Faithful Trip Recommender Using Diffusion Guidance (Student Abstract)

Wenzheng Shu[1], Yanlong Huang[1], Wenxin Tai[1,2], Zhangtao Cheng[1,2], Bei Hui[1,*] Goce Trajcevski[3],

[1]University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China
[2]Kashi Institute of Electronics and Information Industry, Kashgar 844000, China
[3]Iowa State University, Ames, IA 50011, USA
shuwenzheng926@gmail.com, hyloong77@gmail.com, wxtai@std.uestc.edu.cn, zhangtao.cheng@outlook.com,
bhui@uestc.edu.cn, gocet25@iastate.edu

Abstract

Trip recommendation aims to plan user's travel based on their specified preferences. Traditional heuristic and statistical approaches often fail to capture the intricate nuances of user intentions, leading to subpar performance. Recent deep-learning methods show attractive accuracy but struggle to generate faithful trajectories that match user intentions. In this work, we propose a DDPM-based incremental knowledge injection module to ensure the faithfulness of the generated trajectories. Experiments on two datasets verify the effectiveness of our approach.

Introduction

When users share their desire to plan a trip, specifying sources, destinations, and the number of attractions they wish to visit, the goal of the trip recommender is to provide them with a carefully crafted sequence of points of interest (POIs) that align with their preferences.

Traditional approaches, as highlighted in pioneering studies (Chen, Ong, and Xie 2016), primarily rely on heuristic or statistical algorithms. These methods suggest trajectories based on the popularity of POIs and the frequency of their interactions. However, these techniques often fall short of capturing the intricate nuances of user intentions, leading to suboptimal recommendations. In contrast, recent advances (Zhou et al. 2021; Gao et al. 2021) have introduced neural networks (NNs) for POI recommendations. Models such as recurrent neural networks and generative adversarial networks have been leveraged to model high-order interactions among a user's preferences and corresponding trajectories. Nevertheless, despite their promising performance, these prevailing methods still encounter difficulties in producing recommendations that truly align with users' intentions. An example of this challenge is the inconsistency between the source and destination generated by NNs and the user's query (cf. Table 2).

To address this challenge, current methods choose to generate only the intermediate sub-trajectory and merge it with known sources and destinations. However, adopting such a simple clipping-merging strategy inevitably leads to distortion issues. In this study, we introduce Denoising Diffusion Probabilistic Models (DDPMs) to assist the trip recommender in generating faithful recommendations. In contrast to simply clipping-concatenation strategy, DDPM facilitates the alignment of the generated trajectory with the user's intention in a "soft and gradual" manner. Experimental results demonstrate that our approach can ensure faithful recommendation while avoiding trajectory distortion.

Methodology

Problem Definition and Solution. Let $\mathcal{P}$ denote a set of POIs annotated by the location-based system, where each POI $p \in \mathcal{P}$ is associated with its longitude and latitude coordinates (p^o, p^a) and its category $p^c \in \mathcal{C}$. Let quintuple $\mathcal{Q} = (p_s, t_s, p_e, t_e, N)$ denote a tourist query, which includes the desired start POI p_s, start time t_s, the length budget N (a.k.a the number of POIs the tourist wants to visit), the end POI p_e, and end time t_e. When a tourist provides a query $\mathcal{Q}$, the trip recommender will return a trip route $\mathcal{T} : \{p_1 \to p_2 \to \cdots \to p_N\}$, where $p_1 = p_s$ and $p_N = p_e$.

We have two ways for generating trajectories: (1) generating the full trajectory $\{p_1 \to p_N\}$ directly, or (2) generating a sub-trajectory $\{p_2 \to p_{N-1}\}$ and then merging it with the provided source and destination. The majority of existing deep learning approaches opt for the second way as it ensures that the model-generated trajectories align coherently with the user's intentions – at least the model can produce faithful recommendations.

Relying solely on the simple clipping-merging strategy for aligning with the user's intent is self-deception. If the model generates mismatched trajectories initially, merging with sources and destinations won't rectify the deviation from the ground truth. This is due to the model's lack of genuine understanding of the user's intent. A fundamental drawback is that substitution is a post-hoc operation, incapable of influencing the model's inference process.

One of the properties of diffusion models is that multiple steps are required to generate the sample. Based on this property, we can gradually inject ground-true knowledge (sources and destinations) to steer the generation process. Specifically, let's consider $\boldsymbol{X}_0 \in \mathbb{R}^{N\times d}$ as the concatenation of the trajectory embedding. Given the inference of DDPM:

$$p_\theta(\boldsymbol{X}_0) = \int p_\theta(\boldsymbol{X}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t) d(\boldsymbol{X}_{1:T}), \quad (1)$$

[*]Corresponding Author.

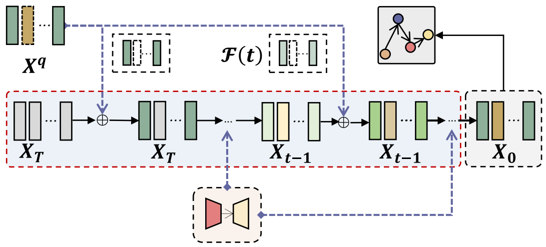

Figure 1: Workflow of the proposed method.

We iteratively inject knowledge into each step:

$$\boldsymbol{X}_{t-1} = \mathcal{F}(t) \odot \boldsymbol{X}_{t-1} + (1 - \mathcal{F}(t)) \odot \boldsymbol{X}^q, \tag{2}$$

where $\boldsymbol{X}_{t-1} \in \mathbb{R}^{N\times d}$ is sampled from $\sim p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$ and $\boldsymbol{X}^q \in \mathbb{R}^{N\times d}$ is generated by users' query – we use the ground-true embedding of start and end for index 0 and -1, while others are feed with zero vector. We define $\mathcal{F}(t)$ as a monotonic function $e^{-\lambda t}$ to adjust the weight of the incorporation of X^q, where λ is a pre-defined hyper-parameter.

Note that the proposed method has some appealing properties: (i) when $t \to T$, $\mathcal{F}(t)$ limits to 0, thus the condition factor will (almost) have no effect on the initial state of DDPM; (2) when $t \to 0$, we can guarantee that the positions of the start and end points will be completely substituted by the ground-true values.

Implementation. Drawing inspiration from prior research (Sun et al. 2019), we employ one transformer layer as the basic architecture of our trip recommender. We train DDPM to learn the distribution of trajectory embeddings, using mean square error (MSE) as the loss function. Furthermore, to ensure these embeddings carry meaningful information, we use one non-linear layer to project embeddings to trajectories. Then we optimize the POI embeddings using cross-entropy loss – the model's needs to accurately reconstruct the trajectory $p_1 \to \cdots \to p_N$ given $\boldsymbol{X}_0$. We adopt multi-task learning to jointly optimize the model:

$$\mathcal{L} = \alpha * \mathcal{L}_{pre} + \mathcal{L}_{rec}, \tag{3}$$

where $\mathcal{L}_{pre}$ represents the prediction loss (MSE) of diffusion training, and $\mathcal{L}_{rec}$ represents the classification loss of POI embedding. Figure 1 shows the details of our method.

Experiments

Baselines. We compare our method in two real-world datasets with five baselines, including two statistical methods: Popularity and PoiRank (Chen, Ong, and Xie 2016), and three deep learning methods: DeepTrip (Gao et al. 2021), CTLTR (Zhou et al. 2021), and SelfTrip (Gao et al. 2022).

Metrics. Following prior studies, we compare our model with baselines using F_1 score. To better assess efficiency in addressing distortion issues, we introduce three new metrics: D_{med} (average Manhattan distance excluding source and destination), S_{ratio} (faithful sources probability), and E_{ratio} (faithful destinations probability). Higher F_1 score, S_{ratio}, and E_{ratio}, or lower D_{med} indicate superior performance.

Popularity	PoiRank	SelfTrip	DeepTrip	CTLTR	**Ours**
0.701	0.700	0.728	0.638	0.728	**0.774**
0.744	0.769	0.834	0.714	**0.838**	0.830

Table 1: F_1 scores for Edinburgh and Glasgow (upper and lower rows) are provided. For deep-learning methods, note that F_1 considers both start and end points, ensuring fairness in comparison with statistical methods.

Method	**Edinburgh**			**Glasgow**		
	D_{med}	S_{ratio}	E_{ratio}	D_{med}	S_{ratio}	E_{ratio}
Popularity	0.485	—	—	0.962	—	—
PoiRank	0.508	—	—	0.612	—	—
SelfTrip	0.485	21.9%	25.4%	0.813	29.5%	29.5%
DeepTrip	0.485	42.4%	31.1%	0.818	36.6%	33.0%
CTLTR	0.442	97.5%	75.1%	0.782	82.1%	54.5%
Ours	**0.411**	**100%**	**100%**	**0.386**	**100%**	**100%**

Table 2: Conclusive results of trajectory distortion problems.

Performance Analysis. Our comparison in Tables 1 and 2 reveals two main findings: (1) Existing deep-learning methods show distortion issues. Despite CTLTR can generate relatively accurate start points, it also falls short in reaching destinations, while our methods solve this problem effectively. (2) While keeping a balanced $F1$ score, we minimize sub-trajectory recommendation error D_{med}, indicating that with DDPM guidance, our model better understands user preferences.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (Grant No.62176043, No.62072077, and No.62273071), and National Science Foundation SWIFT (Grant No.2030249).

References

Chen, D.; Ong, C. S.; and Xie, L. 2016. Learning points and routes to recommend trajectories. In *CIKM*, 2227–2232.

Gao, Q.; Wang, W.; Zhang, K.; Yang, X.; Miao, C.; and Li, T. 2022. Self-supervised representation learning for trip recommendation. *KBS*, 247: 108791.

Gao, Q.; Zhou, F.; Zhang, K.; Zhang, F.; and Trajcevski, G. 2021. Adversarial human trajectory learning for trip recommendation. *TNNLS*, 34: 1764–1776.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 1441–1450.

Zhou, F.; Wang, P.; Xu, X.; Tai, W.; and Trajcevski, G. 2021. Contrastive trajectory learning for tour recommendation. *TIST*, 13: 1–25.