Large Language Models for Next Point-of-Interest Recommendation

Peibo Li
University of New South Wales
Sydney, Australia
peibo.li@student.unsw.edu.au

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Hao Xue
University of New South Wales
Sydney, Australia
hao.xue1@unsw.edu.au

Shuang Ao
University of New South Wales
Sydney, Australia
shuang.ao@unsw.edu.au

Yang Song
University of New South Wales
Sydney, Australia
yang.song1@unsw.edu.au

Flora D. Salim
University of New South Wales
Sydney, Australia
flora.salim@unsw.edu.au

ABSTRACT

The next point-of-interest (POI) recommendation task is to predict users' immediate next POI visit given their historical data. Location-based social network data, which is often used for the next POI recommendation task, comes with challenges. One frequently disregarded challenge is how to effectively use the abundant contextual information present in location-based social network data. Previous methods are limited by their numerical nature and fail to address this challenge. In this paper, we propose a framework that uses pretrained large language models to tackle this challenge. Our framework allows us to preserve heterogeneous location-based social network data in its original format, hence avoiding the loss of contextual information. Furthermore, our framework is capable of comprehending the inherent meaning of contextual information due to the inclusion of commonsense knowledge. In experiments, we test our framework on three real-world location-based social network datasets. Our results show that the proposed framework outperforms the state-of-the-art models in all three datasets. Our analysis demonstrates the effectiveness of the proposed framework in using contextual information as well as alleviating the commonly encountered cold-start and short trajectory problems. Our source code is available at: https://github.com/neolifer/LLM4POI

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Large language models, Point-of-interest recommendation

ACM Reference Format:
Peibo Li, Maarten de Rijke, Hao Xue, Shuang Ao, Yang Song, and Flora D. Salim. 2024. Large Language Models for Next Point-of-Interest Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3626772.3657840

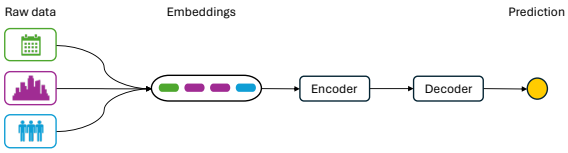
SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657840

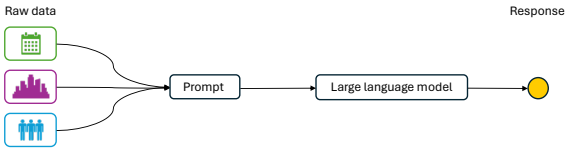
1 INTRODUCTION

Location-based social networks (LBSNs) have experienced massive growth, capitalizing on developments in mobile and localization techniques, as they provide rich location-based geo-information. Next Point-of-interest (POI) recommendation, as one of the applications that use LBSN data, predicts users' next POI visit, given their historical trajectories. Existing next POI methods [18, 30, 32, 33] focus on the short trajectory and cold-start problem, where users with a small amount of data and short trajectories are harder to predict. While these methods alleviate the short trajectory and cold-start problem, they do not fully explore the potential of LBSN data. In particular, the rich contextual information contained in LBSN data has the potential to precisely model users' behavior. By using contextual information, we can understand the data in a way beyond statistics and even derive patterns that do not exist in the data. For example, the data showing a user who frequently visits college buildings during teaching periods could indicate the user's identity as a student or a college staff. Therefore, it is likely that this user will behave differently during vacation between teaching periods.

To exploit such contextual information in LBSN data, there are some substantial **challenges**: **(i)** How to extract the contextual information from the raw data? And **(ii)** How to connect contextual information with commonsense knowledge to effectively benefit next POI recommendation? Here, we consider *contextual information* as time, POI category, and geo-coordinates. And we define *commonsense knowledge* in the context of the next POI recommendation as the capability to understand the semantics of contextual information without additional data, and to connect certain joint patterns of contextual information with behaviors in the real world. Existing next POI recommendation methods [18, 30, 32, 33] have two important **limitations** when dealing with contextual information: **(i)** Due to their numerical nature, they have to transform heterogeneous LBSN data into numbers. For example, POI categories are often encoded from text into IDs. This transformation can result in a loss of inherent meaning associated with contextual information. **(ii)** They rely exclusively on statistics and human designs to understand contextual information and lack an understanding of the semantic concepts provided by the contextual information.



(a) Numeric paradigm.



(b) Large language model-based paradigm.

Figure 1: Comparison of two paradigms for the next POI task: (a) a typical numerical paradigm and (b) the proposed language model-based paradigm.

Large language models (LLMs) have demonstrated capabilities in a variety of tasks. Question-answering, in particular, has benefited from the commonsense knowledge embedded in LLMs [1, 36]. LLMs have a basic grasp of the concepts in daily life and can respond to users' questions using these concepts. Inspired by this and the textual nature of LBSN data, leveraging LLMs for the next POI recommendation task seems a natural step. In our work, we adopt the pretraining and fine-tuning paradigm, and fine-tune pretrained LLMs on LBSN data. As we will see below, by doing so, we are able to use a single LLM to deal with all types of LBSN data and better use contextual information.

More specifically, to address **challenge (i)**, we transform the next POI task into a question-answering task. We convert text-formed raw data into prompts constructed by blocks of sentences. Each block serves as a different module, and each sentence in it contains the necessary information for that module in its original format. Therefore, all heterogeneous data can be fed into a single model with tokenization that still keeps the contextual information. As illustrated in Figure 1, unlike traditional numerical methods, where data needs to be transformed and fed into different embedding layers, our method allows the data to be directly used in its original format. We also propose a notion of trajectory similarity based on prompts, which is used for the cold-start problem.

For **challenge (ii)**, we use pretrained LLMs that have been trained on a large corpus with rich commonsense knowledge. The contextual information in the tokenized data can be understood with its inherent meaning rather than being treated just as a code. As an example, in Table 1, we present the POI category names in a real-world dataset, categorized by their context, as done by ChatGPT.[1] This demonstrates that LLMs are capable of understanding the inherent meaning of contextual information in LBSN data.

Contributions. The main contributions of our work are as follows:

(1) We propose a framework to use pretrained large language models for the next POI recommendation task, which brings commonsense knowledge for making use of the rich contextual information in the data. To the best of our knowledge, we are the first to fine-tune language models on standard-sized datasets to exploit commonsense knowledge for the next POI recommendation task.
(2) We propose a prompt-based trajectory similarity to combine information from historical trajectories and trajectories from different users. With that, our proposed recommendation model is able to alleviate the cold-start problem and make predictions with improved accuracy over trajectories of various lengths.
(3) We conduct an extensive experimental evaluation on three real-world LBSNs datasets, which shows that our proposed next POI recommendation model substantially outperforms state-of-the-art next POI recommendation models in all three datasets.

[1]We provide ChatGPT with the dataset file and ask it to find the unique POI category names. Then we use the prompt "Can you list the category names that have intersections by their context?" to get the content in Table 1. https://chat.openai.com

Table 1: POI categories in a real-world dataset summarized by ChatGPT.

Category	Subcategories/Category Names
Food and Dining	Restaurant (General) **Specific Restaurants**: American, Asian, Italian, Mexican, Korean, Thai, Mediterranean, Caribbean **Specific Food Types**: Seafood Restaurant, BBQ Joint, Steakhouse, Pizza Place, **Other Dining**: Café, Bistro, Diner, Bakery, Food Truck, Deli / Bodega, Dessert Shop
Beverages	Bar, Beer Garden, Coffee Shop, Brewery, Tea Room, Juice Bar
Accommodations	Hotel, Motel, Hostel, Bed and Breakfast
Shopping and Retail	Department Store, Clothing Store, Electronics Store, Bookstore, Market, Mall, Miscellaneous Shop
Outdoor and Recreation	Park, Beach, Zoo, Garden, Plaza, Other Great Outdoors, Playground, Campground
Arts and Entertainment	Museum, Art Museum, Theater, Cinema, Concert Hall, Music Venue, Art Gallery, Comedy Club, Performing Arts Venue
Health and Fitness	Gym / Fitness Center, Spa / Massage, Medical Center, Yoga Studio
Travel and Transport	Airport, Train Station, Bus Station, Subway, Ferry, Taxi
Educational Institutions	School, University, Library, Museum, College Academic Building
Professional and Office	Office, Corporate Building, Conference Room
Residential	Residential Building (Apartment / Condo), Home (private), Housing Development
Cultural and Religious	Church, Temple, Shrine, Synagogue

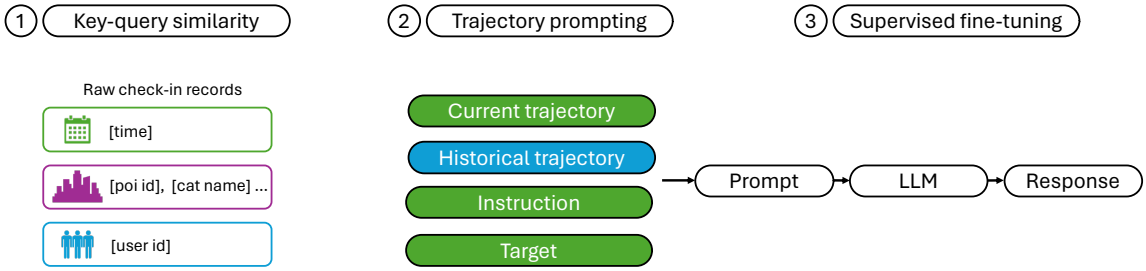


Figure 2: Our overall large language model-based framework for next POI recommendation.

2 RELATED WORK

2.1 Next POI Recommendation

Sequence-based models. Early work on next POI recommendation often treated the next POI recommendation as a sequential recommendation task. Therefore, methods that had been widely used for other sequential recommendation tasks were adapted. For instance, the next POI recommendation task was first introduced by Cheng et al. [4], and they adapted FMPC [21], implementing a localized region constraint where only neighborhood locations are considered for each user. He et al. [14] combined the personalized Markov chain with the latent pattern by incorporating the softmax function. However, these methods are less capable of capturing complex sequential patterns compared to deep neural networks.

Subsequent work has begun to apply RNN-based models with the rise of deep learning. Kong and Wu [17] proposed HST-LSTM, where they added spatial-temporal factors into LSTM gates to guide the learning and further employed a hierarchical extension to model the periodicity of the visit sequence. LSTPM [22] employed three LSTM modules, utilizing non-local neural operations and a short-term preference modeling module using geo-dilated LSTM. PLSPL [27] combined an embedding layer with the attention mechanism to learn long-term preferences and leveraged two LSTM models to model short-term preferences at both the location and category levels. STAN [19] uses a multimodal embedding layer to learn the representation of user, location, time, and spatial-temporal effect, with a bi-layer attention architecture to learn the explicit spatial-temporal relevance within the trajectories. CFPRec [33] focused on the multi-step future plan of users by adopting an attention mechanism to extract future references from past and current preference encoders that are transformer and LSTM encoders. These sequence-based models often become confined to a local view and also suffer from short trajectories and the cold-start problem where inactive users have limited data. Our method deploys key-query similarity, which allows us to combine information from different users, alleviating the cold-start problem.

Graph-based models. More recently, graph-based methods have been incorporated to address the limitations of sequence-based models. STP-UDGAT [18] were the first to use a graph attention network [24], enabling users to selectively learn from others in a global view. Zhang et al. [33] proposed a hierarchical multi-task graph recurrent network (HMT-GRN) to learn user-POI and user-region distribution, employing a GRN to replace the LSTM unit to learn both sequential and global spatial-temporal relationships between POIs. DRGN [26] investigated the intrinsic characteristics of POIs by learning disentangled representation from both distance-based and transition-based relation graphs through a GCN layer. GETNEXT [32] addressed the cold start problem by exploiting collaborative signals from other users and proposing a global trajectory flow map and a novel graph-enhanced transformer model. STHGCN [30] alleviated cold-start issues by constructing a hypergraph to capture higher-order information, including user trajectories and collaborative relations. Although graph-based models handle the cold-start problem, they are not capable of combining contextual information with commonsense knowledge. Our method avoids contextual information loss by trajectory prompting, and the LLMs that we use contains commonsense knowledge to understand contextual information.

2.2 LLMs for Time-series Data

LLMs have proven to be effective for time-series data. The study by SHIFT [28] approached human mobility forecasting as a language translation problem rather than a traditional time-series problem, utilizing a sequence-to-sequence language model complemented by a mobility auxiliary branch. AuxMobLCast [28] further investigated prompt engineering on time-series data. LLM4TS [2] employs a two-stage fine-tuning approach, initially applying supervised fine-tuning to align the LLM with time-series data, followed by downstream task-specific fine-tuning. Inspired by these works, we design trajectory prompting specifically for LBSN data, allowing us to transform the next POI recommendation task into a question-answering task.

2.3 LLMs for Recommender Systems

Recently, many works have adopted LLMs on recommender systems. Zhang and Wang [34] designed multiple prompt templates for different perspectives of news data and did prompt-learning BERT [10] to produce binary answers to templates. Then multi-prompt ensembling was applied to get final predictions. Harte et al. [13] proposed three approaches to leverage LLMs for sequential recommendation. They first compute the embeddings of items and then make recommendations based on the similarity of item embeddings. They also directly fine-tune LLMs to do a prompt completion where the prompt contains a list of item names without the last item and LLMs are asked to complete the prompt with the name of the last product. They also enhanced existing sequential models with embeddings from LLMs. Wang et al. [25] applied in-context learning with LLMs for next POI recommendation. Our method not only fine-tune LLMs for the next POI recommendation, but also

has carefully designed task-specific trajectory similarity to further utilize the power of LLM.

3 PROBLEM DEFINITION

The research problem that we address in this paper is to fine-tune LLMs for the task of next POI recommendation. The problem can be formalized as follows. Consider a dataset $\mathcal{D}$ of user check-in records. Each record is represented by a tuple $q = (u, p, c, t, g)$, where:

- u denotes a user from the set $U = \{u_1, u_2, \ldots, u_N\}$, where N is the total number of users;
- p is a point of interest (POI) from the set $P = \{p_1, p_2, \ldots, p_M\}$, where M is the number of distinct POIs;
- c specifies the category of the POI;
- t represents the timestamp of the check-in; and
- g signifies the geometric coordinate of the POI.

Given a time interval Δt, trajectories for a user u are formed by splitting the check-in records based on this interval. Each trajectory T_i^u up to timestamp t for user u is given by:

$$T_i^u(t) = \{(p_1, c_1, t_1, g_1), \ldots, (p_k, c_k, t_k, g_k)\},$$

where $t_1 < t_2 < \cdots < t_k = t$ and $t_k - t_1 \leq \Delta t$.

Given this set of historical trajectories $\mathcal{T}_u = \{T_1^u, T_2^u, \ldots, T_L^u\}$ for user u, where L represents the number of trajectories for u, the objective is to predict the POI p_{k+1} for a new trajectory $T'^u_i(t)$, where user u will check in at the immediate subsequent timestamp t_{k+1}.

4 METHODOLOGY

The overall framework of our work is presented in Figure 2. Our method includes three components: trajectory prompting, key-query similarity, and supervised fine-tuning for LLMs. First, the raw data is used to construct the prompt and compute the prompt-based key-query similarity. Trajectory prompting uses both raw data and the key-query similarity to form the prompts for the LLM. The LLM is then trained with supervised fine-tuning using the prompts.

4.1 Trajectory Prompting

Inspired by [20, 29, 37], we propose trajectory prompting to convert sequences of user check-in data into a natural language question-answering format for LLMs to follow the instruction from the prompt and generate the POI recommendation. This transformation is crucial in leveraging the power of pretrained LLMs. The idea of trajectory prompting is to unify heterogeneous LBSN data into meaningful sentences that can be fed into LLMs. Specifically, we construct prompts by designing different blocks of sentences for their respective purposes. As shown in Table 2, a prompt consists of the current trajectory block, the historical trajectory block, the instruction block, and the target block.

There are *check-in record* sentences for both the current trajectory block and the historical trajectory block. These sentences contain the necessary information in a check-in record (e.g., user ID, timestamp, POI category name, POI category ID). Specifically, for each check-in record $q = (u, p, c, t, g)$, we form the sentence as "At [time], user [user id] visited POI id [poi id] which is a/an [poi category name] with category id [category id]." We do not include geo-coordinates in the sentence to save the number of tokens and we also find that LLMs, without specifically fine-tuning on map data, are not able to distinguish geo-coordinates well. The check-in records then form trajectories. Note that for the current trajectory block, there will only be one trajectory from the current user, and there can be multiple trajectories from arbitrary users for the historical trajectory block.

Table 2: Structure of prompts and check-in record. Red indicates the current trajectory block. Purple indicates the historical trajectory block. Orange indicates the instruction block. Blue indicates the target block.

prompt	<question> The following is a trajectory of user [user id]: [check-in records]. There is also historical data: [check-in records]. Given the data, at [time], which POI id will user [user id] visit? Note that POI id is an integer in the range from 0 to [id range]. <answer>: At [time], user [user id] will visit POI id [poi id].
check-in record	At [time], user [user id] visited POI id [poi id] which is a/an [poi category name] with category id [category id].

The *current trajectory block* provides information for the current trajectory, excluding the last entry. The *historical trajectory block* incorporates historical information from both the current user and other users who have similar behavior patterns to the current user, which is used for dealing with short trajectory and cold start problems. The details of selecting historical trajectories will be explained in Section 4.2. The *instruction block* guides the model on what to focus on and also reminds the model of the range of POI IDs since the POI IDs generated by LLMs are not by simple argmax over the output probabilities of LLMs that are for the entire vocabulary. The *target block* contains the timestamp, user ID, and POI ID for the check-in record to be predicted, which serves as the ground truth for fine-tuning and evaluation. The target block is excluded from the input during prediction. We have tested adding POI category information in both the instruction block and target block, expecting to encourage the model to pay more attention to the relation between the POI ID and the POI category. However, it turned out that the performance did not show a significant difference, and the model might have already learned that inner relationship.

Our approach of prompting with blocks of sentences allows us to integrate the heterogeneous LBSN data into meaningful sentences in its original format. The design of using blocks is easy to modify and extend.

4.2 Key-Query Pair Similarity

To capture patterns of users' behaviors from their historical trajectories and different users' trajectories, we propose a key-query pair similarity computation framework that suits trajectories in a natural language format. We treat each trajectory differently based on its respective position in the prompt. When a trajectory is considered for the current trajectory block, it is treated as the key, while any trajectory whose end time is earlier than this trajectory is treated as a query. We compute the similarity for all key-query pairs. Subsequently, we select queries with high similarity values

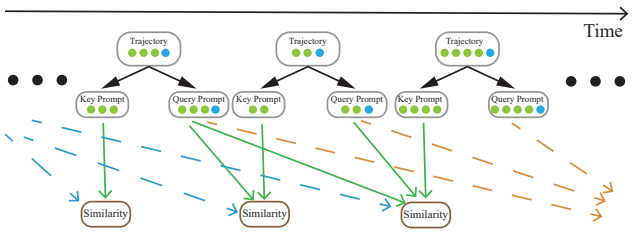


Figure 3: The process of forming and pairing key and query prompts. Each trajectory is made into a key prompt and a query prompt. The key prompt contains the check-in records excluding the last entry of the trajectory, while the query prompt contains the entire trajectory. A key prompt is paired with every query prompt representing the trajectories before the current trajectory.

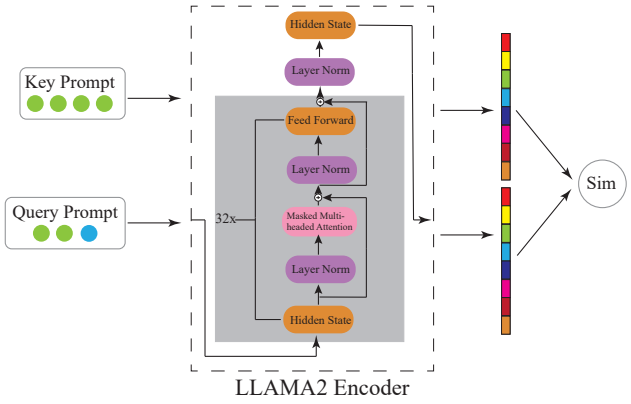


Figure 4: Similarity computation for each pair of key and query. Each pair of key and query prompts are fed into a LLAMA2 encoder separately. We use the last hidden layer embeddings to compute their cosine similarity.

for the historical trajectory block. This approach allows us to incorporate information from other users' trajectories that exhibit similar behavior patterns into the current trajectory.

As illustrated in Figure 3 and 4 , we first form the key and query prompts for every trajectory. Specifically, we use the template for the current trajectory block. For the key prompts, we use the trajectories without their last entry, and for the query prompts, we use the entire trajectories. This is because when the query prompts are used as historical data, the historical trajectories are known, whereas the key prompts are treated as the current trajectories. For every key prompt, we compute the pairwise similarity for it and all the query prompts representing trajectories that have an end time earlier than the start time of the trajectories represented by the key prompt.

For each key and query prompt, we feed it into an LLM encoder and get the embeddings from the last hidden layer. The process can be formulated as:

$$\begin{aligned} h_{k_1} &= \text{Transformer Block}_{(0)}(\text{Tokenizer}(\text{Key})), \\ h_{k_l} &= \text{Transformer Block}_{(l-1)}(h_{k_{l-1}}), \\ E_k &= \text{LN}(h_{k_n}), \end{aligned} \tag{1}$$

$$\begin{aligned} h_{q_1} &= \text{Transformer Block}_{(0)}(\text{Tokenizer}(\text{Query})), \\ h_{q_l} &= \text{Transformer Block}_{(l-1)}(h_{k_{l-1}}), \\ E_q &= \text{LN}(h_{k_n}), \end{aligned} \tag{2}$$

where Tokenizer is used for converting the prompt into a sequence of tokens, Transformer Block$_{(i)}$ represents the i-th Transformer block in the model, LN is the layer normalization, E_k and E_q are the final embeddings of the key and query, respectively.

After getting embeddings for every key and query, we compute the cosine similarity for each key-query pair. Formally,

$$\text{Sim}(\mathbf{E_k}, \mathbf{E_q}) = \frac{\mathbf{E_k} \cdot \mathbf{E_q}}{\|\mathbf{E_k}\|\|\mathbf{E_q}\|}. \tag{3}$$

For each key, we select the top-k queries with the highest similarity to the key. The trajectories represented by these queries are then used in the historical trajectory block for the key trajectory. The process can be expressed as

$$S(\text{key}) = \arg\text{top}_k\{\text{Sim}(q_i, \text{key}) \mid q_i \in Q\}, \tag{4}$$

where Q is the set of queries with the end time earlier than the start time of the key.

4.3 Supervised Fine-tuning

Fine-tuning LLMs can be costly. We apply Parameter-Efficient-Fine-Tuning (PEFT) techniques during the fine-tuning stage.

Low-rank adaptation. We apply LoRA [16], freezing dense layers in LLMs and updating weights with rank decomposition matrices. For a pretrained weight matrix $W_0 \in R^{d\times k}$, we replace weight updates $W_0 + \Delta W$ with low-rank decomposition $W_0 + L_1L_2$, where $L_1 \in R^{d\times r}$, $L_2 \in R^{r\times k}$, and $r \ll min(d, k)$. During training, only B and A receive gradient updates. For $h = W_0x$, the modified forward pass is $h = W_0x + L_1L_2x$. LoRA is applied only to attention layers; MLP layers are frozen during fine-tuning. For an attention layer with 4096 elements, LoRA reduces trainable parameters to 0.78% with a rank of 16, compared to full fine-tuning.

Quantization. To reduce GPU memory usage, we apply quantization techniques [8, 9]. Quantization involves converting high-bit data types into lower-bit ones. We use the 4-bit NormalFloat (NF) proposed by [9], optimal for zero-mean normal distributions in [-1,1]. This process involves rescaling input tensors and applying quantization constants. To reduce memory overhead, double quantization is applied. NF4 is used for storage, while 16-bit BrainFloating (BF) is used for forward and backward passes. Note that the weight gradient is still only computed over the Low-Rank Adaptation (LoRA) parameters.

FlashAttention. Long trajectories and the historical trajectory block in the prompt require long context length from LLMs. A typical context length of 4096 is not enough for our purposes. Therefore, we apply FlashAttention-2 [6, 7], which allows transformers to have long context lengths.

5 EXPERIMENT

5.1 Experimental Setup

5.1.1 Datasets. We conduct experiments on three public datasets: Foursquare-NYC, Foursquare-TKY [31], and Gowala-CA [5]. The

first two datasets, collected over 11 months, comprise data from New York City and Tokyo, sourced from Foursquare. The Gowala-CA dataset, from the Gowalla platform, covers a broader geographical area and time period, encompassing California and Nevada. We utilize data that has been preprocessed as per the methods detailed by Yan et al. [30]. The data is preprocessed as follows: (i) Filter out Points of Interest (POIs) with fewer than 10 visit records in history; (ii) Exclude users with fewer than 10 visit records in history; (iii) Divide user check-in records into several trajectories with 24-hour intervals, excluding trajectories that contain only one check-in record. The check-in records are also sorted chronologically: the first 80% are used for the training set, the middle 10% are defined as the validation set, and the last 10% are defined as the test set. Note that the validation and test set has to contain all users and POIs that appear in the training set. The unseen users and POIs would be removed from the validation and test set.

5.1.2 Baselines. We compare our model with the following baselines:

- FPMC [21]: rooted in the Bayesian Personalized Ranking framework, employs a typical Markov chain combined with matrix factorization to predict location transitions effectively.
- LSTM [15]: A variant of RNN, is designed for processing sequential data. Unlike standard RNNs, LSTMs are capable of capturing both short-term and long-term dependencies in sequential patterns, making them more effective for a range of sequential data tasks.
- PRME [12]: Utilizing a pairwise ranking metric embedding, this personalized ranking model effectively learns sequential transitions of POIs along with capturing user-POI preferences in latent space.
- STGCN [35]: Based on LSTM, this model incorporates gating mechanisms to effectively model temporal and spatial intervals in check-in sequences, thereby capturing both short-term and long-term user preferences.
- PLSPL [27]: This recurrent model employs an attention mechanism to learn short-term preferences and two parallel LSTM structures for long-term preferences, integrating both through a user-specific linear combination.
- STAN [19]: Leveraging a bi-layer attention architecture, STAN aggregates spatio-temporal correlations within user trajectories, learning patterns across both adjacent and non-adjacent locations as well as continuous and non-continuous visits.
- GETNext [32]: A transformer-based approach, GETNext uses a global trajectory flow map that is user-agnostic to enhance next-POI predictions, alongside proposing a GCN model for generating effective POI embeddings.
- STHGCN [30]: Constructing a hypergraph to capture inter and intra-user relations, STHGCN proposes a hypergraph transformer and solves the cold-start problem.

5.1.3 Our Models. In our experiments we consider three versions of what we call "our model": (i) LLAMA2-7b: Our model, using prompts only the current trajectory block without the historical trajectory block, where we use Llama-2-7b-longlora-32k [3, 23] as our base LLM. (ii) LLAMA2-7b*: A variation on LLAMA2-7b where we use prompts with the historical trajectory block without applying key-query similarity, where only the historical trajectories from the current users are considered. (iii) LLAMA2-7b**: A second variation on LLAMA2-7b where we use prompts with the historical trajectory block combined with key-query similarity, where historical trajectories from both the current users and other users are considered. Below, unless stated otherwise, when we say "our model," we refer to the LLAMA2-7b** variant.

5.1.4 Evaluation Metrics. For evaluation we regard the next POI recommendation task as a top-1 recommendation problem. The scenario we have in mind is one with "extreme position bias" [11], where only a small amount of information can be presented to the user during a single interaction. For example, a user who is on a business trip wishes to explore the new city before the meeting. In such a scenario, the user does not have the leisure to review multiple options. This is a scenario that is usually considered in next POI recommendation, often as the primary scenario [see, e.g., 30, 32]. Given this choice of scenario, our approach to the task as a type of question-answering problem is a natural fit. We prioritize the delivery of the most pertinent and contextually suitable recommendation, mirroring the objective of providing a single, correct answer to a user's query.

The evaluation metric we use is Accuracy@1. It looks at what proportion of the test items would have been retrieved with the top-1 recommended list and can be formalized as:

$$\text{Acc@1} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}(\text{rank} \leq 1), \tag{5}$$

where $\mathbb{1}$ is the indicator function. Rank is the rank of the order of the correct prediction in the recommendation list. A larger value represents better performance.

5.1.5 Implementation Details. For fine-tuning, we use a constant learning rate schedule with a learning rate of 2×10^5, combined with a warm-up of 20 steps, a weight decay of 0, a batch size of 1 per GPU, and a sequence length of 32,768 tokens. For each dataset, we fine-tune the model for 3 epochs. We use approximately 300 historical check-in records to construct the historical trajectory block in the prompt. Our experiments are conducted on servers with Nvidia A100 GPUs.

5.2 Main Results

We compare the performance of our models and the baselines on three datasets, as shown in Table 3.

Our model substantially outperforms all baselines. Specifically, we observe improvements of 23.3%, 2.8%, and 19.3% in top-1 accuracy on the NYC, TKY, and CA datasets, respectively, compared to the state-of-the-art STHGCN. Models utilizing historical data perform better than those that do not, and those incorporating data from other users see further performance boosts, highlighting the significance of short trajectory and cold-start problems in next POI recommendation tasks. All models except STHGCN perform best in NYC, with noticeable performance drops in TKY and CA. NYC has the smallest number of users and POIs but a larger number of POI categories than the other datasets, suggesting it has the easiest data to learn. In contrast, CA covers a much wider area than NYC

Table 3: Performance comparison in terms of Acc@1 on three datasets. ✓and × in the History and Other Users columns indicate whether the model uses historical data or data from other users, respectively.

Model	History	Other users	NYC	TKY	CA
			Acc@1	Acc@1	Acc@1
FPMC	×	×	0.1003	0.0814	0.0383
LSTM	×	×	0.1305	0.1335	0.0665
PRME	×	×	0.1159	0.1052	0.0521
STGCN	×	×	0.1799	0.1716	0.0961
PLSPL	×	×	0.1917	0.1889	0.1072
STAN	×	×	0.2231	0.1963	0.1104
GETNext	✓	✓	0.2435	0.2254	0.1357
STHGCN	✓	✓	0.2734	0.2950	0.1730
LLAMA2-7b	×	×	0.2356	0.1517	0.1016
LLAMA2-7b*	✓	×	0.3171	0.2836	0.1683
LLAMA2-7b**	✓	✓	**0.3372**	**0.3035**	**0.2065**

and TKY, leading to data scarcity and significantly lower model performance.

5.3 Analysis

5.3.1 User Cold-start Analysis. Our approach incorporates the historical trajectory block and key-query trajectory similarity to tackle the cold-start problem by leveraging knowledge from diverse users. User activity status greatly impacts model performance, with active users providing more historical data and generally easier behavior patterns to learn. To assess our method's effectiveness with inactive users, we categorize users into inactive, normal, and very active groups based on the number of trajectories in the training set, designating the top 30% users ranked by their number of trajectories as very active and the bottom 30% as inactive.

We compare our model with STHGCN, which is designed to address the cold-start problem and has shown effectiveness with inactive users. The comparison, shown in Table 4, reveals our model significantly improves performance for inactive users, more than doubling the baseline in NYC and increasing by over half in TKY and CA. This improvement underscores our method's ability to leverage information from similar users effectively, especially for those with limited historical data.

Table 4: User cold-start analysis on the NYC, TKY, and CA datasets.

User groups	Model	NYC	TKY	CA
		Acc@1	Acc@1	Acc@1
Inactive	STHGCN	0.1460	0.2164	0.1117
Normal	STHGCN	0.3050	0.2659	0.1620
Very active	STHGCN	0.3085	0.3464	0.2040
Inactive	Ours	0.3417	0.3478	0.2132
Normal	Ours	0.3841	0.3516	0.2057
Very active	Ours	0.3088	0.2727	0.1920

Interestingly, our model performs better for inactive users than very active ones, contrasting the baselines' better performance with very active users. This suggests that the similar trajectories we identify for very active users are often from their own, leading to less diverse behavior patterns and less effective prediction of difficult data points. The less significant improvement in TKY and CA compared to NYC might be due to the higher user count in these datasets, limiting the collaborative information our method can utilize within the model's context length constraints.

5.3.2 Trajectory Length Analysis. The varying lengths of trajectories in the next POI recommendation task, reflecting different user behaviors, pose another significant challenge. Short trajectories, with their limited spatio-temporal information and non-significant patterns, are particularly challenging, especially those with only one or two check-ins. While long trajectories contain more information, extracting useful patterns from them is also difficult. Our method's effectiveness varies with the trajectory length due to the context length limit, allowing fewer historical trajectories to be added for long trajectories compared to short ones. To explore the trade-off between long and short trajectories, we rank the lengths of trajectories in the test set, defining the top 30% as long trajectories and the bottom 30% as short trajectories, with the rest classified as middle trajectories.

Comparing our method to STHGCN, Table 5 highlights our substantial improvement for short trajectories in NYC. We achieve an improvement of 24.4% in top-1 accuracy for short trajectories and 31.6% for middle trajectories in NYC, demonstrating our method's strong capability to integrate historical data for short trajectories. Interestingly, while we perform better for short trajectories in NYC, the opposite is true in TKY and CA. However, our model's performance does not vary significantly across different trajectory lengths, indicating a balanced trade-off between long and short trajectories.

Table 5: Trajectory length analysis on the NYC, TKY, and CA datasets.

Trajectory types	Model	NYC	TKY	CA
		Acc@1	Acc@1	Acc@1
Short	STHGCN	0.2703	0.2787	0.1727
Middle	STHGCN	0.2545	0.2823	0.1785
Long	STHGCN	0.3184	0.3116	0.1742
Short	Ours	0.3364	0.2876	0.1955
Middle	Ours	0.3350	0.3013	0.1998
Long	Ours	0.3271	0.3083	0.2037

5.3.3 Number of Historical Data Variants. An important factor of our method is the number of historical trajectories used for the historical trajectory block. Since there is a token limit for the prompt, we cannot use all the historical trajectories but only the ones with the highest similarity to the current trajectory. Because trajectories vary in length, we use the number of check-in records in the historical trajectory block to evaluate the effect of the number of historical trajectories on performance.

We compare our model trained and evaluated with different numbers of historical check-in records. As shown in Table 6, we observe that the model archives the best performance in NYC with 100 historical check-in records and decreases as the number of check-in records grows. On the other hand, the model has the best performance in TKY with 300 historical check-in records, and the performance is positively correlated to the number of historical

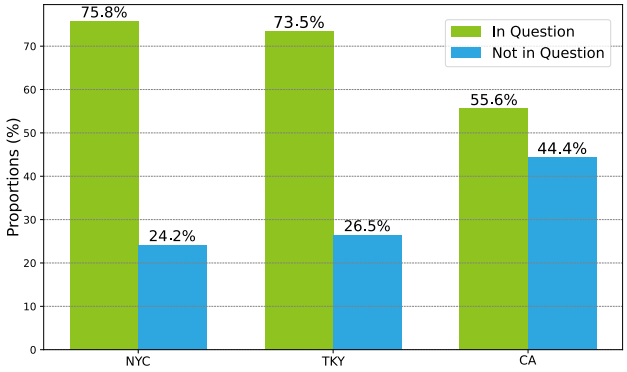


Figure 5: The proportion of test set prompts where the answer POI IDs are included within the questions, in their respective datasets.

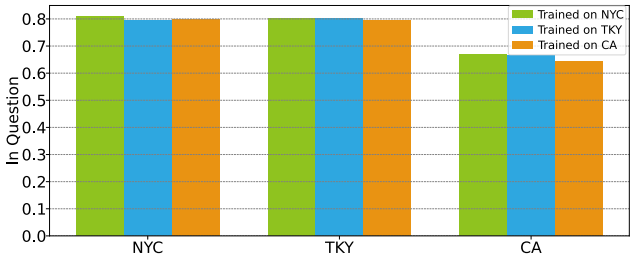


Figure 6: The proportion of test set prompts where the answer POI IDs are included within the questions, in their respective datasets, for the correct predictions made by the models trained on the NYC, TKY, and CA.

check-in records. Note that the performance in TKY only improves by a tiny margin when the number of historical check-in records increases from 200 to 300. The model performance remains closely in CA, with different numbers of different historical check-in records. The results indicate that using more historical data does not necessarily improve the model performance. We can use less historical data to reduce the token size of prompts but still achieve competitive model performance, which also speeds up the training and inference.

Table 6: Analysis on NYC, TKY, and CA dataset for LLAMA2-7b trained on prompts with different numbers of historical trajectories.**

Number of historical check-ins	NYC	TKY	CA
	Acc@1	Acc@1	Acc@1
100	0.3420	0.2166	0.2056
200	0.3400	0.3023	0.2035
300	0.3372	0.3035	0.2065

5.3.4 Generalization to Unseen Data Analysis. The fact that our approach does not rely on a linear classifier to output the POI IDs but predicts with purely language modeling allows us to evaluate our models, fine-tuned on one dataset, on unseen data without any further training. We fine-tune our models on one of the NYC, TKY, and CA and then evaluate them on the rest.

As shown in Table 7, interestingly, the models achieve competitive performance on datasets which they are not fine-tuned on.

Table 7: The models are LLAMA2-7b trained only on one of the NYC, TKY, and CA datasets and evaluated on the rest.**

Trained on	NYC	TKY	CA
	Acc@1	Acc@1	Acc@1
NYC	0.3372	0.2594	0.1885
TKY	0.3463	0.3035	0.1960
CA	0.3344	0.2600	0.2065

Table 8: The results for LLAMA2-7b tested with different prompts.**

Prompts	NYC	TKY	CA
	Acc@1	Acc@1	Acc@1
w/ context	0.3372	0.3035	0.2065
w/o context	0.3310	0.2840	0.1935

Specifically, the model trained in NYC has a top-1 accuracy lower than STHGCN for TKY and better than the state-of-the-art models in CA. The model trained on TKY performs even better in NYC than in NYC and is also better than the state-of-the-art models in CA. The model trained on CA is better than the state-of-the-art models in NYC and has a top-1 accuracy lower than STHGCN for TKY. This suggests that our models generalize well to unseen data.

We look into the prompts to further investigate the reason for the generalization ability. As shown in Figure 5, we find that 75.8%, 73.5%, and 55.6% of the prompts in test sets of NYC, TKY, and CA, respectively, have the answer POI IDs appear within the questions. These portions are positively correlated to the overall model performance on each dataset. We also observe that the models trained on different datasets behave closely on each dataset, as shown in Figure 6. Specifically, in their correct predictions, the portions of the prompts with answer POI IDs that appear within the questions are almost identical. This suggests: (i) the reason why our models have good performance is that our historical trajectory block combined with key-query similarity accurately captures the useful information from the current user's historical trajectories and other users' trajectories; (ii) the models learn to extract the correct POI IDs from the prompts directly, which helps them to generalize to unseen data.

5.3.5 Contextual Information Analysis. What distinguishes our method from other works is that we use models embedded with commonsense knowledge to exploit contextual information. To evaluate how contextual information helps our model, we replace the POI category names in the prompts with texts of the same lengths with no meaning to mask the contextual information. We analyze the model on the NYC, TKY, and CA datasets and compare the results with prompts with and without contextual information.

Table 8 shows the overall results for the models trained on NYC, TKY, and CA and tested with different prompts. Model performance drops by a small margin when we remove the contextual information from POI category names in NYC. For TKY and CA, the model performance decreases by 6.4% and 6.2%, respectively.

To further investigate the effectiveness of the contextual information, we evaluate the model on different levels of user activity with and without contextual information, similar to Section 5.3.1. From Table 9, we observe a significant drop in performance for inactive and normal users and an increase in performance for active

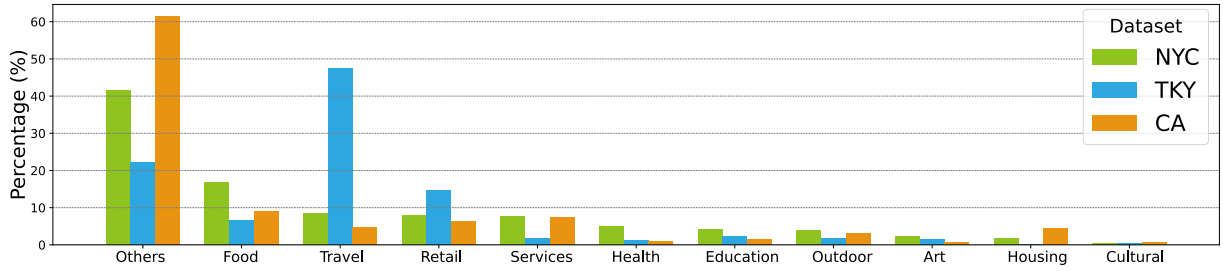


Figure 7: Statistics for different POI categories in the NYC, TKY, and CA datasets.

Table 9: The results for LLAMA2-7b tested with different prompts in terms of users with different levels of activity.**

User groups	Prompts	NYC	TKY	CA
		Acc@1	Acc@1	Acc@1
Inactive	w/ context	0.3417	0.3478	0.2132
Normal	w/ context	0.3841	0.3516	0.2057
Very active	w/ context	0.3088	0.2727	0.1920
Inactive	w/o context	0.3493	0.2751	0.2148
Normal	w/o context	0.3623	0.2715	0.1951
Very active	w/o context	0.3025	0.2884	0.1732

users in TKY. The performance increases for inactive users and decreases for normal and active users in NYC and CA. The contextual information affects the model performance in two opposite directions for datasets in two different countries. Figure 7 shows the stats for POI categories of NYC, TKY, and CA. We can see that the distributions of POI categories are indeed different between the two countries, where almost half of the POIs in TKY are for travel and transportation. Because POI contextual information differs from two types of datasets, the models behave differently from TKY to NYC and CA. This supports the models' understanding of the inherent meaning behind POI contextual information for the three datasets.

5.3.6 Effect of Different Components. We consider the performance of our model as the joint effect of (i) the historical trajectory block; (ii) key-query similarity; (iii) contextual information. To investigate the effect of each component, we remove the historical trajectory block in the prompt and only put the current user's historical trajectories in the historical trajectory block, respectively.

As shown in Table 10, the results suggest that each component contributes to the full model performance. Specifically, the historical trajectory block plays a critical role, where the top-1 accuracy drops as much as 50% in TKY and CA. Because without any historical trajectories, the model suffers from short trajectories, which the datasets mostly consist of. With key-query similarity being removed, the collaborative information from other users is missing from the historical block, which leads to the inability to deal with the cold-start problem. On top of these two components, contextual information provides a further understanding of the data, improving the model's performance.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a framework to deploy large language models for the next point-of-interest recommendation task, which is the first to utilize models with commonsense knowledge for the task. We developed trajectory prompting to transform the task into a question-answering. We also introduce key-query similarity to alleviate the cold-start problem.

Table 10: Ablation study results for LLAMA2-7b over three datasets.**

Model	NYC	TKY	CA
	Acc@1	Acc@1	Acc@1
Full model	0.3372	0.3035	0.2065
w/o history	0.2356	0.1517	0.1016
w/o similarity	0.3171	0.2836	0.1683
w/o context	0.3310	0.2840	0.1935

Our comprehensive experiments conducted on three real-world datasets show that we outperform all baseline models by a large margin. Our analysis supports that our method is able to handle the cold-start problem and various lengths of trajectories. It also shows the effectiveness of contextual information in our model. We have also shown the potential of developing foundation models for the next POI recommendation task, given the ability of large language models to generalize to unseen data.

Because of the nature of large language models, we have limitations with efficiency regarding model training and inference time. Our design of the prompt is also limited by the context length and pretrained corpus of the model, e.g., the deprecation of geo-coordinates. For future work, we plan to address the limitations just mentioned. In addition, we will investigate chain-of-thought reasoning for the next point-of-interest recommendation task, to both further boost the performance and provide explanations for the prediction. Another line of future work is to extend our models to scenarios without the extreme focus on a single best item, going beyond a question-answering context that we focused on here.

Acknowledgements. This research was conducted by the ARC Centre of Excellence for Automated Decision-Making and Society (CE200100005), and funded by the Australian Government through the Australian Research Council. In addition, this research was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI), which is supported by the Australian Government. Peibo Li is supported by Google Conference Scholarships (APAC). Maarten de Rijke was partially supported by the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-shot Learners. In *Advances in neural information processing systems*, Vol. 33. 1877–1901.

[2] Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. 2023. LLM4TS: Two-Stage Fine-Tuning for Time-Series Forecasting with Pre-Trained LLMs. arXiv preprint arXiv:2308.08469.

[3] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. *arXiv:2309.12307*.

[4] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, Francesca Rossi (Ed.). IJCAI/AAAI, 2605–2611.

[5] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and Mobility: User Movement in Location-based Social Networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1082–1090.

[6] Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arXiv preprint arXiv:2307.08691*.

[7] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-efficient Exact Attention with IO-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.

[8] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit Optimizers via Block-wise Quantization. In *International Conference on Learning Representations*.

[9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[11] Yaron Fairstein, Elad Haramaty, Arnon Lazerson, and Liane Lewin-Eytan. 2022. External Evaluation of Ranking Models under Extreme Position-Bias. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, New York, NY, USA, 252–261.

[12] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, and Yeow Meng Chee. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence*. ACM, 2069–2075.

[13] Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging Large Language Models for Sequential Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1096–1102.

[14] Jing He, Xin Li, Lejian Liao, Dandan Song, and William Cheung. 2016. Inferring a Personalized Next Point-of-interest Recommendation Model with Latent Behavior Patterns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30. Issue: 1.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*.

[17] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-short Term Memory Network for Location Prediction.. In *IJCAI*, Vol. 18. 2341–2347. Issue: 7.

[18] Nicholas Lim, Bryan Hooi, See-Kiong Ng, Xueou Wang, Yong Liang Goh, Renrong Weng, and Jagannadan Varadarajan. 2020. STP-UDGAT: Spatial-Temporal-Preference User Dimensional Graph Attention Network for Next POI Recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 845–854.

[19] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. 2021. Stan: Spatio-Temporal Attention Network for Next Location Recommendation. In *Proceedings of the web conference 2021*. 2177–2185.

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning Transferable Visual Models from Natural Language Supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[21] 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.

[22] Ke Sun, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. Where to Go Next: Modeling Long- and Short-term User Preferences for Point-of-interest Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 214–221. Issue: 01.

[23] 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*.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[25] Xinglei Wang, Meng Fang, Zichao Zeng, and Tao Cheng. 2023. Where Would I Go Next? Large Language Models as Human Mobility Predictors. *arXiv preprint arXiv:2308.15197* (2023).

[26] Zhaobo Wang, Yanmin Zhu, Haobing Liu, and Chunyang Wang. 2022. Learning Graph-based Disentangled Representations for Next POI Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1154–1163.

[27] Yuxia Wu, Ke Li, Guoshuai Zhao, and Xueming Qian. 2020. Personalized Long-and Short-term Preference Learning for Next POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 4 (2020), 1944–1957.

[28] Hao Xue, Flora D. Salim, Yongli Ren, and Charles L.A. Clarke. 2022. Translating Human Mobility Forecasting through Natural Language Generation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1224–1233.

[29] Hao Xue, Bhanu Prakash Voutharoja, and Flora D. Salim. 2022. Leveraging Language Foundation models for Human Mobility Forecasting. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. 1–9.

[30] Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. 2023. Spatio-Temporal Hypergraph Learning for Next POI Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 403–412.

[31] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2014. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2014), 129–142.

[32] 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 (SIGIR '22)*. 1144–1153.

[33] Lu Zhang, Zhu Sun, Ziqing Wu, Jie Zhang, Yew Soon Ong, and Xinghua Qu. 2022. Next Point-of-interest Recommendation with Inferring Multi-step Future Preferences. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*. 3751–3757.

[34] Zizhuo Zhang and Bang Wang. 2023. Prompt Learning for News Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 227–237.

[35] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. 2020. Where to Go Next: A Spatio-tTemporal Gated Network for Next POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2512–2524.

[36] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223*.

[37] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision* 130, 9 (2022), 2337–2348.