Meta-Learned Spatial-Temporal POI Auto-Completion for the Search Engine at Baidu Maps

Miao Fan, Yibo Sun, Jizhou Huang*, Haifeng Wang, Ying Li
Baidu Inc., Beijing, China
{fanmiao,sunyibo,huangjizhou01,wanghaifeng,liying}@baidu.com

ABSTRACT

Point Of Interest Auto-Completion (abbr. as POI-AC) is one of the featured functions for the search engine at Baidu Maps. It can dynamically suggest a list of POI candidates within milliseconds as a user enters each character (e.g., English, Chinese, or Pinyin character) into the search box. Ideally, a user may need to provide only one character and immediately obtain the desired POI at the top of the POI list suggested by POI-AC. In this way, the user's keystrokes can be dramatically saved, which significantly reduces the time and effort of typing, especially on mobile devices that have limited space for display and user interfaces. Despite using a user's profile and input prefixes for personalized POI suggestions, however, the state-of-the-art approach, i.e., P^3AC [5], still has a long way to go so as to generate not only personalized but, more importantly, time- and geography-aware suggestions. In this paper, we find that 17.9% of users tend to look for diverse POIs at different times or locations using the same prefix. This insight drives us to establish an end-to-end spatial-temporal POI-AC (abbr. as *ST-PAC*) module to replace P^3AC at Baidu Maps. To alleviate the problem of the long-tail distribution of time- and location-specific data on POI-AC, we further propose a meta-learned *ST-PAC* (abbr. as *MST-PAC*) updated by an efficient MapReduce algorithm. *MST-PAC* can significantly overcome the "long-tail" issue and rapidly adapt to the cold-start POI-AC tasks with fewer examples. We sample several benchmark datasets from the large-scale search logs at Baidu Maps to assess the offline performance of *MST-PAC* in line with multiple metrics, including Mean Reciprocal Rank (MRR), Success Rate (SR), and normalized Discounted Cumulative Gain (nDCG). The consistent improvements on these metrics give us more confidence to launch this meta-learned POI-AC module online. As a result, the critical indicator on online user satisfaction, i.e., the average number of keystrokes in a POI-AC session, significantly decreases as well. For now, *MST-PAC* has already been deployed in production at Baidu Maps, handling billions of POI-AC requests every day. It confirms that *MST-PAC* is a practical and robust industrial solution for large-scale POI Search.

*Corresponding author: Jizhou Huang.

CCS CONCEPTS

• **Information systems → Mobile information processing systems**; **Information retrieval query processing**; **Spatial-temporal systems**.

KEYWORDS

POI auto-completion, POI-AC, POI Search, query-POI matching, meta-learning, spatial-temporal, Baidu Maps

ACM Reference Format:
Miao Fan, Yibo Sun, Jizhou Huang*, Haifeng Wang, Ying Li. 2021. Meta-Learned Spatial-Temporal POI Auto-Completion for the Search Engine at Baidu Maps. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467058

1 INTRODUCTION

As one of the featured functions built in the search engine at Baidu Maps, Point Of Interest Auto-Completion (POI-AC), illustrated by Figure 1, can dynamically suggest a list of POI candidates within milliseconds whenever a user enters an English, a Chinese, or even a Pinyin character, into the search box. It keeps serving billions of search requests on POIs from people all over the world every day. Ideally, a user needs to enter only one character, and the POI-AC function can immediately display the desired POI at the top of the suggested POI list. As a result, a massive amount of time and effort of typing can be significantly reduced. This advantage is beneficial for map applications on mobile devices that have limited space for display and user interfaces.

In order to constantly save users' keystrokes, we have upgraded the POI-AC technology at Baidu Maps several times, contributing both LTR-based [12] and NN-based [5] solutions to the industry. The LTR-based POI-AC, named P^2AC [12], is the 2nd generation POI-AC technology, which uses the canonical pairwise learning to rank [13] framework with handcraft features for POI suggestions. To provide personalized POI suggestions, we further propose an end-to-end neural framework as the 3rd generation POI-AC, i.e., P^3AC [5], which takes a user's profile and the input prefixes into consideration and shows the state-of-the-art performance.

However, as a spatial-temporal mapping function, it is generally admitted that POI-AC should generate time- and location-aware suggestions. This conjecture has been recently confirmed by a data-driven study which shows that 17.9% of users at Baidu Maps tend to look for diverse POIs at different times or locations, even though they enter the same prefix as POI-AC requests. Figure 1 illustrates a representative use case in Shanghai, China. In this case, a user entered the same prefix into the search box in different districts or at different times, but have searched for highly diverse POIs.

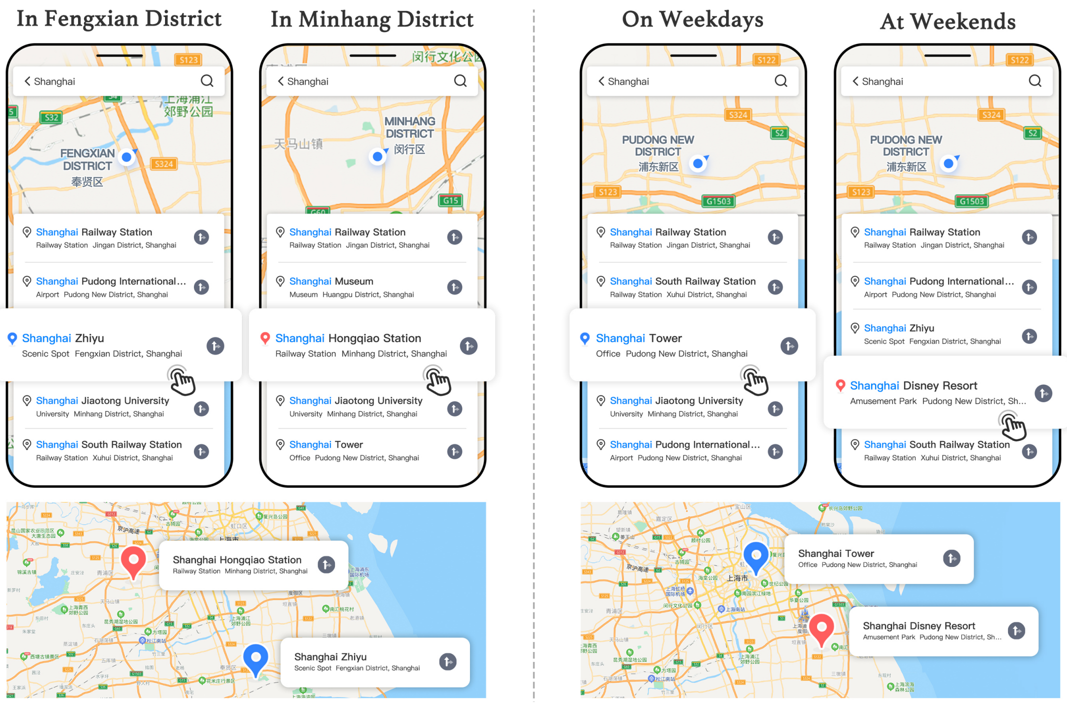
In Fengxian District
In Minhang District
On Weekdays
At Weekends
Shanghai
Shanghai Railway Station
Shanghai Pudong International...
Shanghai Zhiyu
Shanghai Jiaotong University
Shanghai South Railway Station
Shanghai Museum
Shanghai Hongqiao Station
Shanghai Tower
Shanghai Disney Resort
Shanghai Hongqiao Station
Railway Station Minhang District, Shanghai
Shanghai Zhiyu
Scenic Spot Fengxian District, Shanghai
Shanghai Tower
Office Pudong New District, Shanghai
Shanghai Disney Resort
Amusement Park Pudong New District, Sh...

Figure 1: A use case of the POI-AC module, i.e., P^3AC, at Baidu Maps, where a user tended to look for diverse POIs at different times or locations, although she entered the same prefix into the search box as POI-AC requests. We can see that her desired POIs, which were positively related to time and location, generally ranked at 3rd and 4th places. It leaves us room to improve the POI-AC module by leveraging the spatial-temporal characteristics of POI-AC requests.

For example, the POIs "Shanghai Zhiyu"/"Shanghai Hongqiao Station" were clicked on for the same prefix "Shanghai" by the user in Fengxian/Minhang Districts. In addition, if the prefix "Shanghai" was searched on weekdays or at weekends by the user, she was interested in the POIs "Shanghai Tower"/"Shanghai Disney Resort". This observation reveals that a user's search time and location could be beneficial for generating suggestions of POIs.

These insights drive us to build a novel POI-AC module that can generate not only prefix-specific POI candidates but, more importantly, time- and geography-aware POI suggestions. Therefore, we propose an end-to-end spatial-temporal POI-AC (abbr. as *ST-PAC*) module (see Section 2) to replace the previously deployed one (i.e., P^3AC [5]) at Baidu Maps. To be specific, *ST-PAC* is an end-to-end framework fed by five kinds of neural components: a time discretization layer to convert continuous time into discrete buckets, a geocoding layer to represent the coordinates of the POI-AC request, a user profile encoding layer, a prefix embedding network, and an enriched POI representation network. This design enables us to establish connections among the five key elements of POI-AC: time, location, user, prefix, and POI.

As the same as P^2AC and P^3AC, *ST-PAC* is a single and unique model deployed online serving all kinds of POI-AC requests. These models naturally perform well on high-frequency requests because adequate data help them rapidly adapt to "many-shot" scenarios. However, they are unable to serve a majority of POI-AC requests at "long-tailed" times and locations. It is apparently due to the fact that there are fewer examples in low-frequency times and regions, incapable of adapting the "giant" model for "few-shot" scenarios.

To alleviate the problem of the "long-tail" issue, we further propose a meta-learned *ST-PAC* (abbr. as *MST-PAC*) in Section 3. It can significantly overcome the "long-tail" issue by rapidly adapting to the cold-start POI-AC tasks with fewer examples. To make full use of the huge amount of POI-AC logs at Baidu Maps, we further propose a distributed training algorithm named $M^2ST\text{-}PAC$ based on MapReduce [4], which has exactly the same optimization effect, but is highly efficient.

We have sampled several benchmark datasets from the large-scale search logs at Baidu Maps to assess the offline performance of *MST-PAC* with several canonical metrics, including Mean Reciprocal Rank (MRR), Success Rate (SR), and normalized Discounted Cumulative Gain (nDCG). The experimental results demonstrate that *MST-PAC* can achieve consistent improvements on these offline metrics. In addition, the results give us more confidence to launch this meta-learned POI-AC module online. It turns out that the critical indicator on online user satisfaction, i.e., the average number of keystrokes in a POI-AC session, significantly decreases as well. For now, *MST-PAC* has already been deployed in production at Baidu Maps, serving billions of POI-AC requests every day, which confirms that *MST-PAC* is a practical and robust industrial solution for large-scale POI Search.

To summarize, we believe that hundreds of millions of users who search POIs at Baidu Maps will benefit from this new technology on POI-AC. Our key contributions to both research and industrial communities are listed as follows:

- **Potential impact**: POI-AC is an indispensable feature in the search engine of Baidu Maps, serving billions of requests every day. *MST-PAC* is our first attempt to develop a meta-learned POI-AC module, which outperforms the previously state-of-the-art module P^3AC [5] and has been successfully deployed online at Baidu Maps for months.
- **Novelty**: *MST-PAC* is able to not only provide time- and location-aware POI suggestions but also alleviate the problem of long-tail distribution of time- and location-specific data on POI-AC.
- **Scalability**: We further propose a distributed training algorithm (i.e., M^2ST-PAC) based on MapReduce to update our meta-learned POI-AC model. It can efficiently make full use of all the POI-AC logs at Baidu Maps and have the same effectiveness with the serial version, i.e., *MST-PAC*.
- **Technical quality**: Extensive assessments have been conducted both offline and online. Compared with the state-of-the-art model P^3AC, the experiments demonstrate that *MST-PAC* can consistently achieve significant improvements on multiple metrics of the POI-AC task.
- **Reproducibility**: We have released both the source code and the datasets collected from the real-world, large-scale POI-AC logs at Baidu Maps to the public at https://github.com/PaddlePaddle/Research/tree/master/ST_DM/KDD2021-MSTPAC/ to ensure the reproducibility of this work.

2 SPATIAL-TEMPORAL POI-AC

Statistics show that 17.9% of users at Baidu Maps tend to look for diverse POIs at different times or locations with the same prefix. It drives us to build an end-to-end POI-AC model that can generate not only prefix-specific POI candidates but, more importantly, time- and location-aware POI suggestions. In this section, we introduce an end-to-end spatial-temporal POI-AC (abbr. as *ST-PAC*) module, which is illustrated by Figure 2. It is composed of five neural components: a time discretization layer to convert continuous time into discrete buckets, a geocoding approach on representing the pair of coordinates of the POI-AC request, a user profile encoding layer, a prefix embedding network, and an enriched POI representation network. This design enables us to establish connections among the five critical elements of POI-AC: time (denoted by t), location (denoted by g), user (denoted by u), prefix (denoted by p), and POI (denoted by v).

2.1 Time Bucket Strategy

The temporal feature, i.e., $\mathbf{t}$, contains the date and the time that a user initializes a POI-AC request. Figure 1 illustrates an example that a user preferred looking for recreation places at weekends rather than on weekdays. This finding inspires us that the date of POI-AC requests can be first divided into 7 categories, and each class represents one day (from Monday to Sunday) in a week. However, time is a continuous feature that needs to be discretized to enhance the generalization ability of the temporal feature. In our

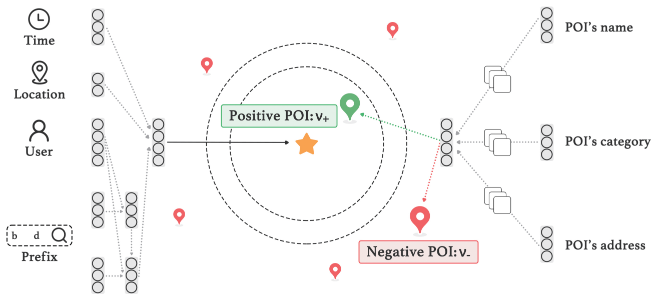

Figure 2: The end-to-end spatial-temporal POI-AC (ST-PAC) framework, which is composed of five neural components: a time discretization layer to convert continuous time into discrete buckets, a geocoding approach on representing the pair of coordinates of the POI-AC request, a user profile encoding layer, a prefix embedding network, and an enriched POI representation network.

work, we decide to use two buckets to represent a whole day. One bucket contains the POI-AC requests ranging from 6:00 a.m. to 6:00 p.m. to denote the daytime, and the other bucket covers the rest of the time to represent the night.

As a result, the temporal feature, i.e., $\mathbf{t}$, is a one-hot vector and its dimension is 14 (i.e., 7×2). For example, if a POI-AC request occurs at 8 o'clock on Monday morning, the temporal feature of this request is denoted by the vector as follows:

$$\mathbf{t} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). \tag{1}$$

2.2 Geocoding Approach

The spatial feature, i.e., $\mathbf{g}$, is denoted by a pair of coordinates that contains the longitude (x) and the latitude (y) of a POI-AC request. The numerical range of longitude is $(-180.000000, 180.000000)$ and that of latitude is $(-90.000000, 90.000000)$. Positive coordinates are to the east and north, which means the Western and Southern Hemispheres have negative longitudes and latitudes, respectively. Moreover, we usually get coordinates that are 6 decimal places long at Baidu Maps. For example, the coordinates of "Baidu Campus" are $(116.307916, 40.057063)$.

Although the spatial feature vector is composed of two continuous values, it will make the feature lack of generalization ability if we directly feed them into the POI-AC model as inputs. Therefore, we propose a geocoding approach to generate the 2-dimensional spatial feature (i.e., $\mathbf{g}$) of a POI-AC request as follows,

$$\mathbf{g} = (\lfloor x \times 10^3 \rfloor, \lfloor y \times 10^3 \rfloor), \tag{2}$$

where x and y stand for the longitude and latitude of the POI-AC request, respectively. As a result, POI-AC requests are aggregated by the same 3 digits after the decimal point, and the generalization ability of this spatial feature can be enhanced in this way.

2.3 Personalized Prefix Embedding

The POI-AC function can simultaneously update the suggested POI list as different users enter different prefixes into the search box of Baidu Maps. In other words, the search intent is composed of a user's characteristics and the prefix that she has typed in. Such observation inspires us to design a neural component for learning a personalized representation for each prefix.

As we know that a prefix p is a sequence of characters $(c_1, c_2, ..., c_n)$, we can adopt a Bi-LSTM [15] network to model the prefix. However, we need to "inject" the user's characteristics into each character of the prefix for the sake of obtaining a "personalized" prefix. Specifically, we need to first build a character set that contains about 8,000 Chinese characters, 10 Arabic numbers, and 52 kinds of English alphabets. Next, each character can be initialized by a random vector. Given the prefix $p = (c_1, c_2, ..., c_i, ..., c_n)$, we can concatenate each character embedding $\mathbf{c}_i$ with the user's feature vector $\mathbf{u}$. Then, each concatenated vector is fed into the Bi-LSTM network step by step:

$$\overrightarrow{\mathbf{s}_i} = \overrightarrow{\text{LSTM}}(\mathbf{c}_i \oplus \mathbf{u}, \overrightarrow{\mathbf{s}_{i-1}}), \tag{3}$$

and

$$\overleftarrow{\mathbf{s}_i} = \overleftarrow{\text{LSTM}}(\mathbf{c}_i \oplus \mathbf{u}, \overleftarrow{\mathbf{s}_{i+1}}), \tag{4}$$

where $\overrightarrow{\mathbf{s}_i}$ and $\overleftarrow{\mathbf{s}_i}$ are the inner states of the Bi-LSTM network at i-th step, and $\oplus$ is the concatenation operator. In this way, we can encode the past and future states (i.e., $\overrightarrow{\mathbf{s}_i}$ and $\overleftarrow{\mathbf{s}_i}$) of the entire user-specific prefix into each character c_i by

$$\mathbf{s}_i = \overrightarrow{\mathbf{s}_i} \oplus \overleftarrow{\mathbf{s}_i}, \tag{5}$$

where $\mathbf{s}_i \in \mathbb{R}^d$ stands for the personalized context embedding of the i-th input character c_i.

To re-weight the contribution of each contextual embedding to the entire prefix p, we use a single-time attention mechanism [1] formulated as follows:

$$a_i = \text{Softmax}(\mathbf{z}^{\text{T}} \tanh(\mathbf{W}\mathbf{s}_i)), \tag{6}$$

in which $\mathbf{W} \in \mathbb{R}^{l \times d}$ and $\mathbf{z} \in \mathbb{R}^l$ are tunable parameters.

We use $\mathbf{p} \in \mathbb{R}^l$ to denote the personalized prefix embedding, which equals to the weighted sum of the contextual embeddings:

$$\mathbf{p} = \sum_i a_i \mathbf{s}_i. \tag{7}$$

2.4 Enriched POI Representation

A POI, suggested by the POI-AC module at mapping applications, typically contains multi-sourced information such as its name, category, and address. This drives us to devise a fusion model to represent the enriched information of the POI.

Both the POI's name and address are mainly sequences of characters. Some representative characters include Chinese characters, Arabic numbers, and English alphabets. Given the different granularity of vocabulary, we adopt a canonical approach on CNN-based sentence embedding [11] to encode the textual information on the POI's name and its address. To be specific, each character embedding of the POI is leveraged as the input fed into the model of CNN-based sentence embedding to produce the distributed representations of the POI's name $\mathbf{e}_{name} \in \mathbb{R}^m$ and its address $\mathbf{e}_{address} \in \mathbb{R}^m$.

Although the POI category is composed of textual data, we consider that it belongs to categorical data as there are about 40 kinds of POI categories. We assign each kind of POI category an ID and an independent embedding $\mathbf{e}_{category} \in \mathbb{R}^m$ for subsequent calculations.

Then, we sum up the embeddings of the POI's name ($\mathbf{e}_{name} \in \mathbb{R}^m$), address ($\mathbf{e}_{address} \in \mathbb{R}^m$), and category ($\mathbf{e}_{category} \in \mathbb{R}^m$) to obtain an intermediated POI representation $\mathbf{e}_{poi}$:

$$\mathbf{e}_{poi} = \mathbf{e}_{name} + \mathbf{e}_{address} + \mathbf{e}_{category}. \tag{8}$$

Then $\mathbf{e}_{poi}$ is fed into a fully-connected network parameterized by $\mathbf{W}' \in \mathbb{R}^{m \times l'}$ to produce the enriched POI embedding ($\mathbf{v} \in \mathbb{R}^{l'}$):

$$\mathbf{v} = \text{ReLU}(\mathbf{e}_{poi}\mathbf{W}'). \tag{9}$$

2.5 N-Pairs Loss

After we have obtained the spatial-temporal features of a POI-AC request, the personalized prefix embedding, and the enriched representations of candidate POIs, we should establish connections between the spatial-temporal personalized prefix embedding and the ground-truth POI's representation. As the POI-AC function at Baidu Maps has logged large-scale anonymized search records, including the time and the location that a user types in a certain prefix, the POIs that our POI-AC module suggests, and the exact POI that the user clicks on. The click history helps us bridge the gap between the spatial-temporal personalized prefix and the ground-truth POI.

Let's assume that we have accumulated M examples of the search records as the training set Δ. The i-th data example in the training set can be denoted by a tuple which contains five kinds of feature vectors: the temporal feature ($\mathbf{t} \in \mathbf{I}^{14}$ and $I \in \{0, 1\}$), the spatial feature ($\mathbf{g} \in \mathbb{R}^2$), the personalized prefix embedding ($\mathbf{p}^{(i)} \in \mathbb{R}^l$), the distributed representation of the clicked POI ($\mathbf{v}^{(i)} \in \mathbb{R}^{l'}$, where $l' = l + 16$), and a set $\Theta^{(i)}$ that consists of the other suggested but unclicked POIs' embeddings.

By means of distributed representations, a simple way to measure the gap between the spatial-temporal personalized prefix and a POI is to define a distance function, $\text{h}(\cdot)$. In this work, we decide to use the cosine similarity to measure their distance as follows:

$$\text{h}(\mathbf{t} \oplus \mathbf{g} \oplus \mathbf{p}, \mathbf{v}) = \frac{(\mathbf{t} \oplus \mathbf{g} \oplus \mathbf{p}) \cdot \mathbf{v}}{|\mathbf{t} \oplus \mathbf{g} \oplus \mathbf{p}||\mathbf{v}|}. \tag{10}$$

As illustrated by Figure 2, the spatial-temporal personalized prefix embedding is set as the anchor, and we need to pair every anchor with a single positive (clicked) POI embedding and every negative (unclicked) POI embedding in the batch. We expect to "pull" the clicked POI's representation close to the spatial-temporal personalized prefix embedding in the l'-dimensional embedding space, and "push" all the unclicked POIs away. Therefore, we adopt the core idea of N-pairs loss [17] to define the loss function $\mathcal{L}_\Delta$ of *ST-PAC* model, which leverages all the unclicked POIs within a batch to guide the gradient update which speeds convergence. Overall, we need to minimize $\mathcal{L}_\Delta$ over the training set Δ, which is defined as follows:

$$\mathcal{L}_\Delta = \sum_{i=1}^{m} -\log \frac{\mathrm{e}^{\text{h}(\mathbf{p}^{(i)} \oplus \mathbf{g}^{(i)} \oplus \mathbf{t}^{(i)}, \mathbf{v}_+^{(i)})}}{\mathrm{e}^{\text{h}(\mathbf{p}^{(i)} \oplus \mathbf{g}^{(i)} \oplus \mathbf{t}^{(i)}, \mathbf{v}_+^{(i)})} + \sum_{\mathbf{v}_-^{(i,j)} \in \Theta^{(i)}} \mathrm{e}^{\text{h}(\mathbf{p}^{(i)} \oplus \mathbf{g}^{(i)} \oplus \mathbf{t}^{(i)}, \mathbf{v}_-^{(i,j)})}}, \tag{11}$$

where $\mathbf{v}_+^{(i)}$ denotes the positive (clicked) POI's embedding and $\mathbf{v}_-^{(i)}$ represents a negative (unclicked) POI embedding sampled from $\Theta^{(i)}$.

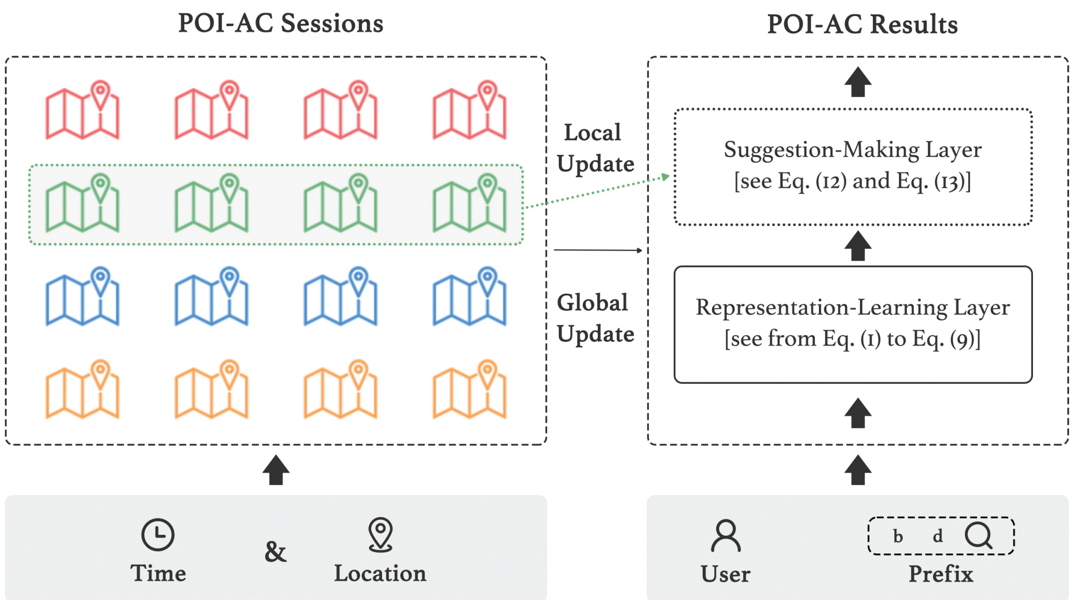
POI-AC Sessions
POI-AC Results
Local Update
Global Update
Suggestion-Making Layer [see Eq. (12) and Eq. (13)]
Representation-Learning Layer [see from Eq. (1) to Eq. (9)]
Time
&
Location
User
b d
Prefix

Figure 3: The meta-learned *ST-PAC* (abbr. as *MST-PAC*) framework, which can significantly overcome the "long-tail" issue by rapidly adapting to the cold-start POI-AC tasks with fewer examples. It is decoupled into two parts: a "suggestion-making" layer locally updated by the partial data of each spatial-temporal segment and a "representation-learning" layer globally updated by the full dataset.

3 META-LEARNED SPATIAL-TEMPORAL POI-AC

The end-to-end models on POI-AC, such as P^3AC and *ST-PAC*, naturally perform well on high-frequency requests as adequate data can facilitate them rapidly adapt to "many-shot" scenarios. The scenarios, in our opinion, refer to the POI-AC requests that occur at different times and locations.

For example, we find that the average number of keystrokes entered by users and the frequency of POI-AC requests in a region shows a significant negative correlation. Statistics show that the regions with top 20% POI-AC requests outperform nearly 30% higher click location than the others. It indicates that the end-to-end models are unable to serve a majority of POI-AC requests at "long-tailed" times and regions. It is apparently due to the fewer examples in low-frequency times and regions, incapable of adapting the "giant" model for "few-shot" [10] scenarios.

To alleviate the problem of "long-tail" issue, we further propose a meta-learned *ST-PAC* (abbr. as *MST-PAC*). It is expected to significantly overcome the "long-tail" issue by rapidly adapting to the cold-start POI-AC tasks with fewer examples. To be specific, we divide the full data into different parts, each of which belongs to a spatial-temporal scenario. As a result, each part of data can be leveraged to acquire a spatial-temporal POI-AC model. As illustrated by Figure 3, after we have obtained the personalized prefix embedding $\mathbf{p}$ (see Equation 7) and the enriched POI representation $\mathbf{v}$ (see Equation 9), we devise another distance function $\mathrm{h}'(\cdot)$ parameterized by $\mathbf{W}^*$ to replace Equation 10 as subsequence:

$$\mathrm{h}'(\mathbf{p},\mathbf{v}) = \tanh(\mathbf{W}^*(\mathbf{p}\oplus\mathbf{v})). \quad (12)$$

Correspondingly, the loss function of *MST-PAC*, denoted by $\mathcal{L}'_\Delta$, is defined as:

$$\mathcal{L}'_\Delta = \sum_{i=1}^{m} -\log \frac{e^{\mathrm{h}'(\mathbf{p}^{(i)},\mathbf{v}_+^{(i)})}}{e^{\mathrm{h}'(\mathbf{p}^{(i)},\mathbf{v}_+^{(i)})} + \sum_{\mathbf{v}_-^{(i,j)}\in\Theta^{(i)}} e^{\mathrm{h}'(\mathbf{p}^{(i)},\mathbf{v}_-^{(i,j)})}}. \quad (13)$$

Algorithm 1 shows the detailed procedure of acquiring *MST-PAC* model. It is quite different from the training procedure of other end-to-end models, such as P^3AC and *ST-PAC*, as the training algorithm of *MST-PAC* mainly contains two phases: local update and global update. Correspondingly, the model is also decoupled into two parts: a "suggestion-making" layer (marked with a red box) locally updated by the partial data of each spatial-temporal scenario and a "representation-learning" layer (marked with a green-dashed box) globally updated by the full data. As a result, the POI-AC preferences in different spatial-temporal scenarios are determined by the parameter $\mathbf{W}^*$ based on the unique POI-AC sessions of each scenario. We do not update the shared parameters such as $\mathbf{W}'$ and $\mathbf{W}$ in the representation-learning layer during the local update to ensure the stability of the learning process.

To make full use of the huge amount of POI-AC logs at Baidu Maps, we further propose a distributed training algorithm (see Algorithm 2) named M^2*ST-PAC* based on MapReduce, which is highly efficient to run on a Hadoop [16] cluster and has the same effectiveness with the serial version, i.e., *MST-PAC*.

Algorithm 1 The learning algorithm of MST-PAC (serial version)

Input: $\Delta = \{(\mathbf{u}^{(j)}, \mathbf{p}^{(j)}, \mathbf{v}^{(j)}, \Theta^{(j)}) \otimes (g,t); j \in \{1,2,3...,n\}, g \in G, t \in T\}$: The full dataset Δ for training is divided into $|G|\times|T|$ scenarios, where G denotes the set of locations and T represents the set of time buckets. In each spatial-temporal scenario, we randomly sample n examples as inputs which contain the user embeddings, the prefix embeddings, the ground-truth POI embeddings, and the negative POI sets. Other hyper-parameters, such as the learning rate ϵ, the covergence threshold τ, and the max step of iterations m.

Output: $\mathbf{W}$, $\mathbf{W}'$, $\mathbf{z}$, and $\mathbf{W}^*$: The parameters that are globally updated by the full dataset Δ. $\{\mathbf{W}^*_{(g,t)}; g \in G, t \in T\}$: The time & location-specific parameters ($|\{\mathbf{W}^*_{(g,t)}\}| = |G| \times |T|$) that are locally updated by the partial dataset $(\mathbf{u}^{(j)}, \mathbf{p}^{(j)}, \mathbf{v}^{(j)}, \Theta^{(j)})$ of each scenario.

```
1:  Randomly initialize $\mathbf{W}$, $\mathbf{W}'$, $\mathbf{z}$, and $\mathbf{W}^*$.
2:  repeat
3:      [Global update]
4:      Compute $(\mathbf{W}, \mathbf{W}', \mathbf{z}, \mathbf{W}^*) = \text{SGD}(\mathcal{L}'_\Delta)$ with Equation (13).
5:      [Local update]
6:      for $\{(g,t); g \in G, t \in T\}$ do prepare $\mathbf{W}^*_{(g,t)}$ and $\Delta_{(g,t)}$
7:          Compute $\mathbf{W}^*_{(g,t)} = \text{SGD}(\mathcal{L}'_{\Delta_{(g,t)}})$ with Equation (13);
8:      end for
9:      Update $\mathbf{W}^* = \frac{\epsilon}{|G|\times|T|} \sum_{g \in G, t \in T} (\mathbf{W}^* - \mathbf{W}^*_{(g,t)})$.
10: until convergence threshold $\tau$ or max step of iterations $m$
```

Algorithm 2 The distributed learning algorithm of MST-PAC, i.e., M^{2}ST-PAC (MapReduce version)

Input: $\Delta = \{(\mathbf{u}^{(j)}, \mathbf{p}^{(j)}, \mathbf{v}^{(j)}, \Theta^{(j)}) \otimes (g,t); j \in \{1,2,3...,n\}, g \in G, t \in T\}$: The full dataset Δ for training is divided into $|G|\times|T|$ scenarios, where G denotes the set of locations and T represents the set of time buckets. In each spatial-temporal scenario, we randomly sample n examples as inputs which contain the user embeddings, the prefix embeddings, the ground-truth POI embeddings, and the negative POI sets. Other hyper-parameters, such as the learning rate ϵ, the covergence threshold τ, and the max step of iterations m.

Output: $\mathbf{W}$, $\mathbf{W}'$, $\mathbf{z}$, and $\mathbf{W}^*$: The parameters that are globally updated by the full dataset Δ. $\{\mathbf{W}^*_{(g,t)}; g \in G, t \in T\}$: The time & location-specific parameters ($|\{\mathbf{W}^*_{(g,t)}\}| = |G| \times |T|$) that are locally updated by the partial dataset $(\mathbf{u}^{(j)}, \mathbf{p}^{(j)}, \mathbf{v}^{(j)}, \Theta^{(j)})$ of each scenario.

```
1:  Randomly initialize $\mathbf{W}$, $\mathbf{W}'$, $\mathbf{z}$, and $\mathbf{W}^*$.
2:  repeat
3:      [Map]
4:      Reserve $|G| \times |T|$ nodes on the Hadoop cluster;
5:      Load $\mathbf{W}$, $\mathbf{W}'$ and $\mathbf{z}$ into each node of the Hadoop cluster;
6:      Concurrently compute $\mathbf{W}^*_{(g,t)} = \text{SGD}(\mathcal{L}'_{\Delta_{(g,t)}})$ with Equation (13) at each node, where $g \in G$ and $t \in T$.
7:      [Reduce]
8:      Update $\mathbf{W}^* = \frac{\epsilon}{|G|\times|T|} \sum_{g \in G, t \in T} (\mathbf{W}^* - \mathbf{W}^*_{(g,t)})$;
9:      Compute $(\mathbf{W}, \mathbf{W}', \mathbf{z}, \mathbf{W}^*) = \text{SGD}(\mathcal{L}'_\Delta)$ as Equation (13).
10: until convergence threshold $\tau$ or max step of iterations $m$
```

4 EXPERIMENTS

In order to validate the effectiveness of our meta-learned POI-AC model, we conducted extensive experiments on *MST-PAC* and compared it with other POI-AC approaches that were previously deployed at Baidu Maps. Section 4.1 elaborates all the prior arts, from the 1st to the 3rd generation POI-AC approaches employed by Baidu Maps. As a featured function widely adopted by the search engine of Web mapping applications for industrial practice, all the approaches need to be tested thoroughly, regardless of offline or online. Therefore, we report the experimental results of offline assessment and online A/B testing in Section 4.2 and Section 4.3, respectively. Both experiments were conducted in December 2020, about one year later than the offline and online evaluations reported by P^3AC [5]. During this period, the POI-AC module was updated several times by the UI team at Baidu Maps. For this reason, the absolute values might be slightly different. However, the relative improvements are consistent.

4.1 Related Work

Distinguished from query auto-completion (abbr. as QAC) [3] which works on completing the user-generated queries, POI-AC aims at suggesting heterogeneous POIs, which are not just composed of their names but also their categories, addresses, coordinates, etc. For this reason, there also exists great challenges in the matching of prefixes and POIs. Therefore, except that we adopted a classical approach on QAC, i.e., MPC [2] as our 1st generation POI-AC approach (named after *PAC*), we have already published two modern techniques before *MST-PAC*: LTR-based POI-AC (abbr. as P^2AC), and neural-based POI-AC (abbr. as P^3AC), specifically deployed into the POI-AC function for industrial use at Baidu Maps. Here we regard them as the prior arts and elaborate on each of them as follows.

4.1.1 1G: PAC. Most Popular Completion (MPC) [2] approach was adopted by the search engine of Baidu Maps as the 1st generation of POI-AC function. i.e., *PAC*. It is mainly based on the statistical results of the large-scale queries entered by all users. In a word, *PAC* statistically builds connections between high-frequency prefixes and user-clicked POIs. However, it relies heavily on the behavioral data of users and cannot meet the individual needs of users.

4.1.2 2G: P^2AC. The 2nd generation of POI-AC function at Baidu Maps focuses on personalized POI-AC, which is short for P^2AC [12]. It attempts to establish connections between user profiles and POIs. The connections are leveraged as features to train an LTR (learning to rank) model to suggest POIs. Specifically speaking, a user profile is composed of not only the basic information on that user but also her historically queried POIs as one of the most important features fed into the LTR model. It refers to the similarity between the user profile and the POI candidates triggered by the prefix entered by the user.

4.1.3 3G: P^3AC. We have recently contributed a novel end-to-end framework for POI-AC, named P^3AC [5]. It focuses on modeling

Table 1: The statistics of the benchmark datasets for POI-AC, which are sampled from the search log of Baidu Maps.

Subset	**Beijing**				**Shanghai**				**Guangzhou**			
	#(Users)	#(Prefixes)	#(POIs)	#(Regions)	#(Users)	#(Prefixes)	#(POIs)	#(Regions)	#(Users)	#(Prefixes)	#(POIs)	#(Regions)
Train	802,240	1,068,075	1,081,544	46	755,536	1,003,356	1,169,132	24	555,757	713,044	961,088	21
Valid	119,516	130,380	332,518	43	113,489	123,595	339,159	24	81,331	88,263	259,452	20
Test	125,730	137,415	345,656	44	117,612	128,415	351,969	24	76,859	83,885	261,618	20
Total	960,737	1,335,870	1,214,630	46	905,182	1,255,366	1,314,709	24	665,820	885,192	1,092,717	21

personalized prefix embeddings for POI-AC without using handcraft features. P^3AC successfully enables the POI-AC function at Baidu Maps to establish connections among three key elements: users, prefixes, and POIs. Therefore, P^3AC renovates the LTR-based framework for POI-AC and shows the state-of-the-art performance.

4.2 Offline Assessment

4.2.1 Benchmark Datasets. Before launching any new models online, we need to conduct an offline evaluation by employing historical records to simulate the results of the new model. It is a fast and low-cost way to achieve the closest online performance of a new model, e.g., *MST-PAC*. Therefore, we collect a large number of POI-AC sessions from the search logs of Baidu Maps as the benchmark datasets for offline evaluation. Table 1 reports the statistics of the benchmark datasets for POI-AC. Each dataset records the POI-AC sessions in a metropolitan city in China and can be leveraged for model training (abbr. *Train*), hyper-parameter tuning (abbr. *Valid*), and performance testing (abbr. *Test*). Each example of data is mainly composed of an anonymized user identifier, the prefix entered by the user, the time when the POI-AC request starts, the region where the POI request begins, the POI list we suggested, and the exact POI that the user clicked. Compared with the datasets released by our previous work [5], we add the time and the location of each POI-AC request as the extra information to study spatial-temporal POI-AC models.

4.2.2 Evaluation Metrics. Ideally, we hope that the POI-AC function would display the user-desired POI in the first place for each request, which can significantly save the time and effort for individual users. Therefore, it becomes the key objective of POI-AC functions to rank the user-desired POI as higher as possible. In practice, we need to use multiple metrics, such as Success Rate (SR), Mean Reciprocal Rank (MRR), and normalized Discounted Cumulative Gain (nDCG), to measure the bias on the ranking position of the user-desired POI.

SR is a kind of macro indicator that computes the average percentage of the user-clicked POIs appearing in the ranked POI lists suggested by the POI-AC function. On mobile devices, we pay more attention to the POIs ranked in top 5 due to the restricted space for display on Baidu Maps, which can only show at most **5** suggested POIs on the first screen. Therefore, SR at top K (i.e., SR@K) is used to denote the average percentage of user-desired POIs that are displayed at or above the position K in the ranked POI lists suggested by a POI-AC module.

MRR and nDCG concern more about the exact position of the user-desired POI in the suggested POI list. They are, to some extent, two kinds of micro metrics to evaluate POI-AC offline. In other words, although two POI-AC models perform equally on SR@K, they probably demonstrate different results assessed by MRR and nDCG. To be specific, MRR calculates the average of the reciprocal ranks of the target POIs, and nDCG further measures the relevance between the suggested POIs and ground-truth POIs.

Table 2: The experimental results of the offline assessment on different approaches/versions of the POI-AC module for the search engine at Baidu Maps.

Model	**Offline Assessment**				
	MRR@5	**nDCG@5**	**SR@1**	**SR@3**	**SR@5**
PAC [2]	0.5487	0.6184	41.39%	65.09%	75.14%
P^2AC [12]	0.5941	0.6496	44.21%	69.80%	79.59%
P^3AC [5]	0.6293	0.7695	48.73%	75.02%	86.57%
ST-PAC	0.6977	0.8326	56.09%	81.96%	91.43%
MST-PAC	**0.7259**	**0.8530**	**59.71%**	**84.28%**	**92.49%**

4.2.3 Experimental Results. All the models are uniformly trained, validated, and tested by the benchmark datasets reported by Table 1. The experimental results of offline assessments reported by Table 2 are the average of the three test sets. From the results, we can see that the performance of the POI-AC approaches gradually increases from the 1st generation POI-AC (i.e., *PAC*) to the 4th (i.e., *MST-PAC*) proposed in this paper. *MST-PAC* outperforms all the previous approaches. Compared with the state-of-the-art method P^3AC [5] that was deployed online last year, *MST-PAC* achieves significant (absolute) improvements by 9.66% MRR@5, 8.35% nDCG@5, as well as 10.98% SR@1, 9.26% SR@3, and 5.92% SR@5, indicating that the new model is capable of achieving the best performance online.

4.3 Online A/B Testing

4.3.1 Real-World Traffic. Before launching the new model for production to serve all the POI-AC requests, we must conduct online A/B testing where both the state-of-the-art and the brand new models should serve an equal proportion of the real-world traffic for at least one week. This is to prevent deviations caused by the historical data leveraged by the offline evaluation. In our experiments, we randomly distributed about 10% POI requests to each model (i.e., *PAC*, P^2AC, P^3AC, *ST-PAC*, and *MST-PAC*) every day and monitor their performance during the period from December 21st, 2020 to December 28th, 2020. After we have confirmed that

Table 3: The experimental results of the online A/B testing on different approaches/versions of the POI-AC module for the search engine at Baidu Maps.

Model	**Online A/B Testing**				
	Avg. #(KS)	**MRR@5**	**SR@1**	**SR@3**	**SR@5**
PAC [2]	4.26	0.4951	39.27%	59.14%	61.35%
P^2AC [12]	4.03	0.5210	43.38%	62.37%	65.26%
P^3AC [5]	3.39	0.5668	46.07%	66.71%	70.24%
ST-PAC	2.84	0.5929	47.69%	67.82%	73.11%
MST-PAC	**2.62**	**0.6178**	**50.28%**	**69.38%**	**74.89%**

the new model can consistently outperform all the other comparison approaches during that period, we start to deploy the upgraded POI-AC module online, fully serving all the POI-AC requests. Even so, we are asked to keep an eye on the performance of the new POI-AC function at the time it begins to serve the full traffic on the POI search for some time, in case of the bias on traffic sampling.

4.3.2 Evaluation Metrics. Besides the position of the user-desired POI, the effort of typing is also concerned by the engineers of POI-AC functions, as it is directly related to user satisfaction, especially on mobile devices. Ideally, POI-AC functions are expected to display the target POI on the first screen even if only one character is entered by a user as the prefix. It drives us to set another objective of the POI-AC function, which is to save the user's time and effort on typing. To quantify the objective, we use the average number of keystrokes (Avg. #(KS)) as the online metric to measure the effort on typing made by users.

In addition, some metrics used by the offline evaluation, such as SR and MRR, are also adopted by A/B testing to measure the online performance of POI-AC functions. The main reasons are two-fold. First, the benchmark datasets for the offline evaluation only contain the POI-AC records in which at least one POI must be clicked. Second, some POI-AC results tend to be ignored by a number of online users, partly because they are used to directly type in the complete name of their desired POI by means of input method editors (IME). Both create an illusion that fewer suggested POIs were clicked online, which leads to much lower performance on SR and MRR.

4.3.3 Experimental Results. For the models mentioned in Section 4.1, we selected the best-performed ones via the offline evaluations to launch online. To conduct online A/B testing, each should further serve 10% POI-AC requests at Baidu Maps and be measured by multiple core indicators (see Section 4.3.2) on user satisfaction. Table 3 shows the experimental results of the online A/B testing. From the results, we can figure out that the type-in effort spent on our POI-AC function gradually decreases from the baseline approach (i.e., *PAC*) to the proposed method *MST-PAC*. Moreover, *MST-PAC* achieves much better performance online compared with P^3AC that has kept serving online over a year. Particularly, Avg.#(KS) is reduced to less than **3** characters for the first time by means of the spatial-temporal characteristics proposed by this paper. It seems that the online models generally achieve lower results on multiple metrics, such as MRR@5, SR@1, SR@3, and SR@5, compared with those reported by the offline evaluation. These discrepancies are due to the fact that nearly 15% POI-AC suggestions are ignored by a number of users at online search, which inevitably leads to lower performance on SR and MRR than that in the offline assessment. However, the online results are credible as the relative improvements on these metrics are consistent with those obtained from the offline evaluation.

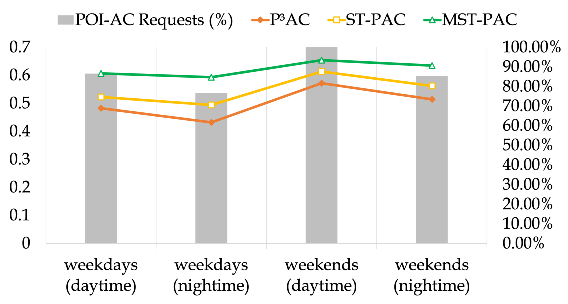

Figure 4: The online performance of P^3AC, *ST-PAC*, and *MST-PAC* on MRR@5 (y-axis on the left) in different time periods (x-axis) in Shanghai with various proportions of POI-AC requests (y-axis on the right).

5 DISCUSSIONS

Although the results in Table 3 have demonstrated that *MST-PAC* generally outperforms all the other models in both online and offline assessments, we still would like to investigate the conditions under which our model has made such significant progress. Therefore, we monitored the online performance of P^3AC, *ST-PAC*, and *MST-PAC* on MRR@5 at different times and regions in Shanghai over a week in 2021 as case studies to discuss the robustness of the models.

5.1 Robustness on Long-Tailed Periods

Figure 4 illustrates the curves of the online performance of P^3AC, *ST-PAC*, and *MST-PAC* on MRR@5 (y-axis on the left) in different time periods in Shanghai. We also plot the proportions of POI-AC requests (y-axis on the right) in the 4 time periods to explore the robustness of those models. It shows that the performance of the three models is positively correlated with the number of POI-AC requests. However, there is a significant difference in the degree of performance change between them. The standard deviation of the performance of *MST-PAC* is 2.74×10^{-2}, which is much lower than those of *ST-PAC* (5.18×10^{-2}) and P^3AC (5.85×10^{-2}). It indicates that *MST-PAC* is able to alleviate the problem of the "long-tail" distribution of time-specific data on POI-AC.

5.2 Robustness on Long-Tailed Regions

Figure 5 illustrates the relation curves between the number of POI-AC requests (x-axis) and the POI-AC performance on MRR@5 (y-axis). It shows that the three models obtain even higher performance than that reported by Table 3 at the regions with high-frequency

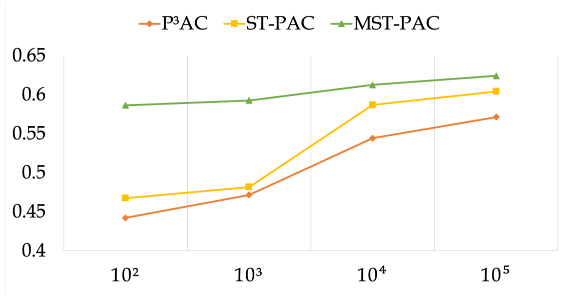

Figure 5: The online performance of P^3AC, *ST-PAC*, and *MST-PAC* on MRR@5 (y-axis) at different regions in Shanghai with various numbers of POI-AC requests (x-axis).

requests (i.e., from 10^4 to 10^5). However, *MST-PAC* obtains a relatively higher performance at "low-frequency" regions (i.e., from 10^2 to 10^3), while the performance of other two end-to-end models (i.e., *ST-PAC* and P^3AC) declines significantly. It verifies our hypothesis that *MST-PAC* can alleviate the problem of the "long-tail" distribution of location-specific data on POI-AC.

6 CONCLUSIONS

Our latest study on POI-AC found that 17.9% of users at Baidu Maps tend to look for diverse POIs at different times or locations with the same prefix. However, the state-of-the-art approach as well as the 3rd generation of POI-AC at Baidu Maps, P^3AC [5], is unable to serve this proportion of users well, as it only takes a user's profile, her personalized prefixes, and correspondingly clicked POIs into consideration. This paper presents our first attempt to deploy a meta-learned POI-AC module online as the 4th generation of POI-AC for the search engine of map applications on mobile devices. It can provide not only personalized but, more importantly, time- and geography-aware suggestions. This modern approach has made several novel contributions to the search engine at Baidu Maps as follows:

- an end-to-end spatial-temporal POI-AC (abbr. as *ST-PAC*) model to make POI suggestions aware of a wide range of factors on search, including the user (who), the time (when), the location (where), and the prefix (what),
- a meta-learned paradigm for *ST-PAC* (abbr. as *MST-PAC*) to alleviate the problem of the long-tail distribution of time- and location-specific data on POI-AC,
- and a distributed algorithm based on MapReduce to implement *MST-PAC* (abbr. as $M^2ST\text{-}PAC$) so as to acquire a deployable model handling billions of POI-AC requests at Baidu Maps every day.

As results, both offline and online assessments demonstrate that $M^2ST\text{-}PAC$ consistently outperforms the prior arts including P^3AC, achieving substantial improvements on multiple canonical metrics such as Mean Reciprocal Rank (MRR), Success Rate (SR), normalized Discounted Cumulative Gain (nDCG), and the average number of keystrokes (Avg. #(KS)). For reproducibility tests, we have also released the source code and the benchmark datasets, which were sampled from the large-scale search logs at Baidu Maps to measure the offline performance of POI-AC approaches.

7 FUTURE WORK

We believe that it still leaves several open issues that deserve more attention in the future:

- *Data preprocessing*: The meta-learned POI-AC model proposed by this paper can rapidly adapt to different parts of the data even though most of them are "long-tailed". We consider that the "long-tailed" phenomenon not only exists in the spatial-temporal scenario but also appears in POI search history. Therefore, we look forward to exploring other meta-learned POI-AC models from the perspective of personalized user preference on POI search.
- *Distributed model training*: To make *MST-PAC* fully acquire the large-scale POI search log at Baidu Maps, we propose an efficient MapReduce algorithm named $M^2ST\text{-}PAC$ to update *MST-PAC* based on Reptile [14], which is a scalable meta-learning algorithm. We believe that $M^2ST\text{-}PAC$ is just one option to train a meta-learned POI-AC model at scale. In the future, we will explore other scalable meta-learning algorithms to optimize our POI-AC function at Baidu Maps.
- *Introducing Context and Explanation*: Previous studies have shown that context [6, 7] and explanation [8, 9] can bring significant improvements in recommendation effectiveness and increase user satisfaction. As future work, we plan to investigate whether POI-AC could benefit from the adoption of such factors.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473* (2014).
[2] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive Query Auto-completion. In *WWW*. 107–116.
[3] Fei Cai and Maarten de Rijke. 2016. A Survey of Query Auto Completion in Information Retrieval. *Found. Trends Inf. Retr.* 10, 4 (Sept. 2016), 273–363.
[4] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*. 137–150.
[5] Jizhou Huang, Haifeng Wang, Miao Fan, An Zhuo, and Ying Li. 2020. Personalized Prefix Embedding for POI Auto-Completion in the Search Engine of Baidu Maps. In *KDD*. 2677–2685.
[6] Jizhou Huang, Haifeng Wang, Wei Zhang, and Ting Liu. 2020. Multi-Task Learning for Entity Recommendation and Document Ranking in Web Search. *ACM Trans. Intell. Syst. Technol.* 11, 5, Article 54 (July 2020), 24 pages.
[7] Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. 2018. Improving Entity Recommendation with Search Log and Multi-Task Learning. In *IJCAI*. 4107–4114.
[8] Jizhou Huang, Wei Zhang, Shiqi Zhao, Shiqiang Ding, and Haifeng Wang. 2017. Learning to Explain Entity Relationships by Pairwise Ranking with Convolutional Neural Networks. In *IJCAI*. 4018–4025.
[9] Jizhou Huang, Shiqi Zhao, Shiqiang Ding, Haiyang Wu, Mingming Sun, and Haifeng Wang. 2016. Generating Recommendation Evidence Using Translation Model. In *IJCAI*. 2810–2816.
[10] Suvarna Kadam and Vinay Vaidya. 2018. Review and Analysis of Zero, One and Few Shot Learning Approaches. In *ISDA*. 100–112.
[11] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. 1746–1751.
[12] Ying Li, Jizhou Huang, Miao Fan, Jinyi Lei, Haifeng Wang, and Enhong Chen. 2020. Personalized Query Auto-Completion for Large-Scale POI Search at Baidu Maps. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 19, 5, Article 70 (2020).
[13] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (March 2009), 225–331.
[14] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On First-Order Meta-Learning Algorithms. *arXiv:1803.02999* (2018).
[15] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.
[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. 2010. The Hadoop Distributed File System. In *MSST*. 1–10.
[17] Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *NIPS*. 1857–1865.