Modeling Spatial Trajectories using Coarse-Grained Smartphone Logs

Vinayak Gupta and Srikanta Bedathur

Abstract—Every user carries their smartphones wherever they go – a crucial aspect ignored by the current models for spatial recommendations. In detail, the current approaches learn the points-of-interest (POI) preferences of a user via the standard spatial features *i.e.* the POI coordinates and the social network, and thus ignore the features related to the smartphone usage of a user. Moreover, with growing privacy concerns, users refrain from sharing their exact geographical coordinates as well as their social media activity. In this paper, we present REVAMP, a sequential POI recommendation approach that uses smartphone app-usage logs to identify the mobility preferences of a user. Our work aligns with the recent psychological studies of online urban users which show that their spatial mobility behavior is largely influenced by the activity of their smartphone apps. Specifically, our proposal of coarse-grained data refers to data logs collected in a privacy-conscious manner consisting only of (a) category of smartphone app-used and (b) category of check-in location. Thus, REVAMP is not privy to precise geo-coordinates, social networks, and the specific app being used. Buoyed by the efficacy of self-attention models, REVAMP learns the POI preferences of a user using two forms of positional encodings – absolute and relative – with each extracted from the inter-check-in dynamics in the check-in sequence of a user. Extensive experiments across two large-scale datasets from China show that REVAMP outperforms the state-of-the-art sequential POI recommendation approaches and can be extended to app- and POI-category prediction.

Index Terms—Sequential Recommendation; Smartphone Apps; Spatial and Temporal Data; Self-Attention;

1 Introduction

THE rapid advancements in the smartphone industry and the ubiquitous internet access have led to an exponential growth in the number of available users and internet-based applications. Specifically, smartphones have become increasingly prevalent across the entire human population with up to 345 million units sold in the first quarter of 2021 [1]. Accordingly, the online footprint of a user spans multiple applications with an average smartphone owner accessing 10 smartphone applications (or apps) every day and 30 apps each month [2]. These footprints can be perceived as the digitized nature of the user's proclivity in different domains. Recent research [1], [22] has shown that the online web activity of a user exhibits *revisitation* patterns *i.e.* a user is likely to visit certain apps repetitively with similar time intervals between corresponding visits. [14] and [30] have shown that these online revisitation patterns are analogous to her spatial mobility preferences *i.e.* the current geographical location can influence the web-browsing activities of a user. Moreover, such cross-domain information of app-preferences of a user can be collected without using any personally identifiable information (PII) [3] and thus maintaining the privacy of a user [12], [49], [53]. Thus, to enhance the performance of a points-of-interest (POI) recommendation system, it is crucial to model the app-revisitation users along with their location preferences.

• *V. Gupta and S. Bedathur are with the Department of Computer Science & Engineering, Indian Institute of Technology, Delhi, New Delhi, India E-mail: {vinayak.gupta, srikanta}@cse.iitd.ac.in*

Manuscript received XX XX, 2021.

1. https://www.canalys.com/newsroom/canalys-worldwide-smartphone-market-Q1-2021
2. https://buildfire.com/app-statistics/
3. https://en.wikipedia.org/wiki/Personal_data

1.1 Limitations of Prior Works

Modern POI recommendation approaches [13], [32], [52] utilize the standard features specific to a user and a POI – social network, geo-coordinates, and category classifications of POI – to learn the mobility patterns of a user. The situation has been exacerbated in recent times due to the advent of restrictions on personal data collection and a growing awareness (in some geopolitical regions) about the need for personal privacy [3], [50]. Moreover, current approaches overlook two crucial aspects of urban computing – the exponential growth of online platforms and the widespread use of smartphones. Undeniably, everyone carries and simultaneously uses a smartphone wherever they go. Thus, standard approaches are inappropriate to design POI recommender systems that must capture the location influence on the apps used by a user. To highlight this importance of the POI-app relationship, we plot the category of the app used by all users and the location-ID of the ten popular locations from our Shanghai-Telecom dataset [53] in Figure 1. The plot shows that the check-in locations can influence a user to visit apps of certain categories more than other apps. Moreover, it also shows that this POI influence over the category of the app being accessed is similar across different users.

The correlation between spatial mobility and smartphone use is essential to address the problems related to user demographics [39], [46], trajectory analysis [35], app recommendation [56], and to identify hotspots for network operators [30]. However, utilizing smartphone usage for sequential POI recommendations is not addressed in the past literature. The papers most similar to our work are by [48] and [45]. [48] utilizes a Dirichlet process to determine the next user location, but it completely disregards the user's privacy *i.e.* requires precise geo-coordinates, and [45] is limited to the cold-start recommendation.

1.2 Our Proposal

In this paper, we present **REVAMP**(**Re**lative position **V**ector for **A**pp-based **M**obility **P**rediction), a sequential POI recommendation model that learns the location and app affinities of smartphone users while simultaneously maintaining their privacy needs. Specifically, we consider each check-in as an event involving a smartphone activity, and the physical presence at a POI and REVAMP models the correlation between the smartphone-app preferences and the spatial mobility preferences of a user. Parallelly, to preserve the privacy restrictions, it solely utilizes two aspects of urban mobility: (a) the types of smartphone apps used during a check-in and (b) the category of the check-in location. Thus, the proposed approach is not privy to any kind of identifiable information (PII) related features such as the precise smartphone-app being accessed, *e.g. 'Facebook', 'Amazon' etc.*, the accurate geo-location, inter-check-in distance, or the social network of a user. Buoyed by the success of self-attention [47] models in sequence modeling, REVAMP encodes the dynamic check-in preferences in the user trajectory as a weighted aggregation of all past check-ins. Moreover, to better capture the evolving POI and app preferences, it models the variation between each check-in in the sequence using *absolute* and *relative* positional encodings [5], [42]. Specifically, we embed three properties associated with each check-in– the smartphone app-category, POI-category, and the time of check-in– and model the temporal evolution as the inter-check-in embedding differences, independently across these features. Figure 2 demonstrates how REVAMP embeds and adaptively learns the inter-check-in dynamics between app and POI categories to determine the next check-in location for a user. Moreover, REVAMP grants the flexibility to predict the category of the most likely smartphone app to be accessed and the POI-category at the next check-in. Predicting the app-preferences of a user has limitless applications ranging from smartphone app recommendation and bandwidth modeling by cellular network providers [45], [46], [56]. To summarize, the key contributions we make in this paper are three-fold:

(i) We propose a self-attention-based neural model, called REVAMP, to learn the POI-preferences of a user via the coarse-grained smartphone usage logs. REVAMP returns a ranked list of candidate POIs and the most likely app and POI category for the next check-in.

(ii) REVAMP preserves the privacy needs of a user by learning a personalized sequence encoding independently for each user and is not privy to accurate geo-locations and social networks. Later, it models the evolving spatial preferences using the variations between each check-in in the sequence based on app category, POI category, and time of the check-in.

(iii) Exhaustive experiments over two large-scale datasets from China show that REVAMP outperforms other state-of-the-art methods for sequential POI recommendation, next app, and location-category prediction tasks. Moreover, we perform a detailed analysis of each component of REVAMP, a convergence analysis, and parameter sensitivity to ascertain its practicability.

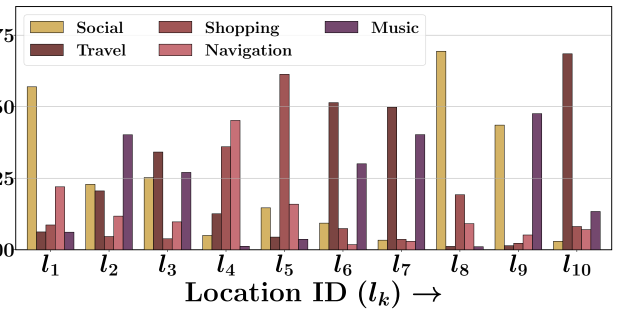


Fig. 1: The probability of a smartphone-app category – among '*Social*', '*Travel*', '*Shopping*', '*Navigation*', and '*Music*' – to be used at 10 most popular locations from Shanghai-Telecom dataset. The plot indicates that the smartphone app-usage depends on the check-in location.

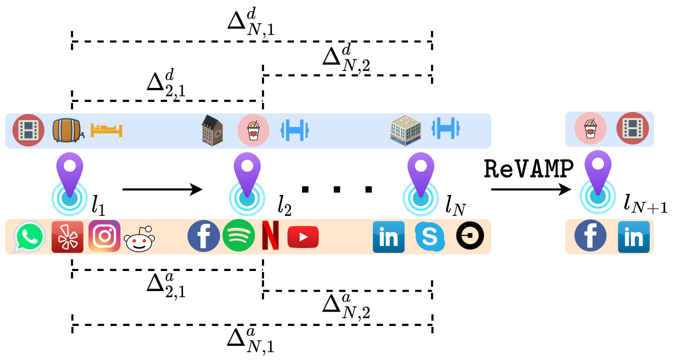


Fig. 2: A schematic diagram of inter-check-in time, app and POI variation-based learning approach of REVAMP. Here, $\Delta^a_{\bullet,\bullet}$ and $\Delta^d_{\bullet,\bullet}$ denote the smartphone-app and POI-based differences respectively.

2 Related Work

In this section, we highlight some relevant works to our paper. It mainly falls into three categories – modeling smartphone and mobility, sequential recommendation, and positional encodings for self-attention models.

2.1 Modeling Smartphone and Mobility

Understanding the mobility dynamics of a user has wide applications ranging from location-sensitive advertisements, social community of user, and disease propagation [8], [33], [35]. Traditional mobility prediction models utilized a function-based learning for spatial preferences but were highly susceptible to irregular events in the user trajectory [7], [32]. Therefore, modern approaches [13], [29], [37] utilize a neural network to model the complex user-POI relationships, geographical features, travel distances, and category distribution. These approaches consider the user trajectory as a check-in sequence and train their model parameters by capturing the influences across different sequences. Other approaches [16], [17], [34], [54] include the continuous-time contexts for modeling the time-evolving preferences of a user. However, prior research has shown that users exhibit *revisitation* patterns on their web activities [1], [22] and these revisitation patterns resonate with the mobility preferences of a user [6], [48]. As per the permissions given by a user to an app, leading corporations, such as Foursquare, utilize smartphone activities to better understand the likes and dislikes of a user to give better POI recommendations [9]. The correlation between spatial mobility and smartphone use is essential to address the problems related to user demographics [39], [46], trajectory analysis [35], app recommendation [56], and to identify hotspots for network operators [30]. However, utilizing smartphone usage for sequential POI recommendations is not addressed in the past literature. The papers most similar to our work are by [48] and [45]. [48] utilizes a Dirichlet process to determine the next user location, but it completely disregards the user's privacy *i.e.* requires precise geo-coordinates, and [45] is limited to *cold-start* POI recommendation rather than sequential recommendations.

2.2 Sequential Recommendation

Standard collaborative filtering (CF) and matrix factorization (MF) based recommendation approaches [19], [46] return a list of most likely items that a user will purchase in the future. However, these approaches ignore the temporal context associated with the preferences *i.e.* it evolves with time. The task of a sequential recommender system is to continuously model the user-item interactions in the past purchases (or check-ins) and predict the future interactions. Traditional sequence modeling approaches such as personalized Markov chains [41] combine matrix factorization with inter-item influences to determine the time-evolving user preferences. However, it has limited expressivity and cannot model complex functions. Neural models such as GRU4Rec [21] utilize a recurrent neural network(RNN) to embed the time-conditioned user preferences which led to multiple developments like GRU4Rec+ [20]. Recent research has shown that including attention [2] within the RNN architecture achieved better prediction performances than standard RNN models even in the case of POI recommendations [28], [34], [55]. However, all these approaches were outperformed by the self-attention-based sequential recommendation models [24], [27]. In detail, the underlying model of [24] is a transformer architecture [47] that embeds user preferences using a weighted aggregation of all past user-item interactions. However, due to largely the heterogeneous nature of data in spatial datasets *e.g.* POI category, geographical distance, *etc.* extending such models for sequential POI recommendation is a non-trivial task.

2.3 Relative Positional Encodings and Self-Attention

The self-attention models are oblivious of the position of events in the sequence and thus, the original proposal to capture the order of events used fixed function-based encodings [47]. However, recent research on positional encodings [5], [42] has shown that modeling the position as a relative pairwise function between all events of a sequence, in addition to the fixed-function encodings, achieves significant improvements over the standard method. Thus, such relative encodings have been used in a wide range of applications – primarily for determining the relative word order in natural language tasks [11], [15] and image order in computer vision problems [4], [51]. Such relative encodings have also been incorporated in item-based recommender systems [27] through *time-interval* based inter-event relevance and in POI recommendation [31] through geographical distance-based variances. However, the former approach cannot be extended to model the heterogeneous nature of smartphone mobility data and the latter requires precise geographical coordinates. Moreover, including the app-category and POI-category-based relevance is not a trivial task as unlike time and distance, these are context-dependent *i.e.* two categories such as '*Burger Joint*' and '*Sushi Restaurant*' differ in terms of the semantic meaning of the category term. Such differences are not explicit and must be learned via natural language embeddings.

3 Problem Formulation

We consider a setting with a set of users, $\mathcal{U}$ and a set of locations (or POIs), $\mathcal{L}$. We embed each POI using a D dimensional vector and denote the embedding matrix as $\boldsymbol{L} \in \mathbb{R}^{|\mathcal{L}| \times D}$. We represent the mobile trajectory of a user u_i as a sequence of check-ins, $e_k^{u_i} \in \mathcal{E}^{u_i}$, with each check-in comprising of the smartphone app and location details. For a better understanding of our model, let us consider a toy sequence with five check-ins to POIs with categories, – '*Bar*', '*Cafe*', '*Burger-Joint*', '*Cafe*', and '*Sushi Restaurant*', while using smartphone apps categories – '*Social*', '*Shopping*', '*Game*', '*Social*', and '*Travel*', respectively. Thus, for this example, REVAMP will use the details of the first four check-ins to predict the last check-in.

Definition 1 (Check-ins)**.** *We define a check-in as a timestamped activity of a user with her smartphone and location details. Specifically, we represent the k-th check-in in $\mathcal{E}$ as $e_k = \{l_k, t_k, \mathcal{A}_k, \mathcal{S}_k\}$ where l_k and t_k denote the POI and check-in time respectively. Here, $\mathcal{A}_k$ denotes the categories set of the smartphone-app accessed by a user $\mathcal{S}_k$ denotes the set of POI categories.*

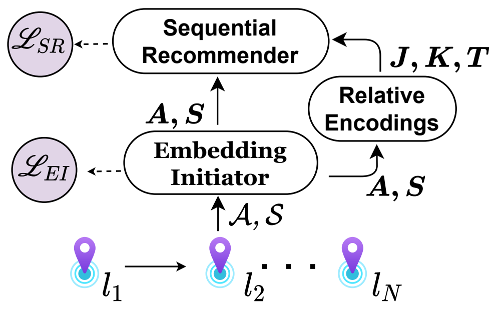
With a slight abuse of notation, we denote a check-in sequence as $\mathcal{E}$ and the set of all app- and location categories till a k-th check-in as $\mathcal{A}_k^* = \bigcup_{i=1}^{k} \mathcal{A}_k$ and $\mathcal{S}_k^* = \bigcup_{i=1}^{k} \mathcal{S}_k$ respectively. Now, we formally define the problem of sequential POI recommendation. For our example, $\mathcal{A}$ will consist of '*Shopping*', '*Game*', '*Social*', and '*Travel*', while $\mathcal{S}$ will include '*Bar*', '*Burger-Joint*', '*Cafe*', and '*Sushi Restaurant*' respectively.

Problem Statement (**Personalized Sequential Recommendation**)**.** *Using the user's past check-in records consisting of app and POI categories, we aim to get a ranked list of the most likely locations the user is expected to visit in her next check-in. Specifically, learn the time-evolving variation in smartphone and physical mobility to estimate her future preference towards different locations in her vicinity.*

Mathematically, Given the first k check-ins in a sequence as $\mathcal{E}_k$, we aim to identify the set of candidate POI for the next check-in *i.e.* e_{k+1}, conditioned on the app- and location-categories of all check-ins in the past history. Specifically, maximize the following probability:

$$\mathbb{P}^* = \arg\max_{\Theta} \{\mathbb{E}[e_{k+1} | \mathcal{E}_k, \mathcal{A}_k^*, \mathcal{S}_k^*]\} \tag{1}$$

where $\mathbb{E}[e_{k+1}]$ calculates the expectation of e_{k+1} being in the sequence of the user, $\mathcal{E}_k$ given the past check-ins of a user. Here, Θ denotes the REVAMP model parameters.



$\mathscr{L}_{SR}$
Sequential Recommender
$\boldsymbol{J}, \boldsymbol{K}, \boldsymbol{T}$
Relative Encodings
$\boldsymbol{A}, \boldsymbol{S}$
$\mathscr{L}_{EI}$
Embedding Initiator
$\boldsymbol{A}, \boldsymbol{S}$
$\mathcal{A}, \mathcal{S}$
l_1
l_2
l_N

Fig. 3: Overview of the neural architecture of ReVAMP.

TABLE 1: Summary of Notations Used.

Notation	Description
$\mathcal{U}, \mathcal{L}$	Set of all user and locations
$e_k \in \mathcal{E}$	k-th check-in in the sequence
$\mathcal{A}, \mathcal{S}$	Set of all smartphone and POI categories
$\boldsymbol{A}, \boldsymbol{S}$	Smartphone and POI category embeddings
D	Embedding Dimension
$\boldsymbol{J}, \boldsymbol{K}, \boldsymbol{T}$	App, POI, and time based relative encodings
$\boldsymbol{P}^{\text{key}}, \boldsymbol{P}^{\text{val}}$	Absolute positional encodings
$\boldsymbol{J}^{\text{key}}, \boldsymbol{K}^{\text{key}}, \boldsymbol{T}^{\text{key}}$	Key matrices for relative embeddings
$\boldsymbol{J}^{\text{val}}, \boldsymbol{K}^{\text{val}}, \boldsymbol{T}^{\text{val}}$	Value matrices for relative embeddings
$\mathscr{L}_{\text{MF}}, \mathscr{L}_{\text{Bert}}$	Trajectory and BERT-based loss
$\mathscr{L}_{\text{Rec}}$	POI recommendation loss for SR
$\mathscr{L}_{\text{App}}, \mathscr{L}_{\text{POI}}$	Smartphone and POI category loss

4 ReVAMP Framework

In this section, we first present a high-level overview of the deep neural network architecture of ReVAMP and then describe component-wise architecture in detail.

4.1 High-level Overview

ReVAMP comprises of two components – (i) an Embedding Initiator (EI) and (ii) Sequential Recommender (SR). Figure 3 shows the overall architecture of ReVAMP with different components and a schematic diagram of both the components is given in Figure 4. The workflow of ReVAMP includes three steps: (i) determining the embeddings of all POI and app categories using the EI module; (ii) calculating the relative positional encodings in terms of app category, POI category, and time of check-in, and determining embedding matrices for each; and (iii) using the category embeddings from EI module and the newly derived relative embeddings, determine the mobility preferences of a user via the SR module.

As we model the differences between the category of POI and smartphone apps across check-ins, we must capture the semantic meaning associated with each category *e.g.* the difference between a '*Sushi restaurant*' and a '*Cafe*'. Accordingly, EI takes the check-in sequence of a user as input, learns the representations of all smartphone app- and POI-categories, and calculates the evolving user preferences as the variation between check-ins.

$$\boldsymbol{A}, \boldsymbol{S} = G_{\text{EI}}(\mathcal{E}_k, \mathcal{A}^*_k, \mathcal{S}^*_k), \tag{2}$$

where $\boldsymbol{A}, \boldsymbol{S}$ denote the learned embeddings for app and POI categories respectively, and $G_{\text{EI}}(\bullet)$ denotes the Embedding Initiator. Moreover, ReVAMP works by modelling the variations between different check-ins in a sequence. Specifically, it learns how the mobility preference of a user has evolved based on the difference in the current and past check-ins. Capturing and feeding these differences to our self-attention model is a non-trivial task as we denote each check-in via its POI and app category embeddings. Therefore, we use our RE module to derive these differences and simultaneously embed them to be fed into a self-attention model. These variations are used to assign relative positional encodings to the check-ins in the self a stacked self-attention architecture in SR.

$$\boldsymbol{J}, \boldsymbol{K}, \boldsymbol{T} = f_{\text{RE}}(\mathcal{E}_k, \boldsymbol{A}, \boldsymbol{S}), \tag{3}$$

where $\boldsymbol{J}, \boldsymbol{K}, \boldsymbol{T}$ are the relative positional encodings for app categories, POI categories, and time respectively. Here, $f_{\text{RE}}(\bullet)$ denotes the function to calculate these relative encodings. Note that these encodings are *personalized i.e.* they are calculated independently for each user. SR then combines these relative encodings with absolute positional encodings to model the sequential POI preference of a user. Through this, we aim to get a ranked list of the most probable candidate POIs for the next check-in of a user.

$$\widehat{l_{k+1}} = G_{\text{SR}}(\mathcal{E}_k, \boldsymbol{J}, \boldsymbol{K}, \boldsymbol{T}), \tag{4}$$

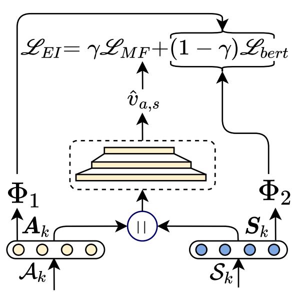
where $\widehat{l_{k+1}}$ is the candidate POI for the $k+1$-th check-in of a user and $G_{\text{SR}}(\bullet)$ denotes the sequential recommender. Figure 4 shows a schematic diagram of ReVAMP architecture. The training process of ReVAMP is divided into two-steps – train the category embeddings using EI and then use them for sequential recommendation in SR section. More details are given in Section 4.5.

4.2 Embedding Initiator (EI)

A major contribution of this paper is to learn the mobility preferences conditioned only on the categories of smartphone apps and POI rather than the exact location coordinates and app preferences. Learning from such coarse data is not a trivial task and training with *randomly-initialized* embeddings may not capture the category semantics, *e.g.* if '*Burger-Joints*' and '*Asian-Restaurants*' are frequently visited then a training process with random initialization will lead to similar the trained embeddings. Therefore, our category embeddings must simultaneously capture the user preferences towards each category and the category semantics via pre-trained word embeddings. We highlight this through an example – a user checks a mobile-app of category '*Social*' frequently at two separate locations, say '*Cafe*' and '*Sushi Restaurant*', the category embeddings should capture the POI influence that persuaded a user to use apps of a similar category('*Social*' in this case) as well as the semantic difference between a coffee joint and an Asian restaurant. Therefore, we use a two-channel training procedure, wherein we use pre-trained embeddings to extract the semantic meaning of all app and location categories and learn user preferences towards these categories via a lightweight matrix-factorization.

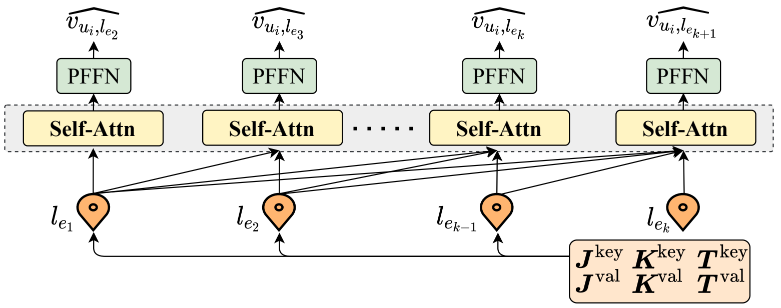
Specifically, given a check-in sequence $e_k \in \mathcal{E}$ we follow a four-layer architecture:

(1) **Input Layer.** We initially embed the app and location categories, $\mathcal{A}^*$ and $\mathcal{S}^*$, as $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{A}^*| \times D}$ and $\boldsymbol{S} \in \mathbb{R}^{|\mathcal{S}^*| \times D}$ respectively. Each row $\boldsymbol{a}_i \in \boldsymbol{A}$ represents a D-dimension



$\mathscr{L}_{EI} = \gamma \mathscr{L}_{MF} + (1-\gamma)\mathscr{L}_{bert}$
$\hat{v}_{a,s}$
Φ_1
Φ_2
A_k
S_k
$\mathcal{A}_k$
$\mathcal{S}_k$

(a) Embedding Initiator (EI)



$\widehat{v_{u_i,l_{e_2}}}$
$\widehat{v_{u_i,l_{e_3}}}$
$\widehat{v_{u_i,l_{e_k}}}$
$\widehat{v_{u_i,l_{e_{k+1}}}}$
PFFN
Self-Attn
l_{e_1}
l_{e_2}
$l_{e_{k-1}}$
l_{e_k}
J^{key} K^{key} T^{key}
J^{val} K^{val} T^{val}

(b) Sequential Recommender (SR)

Fig. 4: Architecture of different components in REVAMP. Panel (a) illustrates an the setup of Embedding Initiator (EI) that learns the category representations. Panel (b) shows the self-attention architecture in Sequential Recommender (SR). Note that only the check-in-sequence till $k-1$-th index is used to predict the next check-in *i.e.* e_k, and the input is an aggregation of all past events and relative positional encodings.

representation of a smartphone app category. Similarly, $\boldsymbol{s}_i \in \boldsymbol{S}$ is a representation for a POI category.

(2) **MF Layer.** To learn the interaction between app and POI categories, we follow a lightweight collaborative filtering approach, wherein we concatenate the entries in $\boldsymbol{A}$ and $\boldsymbol{S}$ that appear together in a check-in $e_k \in \mathcal{E}$. Specifically, we concatenate the app and POI category embeddings for a check-in and then use a feed-forward network.

$$\widehat{v}_{a_i,s_j} = \mathrm{ReLU}\left(\boldsymbol{w}_v(\boldsymbol{a}_i || \boldsymbol{s}_j) + \boldsymbol{b}_v\right), \tag{5}$$

where $\widehat{v}_{a_i,s_j}$ denotes the probability of an app of category a_i to be accessed at a POI of category s_j, $||$ denotes the concatenation operator, and $\boldsymbol{w}_\bullet, \boldsymbol{b}_\bullet$ are trainable parameters. We train our embeddings via a cross-entropy loss:

$$\mathscr{L}_{\mathrm{MF}} = -\sum_{k=1}^{|\mathcal{E}|} \sum_{\substack{a_i \in \mathcal{A}_k \\ s_j \in \mathcal{S}_k}} \Big[\log\left(\sigma(\widehat{v}_{a_i,s_j})\right) + \log\left(1-\sigma(\widehat{v}_{a_i,s'_j})\right) + \log\left(1-\sigma(\widehat{v}_{a'_i,s_j})\right) \Big],$$

where $\widehat{v}_{\bullet,\bullet}$ denotes the estimated access probability (i) $\widehat{v}_{a_i,s_j}$ between a *true* app- and location-category *i.e.* $a_i \in \mathcal{A}_k, s_j \in \mathcal{S}_k$, (ii) $\widehat{v}_{a'_i,s_j}$ for a negatively sampled app-category with a true location-category *i.e.* $a_i \notin \mathcal{A}_k, s_j \in \mathcal{S}_k$, and (iii) $\widehat{v}_{a_i,s'_i}$ for a negatively sampled location-category with a true app-category *i.e.* $a_i \in \mathcal{A}_k, s_j \notin \mathcal{S}_k$.

(3) **BERT Layer.** To capture the real-world semantics of a category, we use a pre-trained BERT [10] model with over 110M parameters. Specifically, we extract the embeddings for each smartphone app and POI category from the pre-trained model. Later, we maximize the similarity between these embeddings and our category representations, $\boldsymbol{A}$ and $\boldsymbol{S}$ by optimizing a mean squared loss.

$$\mathscr{L}_{\mathrm{Bert}} = \frac{1}{|\mathcal{E}|} \sum_{k=1}^{|\mathcal{E}|} \sum_{\substack{a_i \in \mathcal{A}_k \\ s_j \in \mathcal{S}_k}} \left[||\boldsymbol{a}_i - \Phi_1(a_i)||^2 + ||\boldsymbol{s}_i - \Phi_2(s_i)||^2 \right], \tag{6}$$

where, $\boldsymbol{a}_i \in \boldsymbol{A}$ and $\boldsymbol{s}_j \in \boldsymbol{S}$ are our trainable embedding for categories a_i and s_j respectively, and $\Phi_\bullet$ denotes a two-step function that extracts pre-trained embeddings for all categories and uses a feed-forward network to normalize the embedding dimension to D. Specifically,

$$\Phi_1(a_i) = \mathrm{ReLU}\big(\boldsymbol{w}_1 \cdot \mathcal{B}(a_i) + \boldsymbol{b}_1\big), \tag{7}$$

$$\Phi_2(s_i) = \mathrm{ReLU}\big(\boldsymbol{w}_2 \cdot \mathcal{B}(s_i) + \boldsymbol{b}_2\big), \tag{8}$$

where $\mathcal{B}$ denotes the set of all pre-trained embeddings, $\mathcal{B}(a_i)$ and $\mathcal{B}(s_i)$ denote the extracted app and location category embedding, and $\boldsymbol{w}_\bullet, \boldsymbol{b}_\bullet$ are trainable parameters.

(4) **Optimization.** We train our embeddings using a two-channel learning procedure consisting of app-location interaction loss, $\mathscr{L}_{\mathrm{MF}}$, and pre-trained embedding loss, $\mathscr{L}_{\mathrm{Bert}}$, by optimizing a *weighted* joint loss.

$$\mathscr{L}_{\mathrm{EI}} = \gamma \mathscr{L}_{\mathrm{MF}} + (1-\gamma)\mathscr{L}_{\mathrm{Bert}}, \tag{9}$$

where γ denotes a scaling parameter. Later, we use $\boldsymbol{A}$ and $\boldsymbol{S}$ to identify the inter-check-in differences and model the POI preferences of a user.

4.3 Relative or Inter-check-in Variations

Buoyed by the efficacy of relative encodings for self-attention models [5], [42], REVAMP captures the evolving preferences of a user as relative encodings based on three inter-check-in differences: (i) Smartphone App-based dynamics, (ii) Location category distribution, and (iii) Time-based evolution across the event sequence.

Smartphone App-based Variation. Recent research [5], [42] has shown that users' preference towards smartphone apps is influenced by their geo-locations and other POI-based semantics. Seemingly, it is more likely for a user to be active on a multiplayer game at a social joint rather than her workplace. We quantify the differences in the app preferences of a user via the differences in the embeddings of the smartphone-app category being used at a check-in. Specifically, for each check-in e_k, we calculate the *net* app-category as a mean of all category embeddings.

$$\boldsymbol{\mu}_k^a = \frac{1}{|\mathcal{A}_k|} \sum_{a_i \in \mathcal{A}_k} \boldsymbol{a}_i, \tag{10}$$

where $\boldsymbol{\mu}_k^a, a_i \in \mathcal{A}_k, \boldsymbol{a}_i \in \boldsymbol{A}$ represent the net app-category embedding for a check-in e_k, the app-category used in the check-in and the corresponding embedding learned in the EI (see Section 4.2). Following [42], we use these embeddings to calculate a inter-check-in variance matrix $\boldsymbol{J} \in \mathbb{W}^{|\mathcal{E}|\times|\mathcal{E}|}$ for each check-in sequence. Specifically, the i-th row in matrix $\boldsymbol{J}$ denotes the difference between the mean app-category embedding of check-in e_i with all other check-ins in the sequence and is calculated as:

$$\boldsymbol{J}_{i,j} = \left\lfloor \frac{f_{\cos}(\boldsymbol{\mu}^a_i, \boldsymbol{\mu}^a_j) - \min_f(\mathcal{E})}{\max_f(\mathcal{E}) - \min_f(\mathcal{E})} \cdot I_a \right\rfloor, \tag{11}$$

where $f_{\cos}(\bullet,\bullet), \min_f(\mathcal{E}), \max_f(\mathcal{E})$ denote the function for normalized cosine-distance, the minimum and maximum cosine distance between the mean category embedding for any two check-ins in a sequence. We use I_a as a clipping constant and a *floor* operator to discretize the entries in $\boldsymbol{J}$. Such a discretization makes it convenient to extract positional encodings for the self-attention model in SR.

POI-based Variation. We derive the inter-check-in differences between POI categories using a similar procedure for app-based differences. Specifically, we calculate a *net* POI category embedding, $\boldsymbol{\mu}^l_k$ for each check-in as $\boldsymbol{\mu}^l_k = \frac{1}{|\mathcal{S}_k|}\sum_{s_i \in \mathcal{S}_k} \boldsymbol{s}_i$ and similar to Eqn 11, we calculate the POI-based inter-check-in variance matrix $\boldsymbol{K} \in \mathbb{W}^{|\mathcal{E}|\times|\mathcal{E}|}$ using a clipping constant I_l. Here, the i-th row in matrix $\boldsymbol{K}$ denotes the difference between the mean POI-category embedding of check-in e_i with all other check-ins in the sequence.

Time-based Variation. Ostensibly, there may be irregularities in the smartphone app usage of a user, *e.g.* a user browsing '*Amazon*' may receive a message '*Twitter*' that she immediately checks and then later continues her shopping on Amazon. Notably, the '*Amazon*' app did not influence the user to access '*Twitter*' and vice-versa, as such a change between apps was coincidental. To model these nuances in REVAMP we use the time interval between accessing different smartphone apps. Specifically, similar to app- and POI-category based inter-check-in differences, we derive a *time-based* variations matrix, $\boldsymbol{T} \in \mathbb{W}^{|\mathcal{E}|\times|\mathcal{E}|}$, using the absolute time-difference between each check-in.

$$\boldsymbol{T}_{i,j} = \left\lfloor \frac{|t_i - t_j|}{t_{\min}} \cdot I_t \right\rfloor, \tag{12}$$

where $t_i, t_j, t_{\min}$, and I_t denote the time of check-in e_i and e_j, minimum time-interval between check-ins of a user and the normalizing constant for time respectively.

4.4 Sequential Recommender (SR)

In this section, we elaborate on the sequential recommendation procedure of REVAMP that is responsible for modeling the app and POI preferences of a user and then recommend candidate POI for the next check-in. Specifically, it uses a self-attention architecture consisting of five layers:

(1) **Input Layer.** The SR model takes the check-in sequence of a user ($\mathcal{E}$), relative app, POI, and time encodings, ($\boldsymbol{K}, \boldsymbol{J}$, and $\boldsymbol{T}$ respectively), and the *mean* app and location category representations ($\boldsymbol{\mu}^a_\bullet, \boldsymbol{\mu}^l_\bullet$) as input to the self-attention model. Since, the self-attention models require a fixed input sequence, we limit our training to a fixed number of check-ins *i.e.* we consider the N most recent check-ins in $\mathcal{E}$ for training our model and if the number of check-ins is lesser than N, we repeatedly add a *[pad]* vector for the initial check-ins within the sequence.

(2) **Embedding Retrieval Layer.** Since, the self-attention models are oblivious of the position of each check-in in the sequence, we use a trainable positional embedding for each check-in [24], [27]. Specifically, we initialize two distinct vectors denoted by $\boldsymbol{P}^{\text{key}} \in \mathbb{R}^{N\times D}$ and $\boldsymbol{P}^{\text{val}} \in \mathbb{R}^{N\times D}$ where the i-th rows, $\boldsymbol{p}^{\text{key}}_i$ and $\boldsymbol{p}^{\text{val}}_i$, denote the positional encoding for the check-in e_i in the sequence. Similarly, we embed the relative positional matrices $\boldsymbol{K}$, $\boldsymbol{J}$, and $\boldsymbol{T}$ into encoding matrices $\boldsymbol{J}^{\text{key}}, \boldsymbol{J}^{\text{val}} \in \mathbb{R}^{N\times N\times D}$, $\boldsymbol{K}^{\text{key}}, \boldsymbol{K}^{\text{val}} \in \mathbb{R}^{N\times N\times D}$, and $\boldsymbol{T}^{\text{key}}, \boldsymbol{T}^{\text{val}} \in \mathbb{R}^{N\times N\times D}$ respectively.

$$\boldsymbol{K}^{\text{key}} = \begin{bmatrix} \boldsymbol{k}^{\text{key}}_{1,1} & \cdots & \boldsymbol{k}^{\text{key}}_{1,N} \\ \vdots & \vdots & \vdots \\ \boldsymbol{k}^{\text{key}}_{N,1} & \cdots & \boldsymbol{k}^{\text{key}}_{N,N} \end{bmatrix}, \boldsymbol{K}^{\text{val}} = \begin{bmatrix} \boldsymbol{k}^{\text{val}}_{1,1} & \cdots & \boldsymbol{k}^{\text{val}}_{1,N} \\ \vdots & \vdots & \vdots \\ \boldsymbol{k}^{\text{val}}_{N,1} & \cdots & \boldsymbol{k}^{\text{val}}_{N,N} \end{bmatrix}, \tag{13}$$

We use two separate matrices to avoid any further linear transformations [42]. Each entry in $\boldsymbol{K}^{\text{key}}$ and $\boldsymbol{K}^{\text{val}}$ denotes a D dimensional vector representation of corresponding value in in $\boldsymbol{K}$. We follow a similar procedure to initialize $\boldsymbol{J}^{\text{key}}, \boldsymbol{J}^{\text{val}}, \boldsymbol{T}^{\text{key}}$ and $\boldsymbol{T}^{\text{val}}$ for $\boldsymbol{J}$ and $\boldsymbol{T}$ respectively.

(3) **Self-Attention Layer.** Given the check-in sequence of a user, the self-attention architecture learns the sequential preference of a user towards POIs. Specifically, for an input sequence consisting of POI embeddings of locations visited by a user, $\boldsymbol{L}^{\mathcal{E}} = (\boldsymbol{l}_{e_1}, \boldsymbol{l}_{e_2}, \cdots \boldsymbol{l}_{e_N})$ where $l_{e_i} \in e_i$ and $\boldsymbol{l}_{e_i} \in \boldsymbol{L}$ are the location visited in check-in e_i the POI embedding for l_{e_i} respectively, we compute a new sequence $\boldsymbol{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots \boldsymbol{z}_N)$, where $\boldsymbol{z}_\bullet \in \mathbb{R}^D$. Each output embedding is calculated as a weighted aggregation of embeddings of all POIs visited in the past.

$$\boldsymbol{z}_i = \sum_{j=1}^{N} \boldsymbol{\alpha}_{i,j}\left(\boldsymbol{w}_{v,j}\boldsymbol{l}_{e_j} + \overline{\boldsymbol{\mu}}_j + \boldsymbol{p}^{\text{val}}_j + \boldsymbol{j}^{\text{val}}_{i,j} + \boldsymbol{k}^{\text{val}}_{i,j} + \boldsymbol{t}^{\text{val}}_{i,j}\right), \tag{14}$$

where $\boldsymbol{l}_{e_j}$ is the POI embedding, $\overline{\boldsymbol{\mu}}_j = \boldsymbol{\mu}^a_j + \boldsymbol{\mu}^l_j$ is the sum of smartphone app and POI category mean embeddings, and $\boldsymbol{w}_{v,j}$ is a trainable parameter. The attention weights $\boldsymbol{\alpha}_{\bullet,\bullet}$ are calculated using a soft-max over other input embeddings as:

$$\boldsymbol{\alpha}_{i,j} = \frac{\exp\left(x_{i,j}\right)}{\sum_{k=1}^{N}\exp\left(x_{i,k}\right)}, \tag{15}$$

where $x_{i,j}$ denotes the compatibility between two check-ins– e_i and e_j – and is computed using both – relative- as well as absolute-positional encodings.

$$x_{i,j} = \frac{\boldsymbol{w}_{q,i}\boldsymbol{l}_{e_i}\left(\boldsymbol{w}_{k,j}\boldsymbol{l}_{e_j} + \boldsymbol{p}^{\text{key}}_j + \boldsymbol{j}^{\text{key}}_{i,j} + \boldsymbol{k}^{\text{key}}_{i,j} + \boldsymbol{t}^{\text{key}}_{i,j}\right)^\top}{\sqrt{D}}, \tag{16}$$

where $\boldsymbol{w}_{q,\bullet}, \boldsymbol{w}_{k,\bullet}$ and D denote the input *query* projection, *key* projection, and the embedding dimension respectively. We use the denominator as a scaling factor to control the dot-product gradients. As our task is to recommend candidate POI for future check-ins and should only consider the first k check-ins to predict the $(k+1)$-th check-in, we introduce a *causality* over the input sequence. Specifically, we modify the procedure to attention in Eqn.(16) and remove all links between the future check-ins and the current check-in.

(4) **Point-wise Layer.** As the self-attention lacks any non-linearity, we apply a feed-forward layer with two linear-transformation with ReLU activation.

$$\mathrm{PFFN}(\boldsymbol{z}_k) = \mathrm{ReLU}\left(\boldsymbol{z}_k \boldsymbol{w}_{p,1} + \boldsymbol{b}_1\right) \boldsymbol{w}_{p,2} + \boldsymbol{b}_2, \tag{17}$$

where $\boldsymbol{w}_{p,\bullet}, \boldsymbol{b}_\bullet$ are trainable layer parameters.
The combination of a self-attention layer and the point-wise layer is referred to as a self-attention *block* and stacking self-attention blocks gives the model more flexibility to learn complicated dynamics [47]. Thus, we stack M_b such blocks and to stabilize the learning process, we add a residual connection between each such block.

$$\boldsymbol{z}_k^{(r)} = \boldsymbol{z}_k^{(r-1)} + \mathrm{PFFN}(f_{ln}(\boldsymbol{z}_k^{(r-1)})), \tag{18}$$

where $1 \leq r \leq M_b, f_{ln}(\bullet)$ denote the level of the current self-attention block and *layer-normalization* function respectively. The latter is used to further accelerate the training of self-attention and is defined as follows:

$$f_{ln}(\boldsymbol{z}_k) = \beta \odot \frac{\boldsymbol{z}_k - \mu_z}{\sqrt{\sigma_z^2 + \epsilon}} + \gamma, \tag{19}$$

where $\odot, \mu_z, \sigma_z, \beta, \gamma, \epsilon$ denote the element-wise product, mean of all input embeddings, the variance of all input embeddings, learned scaling factor, bias term, and the Laplace smoothing constant respectively.

(5) **Prediction Layer.** A crucial distinction between REVAMP and standard self-attention model is that REVAMP not only predicts the candidate POIs for next check-in, but also the category of the smartphone app and POI category to be used in the next check-in. Here, we describe the prediction procedure for each of them.

POI Recommendation. We predict the next POI to be visited by a user in the check-in sequence using a matrix-factorization [19] based approach between the transformer output $\boldsymbol{Z}^{(M_b)} = (z_1, z_2, \cdots z_k)$ and the embeddings of POIs visited by the user, $(\boldsymbol{l}_{e_2}, \boldsymbol{l}_{e_3}, \cdots \boldsymbol{l}_{e_{k+1}})$

$$\widehat{v_{u_i, l_{e_k}}} = \boldsymbol{z}_{k-1} \boldsymbol{l}_{e_k}^\top, \tag{20}$$

where $\widehat{v_{u_i, l_{e_k}}}$ is the calculated probability of user, u_i, to visit the POI, l_{e_k}, for her next check-in. We learn the model parameters by minimizing the following cross-entropy loss.

$$\mathscr{L}_{\mathrm{Rec}} = -\sum_{u_i \in \mathcal{U}} \sum_{k=1}^{N} \left[\log\left(\sigma(\widehat{v_{u_i, l_{e_k}}}\right) + \log\left(1 - \sigma(\widehat{v_{u_i, l'_{e_k}}})\right)\right] + \lambda ||\Theta||_F^2,$$

where $\widehat{v_{u_i, l'_{e_k}}}$ denotes the check-in probability for a negatively sampled POI *i.e.* a randomly sampled location that will not be visited by a user. λ, σ, Θ denote regularization parameter, *sigmoid* function[4], and the trainable parameters respectively.

Predicting App Categories. Predicting the next smartphone app to be accessed by a user has numerous applications ranging from smartphone system optimization, resource management in mobile operating systems, and battery optimization [39], [56]. Therefore, to predict the category of the next app to be used, we follow a matrix-factorization approach to calculate the relationship between the user preference embedding, $\boldsymbol{z}_k$ and the mean of smartphone app embeddings for the next check-in.

$$\widehat{q_{u_i, \mathcal{A}_k}} = \boldsymbol{z}_{k-1} \boldsymbol{\mu}_k^{a\top}, \tag{21}$$

where $\widehat{q_{u_i, \mathcal{A}_k}}, \boldsymbol{\mu}_k^a$ denote the usage probability of apps of categories in $\mathcal{A}_k$ and the mean embedding for all apps used in check-in e_k. Later, we minimize a cross entropy loss with negatively sampled apps *i.e.* apps that were not used by the user, denoted as $\mathscr{L}_{\mathrm{App}}$.

Predicting Location Categories. As in app-category prediction, we calculate the preference towards a POI-category using the mean of POI category embedding $\boldsymbol{\mu}_k^l$ and learn the parameters by optimizing a similar *cross-entropy* loss denoted as $\mathscr{L}_{\mathrm{POI}}$.

The net loss for sequential recommendation is a weighted combination of POI recommendation loss, app-category loss, and location- category loss.

$$\mathscr{L}_{\mathrm{SR}} = \mathscr{L}_{\mathrm{Rec}} + \kappa(\mathscr{L}_{\mathrm{App}} + \mathscr{L}_{\mathrm{POI}}), \tag{22}$$

Here, κ is a tunable hyper-parameter for determining the contribution of category prediction losses. All the parameters of REVAMP including the weight matrices, relative-position weights, and embeddings are learned using an Adam optimizer [25].

4.5 REVAMP: Training

As mentioned in Section 4.1, REVAMP involves a two-step training procedure. Specifically, it consists of the following steps: (i) learning the app and POI category embeddings using the embedding initiator(EI) and (ii) training the self-attention model in SR to recommend candidate POI to the user. In detail, we first train the parameters of EI by minimizing the $\mathscr{L}_{EI}$ loss for multiple epochs and later use the trained category embeddings in SR and recommend candidate POI by minimizing the recommendation loss, $\mathscr{L}_{SR}$.

We highlight that a *joint* training of both, EI and SR, is not suitable in the presence of relative positional encodings, as they are conditioned on the category embeddings learned in EI. Therefore, during joint training, an update in the category embedding will make the trained parameters of SR across the previous epochs as unsuitable for prediction in the future. Moreover, these encodings are calculated *relatively i.e.* conditioned on the embedding of other categories in a sequence, and thus any change in the category embedding will affect the category embeddings.

5 EXPERIMENTS

In this section, we report a comprehensive empirical evaluation of REVAMP and compare it with other state-of-the-art approaches. We evaluate the POI recommendation performance of REVAMP using two real-world datasets from China. These datasets vary significantly in terms of data sparsity, the no. of app categories, and POI categories. With our experiments, we aim to answer the following research questions:

RQ1 How does REVAMP fare against state-of-the-art models for sequential POI recommendation? Where are the gains and losses?

4. https://en.wikipedia.org/wiki/Sigmoid_function

TABLE 2: Statistics of all datasets used in this paper. Here, $|\mathcal{U}|$, $|\mathcal{P}|$, $|\mathcal{E}|$, $|\mathcal{A}|$, and $|\mathcal{S}|$ denote the number of users, POIs, check-ins, app categories, and POI categories respectively.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{S}\rvert$
Shanghai-Telecom	869	32680	3668184	20	17
TalkingData	14544	37113	438570	30	366

RQ2 What is the contribution of relative positional encodings?

RQ3 What is the scalability of REVAMP and the stability of the learning procedure?

RQ4 What is the impact of different hyper-parameter values on the prediction performance of REVAMP?

All our algorithms were implemented in Tensorflow[5] and to support reproducibility, we pledge to release our implementations to the public upon acceptance.

Shopping
Landmark
Sports
Business_affairs
Medical_care
Restaurant
Government
Life_service
Residence
Factory
Culture
Entertainment_and_leisure
Scenic_spot
Hotel
Education
Chinese_Restaurant
Residential_Building
College_Residence_Hall
Coffee_Shop
Apartment_Condo
Asian_Restaurant
Japanese_Restaurant
Cafe
Hotel
Office
Bus_Station
Building
Fast_Food_Restaurant
Home
private
Pizza_Place
Department_Store
Train_Station
Mall
Korean_Restaurant

(a) Shanghai-Telecom (b) TalkingData

Fig. 5: Word-cloud for POI Categories for both Shanghai-Telecom and TalkingData datasets. Larger the font-size indicates a larger frequency of location of the category. The variance across the datasets is due to the different sources used for extracting location categories – the at-hand location-categories for Shanghai-Telecom and Foursquare-based categories for TalkingData.

5.1 Experimental Setup

Dataset Description. As our goal is to recommend POIs to a user based on her smartphone usage, the mobility datasets used in our experiments must contain the user trajectory data *i.e.* geographical coordinates, time of a check-in, as well as the smartphone-usage statistics – applications used across different locations, the categories of different apps based on online app-stores, *etc.* Therefore we consider two popular large-scale datasets – *Shanghai-Telecom* and *TalkingData* and their statistics are given in Table 2. Moreover, we highlight the high variance between the category semantics of both the datasets by plotting the location category word clouds in Figure 5.

1) **Shanghai-Telecom:** This smartphone usage and the physical-mobility dataset was collected by a major network operator in China [53]. The trajectories were collected from Shanghai in April 2016. It contains the details of a user's physical mobility and the time- and geo-stamped smartphone app usage records. More specifically for each user, we have the time-stamped records of the smartphone apps being used and the different cellular-network base stations to which the smartphone was connected during the data collection procedure. For the region covered by each cellular network base station, we also have the details of the internal POIs and their corresponding categories. For our experiments, we consider each *user$\rightarrow$base-station* entry as a check-in and all the apps and their categories associated with that check-in as the events in the sequence $\mathcal{E}$. We adopt a commonly followed data cleaning procedure [32], [37] and filter out users and POI with less the five check-ins.
2) **TalkingData:** A large-scale public app-usage dataset that was released by TalkingData[6], a leading data intelligence solution provider based in China. The original dataset released by the company [23] consists of location- and time-stamped records of smartphone app usage and physical trajectories of a user. However, in this dataset, we lack the categories associated with each POIs. We overcome this by extracting location categories and geo-coordinates from publicly available check-in records [52] for users in Foursquare – a leading social mobility network, and map each check-in location in Foursquare to a location in the TalkingData within a distance of 50m based on geographical coordinates. For our experiments using this dataset, we restrict our check-in records to only the locations situated in mainland China. As in the Shanghai-Telecom dataset, we filter out the users and POI with lesser than five check-ins.

5. www.tensorflow.org
6. www.talkingdata.com/

System Configuration. All our experiments were done on a server running Ubuntu 16.04. CPU: Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz, RAM: 125GB and GPU: NVIDIA Tesla V100 32GB.

Evaluation Metric. We evaluate REVAMP and the other sequential recommendation baselines, using a widely used *leave-one-out* evaluation, *i.e.* next check-in prediction task. Specifically, for each user, we consider the last check-in of the trajectory sequence as the test check-in, the second last check-in for validation, and all preceding events as the training set [24], [27]. We also follow a common testing strategy wherein we pair each ground truth check-in in the test set with 100 randomly sampled negative events [19], [24]. Therefore, the task becomes to rank the negative check-ins with the ground truth check-in. In this setting, the hit-rate, HR@k, is equivalent to Recall@k and proportional to Precision@k and mean reciprocal rank (MRR) is equivalent to mean average precision (MAP). To evaluate the effectiveness of all approaches, we use Hits@k and NDCG@k, with $k \in \{1, 5, 10\}$ and report the confidence intervals based on five independent runs.

Parameter Settings. For all results in Section 5.2 and 5.4, we set $N = 200$ and $N = 100$ for Shanghai-Telecom and TalkingData respectively. We set $I_a = I_l = I_t = 64$, $D = 64$, and $\lambda = 0.002$, We search the batch-size in $\{128, 256\}$, the no of attention-heads in $\{1, 2, 4, 8\}$, κ, γ are searched in $\{0.2, 0.5, 0.8\}$, and the dropout probability is set to 0.2. However, for parameter sensitivity experiments in Section 5.3, we show the prediction performance across different hyper-parameter values.

Baselines. We compare REVAMP with the state-of-the-art methods based on their architectures below:

(1) **Standard Recommendation Systems.**

TABLE 3: Next check-in recommendation performance of REVAMP and state-of-the-art baselines. As the Shanghai-Telecom dataset lacks precise geographical coordinates for every check-in, we exclude a comparison with STGN [55]. Numbers with bold font and superscript * indicate the best and the second best performer respectively. All results of REVAMP are statistically significant (i.e. two-sided Fisher's test with $p \leq 0.1$) over the best baseline.

	Shanghai-Telecom					**TalkingData**				
Baselines	**NDCG/Hits@1**	**NDCG@5**	**NDCG@10**	**Hits@5**	**Hits@10**	**NDCG/Hits@1**	**NDCG@5**	**NDCG@10**	**Hits@5**	**Hits@10**
FPMC [41]	0.5906	0.6021	0.6402	0.6162	0.6481	0.7224	0.7362	0.7704	0.7408	0.7892
TransRec [18]	0.5437	0.5803	0.6055	0.5839	0.6081	0.6872	0.6892	0.7691	0.6902	0.7784
GRU4Rec+ [20]	0.6291	0.6432	0.6796	0.6443	0.6867	0.7319	0.7654	0.7913	0.7703	0.7962
Caser [44]	0.6418	0.6472	0.6782	0.6507	0.6991	0.7321	0.7802	0.8079	0.8157	0.8482
STGN [55]	-	-	-	-	-	0.6694	0.7981	0.8132	0.8224	0.8549
AUM [48]	0.5718	0.6089	0.6358	0.6097	0.6433	0.7184	0.7395	0.7646	0.7782	0.8179
Bert4Rec [43]	0.7031	0.7346	0.7442	0.7188	0.7301	0.7728	0.8247	0.8281	0.8614	0.8743
SASRec [24]	0.7279	0.7530	0.7562*	0.7583	0.7648	0.8295	0.8621*	0.8680	0.9027*	0.9108*
TiSRec [27]	0.7284*	0.7542*	0.7558	0.7618*	0.7663*	0.8307*	0.8619	0.8693*	0.8998	0.9014
REVAMP	**0.7865**	**0.8021**	**0.8186**	**0.8203**	**0.8340**	**0.8793**	**0.9324**	**0.9371**	**0.9492**	**0.9594**

FPMC [41] FPMC utilizes a combination of factorized first-order Markov chains and matrix factorization for recommendation and encapsulates a user's evolving long-term preferences as well as the short-term purchase-to-purchase transitions.

TransRec [18] A first-order sequential recommendation model that captures the evolving item-to-item preferences of a user through a translation vector.

(2) **POI Recommendation Systems.**

STGN [55] Uses a modified LSTM network that captures the spatial and temporal dynamic user-preferences between successive check-ins using spatio-temporal gates. Hence, it requires the exact location coordinates as input to the model.

(3) **Smartphone App-based.**

AUM [48] Models the user mobility as well as app-usage dynamics using a Dirichlet process to predict next successive check-in locations.

(4) **Recurrent and Convolutional Neural Network.**

GRU4Rec+ [20] A RNN-based approach that models the user action sequences for a session-based recommendation. It is an improved version of GRU4Rec [21] with changes in the loss function and the sampling techniques.

Caser [44] A state-of-the-art CNN-based sequential recommendation method that applies convolution operations on the N-most recent item embeddings to capture the higher-order Markov chains.

(5) **Self-Attention.**

Bert4Rec [43] A bi-directional self-attention [10] based sequential recommendation model that learn user preferences using a Cloze-task loss function, i.e. predicts the artificially *masks* events form a sequence.

SASRec [24] A self-Attention [47] based sequential recommendation method that attentively captures the contribution of each product towards a user's item-preference embedding.

TiSRec [27] A recently proposed enhanced version of SASRec model that uses *relative*-position embeddings using the difference in the time of consecutive purchases made by the user.

We omit comparisons across other approaches for sequential recommendations, such as GRU4Rec [21], MARank [36] as they already have been outperformed by the current baselines. We calculate the confidence intervals based on the results obtained after three independent runs.

5.2 Performance Comparison (RQ1)

In this section, we report the location recommendation performance of different methods across both mobility datasets. The results for Shanghai-Telecom and TalkingData datasets are given in Table 3. From these results, we make the following observations.

- REVAMP consistently outperforms all other baselines for sequential mobility prediction across both datasets. The superior performance signifies the importance of including the smartphone usage pattern of a user to determine her mobility preferences. We also note that the performance gains over other self-attention based models – Bert4Rec [43], SASRec [24], and TiSRec [27] further reinforce our claim that including *relative* positional encodings based on the smartphone, spatial and temporal characteristics enhances the user-modeling ability of a model.
- We also note that the self-attention-based architecture such as Bert4Rec, SASRec, TiSRec, and REVAMP consistently yield the best performance on all the datasets and easily outperform CNN and RNN based models namely Caser [44] and GRU4Rec+ [20]. This further signifies the unequaled proficiency of the transformer [47] architecture to capture the evolution of user preferences across her trajectory sequence. More importantly, it outperforms the state-of-the-art location recommendation model STGN [55] that uses the additional information of precise geographical coordinates of each POI location.
- REVAMP also outperforms the other smartphone-activity-based approach AUM [48] by up to 34% across different metrics.
- We also note that neural baselines such as Caser [44], GRU4Rec+ [20] achieve better results as compared to FPMC [41] and TransRec [18]. It asserts the utmost importance of designing modern recommender systems using neural architectures. Moreover, GRU4Rec+ achieves a similar performance compared to Caser.

To sum up, our empirical analysis suggests the following: (i) the state-of-the-art models, including self-attention

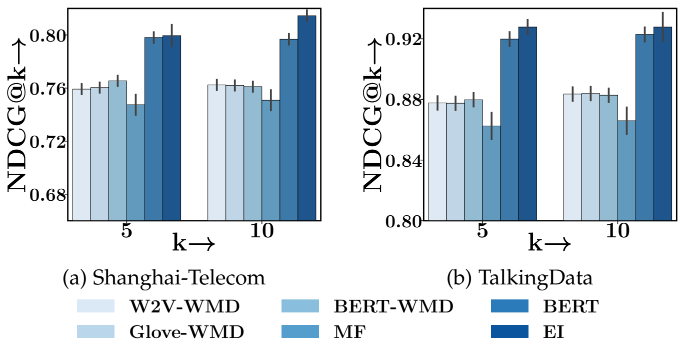


Fig. 6: POI recommendation performance of ReVAMP with different methods for obtaining the relative positional encodings *i.e.* the inter-check-in differences between app- and location embeddings. Here, the time-based representations are kept consistent across all the models.

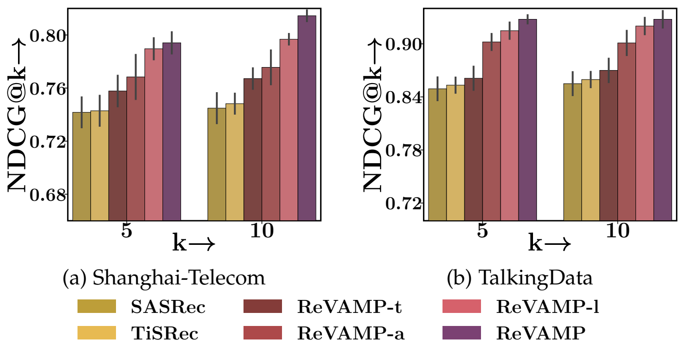


Fig. 7: Ablation study with different *relative* positional encodings used in ReVAMP and their comparison with SASRec [24] and TiSRec [27].

and standard neural models, are not suitable for modeling mobile-user trajectories, and (ii) ReVAMP achieves better recommendation performances as it captures the mobility dynamics as well as the smartphone-activity of a user.

5.3 Ablation Study(RQ2)

We also perform an ablation study to estimate the efficacy of different components in the ReVAMP architecture. More specifically we aim to calculate the contribution of (i) the embedding initiator and (ii) relative positional embeddings.

Analysis of Embedding Initiator. We reiterate that EI, defined in Section 4.2, is used to learn the semantic meaning of each app- and POI category as well as the influence between these embeddings in a mobility sequence. We accomplish this via a joint loss that consists of – minimizing the divergence between the category vector and the pre-trained BERT [10] vectors and a collaborative-filtering (CF) loss. These trained embeddings are later used to learn the inter-check-in differences through relative positional encodings. To emphasize its importance, we compare the prediction performances of ReVAMP with different procedures to learn category embeddings and thus the relative embeddings. Specifically we consider: (i) word-movers-distance(WMD) [26] between the word2vec [38] representations of each category, (ii) WMD on Glove [40] based representations, (iii) WMD based on BERT [10] initialized vectors, (iv) a simple collaborative filtering based parameter training, (v) using pre-trained BERT, and (vi) the EI proposed in the paper. From the results in Figure 6, we note that our proposed EI achieves the best prediction performance compared to other approaches. We also note that standard pre-trained BERT vectors outperform other WMD-based approaches.

Relative Positional Encodings. Relative positional embeddings are a crucial element in our model. We calculate the performance gains due to the different relative encodings – app-, time- and location-based by estimating the recommendation performance of the following approaches: SASRec [24]; (i) TiSRec [27]; (ii) ReVAMP with time-based relative positional encoding called ReVAMP-t; (iii) ReVAMP with app-based encodings, denoted as ReVAMP-a; (iv) ReVAMP with location-based encodings, denoted as ReVAMP-l; and (v) the complete ReVAMP model with all relative encodings.

Figure 7 summarizes our results in which we observe that including relative positional encodings of any form, whether app-based or location-based, leads to better prediction performances. Interestingly, the contribution of location-based relative positional embeddings is more significant than the app-based and could be attributed to *larger* variations in location-category than the app-category across an event sequence. For example, the difference between location categories of a university region and an office space will capture larger dynamics than the differences in the smartphone app usage across these two regions. However, jointly learning all positional encoding leads to the best performance over both datasets. The improvements of ReVAMP-t over TiSRec [27] could be attributed to the inclusion of *absolute* event encodings (both app and location) in ReVAMP.

In addition, we report the results in terms of mean reciprocal rank (MRR) for ReVAMP and the best performing baselines, *i.e.*, SASRec and TiSRec in Figure 8. The results across MRR show a similar trend with ReVAMP easily outperforming other approaches across both datasets. Interestingly, here we note that the performance difference between the baselines SASRec and TiSRec drops, *i.e.*, the performance is similar without any significant differences.

5.4 App and Location Prediction Category

Since our goal via ReVAMP is to understand the smartphone activity of a user and correlate it with her mobile trajectories. Therefore, we perform an additional experiment to evaluate how effectively is ReVAMP able to predict the app- and the location-category for the next user check-in. We also introduce an additional state-of-the-art smartphone-activity modeling baseline, Appusage2Vec [56] which considers the category of the app and the time spent on the app by the user to learn an app-preference embedding of a user. We also compare with the state-of-the-art transformer-based models – SASRec [24] and TiSRec [27]. For an even comparison, we rank the models using the root-mean-squared (RMS) distance between the final user preference

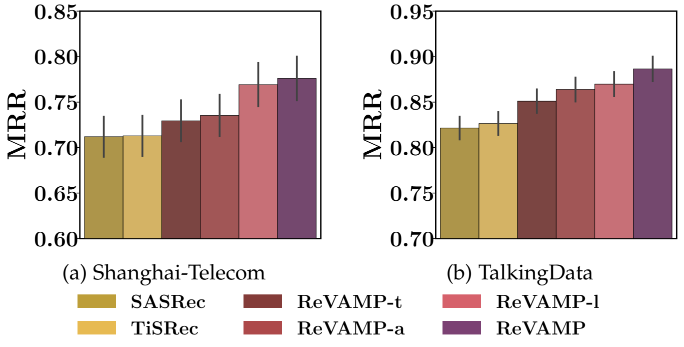


Fig. 8: Next check-in recommendation performance of SASRec, TiSRec, REVAMP, and its variants in terms of mean reciprocal rank (MRR).

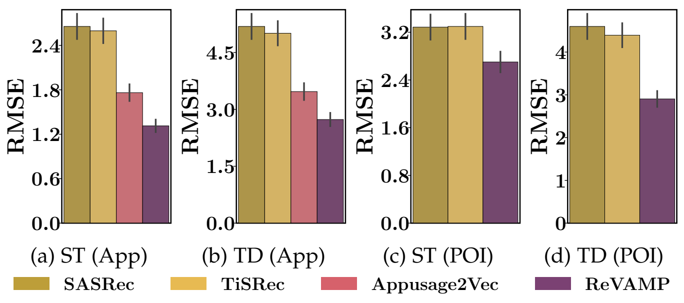


Fig. 9: Root-mean squared distance between the user-preference vector estimated by the model and the mean of app- and location- category vectors of events in the test set. It shows that REVAMP is the best performer for all the datasets.

embedding obtained after learning on N consecutive events of a user and the *mean* of location and category embeddings of the $N + 1$ event in the sequence. Accordingly, we also modify the architectures of SASRec and TiSRec to predict user-affinity across the location and app category affinities. From the results in Figure 9, we make the following observations: (i) REVAMP easily outperforms all other baselines for both apps and location category prediction. This illustrates the better user-preference modeling power of REVAMP over other approaches, (ii) For app-category prediction, Appusage2Vec also outperforms both SASRec and TiSRec even with its shallow neural architecture. However REVAMP easily outperforms Appusage2Vec across both datasets.

5.5 Scalability of REVAMP (RQ3)

To determine the scalability of REVAMP with different positional encodings – absolute and relative, we present the epoch-wise time taken for training REVAMP in Table 4. Note that these running times exclude the time for pre-processing where we calculate the inter-event app and location category-based differences. We note that the run-time of REVAMP is linear with the number of users and secondly, even for a large-scale dataset, like TalkingData, we can optimize all parameters in REVAMP well within 170 minutes. These run times are well in range for designing recommender systems.

TABLE 4: Run-time Statistics of training REVAMP in minutes on a 32GB Tesla V100 GPU with 256 batch-size.

Epochs	20	60	100	160	200	300
Shanghai-Telecom	2.43	6.31	10.48	16.72	21.47	32.01
TalkingData	11.39	33.73	56.12	89.81	112.35	168.49

TABLE 5: Run-time statistics of REVAMP in minutes for different subsets of data – 40%, 60%, and 80%.

	Shanghai-Telecom			TalkingData		
	40%	60%	80%	40%	60%	80%
Time	17.8	22.3	28.4	71.3	105.2	141.6

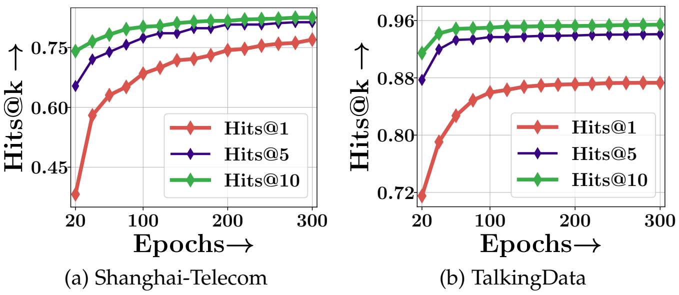


Fig. 10: Epoch-wise recommendation performance of REVAMP for both datasets in terms of Hits@1, 5, 10.

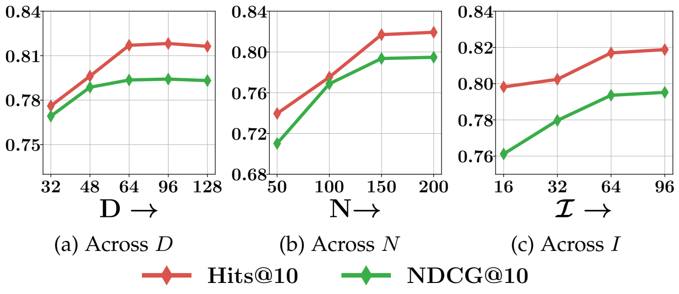


Fig. 11: Parameter Sensitivity for Shanghai-Telecom Dataset

In addition, we report the training times for different subsets of data in Table 5. Specifically, we show results where we use 40%, 60%, 80% of all users in the dataset. These users are selected randomly among all users in the complete dataset and the training-test sets are modified accordingly. All other parameters are same as before. From the results, we make the following observations: (i) the run-times increase linearly as per the subset of users; and (ii) the training times of REVAMP are well within the acceptable range for practical deployment.

Convergence of REVAMP Training. As we propose the first-ever application of the self-attention model for smartphones and human mobility, we also perform a convergence analysis during training REVAMP. To emphasize on the stability of REVAMP training procedure, we plot the epoch-wise best prediction performance of REVAMP across both datasets in Fig 10. From the results, we note that despite the multi-variate nature of data and the disparate positional encodings, REVAMP converges only in a few training iterations. It is also important to note that the REVAMP significantly outperforms other RNN based baselines even with limited training of 40 iterations.

5.6 Parameter Sensitivity (RQ4)

Finally, we perform the sensitivity analysis of REVAMP. The key parameters we study are (i) D, the dimension of embeddings; (ii) N, no. of latest events considered for training; and (iii) $I_{a,l,t}$, predefined normalizing constant for cosine-similarity for all relative encodings. (see Table I). In this section, we evaluate the model on NDCG@10 and Hits@10. We report the recommendation performance across different hyperparameter values for the Shanghai-Telecom dataset and omit results for TalkingData for brevity. However, we noted a similar behavior for the TalkingData dataset as well.

From the results in Figure 11, we note that as we increase the embedding dimension, D, the performance first increases since it leads to better modeling. However, beyond a point, the complexity of the model increases requiring more training to achieve good results, and hence we see some deterioration in performance. Next, increasing the no. of events for recommendation leads to better results before saturating at a certain point. We found $N = 100$ and $N = 200$ to be the optimal point across Shanghai-Telecom and TalkingData in our experiments. Finally, for normalizing constant I, an interesting insight is that on increasing the constant value the performance increases and later plateaus after a certain point. This could be due to saturation after a further increase in no. of distinct positional encodings.

6 CONCLUSION

In this paper, we highlighted the drawbacks of modern POI recommender systems that ignore the smartphone usage characteristics of users. We also proposed a novel sequential POI recommendation model, called REVAMP, that incorporates the smartphone usage details of a user while simultaneously maintaining user privacy. Inspired by the success of relative positional encodings and self-attention models, REVAMP uses relative as well as absolute positional encodings determined by the inter-check-in variances in the smartphone app category, POI category, and time over the check-ins in the sequence. Our experiments over two diverse datasets from China show that REVAMP significantly outperforms other state-of-the-art baselines for POI recommendation. Moreover, we also show that the contribution of each component in the REVAMP architecture, analyze the learning stability of the model, and the performance sensitivity across different hyperparameter values. A drawback of the current REVAMP formulation is the need for the entire data together. Specifically, modern privacy-conscious techniques use a federated learning approach to train the model parameters with decentralized data. As future work, we plan to expand REVAMP to such an architecture.

REFERENCES

[1] Eytan Adar, Jaime Teevan, and Susan T. Dumais. Large scale analysis of web revisitation patterns. In *SIGCHI*, 2008.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Louise Barkhuus and Anind K Dey. Location-based services for mobile telephony: a study of users privacy concerns. In *Interact*, 2003.

[4] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[5] Ivan Bilan and Benjamin Roth. Position-aware self-attention with relative positional encodings for slot filling, 2018.

[6] Hancheng Cao, Zhilong Chen, Fengli Xu, Yong Li, and Vassilis Kostakos. Revisitation in urban space vs. online: A comparison across pois, websites, and smartphone apps. In *IMWUT*, 2018.

[7] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, 2013.

[8] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *KDD*, 2011.

[9] Nield David. All the ways your smartphone and its apps can track you, 2017. Available At: https://gizmodo.com/all-the-ways-your-smartphone-and-its-apps-can-track-you-1821213704.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *EMNLP*, 2018.

[12] Aleksandr Farseev, Ivan Samborskii, Andrey Filchenkov, and Tat-Seng Chua. Cross-domain recommendation via clustering on multi-layer graphs. In *SIGIR*, 2017.

[13] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *WWW*, 2018.

[14] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. In *Nature*, 2008.

[15] Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. In *TACL*, 2019.

[16] Vinayak Gupta and Srikanta Bedathur. Region invariant normalizing flows for mobility transfer. In *CIKM*, 2021.

[17] Vinayak Gupta, Srikanta Bedathur, Sourangshu Bhattacharya, and Abir De. Learning temporal point processes with intermittent observations. In *AISTATS*, 2021.

[18] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation: A scalable method for modeling sequential behavior. In *IJCAI*, 2018.

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

[20] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, 2018.

[21] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[22] Simon L. Jones, Denzil Ferreira, Simo Hosio, Jorge Goncalves, and Vassilis Kostakos. Revisitation analysis of smartphone app use. In *UbiComp*, 2015.

[23] Kaggle. Talkingdata mobile user demographics, 2018. Available At: https://www.kaggle.com/c/talkingdata-mobile-user-demographics.

[24] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[26] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[27] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *WSDM*, 2020.

[28] Ranzhen Li, Yanyan Shen, and Yanmin Zhu. Next point-of-interest recommendation with temporal and multi-level context attention. In *ICDM*, 2018.

[29] Ruirui Li, Xian Wu, and Wei Wang. Adversarial learning to compare: Self-attentive prospective customer recommendation in location based social networks. In *WSDM*, 2020.

[30] Tong Li, Mingyang Zhang, Hancheng Cao, Yong Li, Sasu Tarkoma, and Pan Hui. ”what apps did you use?”: Understanding the long-term evolution of mobile app usage. In *WWW*, 2020.

[31] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. Geography-aware sequential location recommendation. In *KDD*, 2020.

[32] Ankita Likhyani, Srikanta Bedathur, and Deepak P. Locate: Influence quantification for location promotion in location-based social networks. In *IJCAI*, 2017.

[33] Ankita Likhyani, Vinayak Gupta, PK Srijith, P Deepak, and Srikanta Bedathur. Modeling implicit communities from geo-tagged event traces using spatio-temporal point processes. In *WISE*, 2020.

[34] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: a recurrent model with spatial and temporal contexts. In *AAAI*, 2016.

[35] Zheng Lu, Yunhe Feng, Wenjun Zhou, Xiaolin Li, and Qing Cao. Inferring correlation between user mobility and app usage in massive coarse-grained data traces. In *IMWUT*, 2018.

[36] Chuxu Zhang Lu Yu, Shangsong Liang, and Xiangliang Zhang. Multi-order attentive ranking model for sequential recommendation. In *AAAI*, 2019.

[37] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. A contextual attention recurrent architecture for context-aware venue recommendation. In *SIGIR*, 2018.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

[39] Adam J Oliner, Anand P Iyer, Ion Stoica, Eemil Lagerspetz, and Sasu Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. In *SenSys*, 2013.

[40] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[41] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 2010.

[42] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT*, 2018.

[43] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 2019.

[44] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 2018.

[45] Zhen Tu, Yali Fan, Yong Li, Xiang Chen, Li Su, and Depeng Jin. From fingerprint to footprint: Cold-start location recommendation by learning user interest from app data. In *IMWUT*, 2019.

[46] Zhen Tu, Runtong Li, Yong Li, Gang Wang, Di Wu, Pan Hui, Li Su, and Depeng Jin. Your apps give you away: Distinguishing mobile users by their app usage fingerprints. In *IMWUT*, 2018.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[48] Huandong Wang, Yong Li, Sihan Zeng, Gang Wang, Pengyu Zhang, Pan Hui, and Depeng Jin. Modeling spatio-temporal app usage for a large user population. In *IMWUT*, 2019.

[49] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In *SIGIR*, 2017.

[50] Ying Wei, Yu Zheng, and Qiang Yang. Transfer knowledge between cities. In *KDD*, 2016.

[51] Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *ICLR*, 2019.

[52] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *WWW*, 2019.

[53] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. Smartphone app usage prediction using points of interest. In *IMWUT*, 2018.

[54] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. Time-aware point-of-interest recommendation. In *SIGIR*, 2013.

[55] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. Where to go next: a spatio-temporal gated network for next poi recommendation. In *AAAI*, 2019.

[56] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. Appusage2vec: Modeling smartphone app usage for prediction. In *ICDE*, 2019.

Vinayak Gupta is a Ph.D. student in the Department of Computer Science and Engineering at the Indian Institute of Technology (IIT) Delhi. His research interests are in designing neural models for time series and graphs in the presence of missing and scarce data. He received the outstanding doctoral paper award at the First International Conference on AI-ML Systems 2021. Previously, he completed his Bachelors in Computer Science from the Indian Institute of Information Technology (IIIT) Jabalpur.

Srikanta Bedathur is an associate professor and DS Chair Professor of Artificial Intelligence at the Department of Computer Science and Engineering at IIT Delhi. He previously worked as a research scientist at IBM Research as part of their AI Research team, as a faculty member at IIIT Delhi, and senior researcher at the Max-Planck Institute for Informatics (MPII) with Gerhard Weikum and his Ph.D. (2005) was from the Indian Institute of Science where he worked with Prof. Jayant Haritsa.