Personalized Prefix Embedding for POI Auto-Completion in the Search Engine of Baidu Maps

Jizhou Huang, Haifeng Wang, Miao Fan, An Zhuo, Ying Li
Baidu Inc., Beijing, China
{huangjizhou01,wanghaifeng,fanmiao,zhuoan,liying}@baidu.com

ABSTRACT

Point of interest auto-completion (POI-AC) is a featured function in the search engine of many Web mapping services. This function keeps suggesting a dynamic list of POIs as a user types each character, and it can dramatically save the effort of typing, which is quite useful on mobile devices. Existing approaches on POI-AC for industrial use mainly adopt various learning to rank (LTR) models with handcrafted features and even historically clicked POIs are taken into account for personalization. However, these prior arts tend to reach performance bottlenecks as both heuristic features and search history of users cannot directly model personal input habits. In this paper, we present an end-to-end neural-based framework for POI-AC, which has been recently deployed in the search engine of Baidu Maps, one of the largest Web mapping applications with hundreds of millions monthly active users worldwide. In order to establish connections among users, their personal input habits, and correspondingly interested POIs, the proposed framework (abbr. P^3AC) is composed of three components, i.e., a multi-layer Bi-LSTM network to adapt to personalized prefixes, a CNN-based network to model multi-sourced information on POIs, and a triplet ranking loss function to optimize both personalized prefix embeddings and distributed representations of POIs. We first use large-scale real-world search logs of Baidu Maps to assess the performance of P^3AC offline measured by multiple metrics, including Mean Reciprocal Rank (MRR), Success Rate (SR), and normalized Discounted Cumulative Gain (nDCG). Extensive experimental results demonstrate that it can achieve substantial improvements. Then we decide to launch it online and observe that some other critical indicators on user satisfaction, such as the average number of keystrokes and the average typing speed at keystrokes in a POI-AC session, which significantly decrease as well. In addition, we have released both the source codes of P^3AC and the experimental data to the public for reproducibility tests.

CCS CONCEPTS

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '20, August 23–27, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403318

ACM Reference Format:
Jizhou Huang, Haifeng Wang, Miao Fan, An Zhuo, Ying Li. 2020. Personalized Prefix Embedding for POI Auto-Completion in the Search Engine of Baidu Maps. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403318

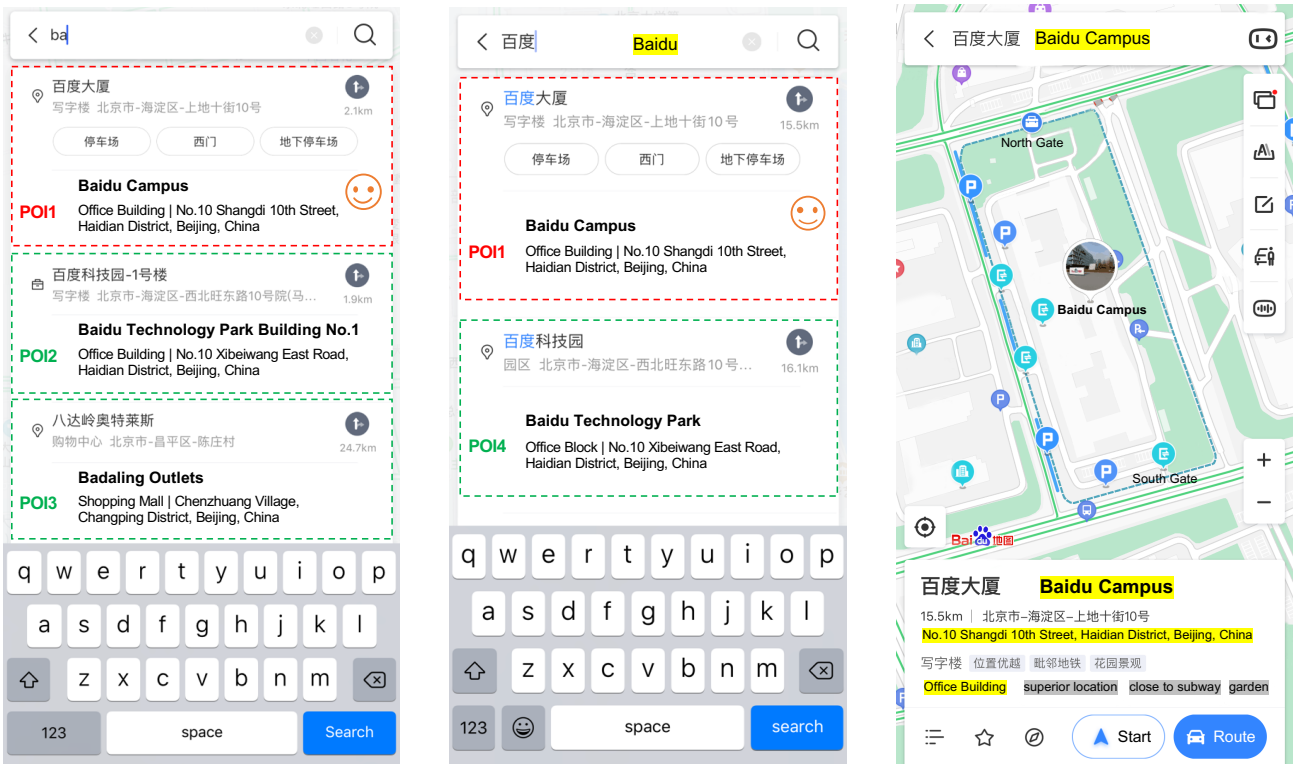
1 INTRODUCTION

Point of interest auto-completion (POI-AC) is a featured module in the search engine of many Web mapping services such as Baidu Maps. Baidu Maps is one of the largest Web mapping applications with over 340 million monthly active users worldwide by the end of December 2016.[1] Figure 1a shows an example of POI-AC results for the prefix "ba" at Baidu Maps. As illustrated in this example, it provides a user-friendly function that keeps suggesting a dynamic list of POIs as a user types each character. POI-AC plays an essential role in improving the usability of POI retrieval in mobile map apps. The objective of this module is to precisely suggest the user-desired POI and display it at the top of the ranking results with as few keystrokes as possible. In this way, the number of typed characters can be dramatically reduced [30], as well as spelling mistakes could be avoided [8], which in turn greatly improves the user's satisfaction [26].

To the best of our knowledge, existing approaches on POI-AC for industrial use mainly adopt similar framework for query auto-completion (QAC) [4, 7] in Web search engines, where various learning to rank (LTR) [19] models are leveraged with handcrafted features to surface the target POI or query. However, POIs are quite distinct from user-generated queries, as a POI is a geographic entity (see Figure 1c) that contains multi-sourced and heterogeneous information such as its GPS coordinates, address, and even photographs, leading to a consideration of more aspects on POIs. In addition, historically clicked queries or POIs can be further taken into consideration to generate personalized suggestions [13]. However, these prior arts tend to easily reach performance bottlenecks as both heuristic features and search history of users cannot directly model personal input habits. To be specific, the personal input habits in POI-AC are commonly regarded as the input method editor (IME) a user prefers and the specific prefix she usually types in when she wants to find a certain POI. Therefore, the key objective of personalized POI-AC, in our opinion, is to tailor suggested POIs to individual user preferences and input habits, which drives us to build up connections among users, their personalized prefixes, and correspondingly clicked POIs.

In this paper, we present our recent breakthrough on the design and implementation of a new personalized POI-AC module, which

[1] http://ir.baidu.com/static-files/e249a0f8-082a-4f8a-b60d-7417fa2f8e7e



(a) The list of POIs provided by the POI-AC module in the search engine of Baidu Maps app. The auto-completed POIs were triggered by a user-preferred prefix "ba", which is composed of two Pinyin characters "b" and "a".

(b) The list of POIs provided by the POI-AC module in the search engine of Baidu Maps app. The auto-completed POIs were triggered by another user-preferred prefix "百度", which is composed of two Chinese characters "百" and "度".

(c) The screenshot of the detailed information on the target POI "百度大厦" (i.e., POI1: "Baidu Campus") at Baidu Maps app. Besides GPS coordinates, the multi-sourced and heterogeneous information on a POI includes its name, category, and address.

Figure 1: Two screenshots of the auto-completed points of interest (POIs) triggered by two mainstream forms of prefixes (Pinyin characters and Chinese characters) at Baidu Maps app, and one screenshot of the multi-sourced and heterogeneous information on the target POI named "百度大厦" ("Baidu Campus"). In both cases, our POI-AC module helped the two users save much effort on searching their target POI (i.e., POI1: "Baidu Campus"), which is displayed in advance in the 1st place, given a short prefix of either "ba" or "百度".

has already been deployed in the search engine of Baidu Maps to achieve the goal of significantly saving users' effort of typing, facilitating tens of millions of users to find their desired POIs every day. This new module is powered by an end-to-end neural-based framework (abbr. P^3AC)[2], which is composed of three components: a personalized prefix embedding network, an enriched POI embedding network, and a triplet ranking loss function. This design enables us to establish connections between personalized prefix embeddings and distributed representations of POIs by leveraging large-scale search logs at Baidu Maps.

To be specific, the personalized prefix embedding network takes the combination of user features and each character embedding as input and can generate distributed representations of personalized prefixes at each step by means of a multi-layer Bidirectional Long Short-Term Memory (Bi-LSTM) [10] network. We use Convolutional Neural Nets (CNN) [15] to capture semantic features from a POI's name as well as its address and integrate these linguistic features with other types of information such as its GPS coordinates via Feed Forward Neural Nets (FFNN) [16] to produce an enriched POI embedding. After acquiring both the personalized prefix embedding and enriched POI embeddings, we need a both effective and efficient loss function to distinguish the user-desired POI from the other suggested POIs. The triplet ranking loss function [23, 27] seems a wise choice as it conventionally outperforms most point-wise LTR models [11] and does not require too many computation resources like list-wise LTR models [29].

The large-scale search logs at Baidu Maps are real-world data that can be employed to train the parameters of P^3AC, select the hyperparameters of P^3AC, and test its offline performance measured by multiple metrics, including Mean Reciprocal Rank (MRR), normalized Discounted Cumulative Gain (nDCG), and Success Rate (SR). The training and test sets consist of millions of users' search records for several months, covering hundreds of cities and tens

[2] P^3AC is short for ***P***ersonalized ***P***refix embedding for ***P***OI ***A***uto-***C***ompletion.

of millions of POIs in China. Experimental results of the offline evaluation demonstrate that P^3AC achieves substantial (absolute) improvements on those metrics compared with the strongest baseline method.

Then we decided to launch this end-to-end framework online to serve a portion of the search traffic for a couple of weeks. This A/B testing was conducted between our P^3AC and the existing LTR-based POI-AC module. We observed that some other critical indicators such as the average number of keystrokes (Avg. #(KS)) and the average typing speed at keystrokes (Avg. Sp.@KS) in a POI-AC session, which are also related to user satisfaction but can hardly be monitored by the offline evaluation, significantly decreased as well, indicating that P^3AC is a better framework to replace the LTR-based module.

P^3AC is our first attempt to deploy an end-to-end neural framework for POI-AC for industrial use. It is entirely implemented by *PaddlePaddle*[3], an open-source deep learning platform maintained by Baidu. Moreover, we have released both the source codes and the datasets to the public for reproducibility tests.[4]

Overall, this paper presents that we have made multiple contributions to POI-AC for industrial practice:

- **Potential impact**: We propose an end-to-end neural framework as an industrial solution to the personalized POI-AC module in the search engine of Web mapping services. It is our first attempt to deploy the framework into Baidu Maps to serve tens of millions of users every day.
- **Novelty**: The design and implementation of this framework are driven by the novel idea that establishes the connections among users, their personalized prefixes, and correspondingly clicked POIs.
- **Technical quality**: Extensive experimental results from both offline and online evaluations using large-scale real-world data demonstrate that the proposed framework is superior to the mainstream approaches because it achieves dramatic improvements on multiple indicators, saves much typing and brings much better user experience.
- **Reproducibility**: We have also released both the source codes of P^3AC and the large-scale real-world datasets collected from Baidu Maps to the public to ensure the reproducibility of our work.

For the rest of this paper, we review related work on POI-AC from the perspectives of academic research and industrial practice in Section 2. After having considered the pros and cons of the existing approaches on POI-AC, we present our novel neural framework, i.e., P^3AC, for POI-AC in Section 3. We conducted extensive experiments in which the performance of P^3AC and several other competitive models were evaluated both online and offline. Section 4 reports the experimental results, and Section 5 discusses the reason why this new framework is more effective on POI-AC. In the last two sections, i.e., Section 6 and Section 7, we conclude our work and point out some promising directions that are deserved to be explored in the future.

[3]The official site is at https://www.paddlepaddle.org.cn/.

[4]Both the source codes of P^3AC and the datasets are available at https://github.com/PaddlePaddle/Research/tree/master/ST_DM/KDD2020-P3AC.

2 PRIOR ARTS

To the best of our knowledge, there has been very little work on POI-AC, but we consider to review related academic research on query auto-completion (QAC) [4], which may inspire the work on POI-AC to some extent. And then, we introduce our existing work on POI-AC from the perspective of industrial practice in the search engine of Baidu Maps.

2.1 Academic Exploration

2.1.1 Heuristic Query Auto-Completion. Heuristic QAC approaches aim to compute a probability of a query completion after typing a prefix. A straightforward model on ranking query completions is to use Maximum Likelihood Estimation (MLE) according to the past popularity or frequency of queries. Bar-Yossef and Kraus [2] named this type of ranking methods as the most popular completion (MPC) model. The MPC model, in essence, assumes that the current query popularity distribution will be the same as that previously observed. In other words, completed queries are suggested by their past popularity or frequency for the sake of maximizing the effectiveness of QAC for all users on average. However, it is a common sense that query popularity strongly depends on time. Therefore, some studies focus on exploring the time-sensitive query auto-completion models based on MPC [5, 6, 26, 28].

2.1.2 LTR-Based Query Auto-Completion. The rise of learning to rank (LTR) [19] inspired the following work of QAC to understand ranking principles based on user interactions with search engines. In a canonical LTR-based framework for query auto-completion [17, 30], the input is a prefix, i.e., a string of several characters. And the input can trigger a pool that consists of query completions that start with that prefix. Each completion can be represented as a prefix-query feature vector, which can be associated with binary labels, i.e., "clicked" or "not-clicked" for training. Learning-based QAC models mainly focus on extracting useful features to seek a correct prediction of the user's intended query. Such QAC models eventually learn a ranking function from the extracted features.

2.1.3 Personalized Query Auto-Completion. Besides the input prefix, research on personalized QAC expected to use the personalized profile as the ranking signal. Therefore, the key challenge of personalized QAC is how to model the specific profile of a user. The user's long-term search history is leveraged in [3] to selectively personalize QAC. Shokouhi [25] studies QAC personalization using a combination of demographic features and context-based textual features, and shows that the user's long-term search history and location are the most effective evidence for QAC personalization. Jiang and Cheng [12] further point out that both long-term and short-term search intents are important to QAC. Generally speaking, those studies remind us to consider the long-term and short-term search intents and histories, as well as the demographic features of users for personalized POI-AC.

2.2 Industrial Practice

Baidu Maps employs a two-layer framework for POI-AC in the POI search engine [18]. As illustrated by Figure 2, this funnel-shaped framework is composed of two layers: the layer of *candidate POI generation* on the top and the layer of *personalized POI suggestion* at

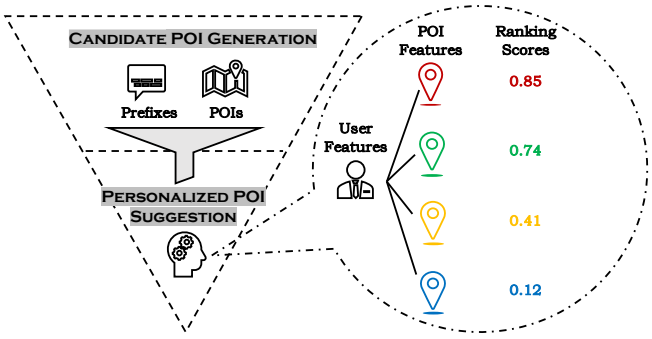


Figure 2: The existing LTR-based framework for POI-AC in the search engine of Baidu Maps [18]. This funnel-shaped framework is composed of two layers: the layer of *candidate POI generation* on the top and the layer of *personalized POI suggestion* at the bottom. We adopt GBRank as the suggestion model at the bottom, which takes hundreds of POI candidates from the top layer as input and learns to rank them in accordance with the features of both users and POIs.

the bottom. The reason why we adopt this canonical strategy is that it can take into account both efficiency and effectiveness, which is especially useful for a mapping service on the Web, maintaining hundreds of millions of POIs and serving a significant number of users every day.

Most academic explorations focus on the bottom layer, which takes hundreds of POI candidates from the top layer as input and learns to rank them in accordance with the features of both users and POIs. Baidu Maps adopts GBRank [31] as the specific LTR model because it is able to provide a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model.

3 PROPOSED FRAMEWORK

Regardless of the purpose of academic research or industrial use, the prior arts, which mainly adopt LTR models for POI-AC, tend to reach performance bottlenecks as both heuristic features and the search history of users cannot directly model the intricate input habits of each user. To address the issue, we present an end-to-end neural-based framework (abbr. P^3AC) for POI-AC in this section, which is elaborately designed to establish connections among users, their personal input habits, and correspondingly interested POIs. As illustrated by Figure 3, P^3AC is composed of three components, i.e., a multi-layer Bi-LSTM network to adapt to personalized prefixes, a CNN-based network to model multi-sourced information on POIs, and a triplet ranking loss function to simultaneously optimize both personalized prefix embeddings and distributed representations of POIs.

3.1 Personalized Prefix Embedding

The POI-AC function is automatically activated whenever a user is entering the first Pinyin or Chinese character into the search box of Baidu Maps. It keeps suggesting self-update ranked lists of completed POIs as the user types in more characters to replenish the prefix, until the user clicks her desired POI in a certain list. In other words, the search intent is composed of the user characteristics and the prefix that she has typed in. Such observation inspires us to design the following network to learn a personalized representation for each prefix.

Given that a prefix p is a sequence of characters $(c_1, c_2, ...c_n)$, we can adopt a Bi-LSTM [9, 24] network to model the prefix. In order to obtain a user-specific prefix, we consider to "inject" the user characteristics into the prefix. To be specific, we first build a character set which contains about 8,000 Chinese characters, 10 Arabic numbers, and 52 kinds of English alphabets. Then, each character is initialized with a random vector. Given a prefix $p = (c_1, c_2, ..., c_i, ..., c_n)$, we concatenate each character embedding $\mathbf{c}_i$ with the user's feature vector $\mathbf{u}$, and feed each concatenated vector into the Bi-LSTM network step by step:

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

and

$$\overleftarrow{\mathbf{s}_i} = \overleftarrow{\text{LSTM}}(\mathbf{c}_i \oplus \mathbf{u}, \overleftarrow{\mathbf{s}_{i+1}}), \tag{2}$$

where $\overrightarrow{\mathbf{s}_i}$ and $\overleftarrow{\mathbf{s}_i}$ are the inner states of the Bi-LSTM network at i-th step, $\oplus$ 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 personal prefix into each character c_i by

$$\mathbf{s}_i = \overrightarrow{\mathbf{s}_i} \oplus \overleftarrow{\mathbf{s}_i}, \tag{3}$$

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 adopt a single-time attention mechanism [1] formulated as follows:

$$a_i = \text{softmax}(\mathbf{z}^{\text{T}} \tanh(\mathbf{W}\mathbf{s}_i)), \tag{4}$$

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

We use $\mathbf{e}_{u.p.} \in \mathbb{R}^l$ to denote the distributed representation of prefix p that user u has typed in, and the vector equals to the weighted sum of the contextual embeddings:

$$\mathbf{e}_{u.p.} = \sum_i a_i \mathbf{s}_i. \tag{5}$$

3.2 Enriched POI Embedding

Different from QAC, the POI-AC module in Web mapping services mainly suggests POIs rather than queries. A complete POI typically contains multi-sourced and heterogeneous information such as GPS coordinates (i.e., longitude and latitude), name, category, and address. This motivates us to employ an appropriate model to encode such information of a POI.

Both POI's name and address are mainly word sequences composed of Chinese characters, Arabic numbers, and English alphabets. To encode the textual information on POI name and address, we adopt a canonical approach on CNN-based sentence embedding [14]. For the sake of convenience, we denote this approach by SenCNN(·). To be specific, we take each character embedding as input, and use SenCNN(·) to produce the distributed representations of a POI's name $\mathbf{v}_{name} \in \mathbb{R}^m$ and its address $\mathbf{v}_{addr.} \in \mathbb{R}^m$.

Even though the POI category is composed of textual data, we regard it as categorical data as there are around 40 kinds of POI

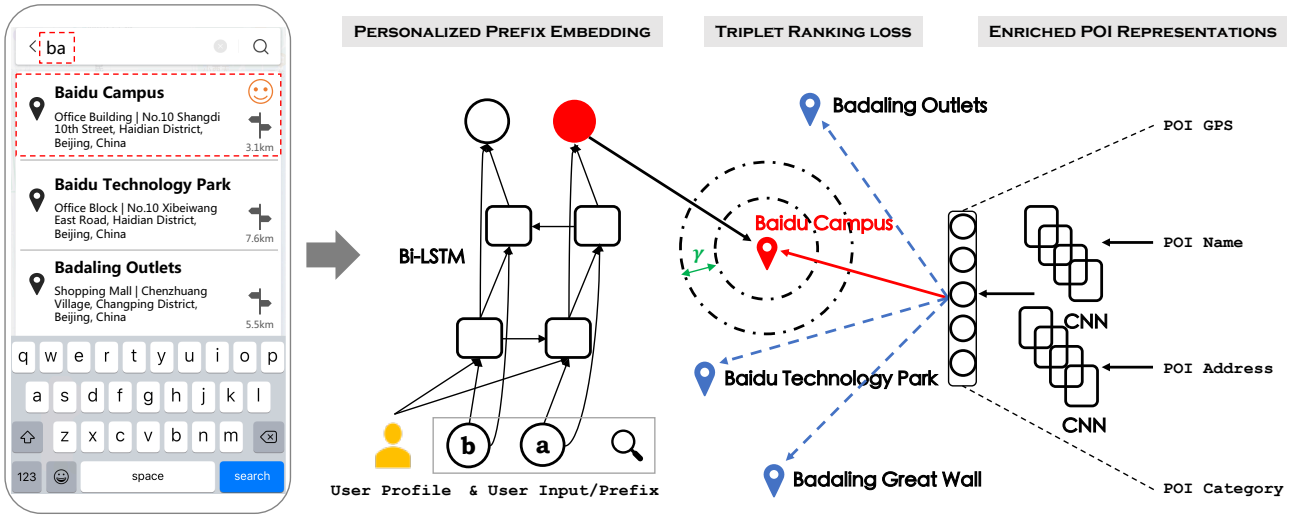


Figure 3: The end-to-end architecture of P^3AC, which is composed of three neural components, i.e., a multi-layer Bi-LSTM network to adapt to the personalized prefix (i.e., a combination of the user profile and the user input/prefix), a CNN-based network to model the multi-sourced information (i.e., GPS, name, address, and category) on POIs, and a triplet ranking loss function to optimize both the personalized prefix embedding and distributed representations of POIs. In this case, just after the user had typed in the prefix "ba", our POI-AC module provided a ranked list of POIs (including "Baidu Campus", "Baidu Technology Park", "Badaling Outlets", etc.) to the user in real time. Then the user clicked the 1st POI ("Baidu Campus") as her desired POI (as a positive example), leaving the other unclicked POIs as negative examples.

categories. Each kind of category has an ID and is initialized by an independent embedding $\mathbf{v}_{cate.} \in \mathbb{R}^m$.

POI GPS coordinates are numeric data. A simple way to take advantage of these data is to feed the 2-dimensional feature vector into the model directly. An alternative way is to use Geohash [22], which encodes a geographic location into a short string of letters and digits. Given that Geohash provides a hierarchical structure for spatial data via subdividing space into buckets of grid shape, the POIs that are near to each other can share the same geo-embedding. We use the latter to get the embedding of POI GPS, i.e., $\mathbf{v}_{GPS} \in \mathbb{R}^m$.

Then, we sum up the embeddings of POI name ($\mathbf{v}_{name} \in \mathbb{R}^m$), POI address ($\mathbf{v}_{addr.} \in \mathbb{R}^m$), POI category ($\mathbf{v}_{cate.} \in \mathbb{R}^m$), and POI GPS ($\mathbf{v}_{GPS} \in \mathbb{R}^m$) to obtain the enriched POI representation $\mathbf{v}_{poi}$:

$$\mathbf{v}_{poi} = \mathbf{v}_{name} + \mathbf{v}_{addr.} + \mathbf{v}_{cate.} + \mathbf{v}_{GPS}. \tag{6}$$

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

$$\mathbf{e}_{poi} = \text{sigmoid}(\mathbf{v}_{poi}\mathbf{W}'). \tag{7}$$

3.3 Triplet Ranking Loss

After we obtain both the personalized prefix embeddings and the enriched representations of POIs, we need to establish connections between them. The POI-AC function of the search engine at Baidu Maps has logged large-scale anonymized search records, including the prefix that a user types in, 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 user prefixes and POIs.

Assume that we have accumulated M examples as the training set Δ, and the i-th data example can be denoted by a triplet which contains: the user prefix embedding: $\mathbf{e}^{(i)}_{u.p.} \in \mathbb{R}^l$, the distributed representation of the clicked POI: $\mathbf{e}^{(i)}_{poi} \in \mathbb{R}^l$, and a set $\Theta^{(i)}$ that consists of the other unclicked POIs' embeddings.

As illustrated by Figure 3, we expect to "pull" the clicked POI close to the personalized prefix in the l-dimensional embedding space, and "push" the unclicked POIs away. To achieve this goal, we need to define the function of distance, i.e., $\text{h}(\cdot,\cdot)$, between the user prefix embedding and the distributed representation of a POI. In this work, we decide to use cosine similarity to measure the distance between $\mathbf{e}_{u.p.}$ and $\mathbf{e}_{poi}$ as follows:

$$\text{h}(\mathbf{e}_{u.p.}, \mathbf{e}_{poi}) = \frac{\mathbf{e}_{u.p.} \cdot \mathbf{e}_{poi}}{|\mathbf{e}_{u.p.}||\mathbf{e}_{poi}|}, \tag{8}$$

because the value range of $\text{h}(\cdot,\cdot)$ is confined between -1 and $+1$.

Overall, we need to minimize the sum of the hinge loss $\mathcal{L}_\Delta$ over the training set Δ, which is defined as follows:

$$\mathcal{L}_\Delta = \sum_{i=1}^{M} \sum_{\mathbf{e}^{*,(i)} \in \Theta^{(i)}} \max\{0, \gamma + \text{h}(\mathbf{e}^{(i)}_{u.p.}, \mathbf{e}^{(i)}_{poi}) - \text{h}(\mathbf{e}^{(i)}_{u.p.}, \mathbf{e}^{*,(i)}_{poi})\}, \tag{9}$$

where γ is a hyperparameter standing for the margin[5] of the hinge loss function, and $\mathbf{e}^{*,(i)}$ is the negative (unclicked) POI embedding sampled from $\Theta^{(i)}$.

4 EXPERIMENTS

POI-AC, as a featured function widely deployed in the search engine of Web mapping applications for industrial practice, needs to be tested thoroughly, regardless of offline or online. In this section,

[5]The margin γ is conventionally set to 1.0 given the value range of $\text{h}(\cdot,\cdot) \in [-1, +1]$.

we will report the results of extensive experiments where the performance of P^3AC and several other competitive/previous models were evaluated by multiple metrics.[6]

4.1 Comparison Models

In line with the up-to-date industrial practice on QAC [13], the POI-AC function in the search engine of Baidu Maps mainly adopts the LTR-based framework for industrial use and has been upgraded three generations as follows.

4.1.1 PAC (V1.x). The 1st generation of POI-AC function at Baidu Maps [18] contains two versions, i.e., *PAC (V1.0)* and *PAC (V1.1)*. *PAC (V1.0)* is the baseline and fundamental model in which the representative features as evidence for POI re-ranking include MPC [2], the popularity of POIs, and demographic information on users. *PAC (V1.1)* further considers the spatial-temporal characteristics of each POI and uses an additional feature vector to represent the frequency of search on specific types of POIs at different times as well as locations. However, the features of the 1st generation of POI-AC are heuristically proposed and separately optimized in the LTR models, which could hardly establish the connections between users and POIs.

4.1.2 P^2AC (V2.x). The 2nd generation of POI-AC function at Baidu Maps [18] focuses on personalized POI-AC, which is short for P^2AC. It attempts to model the connections between users and POIs. To be specific, the connection refers to the similarity between the currently queried POI and the historically queried POIs of individual users. Therefore, the key problem of the 2nd-generation POI-AC module is how to measure the similarity between POIs as additional features, which further comes up with an interesting topic of POI embedding. To obtain the distributed representations of POIs, P^2AC *(V2.0)* adopts the same idea of Word2Vec [20, 21]. To solve the problem of out-of-POI, which is similar to the problem on out-of-vocabulary in Word2Vec, P^2AC *(V2.1)* uses the advanced approach mentioned in Section 3.2 to increase information overlap between POIs, and eventually enhance the effectiveness of POI embeddings.

4.1.3 P^3AC (V3.x). In this paper, we contribute a novel end-to-end framework for POI-AC, which renovates the existing LTR-based framework for POI-AC and connects the dots of users, prefixes, and POIs. This neural framework, denoted as P^3AC *(V3.0)*, can be independently used as the POI-AC function to suggest POIs. Moreover, we can leverage the two kinds of intermediate feature vectors, i.e., the personalized prefix embeddings and the enriched POI embeddings, to produce a cosine similarity score between the two vectors as a reliable indicator. P^3AC *(V3.1)* takes the indicator as an additional feature beyond P^2AC *(V2.1)*.

4.2 Offline Assessment

4.2.1 Real-World Datasets. Baidu Maps has logged large-scale and real-world search records, where the sessions on POI-AC dominate about **80%** search traffic. A POI-AC session refers to a sequence of interactions between a user and our POI-AC function during the POI search. To be specific, it starts whenever the user is activating the search box. During the search session, our POI-AC function keeps suggesting self-update ranked lists of completed POIs as the user types in more characters to replenish the prefix, until the user clicks her desired POI in a certain list.

Table 1: The statistics of the real-world datasets for model training (abbr. *Train*), hyper-parameter tuning (abbr. *Valid*), and performance testing (abbr. *Test*). Each example of data is mainly composed of the anonymized user identifier, the prefix typed by the user, the POI list we suggested, and the exact POI that the user clicked.

Subset	#(Users)	#(Prefixes)	#(Suggested POIs)/Prefix
Train	567, 202	4, 105, 007	16.03
Valid	157, 877	385, 158	16.48
Test	30, 076	60, 426	16.02
TOTAL	663, 364	4, 311, 352	16.07

We collect a large number of POI-AC sessions from the search logs of Baidu Maps for offline evaluation. Each example of data is composed of the anonymized user identifier, the prefix typed by the user, the POI list that the POI-AC function suggested, and the exact POI that the user clicked. Table 1 shows the statistics of large-scale real-world datasets sampled from one-month search logs for model training, hyper-parameter tuning, and performance testing. We can also observe from Table 1 that some users and prefixes may appear in the training, the validation, and the test sets, simultaneously. This observation also gives us a strong intuition that the idea of *personalized prefix* tends to work on the POI-AC module.

4.2.2 Evaluation Metrics. One of the key objectives of the POI-AC function in the search engine of Web mapping services is to display the user-desired POI as high as possible, ideally to be ranked as the first one, on the suggested list. As a result, it can significantly save time and effort for many users. Ideally, we hope that the POI-AC function would display the user-desired POI in the first place for each user request. The bias of display position between the suggested POI and the user-desired POI leads to two groups of offline evaluation metrics as follows.

Success Rate (SR) is a kind of coarse metric that calculates the average percentage of the user-clicked POI appearing in the ranked list provided by the POI-AC function. Because of the limited space for display on mobile phones, Baidu Maps app can only display at most **5** suggested POIs at the first screen. Therefore, we use the metric, Success Rate at Top-K (SR@K), to denote the average percentage of ground-truth POIs that are ranked at or above the position K in the suggested lists of POI-AC. Here we consider SR@1, SR@3, and SR@5 for offline evaluation.

Another group of fine-grained metrics, including Mean Reciprocal Rank (MRR) and normalized Discounted Cumulative Gain (nDCG), concerns more about the exact position where our POI-AC function arranges the ground-truth POI in the suggested list. For example, MRR is the average of the reciprocal ranks of the

[6] The experiments were conducted in January, 2020, about half a year later than the offline and online evaluations reported by Li et al. [18]. During this period, the POI-AC function was updated several times by the UI and PM team. Because of this, the absolute values might be slightly different, however, the relative improvements are consistent.

Table 2: The experimental results of the offline evaluations on different models/versions of our POI-AC module in the search engine of Baidu Maps. All the models/versions are tested by the real-world dataset (see Table 1) and measured by multiple metrics including MRR, nDCG, and SR.

Model	Evaluation Metrics (Offline)				
	MRR	nDCG@5	SR@1	SR@3	SR@5
PAC (V1.0)	0.5991	0.6275	45.66%	68.60%	77.53%
PAC (V1.1)	0.6168	0.6478	47.34%	70.90%	79.79%
P^2AC (V2.0)	0.6316	0.6613	49.36%	72.07%	80.50%
P^2AC (V2.1)	0.6338	0.6641	49.42%	72.41%	81.02%
P^3AC (V3.0)	0.6530	0.6834	51.52%	74.32%	82.74%
P^3AC (V3.1)	**0.6759**	**0.7079**	**54.17%**	**76.78%**	**84.91%**

target POIs. As a consequence, even though two POI-AC models perform equally evaluated by SR@K, they probably show different results measured by MRR and nDCG.

4.2.3 Experimental Results. Table 2 shows the experimental results of offline assessments on the models mentioned in Section 4.1. All these models are trained, developed, and tested by the real-world datasets shown in Table 1, and are measured by the evaluation metrics mentioned in Section 4.2.2, i.e., MRR, nDCG, and SR. From the results of offline assessments, we can see that the performance of the POI-AC module at Baidu Maps gradually increases from the baseline model (i.e., *PAC (V1.0)*) to the neural framework (i.e., *P^3AC (V3.0)*) proposed in this paper. *P^3AC (V3.0)* takes a great leap forward compared with the previously deployed model *P^2AC (V2.1)*. Moreover, the best performing model *P^3AC (V3.1)* obtains significant (absolute) improvements by 4.21% MRR, 4.38% nDCG@5, as well as 4.75% SR@1, 4.37% SR@3, and 3.89% SR@5 compared with *P^2AC (V2.1)*.

4.3 Online A/B Testing

4.3.1 Traffic of POI-AC. We have updated the POI-AC module in the search engine of Baidu Maps for multiple times (i.e., from *PAC (V1.x)* to *P^3AC (V3.x)*). For each time, before being launched in production, we would routinely deploy the new model online and make it randomly serve **5%** traffic of the POI search. During the period of A/B testing, we monitor the performance of the new model and compare it with the performance of the model that has been deployed successfully online. This period conventionally lasts for at least one week. Moreover, even if the results of A/B testing show that the new model outperforms the previous model, we still need to keep an eye on the performance of the upgraded POI-AC module when 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. The other one of the critical objectives of the POI-AC function is to save users' time and effort in typing. It is quite related to user satisfaction, especially on mobile devices. Ideally, we look forward to displaying the user-desired POI at the first screen once a user activates the search box without typing in any prefix.

Table 3: The experimental results of the online A/B testing on different models/versions of our POI-AC module in the search engine of Baidu Maps. All the models/versions are tested by the real traffic of the POI search engine at Baidu Maps. Therefore, except for MRR and nDCG@5, they can be further measured by multiple core indicators on user satisfaction, such as the average number of keystrokes (Avg. #(KS)) and the average typing speed at keystrokes (Avg. Sp.@KS) in a POI-AC session, as well as Success Rate (SR).

Model	Evaluation Metrics (Online)				
	Avg. #(KS)	Avg. Sp.@KS	SR@1	SR@3	SR@5
PAC (V1.0)	4.37	14.82s	39.96%	59.81%	60.15%
PAC (V1.1)	4.15	14.77s	41.28%	60.90%	61.73%
P^2AC (V2.0)	4.08	14.68s	42.71%	61.24%	63.93%
P^2AC (V2.1)	4.02	14.57s	43.29%	61.85%	64.08%
P^3AC (V3.0)	3.78	14.32s	44.62%	63.67%	66.38%
P^3AC (V3.1)	**3.46**	**14.25s**	**45.59%**	**64.57%**	**67.39%**

In practice, we use the average number of keystrokes (Avg. #(KS)) to measure the effort on typing made by users, and the average typing speed at keystrokes (Avg. Sp.@KS) to calculate the general time spent on POI-AC.

In addition, we need to explain the reason why we still need to monitor the online performance of all the models measured by Success Rate (SR), which is also adopted by offline evaluation. This is because the real-world dataset that we used to conduct the offline evaluation only contains the POI sessions where at least one POI must be clicked.

However, some POI-AC suggestions might be ignored by a number of online users, mainly because they are in favor of directly typing in the complete name of their desired POI by means of input method editors (IME) and then click the search button[7]. This results in a phenomenon that none of the suggested POIs was clicked, which may lead to much lower performance on Success Rate (SR).

4.3.3 Experimental Results. Table 3 shows the experimental results of the online A/B testing on different models mentioned in Section 4.1. All these models were selected by the test set for offline evaluations, and we launched the best-performed ones. They are tested by 5% search traffic of Baidu Maps and measured by multiple core indicators on user satisfaction, including Avg. #(KS) and Avg. Sp.@KS in a POI-AC session. From the results, we can figure out that the time and the effort spent on our POI-AC function gradually decreases from the baseline model (i.e., *PAC (V1.0)*) to the new models that we propose in this paper. Compared with the mainstream model (i.e., *P^2AC (V2.1)*), which has kept serving online for several months, *P^3AC (V3.1)* achieves much better performance, with Avg. #(KS) and Avg. Sp.@KS reducing by 13.93% and 2.20%, respectively.

Compared with the results of the offline evaluation, it shows that we gain much lower results on SR@1, SR@3, and SR@5 in the online A/B testing. These differences verify our hypothesis that

[7]Please refer to the two search buttons located on the right-top and right-bottom positions in both Figure 1a and Figure 1b.

some POI-AC suggestions are ignored by a number of users in online search, which inevitably leads to much lower performance on SR. However, the relative improvements on these metrics are consistent with those obtained by the offline evaluation.

5 DISCUSSION

The extensive experimental results shown by the previous section have demonstrated that the proposed model, P^3AC, can achieve significant improvements on user satisfaction. In this section, we will explore the reason why P^3AC can boost both offline and online performance of the POI-AC module for Web mapping services.

Generally speaking, as an end-to-end framework for POI-AC, P^3AC can produce two kinds of intermediate feature vectors. They represent personalized prefixes typed by different users and multi-sourced information (i.e., GPS, name, category, etc.) on POIs, respectively. The cosine similarity between the two vectors is a reliable indicator to decide the rank of candidate POIs in the model of P^3AC *(V3.0)*. The significance of this indicator has already been proved by the experimental results of both offline and online evaluations, which are reported by Table 2 and Table 3, respectively. As the one and only indicator in the model of P^3AC *(V3.0)* to rank candidate POIs, it achieves the best performance compared with all previous models/versions (*PAC (V1.x)* and P^2AC *(V2.x)*).

Moreover, we are curious about how much this cosine similarity between the two intermediate vectors, as a feature, can contribute to the GBRank-based POI-AC function (i.e., P^2AC *(V2.1)*), which has kept serving online in the search engine of Baidu Maps. From the perspective of industrial practice, we need to figure out the importance of the proposed similarity feature among all features that are leveraged by the GBRank model for POI-AC. GBRank is able to provide a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more a feature is used to make critical decisions with decision trees, the higher its relative importance is allocated. After we take the new feature (i.e., P^3AC *(V3.0)*) as additional evidence and feed into the new GBRank model (i.e., P^3AC *(V3.1)*), it shows that the weight of this new feature is 19.04%. Figure 4 illustrates the histogram of the feature importance in P^3AC *(V3.1)*. It shows that this new feature ranks 2nd among all features[8], which demonstrates the effectiveness of our proposed neural framework.

6 CONCLUSIONS

This paper has presented our recent breakthrough on the industrial solution to the personalized POI-AC module in the search engine of Web mapping services. As the next-generation design and implementation of POI-AC, we propose an end-to-end neural framework (i.e., P^3AC), which has already been deployed at Baidu Maps to successfully serve hundreds of millions of users every month.

The framework is composed of three neural components, i.e., a multi-layer Bi-LSTM network to adapt to personalized prefixes, a CNN-based network to model multi-sourced information on POIs, and a triplet ranking loss function to optimize both personalized prefix embeddings and distributed representations of POIs. It realizes the novel idea that builds up the connections among users, their personalized prefixes, and correspondingly clicked POIs. As a result, both the customized prefix embeddings and the distributed representations of POIs can be updated simultaneously to achieve the objective of suggesting the user-desired POI at the top of the suggested POI list with as few keystrokes as possible.

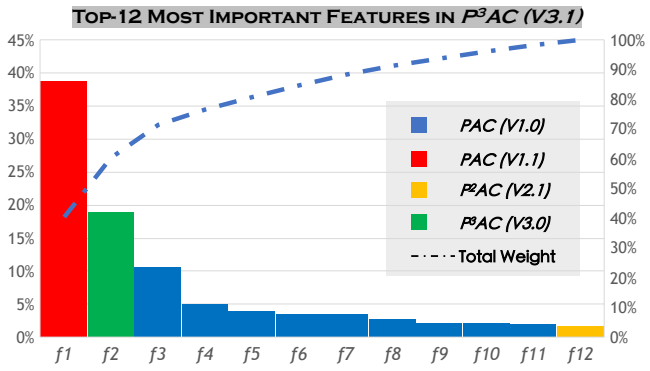


Figure 4: Histogram of the feature importance as resulted from the gradient boosting trees in the GBRank model of P^3AC *(V3.1)*. There are altogether more than 50 kinds of features as evidence to support the GBRank model for ranking candidate POIs in the POI-AC function of Baidu Maps. This figure illustrates the weights of the top-12 most important features, and the total weight of them is 95.95%. Among these features, the importance of our neural feature (i.e., $f2$, colored in green and labeled by "P^3AC *(V3.0)*") ranks 2nd with the weight of 19.04%, which demonstrates the effectiveness of our proposed P^3AC framework.

Baidu Maps has logged large-scale and real-world search records, including the prefix that a user types in, the POIs that our POI-AC module suggests, and the exact POI that the user clicks on. We can use these logs to assess the performance of P^3AC offline measured by multiple metrics, including MRR, nDCG, and SR. Extensive experiments demonstrate that P^3AC has achieved significant (absolute) improvements compared with the previously deployed model at Baidu Maps.

The search logs can just help us tell whether P^3AC works on historical data. Some other critical indicators on user satisfaction, such as the average number of keystrokes (Avg. #(KS)) and the average typing speed at keystrokes (Avg. Sp.@KS) in a POI-AC session, however, need to be monitored online. It requires us to launch this new module at Baidu Maps, serving real-world search traffic for a period of time. After deploying the upgraded model into the search engine of Baidu Maps for A/B testing online, we observe that those core indicators on user satisfaction have significantly decreased as well.

P^3AC is a multi-purpose neural framework in which both its final outputs (i.e., ranking scores of POIs) and intermediate embeddings (e.g., personalized prefix embeddings and POI embeddings) can be leveraged, not only to enhance the performance of POI-AC task, but also to improve the effectiveness of semantic search in POI retrieval.

[8]There are more than 50 kinds of features in total. For example, the MPC [2] as a feature is denoted by $f5$ in Figure 4.

7 FUTURE WORK

In this paper, we mainly propose a novel method on suggesting a list of personalized POIs for the POI-AC function in Web mapping services. The ranking results of *personalized POI suggestion* highly rely on the hundreds of candidate POIs that have been filtered by another sub-module of *candidate POI generation* for industrial practice. Therefore, we consider the future work on POI-AC for industrial practice, from a high-level view, can be developed along two lines: *candidate POI generation* and *personalized POI suggestion*.

Tasks listed below are open problems on the two lines that are deserved to explore in the future:

- *Candidate POI generation*: This sub-module is in charge of winnowing hundreds of POIs from hundreds of millions of POIs in the underlying database. It is first required to optimize the efficiency of the index strategy under the restrictions of fast response and low latency. Besides that, we also concern about the diversity of the candidate POIs based on the premise that the generated candidate POIs need to fulfill the basic intent of the search query.
- *Personalized POI suggestion*: This sub-module is responsible for providing a ranked list of the hundreds of candidate POIs. To achieve this goal, more sophisticated features need to be considered, such as a user's individual profile, her search preferences, her input habits, and even her spatio-temporal characteristics on using the POI-AC function. In addition, more effective models need to be explored to capture these personalized features that bridge users, prefixes, and POIs.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). arXiv:1409.0473 http://arxiv.org/abs/1409.0473

[2] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive Query Auto-completion. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. ACM, New York, NY, USA, 107–116. https://doi.org/10.1145/1963405.1963424

[3] Fei Cai and Maarten de Rijke. 2016. Selectively Personalizing Query Auto-Completion. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 993–996. https://doi.org/10.1145/2911451.2914686

[4] 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. https://doi.org/10.1561/1500000055

[5] Fei Cai, Shangsong Liang, and Maarten de Rijke. 2014. Time-sensitive Personalized Query Auto-Completion. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. ACM, New York, NY, USA, 1599–1608. https://doi.org/10.1145/2661829.2661921

[6] Fei Cai, Ridho Reinanda, and Maarten De Rijke. 2016. Diversifying Query Auto-Completion. *ACM Transactions on Information Systems (TOIS)* 34, 4, Article 25 (June 2016), 33 pages. https://doi.org/10.1145/2910579

[7] V. S. Dandagi and N. Sidnal. 2018. Review on Query Auto-Completion. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems*. 119–123.

[8] Huizhong Duan and Bo-June (Paul) Hsu. 2011. Online Spelling Correction for Query Completion. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. ACM, New York, NY, USA, 117–126. https://doi.org/10.1145/1963405.1963425

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

[10] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[11] Muhammad Ibrahim and Mark Carman. 2016. Comparing pointwise and listwise objective functions for random-forest-based learning-to-rank. *ACM Transactions on Information Systems (TOIS)* 34, 4 (2016), 20.

[12] Jyun-Yu Jiang and Pu-Jen Cheng. 2016. Classifying User Search Intents for Query Auto-Completion. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval (ICTIR '16)*. Association for Computing Machinery, New York, NY, USA, 49–58. https://doi.org/10.1145/2970398.2970400

[13] Manojkumar Rangasamy Kannadasan and Grigor Aslanyan. 2019. Personalized Query Auto-Completion Through a Lightweight Representation of the User Context. In *Proceedings of the SIGIR 2019 Workshop on eCommerce, co-located with the 42st International ACM SIGIR Conference on Research and Development in Information Retrieval, eCom@SIGIR 2019, Paris, France, July 25, 2019.*

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. Association for Computational Linguistics, Doha, Qatar, 1746–1751. https://doi.org/10.3115/v1/D14-1181

[15] Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 521, 7553 (2015), 436–444.

[17] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Hongyuan Zha, and Ricardo Baeza-Yates. 2015. Analyzing User's Sequential Behavior in Query Auto-Completion via Markov Processes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, New York, NY, USA, 123–132. https://doi.org/10.1145/2766462.2767723

[18] 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 (May 2020), 16 pages. https://doi.org/10.1145/3394137

[19] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (March 2009), 225–331.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119.

[22] Guy M Morton. 1966. *A computer oriented geodetic data base and a new technique in file sequencing*. Technical Report. International Business Machines Company New York.

[23] F. Schroff, D. Kalenichenko, and J. Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[24] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[25] Milad Shokouhi. 2013. Learning to Personalize Query Auto-completion. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*. ACM, New York, NY, USA, 103–112. https://doi.org/10.1145/2484028.2484076

[26] Yang Song, Dengyong Zhou, and Li-wei He. 2011. Post-ranking Query Suggestion by Diversifying Search Results. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. ACM, New York, NY, USA, 815–824. https://doi.org/10.1145/2009916.2010025

[27] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning Fine-Grained Image Similarity with Deep Ranking. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*. IEEE Computer Society, Washington, DC, USA, 1386–1393. https://doi.org/10.1109/CVPR.2014.180

[28] Stewart Whiting, Andrew James McMinn, and Joemon M Jose. 2013. Exploring Real-Time Temporal Query Auto-Completion. In *DIR Workshop*. 12–15.

[29] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 1192–1199. https://doi.org/10.1145/1390156.1390306

[30] Aston Zhang, Amit Goyal, Weize Kong, Hongbo Deng, Anlei Dong, Yi Chang, Carl A. Gunter, and Jiawei Han. 2015. adaQAC: Adaptive Query Auto-Completion via Implicit Negative Feedback. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, New York, NY, USA, 143–152. https://doi.org/10.1145/2766462.2767697

[31] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A Regression Framework for Learning Ranking Functions Using Relative Relevance Judgments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. ACM, New York, NY, USA, 287–294. https://doi.org/10.1145/1277741.1277792