different tasks and existing practice, not all the model optimization are necessary for each task. If possible, skipping the optimization step which is time-consuming and costly can accelerate the whole procedure without sacrificing performance. For example, while the data augmentation is very important for sparse check-in trajectory data, it is not so useful for dense GPS trajectory in many applications. Thus, we can generate simplified execution plan for dense GPS trajectory in some tasks to improve the success rate of the whole task and also reduce the cost. After generating the plan, it will start a simple execution step to verify the feasibility of the plan. Once any error occurs during the execution, e.g., the model name is wrong, the planning module will obtain the feedback from *UniEnv* and start to regenerate a new plan with the last plan as the failed history in its memory.

2.2.3 *Task Execution Module.*

Give the execution plan, the task execution module is responsible for invoking UniEnv to execute the experiment plan. In addition to the previously mentioned basic execution interface, another interface of this module is to call *AutOpt* module to complete the model optimization according to the optimization plan. For both interface, task execution module will give the feedback including error information for failed cases and performance metrics for success cases. Detailed designs of *AutOpt* are introduced in the following section 2.3.

2.2.4 *Task Summary Module.*

After the execution module, we design a task summary module to analyze the execution records to generate the optimization summary of the task. The summary contains the optimization path during the experiment and the final optimization result for the given task. User can also directly utilize the optimized model from the experiments via APIs for further applications.

[1]https://www.txyz.ai/
[2]https://optuna.org/
[3]https://github.com/movingpandas/movingpandas
[4]https://github.com/gboeing/osmnx
[5]https://openai.com/
[6]https://deepinfra.com

2.3 AutOpt

Due to the geospatial heterogeneity and the diversity of trajectory data, the trajectory models usually cannot be directly transferred between data and regions. In other words, for various data in different region, the model needs to be trained from scratch. Thus, the sufficient optimization of various models becomes emergent. In *TrajAgent*, we design *AutOpt* module to complete this task. As shown in the right part of Figure 1, *AutOpt* provides several optimization mechanisms: data augmentation, parameter optimization, joint optimization, prompt optimization and optimization via tools. Following Reflexion [30], we design optim agent for each kind of optimization mechanism. The standard optim agent utilizes the history operation and related results as the feedback to update its action in the next step. Specifically, it works as two stages, including "think" and "action". To support the "think then action", it builds a long-term memory for recording all experimental data and a short-term memory for historical actions. In the "think" stage, optim agent analyzes the long-term memory and meta information of experiment and generating the guidance of action in the short-term memory. In the "action" stage, optim agent analyze the results in short-term memory to generate the action. Different optimization mechanism utilize the same optim agent with different action space and optimization tips.

2.3.1 Data Augmentation. Based on the standard optim agent, we introduce the specific action space for data augmentation. Following practice from existing works [5, 48, 54], we define a fixed set with ten operators for data augmentation, e.g., insert, replace, split and so on. During the optimization, the operator set with simple description is provided to the optim agent, it can select any combination of operators with their simple parameters as the action. After action, optim agent obtain the feedback information, e.g., performance metrics, from *UniEnv* to continue update its memory and action.

2.3.2 Parameter Optimization. The action space of parameter optimization is defined based on the parameters of model itself. We define a parameter configuration file for each model, the optim agent reads the configuration file and generates code as the action to update the parameters in it. To better understand the meaning of each parameter, we add comments for each parameter in the file. While the action space is large, this kind of action space is flexible to adapt with any models.

2.3.3 Joint Optimization. Furthermore, we introduce the joint optimization mechanisms in *AutOpt* to improve the final performance. Due to the different working paradigms, the direct combination of two kinds of optimizations is unsuccessful. We designate the optimization order to prioritize data augmentation first, followed by parameter optimization. This means that once the performance of data augmentation stabilizes, the agent proceeds with parameter optimization. This procedure can be repeated a fixed number of times until it meets the stop criteria.

3 EXPERIMENTS

In this section, we first introduce the experimental settings, including the datasets, task models, and evaluation metrics. Next, we present the experimental results, covering the performance across multiple trajectory tasks as well as the effectiveness of agent execution. Following that, we perform an ablation study to assess the impact of different components within the proposed framework. Subsequently, we investigate the effect of key parameters for TrajAgent execution. Finally, we conclude with a case study to further illustrate the practical application of our approach.

3.1 Settings

3.1.1 Dataset.

- **Foursquare (FSQ)**: This dataset consists of 227,428 check-ins collected in New York City from 04/12/2012 to 02/16/2013, Each check-in is associated with a timestamp, GPS coordinates and corresponding venue-category.
- **Brightkite (BGK)**: This dataset contains 4,491,143 check-ins of 58,228 users collected from BrightKite website.
- **Porto**: This dataset contains 1.7 million taxi trajectories of 442 taxis running in Porto, Portugal from 01/07/2013 to 30/06/2014. Each trajectory corresponds to one completed trip record, with fields such as taxiID, timestamp and the sequence of GPS coordinates.
- **Chengdu**: This dataset contains GPS trajectory records of Chengdu from 01/11/2016 to 30/11/2016. Each record includes taxiID, timestamp, longitude and latitude, collected and released by Didi Chuxing.
- **UserQueries**: To verify the effectiveness of the whole system, we utilize self-instruct method [36] with 5 seed queries to generate 300 user queries as the experiment input.

More detailed information for the Check-in and GPS trajectory datasets are summarized in Table 1 and Table 2, respectively. Note that the raw datasets are typically city-scale and data-intensive. Loading them all into the TrajAgent framework is costly. In this work, we are selecting a part of data for task model training and testing during experiments.

Table 1: Statistics of the Check-in trajectory datasets.

Datasets	Foursquare (FSQ)	Brightkite(BGK)
Num. Users	463	272
Num. POIs	19870	50061
Num. Trajectories	10632	22208

Table 2: Statistics of the GPS trajectory datasets.

Datasets	Porto	Chengdu
Sampling Rate	15s	3s
Num. Trajectories	1,710,670	5,819,383
Avg. Trajectory Length (m)	3522.64	2857.81
Avg. Travel Time (s)	724.20	436.12
Latitude Range	[41.1401, 41.1859]	[30.6529, 30.7277]
Longitude Range	[-8.6902, -8.5491]	[104.042, 104.129]

3.1.2 *Task models.*

In this work, We adopt various AI models and incorporate them into TrajAgent for solving trajectory-related tasks. According to the type of tasks they deal with, these models can be framed in the following categories:

- **Models for Trajectory Prediction**. We consider the classic method RNN [20] and well-performing approaches FPMC [26] and DeepMove [7] for the task of next location prediction.
- **Models for Trajectory User Linkage**. We focus on methods for solving the task of Trajectory User Linkage under check-in trajectories, such as MainTUL [3] and DPLink [9].
- **Models for Travel Time Estimation**. We consider the mainstream model DeepTTE [34] to estimate the travel time for a given GPS trajectory.
- **Models for Trajectory Anomaly Detection**. We considered the well-performing method GMVSAE [21] for detecting anomalous trajectory.

3.1.3 *Metrics.*

To evaluate the performance of all models on multiple trajectory tasks, we employ the following different metrics:

- **Acc@5** and **Hit@5**: Acc@5 refers to the percentage of the first five results predicted correctly. Hit@5 measures whether at least one of the top-5 predictive results is correct.
- **MAE** and **AUC**: Mean Absolute Error (MAE) indicates the amount of deviation from the actual values. Area Under ROC Curve (AUC) measures how well the model correctly distinguishes the type of the sample.

In our experiments, Acc@5 is used to measure the accuracy of trajectory prediction and agent execution, Hit@5 is adopted for evaluating the performance of trajectory user linkage task; MAE is employed to compute the error of travel time estimation, while the metric AUC is used to assess the performance of trajectory anomaly detection.

3.2 Main Results

In this section, we evaluate the effectiveness of TrajAgent from two dimensions: 1) Performance of trajectory modelling tasks accomplished through TrajAgent, and 2) Execution efficiency of each stages in TrajAgent.

3.2.1 *Performance on Trajectory-related Tasks.*

Here, we focus on the performances of proposed TrajAgent framework across four types of trajectory tasks: Trajectory Prediction (TP), Trajectory User Linkage (TUL), Travel Time Estimation (TTE) and Trajectory Anomaly Detection (TAD). The first two tasks, TP and TUL, are based on check-in trajectory data, while TTE and TAD utilize GPS trajectory data. The results for these tasks are summarized in Table 3 and Table 4, respectively.

For each task, we report the model performance under four different configurations: the original model (Origin), with Data Augmentation (+DA), with Parameter Optimization (+PO), and with joint optimization (+JO). This setup allows us to assess the individual and combined effects of data augmentation and parameter optimization on the task performance.

Table 3 presents the performance comparison on check-in trajectory tasks TP and TUL. In the TP task on FSQ dataset, DeepMove improves from 0.3422 (Origin) to 0.4018 (+JO), a 17.42% increase. RNN, as a simpler model, experiences the most significant gains,

Table 3: Comparison of task performance on check-in trajectories for TrajAgent with different configurations. 'DA' represents Data Augmentation, 'PO' denotes Parameter Optimization, 'JO' indicates Joint Optimization. δ represents improvements.

Task		TP			TUL	
Model		RNN	FPMC	DeepMove	MainTUL	DPLink
Metrics			Acc@5		Hit@5	
FSQ	Origin	0.1795	0.2939	0.3422	0.4871	0.7551
	+DA	0.2667	0.2939	0.4018	0.5973	0.7551
	+PO	0.1795	0.3066	0.3422	0.5691	0.7686
	+JO	**0.2717**	**0.3066**	**0.4018**	**0.6121**	**0.8010**
	δ	51.36%	4.32%	17.42%	25.66%	6.08%
BGK	Origin	0.4422	0.5241	0.5570	0.5908	0.8993
	+DA	0.5416	0.5275	0.5647	0.6836	0.9613
	+PO	0.5022	0.5250	0.6041	0.6683	0.9552
	+JO	**0.5470**	**0.5275**	**0.6100**	**0.7145**	**0.9622**
	δ	23.70%	0.65%	9.52%	20.94%	6.99%

with accuracy increasing by 51.36% through joint optimization. On the BGK dataset, DeepMove continues to perform well, achieving 0.6100 (+JO), marking a 9.52% improvement. Across both datasets, TrajAgent's optimization techniques consistently enhance model performance, with the FSQ dataset showing particularly notable improvements. For the TUL task, MainTUL and DPLink exhibit consistent improvements under TrajAgent's optimization strategies. the performance of MainTUL increases from 0.4871 to 0.6121 (+JO) in FSQ, while DPLink achieves a more moderate increase of 6.08%. These results demonstrate the effectiveness of the TrajAgent framework, particularly in automating check-in trajectory tasks. The results also highlight the utility of TrajAgent in terms of integrating data enhancement and parameter optimization strategies for better problem solving.

Table 4 presents the performance comparison on GPS trajectory tasks, specifically focusing on TTE and TAD tasks. The models evaluated are DeepTTE for TTE and GMVSAE for TAD, with results shown for the original configuration (Origin) and after Joint Optimization (+JO). For the Proto dataset, the original model has an MAE of 8.48, which significantly improves to 5.85 after joint optimization, resulting in a 31% reduction in prediction error. On the Chengdu dataset, the MAE decreases from 7.23 to 5.95. As for TAD task, we find that the AUC scores were already high, the small but consistent improvements in both datasets suggest that TrajAgent's joint optimization can further refine model performance, even for tasks where models initially perform well.

3.2.2 *Efficiency of Agentic Workflow.*

We select the trajectory prediction task as an example to demonstrate the efficiency of agentic workflow.

Table 5 compares the execution efficiency of TrajAgent implemented by different LLMs, across each stages in the trajectory modelling workflow. We can observe that Qwen2-72B and GPT-4o-mini demonstrate the highest success rates (i.e., Succ.) across key stages such as Data Extraction, Processing, and Data/Model

Table 4: Comparison of task performance on GPS trajectories for TrajAgent with different configurations.

Task		TTE	TAD
Model Metrics		DeepTTE MAE	GMVSAE AUC
Porto	origin	8.48	0.9892
	+JO	**5.85**	**0.9899**
	δ	31.01%	minor
Chendu	Origin	7.23	0.978
	+JO	**5.95**	**0.984**
	δ	17.70%	0.61%

Selection, with over 90% success in each. They also excel in Joint Optimization, with success rates of 94% and 96%, resulting in superior Acc@5 scores of 0.4333 and 0.3724 respectively. In contrast, models like Gemma-2-9B and LLama3-8B struggle with lower processing and data selection success rates, which results in reduced overall performance. Their weaker performance in key optimization stages, especially in parameter selection, reflects their limitations in effectively supporting TrajAgent.

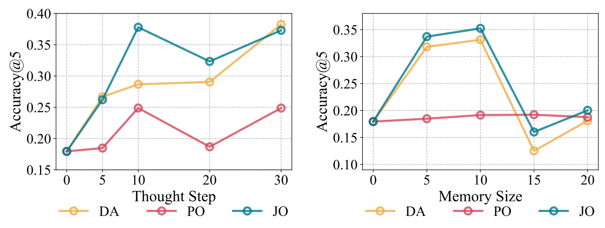
These results show that TrajAgent's efficiency is strongly influenced by the base LLM, with high-performing models like Qwen-72B and GPT-4o-mini significantly enhancing its capabilities. To fully realize TrajAgent's potential, it is essential to improve LLM performance in data augmentation and parameter optimization, as many LLMs currently struggle in these aspects.

3.3 Ablation Study

To assess the effectiveness of the key components in TrajAgent, we conducted an ablation study using execution efficiency as the evaluation criterion. Specifically, we examined the framework by isolating two main designs: the memory unit and the reflection mechanism, resulting in two variants: 1) **w/o Reflection**, where the reflection mechanism is removed, and 2) **w/o Memory**, where the memory unit is excluded. Additionally, it is important to note that GPT-4o-mini was used as the base LLM for all variants of TrajAgent in this study. The experimental results are presented in Table 6. We can find that: 1) Removing either component leads to average performance declines, with the memory unit being especially critical for maintaining execution efficiency. For example, removing the memory unit causes significant drops in accuracy for both Data Augmentation (DA) and Parameter Optimization (PO), underscoring its essential role in these processes. 2) Interestingly, removing a component occasionally results in slight increases in success or accuracy, suggesting that some components may introduce overhead or complexity in specific stages. Overall, the combined use of the memory and reflection mechanisms is crucial for optimizing TrajAgent's performance.

3.4 Parameter Study

In this section, we analyze the effects of two important parameters in the TrajAgent framework: 1) **thought step**: the number of steps that agent thought before taking action, and 2) **memory size**: the size of memory units in TrajAgent. To be specific, we vary the values of these parameters to investigate their impact on agent performance, and the results are illustrated in Figure 2(a) and (b).



(a) Thought Steps. (b) Memory Size.

Figure 2: Effects of thought steps and memory size.

As shown in Figure 2(a), increasing thought steps significantly enhances Joint Optimization (JO) and Data Augmentation (DA). This indicates that giving TrajAgent more reasoning time improves its overall performance. However, the peak performance for Parameter Optimization (PO) and JO at around 10 steps suggests that once sufficient contextual understanding is achieved, further increasing the thought steps provides diminishing returns and may not be beneficial. From Figure 2(b), we find that increasing memory size initially improves performance for JO and DA, with peak accuracy reached at a memory size of 10. This suggests that having sufficient memory allows TrajAgent to store and utilize relevant information effectively. However, beyond this point, performance starts to decline, indicating that excessive memory may introduce unnecessary complexity or irrelevant data, leading to diminished returns for JO and DA. For Parameter Optimization (PO), memory size has little effect, reflecting that this task does not rely as heavily on memory records.

3.5 Case Study

In this section, we analyze some cases where the TrajAgent's optimization performance is suboptimal. The overall process is illustrated in Figure 3. The optimization module is a key component affecting overall performance.

The first issue is *optimization trap* present in data augmentation module of TrajAgent. Specifically, it refers to the situation where agent sometimes ignores the contents of the memory during the thought process. Even when the chosen parameter combination yields poor training results, the model overlooks the error feedback (i.e., the "Not good enough..." in the memory). The "optimization trap" occurs even in the best-performing GPT-4o-mini. As the length of the Memory increases, the impact of the optimization trap on overall accuracy grows. Once an optimization trap occurs, all memories within the same step tend to favor the same ineffective combination. We believe the causes of the optimization trap could be: (1) excessively long prompts leading to truncated inputs; (2) insufficient proportion of memory in the total input.

The second issue is the *sub-optimality* appears in parameter optimization module of TrajAgent. This phenomenon exists across various datasets and model sizes. We believe the reasons for the poor performance might be: (1) the TrajAgent parameter optimization module is overly sensitive to the format of model outputs, treating all responses that do not meet the format requirements as invalid;

Table 5: Execution efficiency of TrajAgent implemented by different LLMs.

LLM	Extraction	Processing	Data/Model Selection		Data Augmentation.		Parameter Optim.		Joint Optim.	
	Succ.	Succ.	Succ.	Acc@5	Succ.	Acc@5	Succ.	Acc@5	Succ.	Acc@5
Qwen2-7B	85.00%	30%	72%	83.33%	15%	0.2015	25%	0.1833	64%	0.2668
Mistral7B-V3	78.89%	42%	88%	90.91%	94%	0.2940	**95%**	**0.2087**	82%	0.2980
LLama3-8B	69.44%	28%	81%	80.25%	18%	0.1790	11%	0.1809	65%	0.2809
Gemma2-9B	83.88%	12%	57%	52.63%	18%	0.1822	15%	0.1848	70%	0.2970
Gemma-2-27B	79.44%	30%	70%	88.57%	78%	0.2507	30%	0.1775	78%	0.3366
GPT3.5-Turbo	88.89%	54%	**100%**	82.00%	88%	0.2846	90%	0.1809	92%	0.3295
LLama3-70B	83.33%	**100%**	95%	86.32%	92%	0.2931	83%	0.1848	95%	0.3473
Qwen2-72B	92.22%	95%	**100%**	95%	**96%**	**0.3925**	70%	0.1816	94%	**0.4333**
GPT-4o-mini	**95.56%**	92%	**100%**	**98.00%**	90%	0.2967	85%	0.1822	**96%**	0.3724

Table 6: Ablation Study of TrajAgent. 'MS' stands for Model Selection, 'DA' represents Data Augmentation, 'PO' denotes Parameter Optimization, 'JO' stands for Joint Optimization. ↓ indicates a decrease in performance, and ↑ indicates an improvement.

Agent Variants	MS		DA		PO		JO	
	Succ.	Acc	Succ.	Acc	Succ.	Acc	Succ.	Acc
TrajAgent	100%	98%	98%	0.3050	89%	0.1895	85%	0.3233
w/o Reflection	100%	95%↓	98%	0.3028↓	90%↑	0.1872↓	82%↓	0.3212↓
w/o Memory	99%↓	80%↓	85%↓	0.1707↓	70%↓	0.2050↑	68%↓	0.1804↓

(2) adjustments to certain parameters result in increased training time, reducing the total number of iterations.

4 RELATED WORK

4.1 Trajectory Modelling and Analytics

In recent year, trajectory modelling [4, 11, 53] makes great progress on its core research questions, including prediction [7, 20, 28], classification [19, 31], and generation [8, 49].

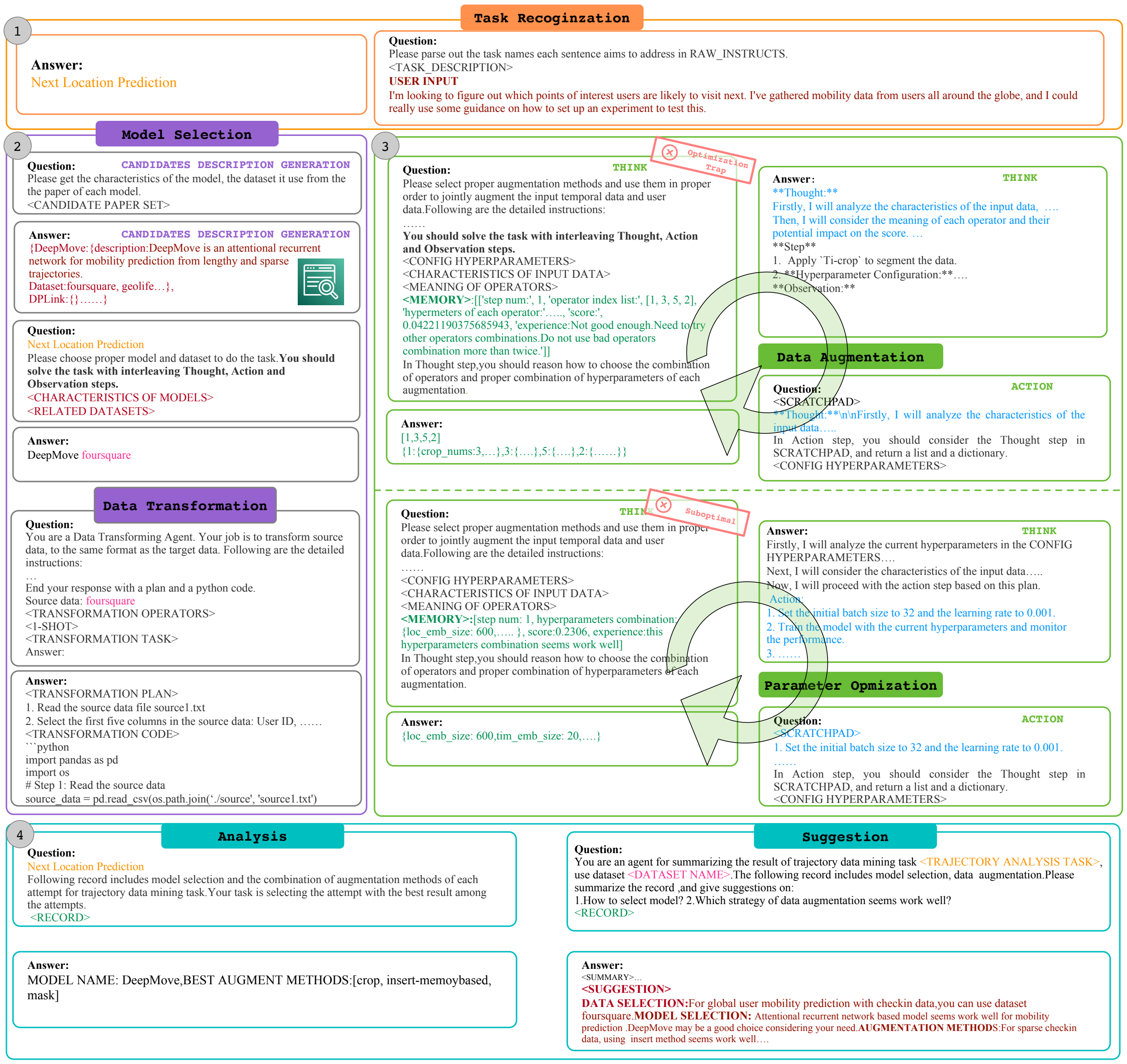
Trajectory prediction is to predict the next location or the trip destination of the trajectory. ST-RNN [20] is the first work for applying recurrent neural network into the trajectory prediction task. Following this work, lots of works have been done to solve the critical issues of trajectory prediction with deep learning, for example, DeepMove [7] for multi-granularity regularity modelling, Flash-back [42] for sparse trajectory prediction, GETNext [45] for utilizing aggregated mobility pattern. Besides, to solve the travel time prediction task, another important prediction problem for trajectory modelling, deep learning models including DeepTTE [34], TTPNet [28] and CARTE [13] are proposed and achieve promising performance in various scenarios. As for the classification problem in trajectory modelling, the most well-known tasks are trajectory mode detection [15] and trajectory user linkage [10] task. Various techniques, such as siamese network [9], distillation learning [3] and transformers [19] are successfully applied to discover the pattern among trajectories to solve the classification tasks. Finally, due to the natural limitation of trajectory data quality, trajectory data recovery [39, 41] and generation [8, 24] are also important tasks in trajectory modelling which are crucial for the practical application in the real world. Thus, follow the the trend of generative modelling, diverse generative models including GAN [8, 24], diffusion models [37, 49] are utilized to improve the modelling of human dynamics.

While these specific methods accelerate the development of trajectory modelling from different aspects, they can only handle one type of task. In other words, the unified model for all the trajectory modelling task is still missing due to the heterogeneity of tasks and trajectory data. In this paper, we propose to utilize the power of LLM and agent to build a unified model framework for diverse trajectory modelling tasks, which can automatically handle various trajectory data and trajectory modelling tasks without human intervention.

4.2 Large Language Model and Agent

With the advent of ChatGPT [23], large language models(LLMs) with extensive commonsense and outstanding reasoning abilities have been widely explored in many domains, such as mathematics [47], question answering [56], and human-machine interaction [33]. Furthermore, agentic framework [32, 35, 40] are proposed to enhance the robustness and task solving abilities of LLMs for real-world complex tasks, such as web-navigation [22, 46, 55] and software development [12, 25, 44]. Following this promising direction, researchers design various LLM based agents for solving specific spatial-temporal tasks, including mobility prediction [6], trajectory generation [18] and traffic signal control [17].

Besides, researchers also explore the potential of applying LLM based agent for automatic research experiments [1, 2, 27] especially machine learning experiments [14, 29, 38, 52]. For example, HuggingGPT [29] is proposed to utilize LLM to manage existing AI models in Huggingface community to solve complicated AI tasks. MLAgentBench [14] designs MLagent to automatically solve simple kaggle competition with ReAct mechanism in terms of success rate. DSBench [52] verifies the LLM based agent for solving classic data science tasks with tabular data as input. However, trajectory data are more complicated than the tabular data or text data and different trajectory modelling tasks also exhibit diverse problem settings and models, which are too challenging for these existing agents designed for general machine learning tasks with simple data input and interface.



Answer:
Next Location Prediction

Question:
Please parse out the task names each sentence aims to address in RAW_INSTRUCTS.
<TASK_DESCRIPTION>
USER INPUT
I'm looking to figure out which points of interest users are likely to visit next. I've gathered mobility data from users all around the globe, and I could really use some guidance on how to set up an experiment to test this.

Question: CANDIDATES DESCRIPTION GENERATION
Please get the characteristics of the model, the dataset it use from the the paper of each model.
<CANDIDATE PAPER SET>

Answer: CANDIDATES DESCRIPTION GENERATION
{DeepMove:{description:DeepMove is an attentional recurrent network for mobility prediction from lengthy and sparse trajectories.
Dataset:foursquare, geolife…},
DPLink:{}……}

Question:
Next Location Prediction
Please choose proper model and dataset to do the task.**You should solve the task with interleaving Thought, Action and Observation steps.**
<CHARACTERISTICS OF MODELS>
<RELATED DATASETS>

Answer:
DeepMove foursquare

Question:
You are a Data Transforming Agent. Your job is to transform source data, to the same format as the target data. Following are the detailed instructions:
…
End your response with a plan and a python code.
Source data: foursquare
<TRANSFORMATION OPERATORS>
<1-SHOT>
<TRANSFORMATION TASK>
Answer:

Answer:
<TRANSFORMATION PLAN>
1. Read the source data file source1.txt
2. Select the first five columns in the source data: User ID, ……
<TRANSFORMATION CODE>
```python
import pandas as pd
import os
# Step 1: Read the source data
source_data = pd.read_csv(os.path.join('./source', 'source1.txt')
```

Question: THINK
Please select proper augmentation methods and use them in proper order to jointly augment the input temporal data and user data.Following are the detailed instructions:
……
You should solve the task with interleaving Thought, Action and Observation steps.
<CONFIG HYPERPARAMETERS>
<CHARACTERISTICS OF INPUT DATA>
<MEANING OF OPERATORS>
<MEMORY>:[['step num:', 1, 'operator index list:', [1, 3, 5, 2], 'hypermeters of each operator:'….., 'score:', 0.04221190375685943, 'experience:Not good enough.Need to try other operators combinations.Do not use bad operators combination more than twice.']]
In Thought step,you should reason how to choose the combination of operators and proper combination of hyperparameters of each augmentation.

Answer:
[1,3,5,2]
{1:{crop_nums:3,…},3:{….},5:{….},2:{……}}

Answer: THINK
Thought:
Firstly, I will analyze the characteristics of the input data, ….
Then, I will consider the meaning of each operator and their potential impact on the score. …
Step
1. Apply `Ti-crop` to segment the data.
2. **Hyperparameter Configuration:**….
Observation:

Question: ACTION
<SCRATCHPAD>
Thought:\n\nFirstly, I will analyze the characteristics of the input data…..
In Action step, you should consider the Thought step in SCRATCHPAD, and return a list and a dictionary.
<CONFIG HYPERPARAMETERS>

Question: THINK
Please select proper augmentation methods and use them in proper order to jointly augment the input temporal data and user data.Following are the detailed instructions:
……
<CONFIG HYPERPARAMETERS>
<CHARACTERISTICS OF INPUT DATA>
<MEANING OF OPERATORS>
<MEMORY>:[step num: 1, hyperparameters combination: {loc_emb_size: 600,….. }, score:0.2306, experience:this hyperparameters combination seems work well]
In Thought step,you should reason how to choose the combination of operators and proper combination of hyperparameters of each augmentation.

Answer:
{loc_emb_size: 600,tim_emb_size: 20,….}

Answer: THINK
Firstly, I will analyze the current hyperparameters in the CONFIG HYPERPARAMETERS….
Next, I will consider the characteristics of the input data…..
Now, I will proceed with the action step based on this plan.
Action:
1. Set the initial batch size to 32 and the learning rate to 0.001.
2. Train the model with the current hyperparameters and monitor the performance.
3. ……

Question: ACTION
<SCRATCHPAD>
1. Set the initial batch size to 32 and the learning rate to 0.001.
……
In Action step, you should consider the Thought step in SCRATCHPAD, and return a list and a dictionary.
<CONFIG HYPERPARAMETERS>

Question:
Next Location Prediction
Following record includes model selection and the combination of augmentation methods of each attempt for trajectory data mining task.Your task is selecting the attempt with the best result among the attempts.
<RECORD>

Answer:
MODEL NAME: DeepMove,BEST AUGMENT METHODS:[crop, insert-memoybased, mask]

Question:
You are an agent for summarizing the result of trajectory data mining task <TRAJECTORY ANALYSIS TASK>, use dataset <DATASET NAME>.The following record includes model selection, data augmentation.Please summarize the record ,and give suggestions on:
1.How to select model? 2.Which strategy of data augmentation seems work well?
<RECORD>

Answer:
<SUMMARY>…
<SUGGESTION>
DATA SELECTION: For global user mobility prediction with checkin data,you can use dataset foursquare.**MODEL SELECTION:** Attentional recurrent network based model seems work well for mobility prediction .DeepMove may be a good choice considering your need.**AUGMENTATION METHODS**:For sparse checkin data, using insert method seems work well.…

Figure 3: A representative example case of *TrajAgent*.

Recently, UrbanLLM [16] follows the idea of HuggingGPT [29] to manage the existing specialized spatial-temporal models and APIs for tackling urban related tasks, which covers some trajectory modelling tasks. However, existing specialized spatial-temporal AI models usually designed for specific task with specific data and format, directly utilizing these models can suffer significant performance drop even can not generate any make sense results. In this paper, different from general machine learning agent and UrbanLLM for static spatial-temporal model, our proposed agentic framework is designed for unified trajectory modelling which can provide automatically model training and optimization for various trajectory data and tasks.

5 CONCLUSION

In this paper, we propose *TrajAgent*, an LLM-based agentic framework for unified trajectory modelling. Supported by *UniEnv*, which provides a unified data and model interface, and *AutOpt* for data and parameter optimization, *TrajAgent* can automatically identify and train the appropriate model, delivering competitive performance across a range of trajectory modelling tasks. *TrajAgent* introduces a new paradigm for unified trajectory modelling across diverse datasets and tasks, paving the way for advancements in trajectory modelling applications.

REFERENCES

[1] Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. 2024. SUPER: Evaluating Agents on Setting Up and Executing Tasks from Research Repositories. *arXiv preprint arXiv:2409.07440* (2024).

[2] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature* 624, 7992 (2023), 570–578.

[3] Wei Chen, Shuzhe Li, Chao Huang, Yanwei Yu, Yongguo Jiang, and Junyu Dong. 2022. Mutual distillation learning network for trajectory-user linking. *arXiv preprint arXiv:2205.03773* (2022).

[4] Wei Chen, Yuxuan Liang, Yuanshao Zhu, Yanchuan Chang, Kang Luo, Haomin Wen, Lei Li, Yanwei Yu, Qingsong Wen, Chao Chen, et al. 2024. Deep learning for trajectory data management and mining: A survey and beyond. *arXiv preprint arXiv:2403.14151* (2024).

[5] Yizhou Dang, Enneng Yang, Guibing Guo, Linying Jiang, Xingwei Wang, Xiaoxiao Xu, Qinghui Sun, and Hong Liu. 2023. Uniform sequence better: Time interval aware data augmentation for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 4225–4232.

[6] Jie Feng, Yuwei Du, Jie Zhao, and Yong Li. 2024. AgentMove: Predicting Human Mobility Anywhere Using Large Language Model based Agentic Framework. *arXiv preprint arXiv:2408.13986* (2024).

[7] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*. 1459–1468.

[8] Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to simulate human mobility. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3426–3433.

[9] Jie Feng, Mingyang Zhang, Huandong Wang, Zeyu Yang, Chao Zhang, Yong Li, and Depeng Jin. 2019. Dplink: User identity linkage via deep neural network from heterogeneous mobility data. In *The world wide web conference*. 459–469.

[10] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying Human Mobility via Trajectory Embeddings.. In *IJCAI*, Vol. 17. 1689–1695.

[11] Anita Graser, Anahid Jalali, Jasmin Lampert, Axel Weißenfeld, and Krzysztof Janowicz. 2024. MobilityDL: a review of deep learning from trajectory data. *GeoInformatica* (2024), 1–33.

[12] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023).

[13] Liping Huang, Yongjian Yang, Hechang Chen, Yunke Zhang, Zijia Wang, and Lifang He. 2022. Context-aware road travel time estimation by coupled tensor decomposition based on trajectory data. *Knowledge-Based Systems* 245 (2022), 108596.

[14] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. In *Forty-first International Conference on Machine Learning*.

[15] Xiang Jiang, Erico N de Souza, Ahmad Pesaranghader, Baifan Hu, Daniel L Silver, and Stan Matwin. 2017. Trajectorynet: An embedded gps trajectory representation for point-based classification using recurrent neural networks. *arXiv preprint arXiv:1705.02636* (2017).

[16] Yue Jiang, Qin Chao, Yile Chen, Xiucheng Li, Shuai Liu, and Gao Cong. 2024. UrbanLLM: Autonomous Urban Activity Planning and Management with Large Language Models. *arXiv preprint arXiv:2406.12360* (2024).

[17] Siqi Lai, Zhao Xu, Weijia Zhang, Hao Liu, and Hui Xiong. 2023. Large language models as traffic signal control agents: Capacity and opportunity. *arXiv preprint arXiv:2312.16044* (2023).

[18] Xuchuan Li, Fei Huang, Jianrong Lv, Zhixiong Xiao, Guolong Li, and Yang Yue. 2024. Be More Real: Travel Diary Generation Using LLM Agents and Individual Profiles. *arXiv preprint arXiv:2407.18932* (2024).

[19] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. 2022. TrajFormer: Efficient trajectory classification with transformers. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1229–1237.

[20] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[21] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. 2020. Online anomalous trajectory detection with deep generative sequence modeling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 949–960.

[22] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[23] OpenAI. 2024. Introducing ChatGPT. https://openai.com/index/chatgpt/ Accessed: 2024-10-14.

[24] Kun Ouyang, Reza Shokri, David S Rosenblum, and Wenzhuo Yang. 2018. A non-parametric generative model for human trajectories.. In *IJCAI*, Vol. 18. 3812–3817.

[25] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15174–15186.

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[27] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (2024), 468–475.

[28] Yibin Shen, Cheqing Jin, Jiaxun Hua, and Dingjiang Huang. 2020. TTPNet: A neural network for travel time prediction based on tensor decomposition and graph embedding. *IEEE Transactions on Knowledge and Data Engineering* 34, 9 (2020), 4514–4526.

[29] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems* 36 (2024).

[30] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2024).

[31] Li Song, Ruijia Wang, Ding Xiao, Xiaotian Han, Yanan Cai, and Chuan Shi. 2018. Anomalous trajectory detection using recurrent neural network. In *Advanced Data Mining and Applications: 14th International Conference, ADMA 2018, Nanjing, China, November 16–18, 2018, Proceedings 14*. Springer, 263–277.

[32] Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427* (2023).

[33] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[34] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[35] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[36] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560* (2022).

[37] Tonglong Wei, Youfang Lin, Shengnan Guo, Yan Lin, Yiheng Huang, Chenyang Xiang, Yuqing Bai, and Huaiyu Wan. 2024. Diff-rntraj: A structure-aware diffusion model for road network-constrained trajectory generation. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[38] Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Wenhai Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, et al. 2024. VisionLLM v2: An End-to-End Generalist Multimodal Large Language Model for Hundreds of Vision-Language Tasks. *arXiv preprint arXiv:2406.08394* (2024).

[39] Dongbo Xi, Fuzhen Zhuang, Yanchi Liu, Jingjing Gu, Hui Xiong, and Qing He. 2019. Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5458–5465.

[40] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864* (2023).

[41] Tong Xia, Yunhan Qi, Jie Feng, Fengli Xu, Funing Sun, Diansheng Guo, and Yong Li. 2021. Attnmove: History enhanced trajectory recovery via attentional network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4494–4502.

[42] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. 2020. Location prediction over sparse user mobility traces using rnns. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence*. 2184–2190.

[43] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 1–23.

[44] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793* (2024).

[45] Song Yang, Jiamou Liu, and Kaiqi Zhao. 2022. GETNext: trajectory flow map enhanced transformer for next POI recommendation. In *Proceedings of the 45th*

International ACM SIGIR Conference on research and development in information retrieval. 1144–1153.
[46] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems* 35 (2022), 20744–20757.
[47] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).
[48] Mingjia Yin, Hao Wang, Wei Guo, Yong Liu, Suojuan Zhang, Sirui Zhao, Defu Lian, and Enhong Chen. 2024. Dataset regeneration for sequential recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3954–3965.
[49] Yuan Yuan, Jingtao Ding, Huandong Wang, Depeng Jin, and Yong Li. 2022. Activity trajectory generation via modeling spatiotemporal dynamics. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4752–4762.
[50] Zhengqing Yuan, Ruoxi Chen, Zhaoxu Li, Haolong Jia, Lifang He, Chi Wang, and Lichao Sun. 2024. Mora: Enabling generalist video generation via a multi-agent framework. *arXiv preprint arXiv:2403.13248* (2024).
[51] Xiaocai Zhang, Zhixun Zhao, Yi Zheng, and Jinyan Li. 2019. Prediction of taxi destinations using a novel data embedding method and ensemble learning. *IEEE Transactions on Intelligent Transportation Systems* 21, 1 (2019), 68–78.
[52] Yuge Zhang, Qiyang Jiang, Xingyu Han, Nan Chen, Yuqing Yang, and Kan Ren. 2024. Benchmarking Data Science Agents. *arXiv preprint arXiv:2402.17168* (2024).
[53] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 1–41.
[54] Peilin Zhou, You-Liang Huang, Yueqi Xie, Jingqi Gao, Shoujin Wang, Jae Boum Kim, and Sunghun Kim. 2024. Is Contrastive Learning Necessary? A Study of Data Augmentation vs Contrastive Learning in Sequential Recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3854–3863.
[55] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* (2023).
[56] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems* 36 (2023), 50117–50143.

A APPENDIX

Parameter Optimization

User

Please select proper combination of hyperparameters of model in CONFIG HYPERPARAMETERS.Adjust the selection, the combination of hyperparameters based on the main function of hyperparameters, the characteristics of the input data, the tuning principles, and memory to get a high score. You should solve the task with interleaving Thought, Action and Observation steps.

<CHARACTERISTICS OF INPUT DATA>

The input temporal data contains a time dictionary(key is the user ID,the value is a list containing all time points when the user is active in chronological order) , the input user data contains a user dictionary(key is the user ID,the value is a list containing all items that the user interacts with in chronological order).

<CONFIG HYPERPARAMETERS>

`{loc_emb_size: 500, tim_emb_size: 10, ...}`

<TUNING PRINCIPLES>

1.Start with a small batch size (32-64) and a small learning rate (0.001-0.01): This will help prevent overshooting and overfitting.

2.Increase batch size and learning rate: If the model is not overfitting, increasing the batch size and learning rate can help improve convergence.

3.Add dropout (0.2-0.5) to prevent overfitting: If the model is overfitting, adding dropout can help regularize the model.

4.Increase embedding size: If the model is not capturing enough information, increasing the embedding size can help improve representational power.

5.Decrease learning rate and increase batch size: If the model is not converging, decreasing the learning rate and increasing the batch size can help improve stability.

<MEMORY>:

<SCRATCHPAD>:

In Thought step,you should reason how to choose the combination of hyperparameters to get a higher score.Please consider following aspects: 1.Observe the hyperparameters with high scores in MEMORY, to determine the optimal hyperparameters.

2.Use a grid search: Perform a grid search over a range of hyperparameters with high scores to find the optimal combination with higher scores.

3.Increase batch size and learning rate, and use dropout to avoid overfitting.

4. Stop or reverse the adjusting trend if the score is decreasing. According to above aspects,please first learn experiences from MEMORY, then make plan for the action step.Please use the sentence structure 'Firstly... Then... Lastly'.Let's think step by step.

Thought:

Assistant

Firstly, I will analyze the current hyperparameters in the CONFIG HYPERPARAMETERS. ...

Then, I will consider the hidden size, which is currently set at 500. ...

Next, the learning rate is currently at 0.001, which is a good starting point. ...

Lastly, I will monitor the model's performance closely....

Now, I will proceed with the action step to adjust the hyperparameters based on this reasoning.

Action:

1. Increase the tim_emb_size from 10 to 20 to capture more temporal information.

2. Keep the loc_size at 500 for now, as it seems adequate.

3. Increase the hidden_size from 500 to 600 to enhance the model's capacity.

4. Increase the dropout_p from 0.3 to 0.4 to further prevent overfitting.

5. Set the learning_rate to 0.005 to improve convergence speed.

6. Start with a batch size of 32 and monitor the performance closely.

Observation:

After implementing these changes, I will evaluate the model's performance on a validation set. ...

User

In Action step, you should consider the Thought step in SCRATCHPAD, and give a dict:{hypermeter name:hypermeter value}.The hypermeter name should be the same with the raw config hypermeter names in CONFIG HYPERPARAMETERS,and hypermeter values should be the same type as the hypermeter values in CONFIG HYPERPARAMETERS.Please do not add any comments to each value.

Action:

Assistant

{'loc_emb_size': 500,...}

Task Understanding

User

The description of each task is in TASK_DESCRIPTION.Please parse out the task name the sentence aims to address in RAW_INSTRUCTS.

1. The task name should match the key in the TASK_DESCRIPTION.

2. Please only output the task name.Do not output other contents.

<TASK_DESCRIPTION>

Next_Location_Prediction: ...,

Trajectory_user_linking:...

assistant

Trajectory$_{completion}$

Data Augmentation

User

<TASK>

Please:

1.select proper augmentation methods and use them in proper order to jointly augment the input temporal data and user data.Please adjust the selection and combination sequence of operators based on MEANING OF OPERATORS, CHARACTERISTICS OF INPUT DATA, and MEMORY to get a high score.

2.select proper combination of hyperparameters of each augmentation method in CONFIG HYPERPARAMETERS.Adjust the selection, the combination of hyperparameters based on the main function of hyperparameters, the characteristics of the input data, and memory to get a high score.

You should solve the task with interleaving Thought, Action and Observation steps.

<CONFIG HYPERPARAMETERS>

1.Ti-crop: crop_nums:(int) the size of cropping. default is 2.

crop_ratio:(float) the ratio of cropping. default is 0.

crop_n_times:(int) the number of cropping for each sequence for default setting. default is 2.

crop_time_sort:(str, choice in [maximum, minimum]) choose the candidate subsequence in a descending/ascending order according to its time interval variance. default is 'minimum'.

...

2.Ti-insert_unvisited: ...

<MEANING OF OPERATORS>

1.Ti-crop: For each user, filter out sessions with fewer trajectory points than ti_threshold + ti_crop_n_times. In each session, randomly extract several segments (ti_crop_n_times) of equal length (ti_threshold). Sample data segments according to the rules.

2.Ti-insert_unvisited:

<MEMORY>:

<SCRATCHPAD>:

In Thought step,you should reason how to choose the combination of operators and proper combination of hyperparameters of each augmentation method to get a higher score.Please consider following aspects:

1.The meaning of each operator,and the impact of adding or removing operators at a specific location on the score.

2.Common characteristics of operator index lists with higher scores in MEMORY.Avoid using the same index list as MEMORY with scores lower than 0.18111313868613138.

3.The meaning of hyperparameters of each operator you select, and how to adjust the hyperparameters based on the characteristics of input data and hyperparameters with high scores in MEMORY.

4.Use a grid search: Perform a grid search over a range of hyperparameters with high scores to find the optimal combination with higher scores.

7. Stop or reverse the adjusting trend if the score is decreasing.

According to above aspects,please first learn experiences from MEMORY, then make plan for the action step.Please use the sentence structure 'Firstly... Then... Lastly'.Let's think step by step.

Thought:

assistant

Thought:

Firstly, I will analyze the characteristics of the input data, which consists of user activity timestamps and the corresponding items they interacted with. This suggests that temporal relationships and user preferences are crucial for effective augmentation. The augmentation methods should enhance these aspects without distorting the underlying patterns.

Then, I will consider the meaning of each operator and their potential impact on the score. For instance, using 'Ti-crop' can help in segmenting the data, which is essential for understanding user behavior over time. ...

Lastly, I will select hyperparameters for each operator based on their functions and the characteristics of the input data.

Action:

1. **Select Augmentation Methods:** - **Ti-crop**:

- 'crop_nums': 3

- 'crop_ratio': 0.1 (to allow for a small portion of the data to be cropped)

- 'crop_n_times': 3

...

2. **Hyperparameter Configuration:**

- For 'Ti-crop', I will keep 'crop_n_times' at 3 to ensure multiple segments are created....

3. **Evaluate and Adjust:**

- After implementing the above configurations, I will evaluate the score...

Observation:

I will monitor the score after each adjustment to ensure that the changes lead to an improvement. If the score decreases, I will revert to the previous settings.

User

<SCRATCHPAD>

Action:

1. **Select Augmentation Methods:**

- **Ti-crop**: ...

In Action step, you should consider the Thought step in SCRATCHPAD, and return a list and a dictionary.

The list should contain the indices of augmentation methods in {1: 'Ti-crop', 2: 'Ti-insert_unvisited...}.For example, if you want to first use Ti-crop, secondly use Ti-insert_random, thirdly use Ti-mask, then the list should be '[1,4,8]'.

The dict:....The hypermeter name should be the same with the corresponding operator hypermeter names in CONFIG HYPERPARAMETERS,and hypermeter values should be the same type as the corresponding operator hypermeter values in CONFIG HYPERPARAMETERS.

Please directly output the list and dictionary.For example:...

Action:

Assistant

[1, 3, 5]

{{crop_nums:3,crop_ratio:0,....},3:{...},5:{...}}